



# HostExplorer®

*Programmer's Guide*

## **HostExplorer Programmer's Guide**

### **Version 12**

**Published in Canada — July 2006**

#### **Hummingbird Ltd. — Corporate Headquarters**

1 Sparks Avenue • Toronto, Ontario • M2H 2W1 • Canada

Toll Free Canada/U.S.A. 1 877 FLY HUMM (359 4866)

Tel +1 416 496 2200 • Fax +1 416 496 2207 • E-mail [getinfo@hummingbird.com](mailto:getinfo@hummingbird.com)

For more information, visit [connectivity.hummingbird.com](http://connectivity.hummingbird.com)

**RESTRICTED RIGHTS LEGEND** Unpublished rights reserved under the copyright laws of the United States. The SOFTWARE is provided with restricted rights. Use, duplications, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) of the Rights in Technical Data and Computer Software clause at DFARS 252.27-7013, subparagraph (c) (1) and (2) (a) (15) of the Commercial Computer Software-Restricted Rights clause at 48 CFR 52.227-19, as applicable, similar clauses in the FAR and NASA FAR Supplement, any successor or similar regulation.

Information in this document is subject to change without notice and does not represent a commitment on the part of Hummingbird Ltd. Not all copyrights pertain to all products.

Copyright © 2006 Hummingbird Ltd. All rights reserved. Trademarks and logos are the intellectual property of Hummingbird Ltd.

Connectivity Kerberos™, Connectivity Secure Shell™, Connectivity SecureTerm®, Connectivity SSL™, Exceed®, Exceed 3D™, Exceed Connectivity Suite™, Exceed onDemand®, Exceed onDemand Client™, Exceed onDemand Server™, Exceed onDemand Server Manager™, Exceed PowerSuite™, Exceed XDK™, HostExplorer®, HostExplorer Connectivity Suite™, Host Access Services™, HostExplorer Print Services™, HostExplorer Web™, Hummingbird Basic™, Hummingbird Certificate Manager™, Hummingbird Connectivity™, Hummingbird Connectivity Suite™, Hummingbird Deployment Packager™, Hummingbird Deployment Wizard™, Hummingbird FTP™, Hummingbird InetD™, Hummingbird Proxy Server™, Hummingbird SOCKS Client™, NFS Maestro™, NFS Maestro Client™, NFS Maestro Gateway™, NFS Maestro Server™, NFS Maestro Server™ Enterprise Edition, NFS Maestro Solo™, NFS Maestro Tuner™, TXP™, TXPM™, and Xweb® are trademarks or registered trademarks of Hummingbird Ltd. and/or its subsidiaries.

All other copyrights, trademarks, and tradenames are the property of their respective owners.

#### **TECHNICAL ACKNOWLEDGEMENTS**

**Exceed® and Exceed PowerSuite™** include the following third-party software:

Portions of the code have been contributed by Massachusetts Institute of Technology.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>) Copyright © 1999-2000 The Apache Software Foundation. All rights reserved.

The technology used by Smart card Manager is derived from the RSA Security Inc. PKCS#11 Cryptographic Token Interface (Cryptoki)

**Exceed 3D™, Exceed PowerSuite™ and Exceed XDK™** include the following third-party software:

OpenGL is a registered trademark of Silicon Graphics Inc.

**Exceed XDK™** includes the following third-party software:

Portions of the code have been contributed by Massachusetts Institute of Technology

**Connectivity Secure Shell™ and Connectivity SecureTerm®** include the following third-party software:

This product includes software developed by Massachusetts Institute of Technology. Copyright 1992-2005 by the Massachusetts Institute of Technology. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>) Copyright© 1998-2005 The OpenSSL Project. All rights reserved.

This product includes cryptographic software written by Eric Young ([eay@cryptsoft.com](mailto:eay@cryptsoft.com)) Copyright © 1995-1998 Eric Young ([eay@cryptsoft.com](mailto:eay@cryptsoft.com)) All rights reserved.

**FONTS** The fonts distributed are included free of charge. Some of the fonts were donated by Adobe Systems Inc., Bitstream Inc., International Business Machines Corporation, Hewlett Packard Company, Massachusetts Institute of Technology, the Open Group and Sun Microsystems Inc. to Hummingbird Ltd. for redistribution in Exceed®, Exceed PowerSuite™ and Exceed onDemand®. Each font contains a copyright message describing the owner of the font.

**DISCLAIMER** Hummingbird Ltd. software and documentation has been tested and reviewed. Nevertheless, Hummingbird Ltd. makes no warranty or representation, either express or implied, with respect to the software and documentation other than what is expressly provided for in the Hummingbird Ltd. Software License Agreement included within the software. In no event will Hummingbird Ltd. be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the software or documentation. In particular, Hummingbird Ltd. shall have no liability for any programs or data used with the software, including the cost of recovering such programs or data.

# **Related Documentation and Services**

## **Manuals**

All manuals are available in both print and PDF format. PDFs require Adobe Acrobat Reader and are installed only if you perform a Complete installation, or if you select them during a Custom installation. If the PDFs are installed, you can access them from the Hummingbird Connectivity application group on the Start menu.

## **Help**

The online Help is a comprehensive, context-sensitive collection of information regarding your Hummingbird product. It contains conceptual and reference information and detailed, step-by-step procedures to assist you in completing your tasks.

## **Release Notes**

The Release Notes contain descriptions of new features and details on release-time issues for all Hummingbird Connectivity products and components. They are available in HTML format and are installed with the software. You can access the Release Notes from the Hummingbird Connectivity application group on the Start menu. You can also access the release notes during installation setup. It is recommended that you read the sections that apply to the applications you want to install before installing them.

## **Hummingbird Exposé Online**

Hummingbird Exposé Online is an electronic mailing list and online newsletter. It was created to facilitate the delivery of Hummingbird product-related information. It also provides tips, help, and interaction with Hummingbird users. To subscribe/unsubscribe, browse to the following web address:

<http://www.hummingbird.com/expose/about.html>

## **User Groups and Mailing Lists**

The user group is an unmoderated, electronic mailing list that facilitates discussion of product-related issues to help users resolve common problems and to provide tips, help, and contact with other users.

### **To join a user group:**

Send an e-mail to `listserv@hummingbird.com`. Leave the Subject line blank. In the body of the e-mail message, type the following:

`subscribe hostexplorer-users Your Name`

### **To unsubscribe:**

Send an e-mail to `listserv@hummingbird.com`. Leave the Subject line blank. In the body of the e-mail message, type the following:

`unsubscribe hostexplorer-users Your Name`

### **To post a messages to the user group:**

Send your e-mail to:

`hostexplorer-users@hummingbird.com`

### **To search the mailing list archives:**

Go to the following web site:

<http://www.hummingbird.com/support/usergroups.html>

---

# Contents

<b>Chapter 1: Introducing HostExplorer Programming</b>	<b>1</b>
Introducing HostExplorer Programming .....	2
<b>Chapter 2: Customizing HostExplorer</b>	<b>3</b>
Introducing HostExplorer APIs .....	4
OLE Support .....	8
OLE Automation .....	8
OLE Objects .....	9
Methods and Properties of the Application Object .....	10
Methods and Properties of the Hosts Object .....	21
Methods and Properties of the Host Object .....	25
Methods and Properties of the Area Object .....	76
Methods and Properties of the Field Object .....	90
Methods and Properties of the Cfg3270, Cfg5250, and CfgVT Objects ..	99
Unsupported OLE Methods and Properties .....	168
Differences Between Hummingbird Basic and WinWrap Basic .....	169
File Transfer Options .....	188
File Transfer Options .....	188
General Options .....	188
CMS-Specific Options on Upload .....	189
TSO-Specific Options on Upload .....	189
MUSIC-Specific Options on Upload .....	190
Special Sequences .....	191
3270/5250 Special Sequences .....	191
VT Special Sequences .....	193
About the Terminal Objects .....	195
Methods of the Terminal Objects .....	195
Properties of the Terminal Objects .....	199

About COM Objects .....	207
Relationship Between COM Objects .....	208
Unsupported COM Methods and Properties .....	208
Renamed or Moved COM Methods, Properties, and Interfaces .....	210
About the Profile Object .....	263
Profile Interface .....	263
ProfileTerminal Interface .....	343
ProfileGraphics Interface .....	406
ProfileKeyboard Interface .....	418
ProfileMouse Interface .....	445
ProfileToolbar Interface .....	449
ProfileTrackMenu Interface .....	465
ProfileTranslationTable Interface .....	470
ProfileVTCharset Interface .....	472
ProfileEdit Interface .....	481
ProfileColor Interface .....	549
ProfileFileTransfer Interface .....	551
Profile Security Interface .....	656
ProfileDisplay Interface .....	669
ProfileCursor Interface .....	700
ProfileFonts Interface .....	708
ProfilePCPrint Interface .....	713
ProfilePrintScreen Interface .....	722
ProfilePrintSession Interface .....	753
ProfileCapture Interface .....	768
ProfileHostPrinting Interface .....	779
ProfileHotspots Interface .....	818
ProfileEvents Interface .....	827
ProfileConnection Interface .....	832
ProfileSessionWindow Interface .....	911
ProfileSound Interface .....	944
Data Types of the Profile Object .....	948
About the Parser Objects .....	949
Methods of the Parser Objects .....	955
Properties of the Parser Objects .....	1060
Data Types of the Parser Objects .....	1175

About the Transport Objects .....	1176
Methods of the Transport Objects .....	1179
Properties of the Transport Objects .....	1203
Data Types of the Transport Objects .....	1282
About OHIO .....	1283
OhioManager Interface .....	1283
OhioSessions Interface .....	1289
OhioSession Interface .....	1293
OhioScreen Interface .....	1302
OhioOIA Interface .....	1324
OhioFields Interface .....	1332
OhioField Interface .....	1338
OhioPosition Interface .....	1353
Data Types of OHIO .....	1357
About Legacy APIs .....	1364
EHLLAPI and WinHLLAPI DLL Support .....	1364
Special EHLLAPI and WinHLLAPI Flags .....	1366
EHLLAPI Support .....	1367
EHLLAPI Module .....	1367
Irma Compatibility Mode .....	1367
EHLLAPI Calls .....	1367
Visual Basic Interface .....	1383
EHLLAPI Support in VT and NVT Modes .....	1388
EHLLAPI Development Files .....	1389
NVT Mode Functions .....	1389
NVT Mode Exceptions .....	1390
Configuration Tips .....	1391
WinHLLAPI Module .....	1392
WinHLLAPI Development Files .....	1392
Dynamic Data Exchange (DDE) .....	1393
What is DDE? .....	1393
How Does DDE Work? .....	1393
DDE Terminology .....	1402
DDE Sample Code .....	1404
System Topic .....	1405

<b>Chapter 3: Customizing FTP</b>	<b>1408</b>
Introducing FTP API .....	1409
Sample Client Source Code .....	1410
Customizing FTP .....	1410
Initialization and Termination Functions .....	1410
Directory Functions .....	1411
Connection and Access Functions .....	1413
File Transfer Functions .....	1415
Deleting and Renaming Functions .....	1417
Tree Functions .....	1418
Transfer Type Settings .....	1420
Local File Creation Settings .....	1422
Remote Command Functions .....	1424
FTP Session Settings .....	1426
Miscellaneous FTP Command Functions and Settings .....	1429
Error Handling Function .....	1432
Customizing FTP OLE .....	1432
Introducing FTP OLE API .....	1432
Creating an OLE Script .....	1433
IHclFtpEngine Object .....	1434
IHclFtpSession Object .....	1435
IHclFtpSessions Object .....	1437
User-Defined Types .....	1470
<b>Chapter 4: Customizing WyseTerm</b>	<b>1471</b>
Introducing WyseTerm API .....	1472
Configuring Telnet Using DDE .....	1473
Conversing with Telnet using DDE .....	1473
Request and Execute Syntax .....	1473
DDE Topics Available in Telnet .....	1474
System Topic Request Items (HTelnet\System) .....	1475
Terminal Topic Items (HTelnet\Terminal) .....	1476
Emulation Type Topic Items (HTelnet\EmulationType) .....	1483
Page Topic Items (HTelnet\Page) .....	1490

Configuring Telnet Using OLE .....	1496
Automating Telnet using OLE .....	1496
Starting Telnet in Hummingbird Basic Using OLE .....	1496
Property and Method Syntax .....	1497
Available OLE Objects in Telnet .....	1498
WyseTerm Object .....	1499
Emulation Object .....	1506
Page Object .....	1521
Title Object .....	1527
<b>Appendix: Accessibility and Technical Support</b>	<b>1529</b>
General Accessibility .....	1531
Microsoft Accessibility Options .....	1532
Technical Support .....	1533
<b>Index</b>	<b>1535</b>



# Chapter 1

---

## Introducing HostExplorer Programming

## 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.

## Chapter 2

---

### Customizing HostExplorer

## 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:

- EHLLAPI (Extended HLLAPI)/ WinHLLAPI (Windows HLLAPI)
- OLE (Object Linking and Embedding)
- COM (Component Object Model)
- OHIO (Open Host Interface Objects)

### OLE Automation

OLE Automation is an application programming interface that lets you control one or more emulation sessions from any standard programming environment such as Visual Basic, Visual C++, Delphi, etc. OLE Automation communicates with emulation sessions running in the main HostExplorer process.

For more information, see “OLE Automation” on page 8.

## HostExplorer COM Objects

COM Objects allow you to embed and control the HostExplorer objects in your application. You can embed the visual Terminal object or simply use the Parser and Transport objects to communicate directly with the host. Unlike OLE Automation where the emulation sessions run in a separate process, when you embed the COM objects, they run in your process.

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

- Profile object
- Parser objects
- Transport objects

The functional diagram, “Relationship Between COM Objects” on page 208, 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 control (for example, HETerminal) 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 Basic, you must create the object as follows:

```
Set OManager=CreateObject ("HEOhio.OhioManager")
```

## Legacy APIs

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

- EHLLAPI (Extended HLLAPI) and WinHLLAPI (Windows HLLAPI)—Allow other Windows programs to communicate and control HostExplorer terminal emulators. HostExplorer provides HLLAPI DLLs compatible with Attachmate(r) Extra!, NetManage Rumba, and IBM Personal Communications.
- DDE (Dynamic Data Exchange)—An API 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 OLE Automation, COM and OHIO, you can still use them to write applications and thus avoid rewriting your own code. HostExplorer’s support of these earlier APIs helps maximize an organization’s investment in its development.

## OLE Support

### 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 HEOleAut.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.

### OLE Automation Reference

#### 3270 5250 VT

You can use OLE Automation clients to write and use a variety of tools including Hummingbird Basic, Visual Basic, C++ and more. HostExplorer provides a number of OLE Automation Objects that you can use to develop scripts to control HostExplorer and its operations. You can view OLE Automation Objects as methods and properties.

## OLE Objects

### 3270 5250 VT

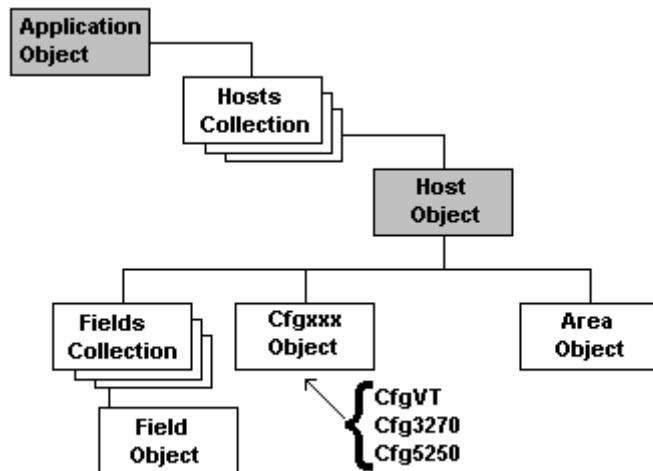
HostExplorer provides methods and properties for the following OLE objects:

- 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.



## 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
- ExitAll
- GetCurrentDir
- GetFilePath
- GetProfileString
- HostFromProfile
- Hosts Collection
- HostFromShortName
- NewSession
- StartSession
- Word
- WriteProfileString

## CurrentHost

### Property, Application Object 3270 5250 VT

The CurrentHost property is a pointer to the current host object, which is the session with the last focus. If there is only one session, this object represents a pointer to that session.

**Note:** Because this object represents the session that last had the focus, the value of CurrentHost may change throughout your program when you start new sessions. To prevent this, use the Host object by assigning a static reference to the current session. For example:

```
Dim Host as Object  
Set Host = HE.CurrentHost  
... Now use Host.* object ...
```

#### Syntax

##### CurrentHost

#### Example

```
Sub Main  
    Dim HE as Object  
    Dim Host as Object  
  
    Set HE = CreateObject( "HostExplorer" )  
    Set Host = HE.CurrentHost  
  
    Host.PrintScreen  
  
End Sub
```

## ExitAll

### Method, Application Object 3270 5250 VT

You can use this method to immediately terminate and close all active sessions. Make this method the last call to HostExplorer because it automatically unloads the emulator.

#### Syntax

##### **ExitAll**

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ...
    HE.ExitAll
End Sub
```

## GetCurrentDir

### Method, Application Object 3270 5250 VT

The GetCurrentDir method returns the current working directory.

#### Syntax

##### **szDir\$ = GetCurrentDir**

where *szDir\$* is the current working directory.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    Path$ = HE.GetCurrentDir
    Print Path$
End Sub
```

## GetFilePath

### Method, Application Object 3270 5250 VT

The GetFilePath method displays a dialog box that prompts you for a file path. The returned string is a complete path and file name. If you click the Cancel button, the system returns a null string.

#### Syntax

```
szPath$ = GetFilePath ( szDefName$, szDefExt$, szDefDir$, szTitle$,  
iOption% )
```

where:

- *szPath\$*—The path string returned from the dialog box if the user pressed OK. Selecting a different directory changes the current directory of the application.
- *szDefName\$*—Used to set the initial file name. If you omit this by specifying an empty string (" "), \*.szDefExt\$ is used.
- *szDefExt\$*—Used to initially show files whose extension matches this string value. You can specify multiple extensions by using a comma (,) as the separator. If you omit this by specifying an empty string (" "), \* is used.
- *szDefDir\$*—Used to set the initial directory. If you omit this by specifying an empty string (" "), the system uses the current directory.
- *szTitle\$*—Used to set the title of the dialog box. If you omit this by specifying an empty string (" "), "Open" is used.
- *iOption%*—The numeric value that determines the file selection options. If you omit this, the system uses a zero. The available options are: 0—Only allow the user to select a file that exists, 1—Confirm creation when the user selects a file that does not exist, 2—Allow the user to select any file whether it exists or not, 3—Confirm overwrite when the user selects a file that exists

#### Example

```
Sub Main  
    Dim HE as Object  
    Set HE = CreateObject( "HostExplorer" )  
  
    Print HE.GetFilePath( "Hostex32.exe", "EXE", "Program Files", "Get File  
Path", 0 )  
  
End Sub
```

## GetProfileString

### Method, Application Object 3270 5250 VT

You can use the GetProfileString method to retrieve a value from any .ini file.

The `szSection$` parameter is the section name. This parameter must always be present. The `szKey$` parameter is the key name. You can pass an empty string as this parameter to read all the keys in the section. The system separates all keys by newline characters. The `szINIFileName$` parameter is the file name. If you omit a path, the system looks for the file in the Windows directory.

## Syntax

```
szValue$ = GetProfileString( szSection$, szKey$, szINIFileName$ )
```

where:

- *szSection\$*—The section of the file (the name in square brackets).
- *szKey\$*—The key to read.
- *szINIFileName\$*—The name of the .ini file to read.
- *szValue\$*—The string value read from the .ini file.

## Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szBeep$ = HE.GetProfileString( "Windows", "Beep", "c:\winnt\win.ini" )
    MsgBox "The content of that key is: " & szBeep$, , "Output"
End Sub
```

## HostFromProfile

### Method, Application Object 3270 5250 VT

The HostFromProfile method is a pointer to the host object that was started from the specified session profile.

## Syntax

```
Set Host = HostFromProfile( strProfile$ )
```

where *strProfile\$* is the name of the session profile.

**Example**

```
Sub Main
    Dim HE as Object
    Dim Host as Object

    Set HE = CreateObject( "HostExplorer" )
    Set Host = HE.HostFromProfile( "c:\winnt\Profiles\AllUsers\_
        Application Data\Hummingbird\Connectivity\10.00\Profile\_
        MyProfile.hep" )

End Sub
```

Note: You do not need to specify an absolute filepath for the strProfile\$ parameter. If you specify only a name.ext, the method will assume that the profile is in the main user profile space.

## **HostFromShortName**

### **Method, Application Object 3270 5250 VT**

The HostFromShortName method is a pointer to the host object that contains the specified session short name.

**Syntax**

```
Set Host = HostFromShortName( strShortName$ )
```

**Parameters**

*strShortName\$*—The name of the session short name.

**Example**

```
Sub Main
    Dim HE as Object
    Dim Host as Object

    Set HE = CreateObject( "HostExplorer" )
    Set Host = HE.HostFromShortName("A")

End Sub
```

## Hosts Collection

**Object, Application Object 3270 5250 VT**

The Hosts Collection object accesses available sessions by their unique session ID.

**Note:** Hosts(n) may or may not be set to Nothing. Each session is assigned a new unique index. You must use the Next method to properly enumerate a Hosts collection.

Refer to the Next property for an example of the suggested way to enumerate all of the current host objects.

### Syntax

```
Set Host = Hosts( UniqueSessID% )
```

where *UniqueSessID%* is the unique ID for the session.

### Example

```
Sub Main
    Dim HE as Object
    Dim Host as Object

    Set HE = CreateObject( "HostExplorer" )

    Set Host = Nothing      ' Prepares 'Next' method to get the first host

    for i = 1 to HE.Hosts.Count
        Set Host = HE.Hosts.Next(Host)
        MsgBox "The current short name is: " + Host.ShortName
    next i

End Sub
```

## NewSession

### Method, Application Object 3270 5250 VT

**Note:** HostExplorer continues to support this method. However, the Open method is more current. Not only can you use the Open method to create a new session, but you can also use it to specify a session profile or terminal type. If there are any programs or macros that used the NewSession method in version 6.x, you must review (and possibly modify) them for versions 7.0 or newer.

You can use the NewSession method to create a new session control block within HostExplorer. Use this method if you want to create a new session from scratch without using a previously saved profile. The NewSession method returns a value representing the unique session ID of the new session. The first available session has a session ID of 0.

The default terminal type is 3270. To change the default, add the following line to the `HEOLEAut.Settings` section in the global `hostex.ini` file, located in the HostExplorer 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. For more information, see “Default Locations for User Files” on page 1407.

```
[HEOLEAut.Settings]
New Session Terminal Type = x
```

where `x` is one of the following values:

- 1—TERMINAL\_3270
- 2—TERMINAL\_TELNET
- 4—TERMINAL\_5250

## Syntax

`UniqueSessID% = NewSession`

where `UniqueSessID%` is the ID of the session started.

**Example**

```
'$Include:"-E\hebasic.ebh"

Sub Main
    Dim Host as Object
    Dim HE as Object
    Dim Cfg as Object
    Set HE = CreateObject( "HostExplorer" )

    UniqueSessID% = HE.NewSession
    Set Host = HE.Hosts(UniqueSessID%)
    Set Cfg = Host.Cfg
    Host.Model =TELNET_VT220' Set terminal model next
    Host.ConnectBy = IO_TELNET' Set transport next

    Cfg.Host = "myhost.mydomain"
    Cfg.ConnectTimeout = 30
    Cfg.TCPPort = 23

    Host.Connect
End Sub
```

## StartSession

### Method, Application Object 3270 5250 VT

You can use the StartSession method to start a new session from a profile saved on disk. This method takes one parameter that is the name of the profile. You can specify the `ProfileName` in one of two ways:

- Specify a complete directory path to the `MyProfile.hep` file in the `Profile\3270` directory where the user files are stored on your machine. For the appropriate directory path for your platform, refer to the list of

- the default locations for the user files. For more information, see “Default Locations for User Files” on page 1407.
- Specify the older format if the profile is in a first-level folder "MyProfile.MyFolder".

**Note:** HostExplorer continues to support this method. However, the Open method is more current. Not only can you use the Open method to create a new session, but you can also use it to specify a session profile or terminal type. If there are any programs or macros that used the StartSession method in version 6.x, you must review (and possibly modify) them for versions 7.x and 8.x.

The value UniqueSessID% returned is the session ID for the host session created. You can use this session ID to access the Host object directly.

## Syntax

*UniqueSessID% = StartSession( szProfileName\$ )*

*szProfileName\$*—The name of the profile to load and connect.

*UniqueSessID%*—The ID of the session started.

## Example

```
Sub Main
    Dim HE as Object
    Dim Host as Object
    Set HE = CreateObject( "HostExplorer" )

    UniqueSessID% = HE.StartSession( "Default 3270" )

    Set Host = HE.Hosts( UniqueSessID% )
End Sub
```

## Word

### Method, Application Object 3270 5250 VT

You can use the Word method to parse out the nth word of a string. The first parameter is the string that you want to parse. The second parameter contains a string of valid delimiters to parse the words. The third parameter is the index of the word to retrieve. Unlike other methods, the first word of a string has an index of 1.

#### Syntax

```
szWord$ = Word( szString$, szParse$, index% )
```

where:

- *szString\$*—The string to parse.
- *szParse\$*—The string of delimiters to parse *szString\$*.
- *index%*—The index of the word to parse out.
- *szWord\$*—The word parsed by the method.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szWord$ = HE.Word("I have a lazy brown dog", " ", 2 )
    MsgBox "The second word in the string is " & """ & szWord$ & """

End Sub
```

## WriteProfileString

### Method, Application Object 3270 5250 VT

You can use the WriteProfileString method to write an INI key value to a file. The *szSection\$* parameter is the section name and is mandatory. The *szKey\$* parameter is the key name. If you pass an empty string as the *szKey\$* and *szValue\$* parameters, the entire section is deleted. The *szValue\$*

parameter is the value for the key. If you pass an empty string as the `szValue$`, then the key is deleted from the section. The `szINIFileName$` parameter is the file name. If you omit a path, the system looks for the file in the Windows directory.

## Syntax

`WriteProfileString szSection$, szKey$, szValue$, szINIFileName$`

where:

- `szSection$`—The section of the file (the name is in square brackets).
- `szKey$`—The key to write.
- `szINIFileName$`—The name of the .ini file to write.

## Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.WriteProfileString "Windows", "Beep", "No", "c:\winnt\win.ini"
End Sub
```

## 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:

- CloseAll
- Count
- Item
- Next
- Open

**Note:** These methods and properties are valid for all terminal types (that is, TN3270, TN5250, and Telnet).

## CloseAll

### Method, Hosts Object 3270 5250 VT

You can use this method to immediately terminate and close all active sessions. Make this method the last call to HostExplorer because it automatically unloads the emulator.

#### Syntax

```
Hosts.CloseAll
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ...
    HE.Hosts.CloseAll
End Sub
```

## Count

### Property, Hosts Object 3270 5250 VT

This property returns the total number of sessions available.

#### Syntax

```
count% = Hosts.Count
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ...
    count% = HE.Hosts.Count
End Sub
```

## Item

### Property, Hosts Object 3270 5250 VT

This property returns the pointer to the session (n) where n is a unique session ID.

#### Syntax

```
Set Host = Hosts.Item (ID%)
```

where ID% is the ID of the session.

**Example**

```
Sub Main
    Dim HE as Object
    Dim Host as Object
    Set HE = CreateObject( "HostExplorer" )

    ...
    Set Host = HE.Hosts(0)
End Sub
```

**Next****Property, Hosts Object 3270 5250 VT**

This property returns the next session after the specified session. If you specify "Null", it returns the first session.

**Syntax**

```
Set Host = Hosts.Next (HostX)
```

where *HostX* is the specified host.

**Example**

```
Sub Main
    Dim HE as Object
    Dim Host as Object
    Set HE = CreateObject( "HostExplorer" )

    Set Host = Nothing      ' Prepares 'Next' method to get the first host

    for i = 1 to HE.Hosts.Count
        Set Host = HE.Hosts.Next(Host)
        MsgBox "The current short name is: " + Host.ShortName
    next i

End Sub
```

**Open****Method, Hosts Object 3270 5250 VT**

You can use the Open method to start a new session from a profile saved on disk or to create a new session from scratch without using a previously saved profile. This method takes one parameter that is the name of the profile. You can specify the ProfileName in one of two ways:

- Specify a complete directory path to the MyProfile.hep file in the Profile\3270 directory where the user files are stored on your machine.

For the appropriate directory path for your platform, refer to the list of the default locations for the user files. For more information, see “Default Locations for User Files” on page 1407.

- Specify the older format if the profile is in a first level folder "MyProfile.MyFolder".

## Syntax

```
Set Host = Hosts.Open( szProfileName$ )
Set Host = Hosts.Open( TERMINAL_TELNET)
where szProfileName$ is the name of the profile to load and connect.
```

## Example

The following example involves opening an existing profile:

```
Sub Main
    Dim HE as Object
    Dim Host as Object
    Set HE = CreateObject( "HostExplorer" )

    Set Host = HE.Hosts.Open( szProfileName$ )
```

```
End Sub
```

The following example involves opening a new profile:

```
'$Include: "-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Dim MyHost as Object

    Set HE = CreateObject( "HostExplorer" )

    'Default connection method - IO_TELNET
    Set MyHost = HE.Hosts.Open( TERMINAL_3270 )
    MyHost.Cfg.Host = "MyHost.MyDomain"

    'Connect with Microsoft SNA Server
    'Set MyHost = HE.Hosts.Open( TERMINAL_3270, IO_MSSNA)
    'MyHost.Cfg.LUName = "MYLUNAME"

    MyHost.Cfg.ConnectTimeout = 30
    MyHost.Connect

End Sub
```

## 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:

- Activate
- AllowClose
- AllowUpdates
- Area
- BFPress
- BFStatus
- Bytes
- Capture
- CaptureOIA
- Close
- Columns
- Connect
- ConnectBy
- ConnectErrorStatus
- ConnectRC
- Cursor
- CursorRC
- Device3279
- ProtectedText
- PSReserved
- PSUpdated
- PutText
- QueryCloseRequest
- QuickKeyFile
- ReceiveFile
- Restore
- Row
- Rows
- RunCmd
- RunQuickKey
- SaveQuickKeyFile
- SaveScreen
- SaveScrollbar
- Search
- SendFile
- SetFont

- Disconnect
- EAB
- FieldID
- Fields Collection
- FontLarger
- FontSmaller
- Hide
- HideToolbar
- HighlightText
- Index
- InsertMode
- IsConnected
- IsXfer
- Keyboard
- Keys
- LoadQuickKeyFile
- Maximize
- Minimize
- Model
- MouseToCursor
- ShortName
- Show
- ShowToolbar
- SilentConnect
- SystemColor
- TerminalMode
- Text
- TextRC
- TN3270
- TrackMenu
- Update
- WaitConnected
- WaitForIO
- WaitForString
- WaitForStringRC
- WaitIdle
- WaitPSUpdated
- WaitXfer
- XferCount

## Activate

**Method, Host Object 3270 5250 VT**

You can use this method to activate the host object. This brings the host object to focus by making the screen window the top-level window on the desktop.

### Syntax

**Activate**

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Bring this window to the top
    HE.CurrentHost.Activate
End Sub
```

## AllowClose

### Property, Host Object 3270 5250 VT

You can use this property to allow or prevent the user from closing the current session. To allow the user to close the session, set the AllowClose property to TRUE. To prevent the user from closing the session, set the property to FALSE.

**Syntax**

**AllowClose** = TRUE/FALSE

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bAllowClose = HE.CurrentHost.AllowClose
    HE.CurrentHost.AllowClose = FALSE
End Sub
```

## AllowUpdates

### Property, Host Object 3270 5250 VT

You can use this property to return or set the screen updates flag. When disabled, the system does not perform screen updates. Use this flag to optimize performance or prevent a user from seeing updates while you are performing transactions.

**Syntax**

**AllowUpdates** = TRUE/FALSE

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bUpdates = HE.CurrentHost.AllowUpdates
    HE.CurrentHost.AllowUpdates = FALSE
End Sub
```

## Area

**Method, Host Object 3270 5250 VT**

This method is used to construct an area object. Specify the top-left and bottom-right coordinates of the area and the type of area (stream or block). See Methods and Properties of the Area Object for more information.

**Syntax**

```
cArea$ = BArea( iTopRow%, iTopLeftColumn%, iBottomRow%, iBottomRightColum
n%, iUnused%, iAreaType% )
where
iTopRow%, iBottomRow = 1 to Rows
iTopLeftColumn, iBottomRightColumn = 1 to Columns
iUnused = 0
iAreaType = 2 for STREAM or 3 for BLOCK
```

**Example**

```
Sub Main
    Dim HE as Object
    Dim Area as Object

    Set HE = CreateObject( "HostExplorer" )

    ' Create a rectangular area from 1,1 to 10,80
    Set Area = HE.CurrentHost.Area( 1, 1, 10, 80, 0, 3)

End Sub
```

## BFPress

### Method, Host Object 3270

You can use this method to send the modified data to the host system. The parameter allows you to specify the actual AID (Attention Identifier) sent to the host system in the 3270 datastream. This method is used to support RPQ 7J0048.

#### Syntax

```
BFPress iAIDValue%
```

where *iAIDValue%* is the AID key value for the host datastream.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.BFPress &h7D' Press Enter key
End Sub
```

## BFStatus

### Property, Host Object 3270

This read-only property is used to return the status of the extended function keys used to support RPQ 7J0048. The string returned contains the status for each extended function key. There are 48 function keys. The status character for each key can be: '0' - Key is Off, '1' - Key is On, '2' - Key is flashing. Therefore, to check the status of BF Key 10, you would check the tenth character in the string.

#### Syntax

```
szStatus$ = BFStatus
```

where *szStatus\$* is the status string for all BF keys.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    status$ = HE.CurrentHost.BFStatus
End Sub
```

## Bytes

### Property, Host Object **3270 5250 VT**

This read-only property returns the number of bytes in the presentation space of the host object. It is equal to the number of columns multiplied by the number of rows.

#### Syntax

```
iBytes% = Bytes
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iBytes% = HE.CurrentHost.Bytes
    MsgBox "The number of Bytes in the presentation space is & iBytes%, , "
Information"
End Sub
```

## Capture

### Property, Host Object **3270 5250 VT**

You can use this read/write property to get or set the capture mode for the host object. You can retrieve the current setting for the capture mode or you can set a new value.

You can enable constant capturing of TN3270 and TN5250 panels as they are received from the host. In Telnet, you can capture data in Text or Raw Mode. The capture file is set using the SaveFile property.

#### Syntax

```
Capture = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bMode = HE.CurrentHost.Capture
    HE.CurrentHost.Capture = TRUE
End Sub
```

## CaptureOIA

### Property, Host Object 3270 5250

You can use this read/write property to get or set the capture operator information area (OIA) flag for the host object. When the value is TRUE, the system captures the OIA in every screen. When the value is FALSE, the system does not capture the OIA.

#### Syntax

**CaptureOIA** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bMode = HE.CurrentHost.CaptureOIA
    HE.CurrentHost.CaptureOIA = TRUE
    HE.CurrentHost.SaveScreen "c:\screen.txt"
End Sub
```

## Close

### Method, Host Object 3270 5250 VT

You can use this method to close (terminate) the host session and object. This method immediately terminates any host connection and closes the session window.

**Note:** Do not place any calls against the host object after this call.

#### Syntax

**Close**

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Close the current host session
    HE.CurrentHost.Close
End Sub
```

## Columns

### Property, Host Object **3270 5250 VT**

This read-only property returns the number of columns currently defined in the Host object.

#### Syntax

```
iCols% = Columns
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iColumns = HE.CurrentHost.Columns
    MsgBox "The number of Columns in presentation space is " & iColumns, ,
    "Information"
End Sub
```

## Connect

### Method, Host Object **3270 5250 VT**

You can use this method to connect a session to a host system. You can only make this call if your current session does not already have a host connection. This method maps directly to the File, Connect menu.

You can set the host system name and other connection parameters through the Cfg3270, Cfg5250, and CfgVT objects.

#### Syntax

```
Connect
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.Host = "MyHost"
    HE.CurrentHost.Cfg3270.TCPPort = 23
    HE.CurrentHost.Connect
End Sub
```

## ConnectBy

**Property, Host Object 3270 5250 VT**

You can use this property to set or retrieve the connection type. The following definitions exist in the hebasic.ebh header:

```
Const IO_TELNET = 0 ' All terminal types
Const IO_MODEM = 1 ' VT Only
Const IO_MSSNA = 2 ' 3270 Only
Const IO_NWSAA = 3 ' 3270 Only
Const IO_DEMOLINK = 4 ' All terminal types
```

### Syntax

```
iMode% = ConnectBy
```

### Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Dim Host as Object

    Set HE = CreateObject( "HostExplorer" )
    Set Host = HE.Hosts.Open(1)      '1 = 3270 terminal

    Host.Model = TN3270_MODEL_2' Set terminal model next
    Host.ConnectBy = IO_TELNET' Set connection method
    Host.Cfg3270.Host = "myhost.mydomain"   ' Set host name
    Host.Connect
End Sub
```

## ConnectErrorStatus

**Property, Host Object VT**

This read-only property returns the error status for the last connect request. Valid return codes are:

- 0—No error.
- 1—Busy signal.
- 2—No answer.
- 3—No dial tone.
- 4—Unspecified problem after dialing.

**Syntax**

```
iRc% = ConnectErrorStatus
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.StartSession Default VT

    If HE.CurrentHost.ConnectRC = 0 Then
        MsgBox "Error occurred during connect. Rc: " + Str$ (HE.CurrentHost.C
onnectErrorStatus)
    End If
End Sub
```

## ConnectRC

### Property, Host Object 3270 5250 VT

This read-only property returns the connection status for the last connect request. Valid return codes are:

- 0—Not connected.
- 1—Connecting.
- 2—Connected.

**Syntax**

```
iRc% = ConnectRC
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.StartSession Default 3270

    If HE.CurrentHost.ConnectRC <> 2 Then
        MsgBox "Error occurred during connect. Rc: " + Str$ (HE.CurrentHost.C
onnectRC)
    End If
End Sub
```

## CURSOR

### Property, Host Object 3270 5250 VT

You can use this property to get or set the cursor location. The location is expressed with an absolute number from 1 to the screen size. If the value is outside this range, the system ignores the call. In VT mode, the Cursor property is read-only. You can also use the CursorRC method to set the cursor position using a row and column value.

#### Syntax

```
Cursor = iPos%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iPos% = HE.CurrentHost.Cursor
    HE.CurrentHost.Cursor = 1760
    MsgBox "The position of the cursor is at " & iPos%, , "Information"
End Sub
```

## CURSORRC

### Method, Host Object 3270 5250

You can use this method to set the cursor location by using row and column values. If a value is outside the valid range, the system ignores the call. The first position on the screen is row 1 column 1.

#### Syntax

```
CursorRC iRow%, iCol%
```

where:

- *iRow%*—The row position.
- *iCol%*—The column position.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.CursorRc 5, 6' Set cursor to Row 5, Col 6
End Sub
```

## Device3279

### Property, Host Object 3270

This read/write property lets you set the 3270 device as FALSE (3278) or TRUE (3279). For this value to take effect, you must first disconnect and reconnect the session. You can only set this property if the host session is currently disconnected.

#### Syntax

```
b3279Device = Device3279
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    b3279Device = HE.CurrentHost.Device3279
    MsgBox "The 3270 Device is set to " & b3279Device, , "Information"
End Sub
```

## Disconnect

### Method, Host Object 3270 5250 VT

You can use this method to disconnect a session from a host system. You can call this method only if your session is currently connected. This method maps directly to the Disconnect item on the File menu.

#### Syntax

```
Disconnect
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Disconnect
End Sub
```

## EAB

### Property, Host Object 3270

You can use this read/write property to get or set Extended Attribute support. If the value is TRUE, the system enables Extended Attributes. Changing this value takes effect only after you disconnect and reconnect the session. You can only set this property if the host session is currently disconnected.

#### Syntax

```
bVal = EAB
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bEAB = HE.CurrentHost.EAB
    MsgBox "The Extended Attribute support has a value of " + bEAB
End Sub
```

## FieldID

### Method, Host Object 3270 5250

This method returns the field index for a given position on the screen. You can use this method to access the field object directly. The first position on the screen is 1, whereas the first Field ID index is 0.

#### Syntax

```
iFieldID% = FieldID( iPos% )
```

where:

- *iPos%*—The position to query.
- *iFieldID%*—The field ID returned for the queried position.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iFldID = HE.CurrentHost.FieldID( 1760 )
    MsgBox "The ID of the field at position 1760 is " & """ + iFldID & """
           , , "Information"
End Sub
```

## Fields Collection

### Object, Host Object 3270 5250

The Fields Collection object lets you manipulate a field through a series of methods and properties. This interface provides a device-independent way of dealing with data on the host screen. The first field is Field 0.

#### Syntax

```
Dim Fld as Object  
Set Fld = CurrentHost.Fields( iFieldNum% )  
where iFieldNum% is the field index to return.
```

#### Example

```
Sub Main  
    Dim Fld as Object  
    Dim HE as Object  
  
    Set HE = CreateObject( "HostExplorer" )  
    Set Fld = HE.CurrentHost.Fields(2)  
  
    MsgBox "Field 2 has length: " + Str$(Fld.Length) + ", Position: " + Str$  
    $(Fld.Pos) + ", Value: " + _ Fld.Text  
End Sub
```

## FontLarger

### Method, Host Object 3270 5250 VT

You can use this method to change the session font to the next largest font.

#### Syntax

#### FontLarger

#### Example

```
Sub Main  
    Dim HE as Object  
    Set HE = CreateObject( "HostExplorer" )  
  
    HE.CurrentHost.FontLarger  
End Sub
```

## FontSmaller

**Method, Host Object 3270 5250 VT**

You can use this method to change the session font to the next smallest font.

### Syntax

**FontSmaller**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.FontSmaller
End Sub
```

## Hide

**Method, Host Object 3270 5250 VT**

You can use this method to hide the host object. This will hide the screen whether it is minimized, normal, or maximized. When the screen is hidden, it no longer appears in the taskbar.

### Syntax

**Hide**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Hide the current host screen
    HE.CurrentHost.Hide
End Sub
```

## HideToolbar

**Method, Host Object 3270 5250 VT**

You can use this method to hide the session toolbar. The reverse of this call is ShowToolbar.

### Syntax

**HideToolbar**

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.HideToolbar
End Sub
```

## HighlightText

**Method, Host Object 3270 5250 VT**

You can use this method to return text that you have highlighted on the screen by using your mouse. The parameter `row%` selects the row of highlighted text to return. Row 1 is the first highlighted row; row 2 is the second, and so on. Passing a row number of 0 will result in the entire highlighted block being returned with newline characters "`\n`" as line separators. If you specify an invalid row number, you will get a null string.

**Syntax**

```
szText$ = HighlightText( row% )
```

where:

- `row%`—The highlighted row to return.
- `szText$`—The string returned.

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szText$ = HE.CurrentHost.HighlightText( 0 )
    MsgBox "The content of the highlighted area is " & szText$, , "Hummingbird Basic Output"
End Sub
```

## Index

### Property, Host Object 3270 5250 VT

You can use this property to return the unique ID for the current session. This is the session control block ID used internally. This corresponds to the value used in the Hosts(*n*) property. The first session has an index value of 0.

#### Syntax

**UniqueSessID**

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    UniqueSessID% = HE.CurrentHost.Index
    MsgBox "The ID of the current session is " & UniqueSessID%
          , , "Information"
End Sub
```

## InsertMode

### Property, Host Object 3270 5250

This property is used to get or set the insert mode for the host session.

#### Syntax

**InsertMode** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bMode = HE.CurrentHost.InsertMode
    HE.CurrentHost.InsertMode = TRUE
End Sub
```

## IsConnected

### Property, Host Object [3270](#) [5250](#) VT

This read-only property returns the value `TRUE` if the session is connected to a host system. It returns the value `FALSE` if the session is not in use.

#### Syntax

```
bVal = IsConnected
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    If HE.CurrentHost.IsConnected = FALSE Then
        MsgBox "Not connected to host system!", , "Information"
    Else
        MsgBox "Connected to host system!", , "Information"
    End If
End Sub
```

## IsXfer

### Property, Host Object [3270](#)

This read-only property returns a `TRUE` value if a file transfer is currently taking place.

#### Syntax

```
bVal = IsXfer
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SendFile "c:\config.sys", "CONFIG SYS A1" , "( ASCII CRL
F"
    If HE.CurrentHost.IsXfer Then
        MsgBox "Wait until File Transfer is complete"
    End If
End Sub
```

## Keyboard

### Property, Host Object 3270 5250 VT

You can use this property to get or set the locked keyboard mode. When set to TRUE, the host keyboard locks, preventing user input. You can unlock the host keyboard by setting this value to FALSE.

**Note:** This corresponds to the x ... indicators in the Operator Information Area. When any x indicator is present, this property returns TRUE. When no indicator is present, meaning that the keyboard is ready for input, this property returns FALSE.

#### Syntax

**Keyboard** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iKeybLocked = HE.CurrentHost.Keyboard
    HE.CurrentHost.Keyboard = FALSE
End Sub
```

## Keys

### Method, Host Object 3270 5250 VT

You can use this method to press keys while in HostExplorer. You can insert special sequences in 3270, 5250, or VT mode to press system keys such as Tab, Reset, and so on. Valid return codes are:

- 0—String processed successfully
- 5—Error processing string. Keyboard locked. If the keystring\$ contains an AID key, the function will always return a value of 5.

**Note:** The Keys method is affected by the Type Ahead setting. If you send a 3270/5250 AID key and Type Ahead is enabled, the function will not return until the keyboard is unlocked by the host system. If Type Ahead is disabled, the function will always return immediately.

You can use the HLLAPI @ style commands to press emulation action keys. For VT, you can enter C-style \x sequences.

## Syntax

*iRc%* = **Keys**( *keystring\$* )

where:

- *keyString\$*—The string to type.
- *iRc%*—The return code.

## Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Send USERID, Tab, PASSWORD, Enter
    HE.CurrentHost.Keys "USERID@TPASSWORD@E"
End Sub
```

## LoadQuickKeyFile

**Method, Host Object 3270 5250 VT**

You can use this method to load a new set of Quick-Keys for this session. Quick-Keys are shared between sessions and loading a new set of Quick-Keys from one session will affect other sessions. Valid return codes are:

- 0—Successfully loaded Quick-Keys.
- 1—Error loading Quick-Keys.

### Syntax

```
iRc% = LoadQuickKeyFile( szFileName$ )
```

where:

- *szFileName\$*—The Quick-Key file to load. No path or extension specified.
- *iRc%*—The return code for the method.

### Example

```
Sub Main
    Dim HE As Object
    Set HE = CreateObject( "HostExplorer" )

    iRc% = HE.CurrentHost.LoadQuickKeyFile( "MYQUICKKEYS" )
End Sub
```

## Maximize

**Method, Host Object 3270 5250 VT**

You can use this method to maximize the host session window.

### Syntax

**Maximize**

### Example

```
Sub Main
    Dim HE As Object
    Set HE = CreateObject( "HostExplorer" )

    ' Maximize the current host screen
    HE.CurrentHost.Maximize
End Sub
```

## Minimize

**Method, Host Object 3270 5250 VT**

You can use this method to minimize the host session window.

### Syntax

**Minimize**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Minimize the current host screen
    HE.CurrentHost.Minimize
End Sub
```

## Model

**Property, Host Object 3270 5250 VT**

This read/write property returns the model type for the session or sets the model for the session. Changing this value will take effect only after you disconnect and reconnect a session.

### Syntax

iModel% = **Model**

### Mode

For 3270, the possible values are: 2, 3, 4, or 5.

For 5250, the possible values are: 2 or 5.

For Telnet, the possible values are:

- TELNET\_VT52
- TELNET\_VT100
- TELNET\_VT101
- TELNET\_VT102
- TELNET\_VT220
- TELNET\_VT320
- TELNET\_VT420
- TELNET\_ANSI
- TELNET\_SCOANSI

**Example**

```
'$Include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iModel = HE.CurrentHost.Model
    MsgBox "This session is of Model Type " + iModel, , "Information"
End Sub
```

**MouseToCursor****Property, Host Object 3270 5250 VT**

You can use this R/O property to return the position of the mouse in screen coordinates. If the mouse is not in the screen area, the value returned is -1. Otherwise, the value returned is between 1 and the screen size. The screen size is the number of rows multiplied by the number of columns.

**Syntax**

```
iPos% = MouseToCursor
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iPos = HE.CurrentHost.MouseToCursor
End Sub
```

**OIA****Property, Host Object 3270 5250 VT**

This property returns the operator information area (OIA) as a text string. This is a read-only property.

**Note:** Do not use this property to check the status of certain flags such as Keyboard Inhibited or Insert Mode. There are Host properties to retrieve these values.

**Syntax**

```
szOIA$ = OIA
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szOIA = HE.CurrentHost.OIA
    MsgBox szOIA, , "Content of Operator Information Area (OIA)"
End Sub
```

**OIAUpdated****Property, Host Object 3270 5250 VT**

If the system returns a TRUE value (Updated), and the operator information area (OIA) was updated since the last call, the system will reset the internal flag to FALSE (Not Updated) after the call. This is a read-only property.

**Syntax**

```
bVal = OIAUpdated
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bUpdated = HE.CurrentHost.OIAUpdated
    MsgBox "OIA Return Value is " & bUpdated, , "Information"
End Sub
```

**Pause****Method, Host Object 3270 5250 VT**

You can use this method to pause the macro for the specified time in milliseconds.

**Syntax**

```
PrintScreen
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Pause for 5 seconds
    HE.CurrentHost.Pause 5000
End Sub
```

## PrintScreen

### Method, Host Object 3270 5250 VT

You can use this method to print the current host session to the Windows printer specified in the profile for the session. This method will not display the Print Setup dialog box.

#### Syntax

##### **PrintScreen**

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.PrintScreen
End Sub
```

## ProtectedText

### Method, Host Object 3270 5250

You can use this method to replace text in protected locations of the screen. Use this method carefully since you can overwrite field attributes.

#### Syntax

##### **ProtectedText iRow%, iCol%, szText\$**

where:

- *iRow%*—The row in which to write text.
- *iCol%*—The column in which to write text.
- *szText\$*—The text to write.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.ProtectedText 2, 2, "Hello World"
End Sub
```

## PSReserved

### Property, Host Object 3270 5250 VT

You can use this property to get or set the session-reserved flag. When you reserve a session (TRUE), the system does not allow user input. When the flag is FALSE (default), the user can type and use the menu. This makes the screen read-only.

#### Syntax

```
PSReserved = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bReserved = HE.CurrentHost.PSReserved
    HE.CurrentHost.PSReserved = FALSE
End Sub
```

## PSUpdated

### Property, Host Object 3270 5250 VT

This read-only property returns a TRUE value if the presentation space was updated since the last call to this property. This property resets the internal flag to FALSE (not updated) after each call.

#### Syntax

```
bVal = PSUpdated
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bUpdated = HE.CurrentHost.PSUpdated
    MsgBox "Presentation Space Value is " & bUpdated, , "Information"
End Sub
```

## PutText

### Method, Host Object 3270 5250 VT

You can use this method to write text to any unprotected location on the screen. In VT mode, the iRow and iCol parameters are ignored and can be omitted. The first location on the screen is Row 1 Column 1. This method is identical to the Keys method, but it allows you to specify the location to write the text. This method does not move the cursor.

#### Syntax

**PutText** *szText\$*, *iRow%*, *iCol%*

where:

- *szText\$*—The text to write.
- *iRow%*—The row in which to write text.
- *iCol%*—The column in which to write text.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.PutText "Hello World", 2, 2
End Sub
```

## QueryCloseRequest

### Property, Host Object 3270 5250 VT

You can use this property to see if the user attempted to close the session. The R/O property returns TRUE if the user attempted to close the session (File/Close Session, File/Exit All, Alt-F4). This query is valid only if you disabled the session close using the AllowClose property.

#### Syntax

*bCloseRequested* = **QueryCloseRequest**

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bCloseRequested = HE.CurrentHost.QueryCloseRequest
End Sub
```

## QuickKeyFile

**Property, Host Object 3270 5250 VT**

You can use this read-only property to return the set of Quick-Keys that are currently loaded.

### Syntax

```
szQuickKey$ = QuickKeyFile
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szQuickKeys$ = HE.CurrentHost.QuickKeyFile
End Sub
```

## ReceiveFile

**Method, Host Object 3270 VT**

You can use this method to initiate a file transfer (download) from the host system.

In 3270 mode, the first parameter specifies the local file to receive the data, the second parameter specifies the host filename to download, and the last parameter specifies the host file transfer options. For more information, see “File Transfer Options” on page 188.

In VT mode, the first parameter specifies the local file to receive the data and the second parameter specifies the file transfer protocol to use. When using ZModem, this parameter should be a path since the filename will be provided by the ZModem protocol.

## Syntax

For 3270 mode, the syntax is:

**ReceiveFile** *pcfilename\$*, *hostfilename\$*, *options\$*

For VT mode, the syntax is:

**ReceiveFile** *pcfilename\$*, *iXferProtocol%*

where:

- *pcfilename\$*—The PC file name to receive the host file.
- *hostfilename\$*—The name of the host file to download.
- *options\$*—Options for this file transfer.
- *iXferProtocol%*—The file transfer protocol to use. Refer to `hebasic.ebh` for values.

## Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' 3270 mode download
    HE.CurrentHost.ReceiveFile "c:\profile.txt", "PROFILE EXEC A1"
    , "( ASCII CRLF"
    HE.CurrentHost.WaitXfer

    ' VT mode download
    HE.CurrentHost.ReceiveFile "c:\vtfile.txt", XFER_TYPE_ZMODEM
End Sub
```

## Restore

### Method, Host Object 3270 5250 VT

You can use this method to restore the session window to the last state.

#### Syntax

##### Restore

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Restore the current host screen
    HE.CurrentHost.Restore
End Sub
```

## Row

### Method, Host Object 3270 5250 VT

You can use this method to retrieve the contents of the screen row-by-row. The numeric parameter specifies which row to retrieve. The value for the parameter can range between 1 and the number of rows in the presentation space. This method will convert all nulls to blanks and will truncate any trailing blanks.

#### Syntax

```
szRowText$ = Row( iRow% )
```

where:

- *iRow%*—The row from which to retrieve text.
- *szRowText\$*—The string retrieved from the row.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    SecondRow$ = HE.CurrentHost.Row(2)
    MsgBox "The content of the specified row in the presentation space is"
    & SecondRow, , "Output"
End Sub
```

## Rows

### Property, Host Object **3270 5250 VT**

This read-only property returns the number of rows currently defined in the host object.

#### Syntax

##### Rows

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iRows = HE.CurrentHost.Rows
    MsgBox "The number of rows in the presentation space is " & iRows
    , , "Information"
End Sub
```

## RunCmd

### Method, Host Object **3270 5250 VT**

You can use this method to run a system command within the session. You can assign system commands to keys and tool items. If you run a command which displays a dialog, there is no programmatic method to enter data into a dialog and exit it.

**Note:** The RunCmd method is affected by the Type Ahead setting. If you send a 3270/5250 AID key (for example, Enter, PFx, and so on) and Type Ahead is enabled, the function will not return until the keyboard is unlocked by the host system. If Type Ahead is disabled, the function will always return immediately.

## Syntax

**RunCmd** *szCommandName\$*

where *szCommandName\$* is the name of command to execute.

## Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Show the Session Profile dialog
    HE.CurrentHost.RunCmd "Dlg-Edit-Session-Profile"
End Sub
```

## RunQuickKey

**Method, Host Object 3270 5250 VT**

You can use this method to run a Quick-Key within the session.

## Syntax

**RunQuickKey** *szQuickKeyName\$*

where *szQuickKeyName\$* is the name of the Quick-Key to execute.

## Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    'Make sure you create a Quick-Key before running this macro
    HE.CurrentHost.RunQuickKey "dothis"
End Sub
```

## SaveQuickKeyFile

**Method, Host Object 3270 5250 VT**

You can use this method to save Quick-Keys to disk. Do not supply a path or extension for the file name.

### Syntax

**SaveQuickKeyFile szQuickKeyName\$**

where *szQuickKeyName\$* is the file name of the Quick-Keys to save.

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SaveQuickKeyFile "VMCMS"
End Sub
```

## SaveScreen

**Method, Host Object 3270 5250 VT**

You can use this method to save the session screen to a file.

### Syntax

**SaveScreen szFileName\$**

where *szFileName\$* is the name of the file to which the screen is saved.

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SaveScreen "c:\screen.txt"
End Sub
```

## SaveScrollbar

### Method, Host Object VT

You can use this method to save the session scrollback buffer to a file.

#### Syntax

**SaveScrollbar szFileName\$**

where *szFileName\$* is the name of the file to which the scrollback buffer is saved.

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SaveScrollbar "c:\scrollback.txt"
End Sub
```

## Search

### Method, Host Object 3270 5250 VT

You can use this method to search the presentation space for the given search string. The *bCaseSensitive* flag specifies whether the search is case-sensitive or not. *iRow%* and *iCol%* specify the starting position for the search. If *iRow%* is not passed, the cursor row is used instead. If *iCol%* is not passed, the cursor column is used instead. Valid return codes are:

- n—The string was found in the presentation space at the location of n, where n is a value of 1 or greater. For example, a returned value of 1 indicates that the string is located at the top left-hand corner of the session screen.
- 0—String was not found in the presentation space.

If you use variables in the parameter list (as opposed to constants), the method will return the row and column where the text was found.

## Syntax

```
iRc% = Search( szSearchString$, bCaseSensitive, iRow%,  
iCol% )
```

where:

- *szSearchString\$*—The string that HostExplorer attempts to find.
- *bCaseSensitive*—The case-sensitive flag – TRUE/FALSE
- *iRow%*—The row in which to start the search.
- *iCol%*—The column in which to start the search.

## Example

```
Sub Main  
    Dim HE as Object  
    Set HE = CreateObject( "HostExplorer" )  
  
    iRow = 1  
    iCol = 1  
  
    iRc% = HE.CurrentHost.Search( "VM/ESA", FALSE, iRow, iCol)  
    If iRc% > 0 Then  
        MsgBox "String found at Row:" & iRow & ", Col: " & iCol  
    End If  
End Sub
```

## SendFile

### Method, Host Object [3270 VT](#)

You can use this method to initiate a file transfer (upload) to the host system.

In 3270 mode, the first parameter specifies the local file to be uploaded, the second parameter specifies the host filename to receive the file, and the last parameter specifies the host file transfer options. For more information, see “File Transfer Options” on page 188.

In VT mode, the first parameter specifies the local file to receive the data and the second parameter specifies the file transfer protocol to use.

## Syntax

```
SendFile pcfilename$, hostfilename$, options$  
SendFile pcfilename$, iXferProtocol%
```

where:

- *pcfilename\$*—The name of the PC file to upload to the host.
- *hostfilename\$*—The name of the host file to receive the transferred file.
- *options\$*—The option for this file transfer.
- *iXferProtocol%*—The file transfer protocol to use. Refer to hebasic.ebh for values.

## Example

```
Sub Main  
    Dim HE as Object  
    Set HE = CreateObject( "HostExplorer" )  
  
    ' 3270 mode upload  
    HE.CurrentHost.SendFile "c:\config.sys", "CONFIG SYS A1"  
    , "( ASCII CRLF"  
    HE.CurrentHost.WaitXfer  
  
    ' VT mode upload  
    HE.CurrentHost.SendFile " c:\config.sys", XFER_TYPE_ZMODEM  
End Sub
```

## SetFont

**Method, Host Object 3270 5250 VT**

You can use this method to change the session font to a specific name, width, and size. If you specify a width of 0, the system will select a visually proportioned font. The font generated may not actually match the characteristics of those you requested.

## Syntax

```
SetFont fontname$, width%, height%
```

where:

- *fontname\$*—The name of the font.
- *width%*—The width of the font (in pixels).
- *height%*—The height of the font (in pixels).

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SetFont "Courier New",0, 12
End Sub
```

## ShortName

**Property, Host Object** [3270](#) [5250](#) [VT](#)

You can use this property to get or set the session short name.

**Note:** The short name can be only one character (between A and Z).

The session short name is used primarily by HLLAPI applications. Changing the short name of a session will take effect immediately and does not require the emulator or session to be restarted.

**Syntax**

```
ShortName = "c"
strVal$ = ShortName
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Assign letter 'A' to this session
    HE.CurrentHost.ShortName = "A"
End Sub
```

## Show

**Method, Host Object 3270 5250 VT**

You can use this method to show the Host object. This displays the screen whether it is minimized, normal, or maximized.

### Syntax

**Show**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Show the current host screen
    HE.CurrentHost.Show
End Sub
```

## ShowToolbar

**Method, Host Object 3270 5250 VT**

You can use this method to show the session toolbar. The toolbar cannot be shown if a mouse is not installed on the Windows system.

### Syntax

**ShowToolbar**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.ShowToolbar
End Sub
```

## SilentConnect

**Property, Host Object 3270 5250 VT**

You can use this property to get or set the Silent Connect option. This option is useful only when you construct a session manually by using the NewSession method.

When this flag is enabled, there are no informational or error messages. These messages appear during the connection process.

### Syntax

**SilentConnect**

### Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
    Dim Host as Object
    Dim HE as Object
    Dim Cfg as Object
    Set HE = CreateObject( "HostExplorer" )

    UniqueSessID% = HE.NewSession
    Set Host = HE.Hosts(index%)
    Set Cfg = Host.CfgVT' Use CfgVT, Cfg3270, or Cfg5250
    Host.TerminalMode =TERMINAL_TELNET' Set terminal mode first!
    Host.Model =TELNET_VT220' Set terminal model next
    Host.ConnectBy = IO_TELNET' Set transport next
    Host.SilentConnect = TRUE' Silent connection

    Cfg.Host = "myhost.mydomain"
    Cfg.ConnectTimeout = 30
    Cfg.TCPPort = 23

    Host.Connect
End Sub
```

## SystemColor

Method, Host Object **3270 5250 VT**

You can use the SystemColor Host method to change the color composition for one of the 16 base colors used by HostExplorer. This performs the same function as accessing the Color Palette category in the Session Profile dialog box. The values for the iRed, iGreen, and iBlue parameters can vary between 0 and 255. The value for the iColor parameter can range from 0 to 15, which maps to the normal VGA base colors, or you can use the following reserved names:

BLACK  
BLUE  
GREEN  
CYAN  
RED  
MAGENTA  
BROWN  
WHITE  
GRAY  
LIGHT\_BLUE  
LIGHT\_GREEN  
LIGHT\_CYAN  
LIGHT\_RED  
LIGHT\_MAGENTA  
YELLOW  
BRIGHT\_WHITE

**Syntax****SystemColor** *iColor%*, *iRed%*, *iGreen%*, *iBlue%*

where:

- *iColor%*—The color index to set the color mix.
- *iRed%*—The red color mix value.
- *iGreen%*—The green color mix value.
- *iBlue%*—The blue color mix value.

**Example**

'\$Include:"-E\hebasic.ebh"

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SystemColor LIGHT_BLUE, 64, 64, 255
End Sub
```

**TerminalMode****Property, Host Object 3270 5250 VT**

This read-only property returns the terminal type that is currently defined for the session. You can use the following literals to test the returned value:

- TERMINAL\_3270
- TERMINAL\_TELNET
- TERMINAL\_5250

**Syntax***iMode%* = **TerminalMode****Example**

'\$Include:"-E\hebasic.ebh"

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iMode = HE.CurrentHost.TerminalMode
    MsgBox "The current Terminal Mode is " + iMode, , "Information"
End Sub
```

## Text

### Property, Host Object **3270 5250 VT**

You can use this property to retrieve and update the entire screen as a string. Nulls are converted to blanks. The size of the string returned is the number of columns \* the number of rows.

#### Syntax

```
szText$ = Text
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szTxt$ = HE.CurrentHost.Text
    MsgBox "Screen Content: " &szTxt$
End Sub
```

## TextRC

### Method, Host Object **3270 5250 VT**

You can use this method to retrieve part of the screen as a string. You must specify a valid row and column. The first position on the screen is Row 1 Column 1. The *iLength%* value can be:

- 0—Copy to End of Field.
- -1—Copy to End of Line.
- -2—Copy to End of Word.
- -3—Copy to End of Screen.
- >0—Exact length specified.

**Syntax**

```
szText$ = TextRC( iRow%, iCol%, iLength% )
```

where:

- *iRow%*—The row that contains the text to retrieve.
- *iCol%*—The column that contains the text to retrieve.
- *iLength%*—The length of text to retrieve.

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szTxt$ = HE.CurrentHost.TextRC( 5, 5, 20 )
End Sub
```

**TN3270****Property, Host Object 3270 5250**

This read-only property returns TRUE if HostExplorer is currently in 3270 or 5250 mode. It returns FALSE if HostExplorer is in VT or NVT terminal mode.

**Syntax**

```
bVal = TN3270
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bMode = HE.CurrentHost.TN3270
    If bMode = TRUE then
        MsgBox "The current session is in 3270 or 5250 Mode."
    Else
        MsgBox "The current session is in VT Mode."
    End If
End Sub
```

## TrackMenu

**Method, Host Object 3270 5250 VT**

You can use this method to display the track menu at the current mouse pointer location.

### Syntax

**TrackMenu**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.TrackMenu
End Sub
```

## Update

**Method, Host Object 3270 5250 VT**

You can use this method to force a repaint of the session window.

### Syntax

**Update**

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Update
End Sub
```

## WaitConnected

**Method, Host Object 3270 5250 VT**

You can use this method to synchronize the connection process to a host. This method will wait until the connection is complete or for up to 'iTimeout' seconds. Valid return codes are:

- 0—Connection complete.
- 1—Timer expired. Emulator is not connected.

**Syntax**

```
iRc% = WaitConnected( iTimeout& )
```

where *iTimeout&* is the timeout for the method (in seconds).

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.StartSession "Default.3270"
    iRc% = HE.CurrentHost.WaitConnected( 60 )
End Sub
```

## WaitForIO

### Method, Host Object 3270 5250

You can use this method (when using the Demo Link) to wait for the user to press an action key such as Enter or PFx. Valid return codes are:

- 0—Action key pressed.
- 1—Timer expired.

**Syntax**

```
iRc% = WaitForIO( iTimeout& )
```

where *iTimeout&* is the timeout for the method (in seconds).

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Wait for 60 seconds for user to press action key
    iRc% = HE.CurrentHost.WaitForIO( 60 )

    If iRc% = 0 Then
        cAID$ = HE.CurrentHost.GetIOBuffer( -1 )
        strField1$ = HE.CurrentHost.GetIOBuffer( 1 )
    End If
End Sub
```

## WaitForString

Method, Host Object 3270 5250 VT

You can use this method to wait for a string to appear on the screen. This function is provided for compatibility purposes and has been superseded by WaitForStringRC method.

Valid return codes are:

- 0—The string was not found.
- >0—The row where the string was found.

### Syntax

```
iRc% = WaitForString (szString$, iRow%, iTimeout&, bCaseSensitive )
```

where:

- *szString\$*—The string that HostExplorer attempts to find.
- *iRow%*—The values are: >0—The row to search for the string, 0—Search the entire screen, -1—Search the row containing the cursor.
- *iTimeout&*—The timeout for the method (in seconds).
- *bCaseSensitive*—TRUE/FALSE if the search is case-sensitive.

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.StartSession "Solaris.VT"
    HE.CurrentHost.WaitConnected 120

    iRc% = HE.CurrentHost.WaitForString ( "login", -1, 1, FALSE )

    If iRc% > 0 Then
        HE.CurrentHost.Keys "myuserid\r"

        iRc% = HE.CurrentHost.WaitForString ( "password", -1, 1, FALSE )

        If iRc% > 0 Then
            HE.CurrentHost.Keys "mypassword\r"
            MsgBox "Login successful!"
        End If
    End If
End Sub
```

## WaitForStringRC

Method, Host Object **3270 5250 VT**

You can use this method to wait for a string to appear on the screen. For example, this method is useful when logging in or writing macros for UNIX systems running in Linemode. Valid return codes are:

- 0—The string was not found.
- >0—The row where the string was found.

### Syntax

```
iRc% = WaitForStringRC( szString$, iRow%, iCol%, iFlag%, iTimeout&, bCaseSensitive )
```

where:

- *szString\$*—The string that HostExplorer attempts to find.
- *iRow%*—The values are: >0—The row to search for the string, 0—Search the entire screen, -1—Search the row containing the cursor.
- *iCol%*—The column in which to start the search.
- *iFlag%*—The values are: 0—Search using the *iRow%* value, 1—Search only at *iRow% / iCol%* (requires *iRow% >= 1*), 2—Search from the beginning at *iRow% / iCol%* (requires *iRow% >= 1* )
- *iTimeout&*—The timeout for the method (in seconds).
- *bCaseSensitive*—TRUE/FALSE if the search is case-sensitive.

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.StartSession "Solaris.VT"
    HE.CurrentHost.WaitConnected 120

    iRc% = HE.CurrentHost.WaitForStringRC ( "login", -1, 1, 0, 10, FALSE )

    If iRc% > 0 Then
        HE.CurrentHost.Keys "myuserid\r"
        iRc% = HE.CurrentHost.WaitForStringRC ( "password"
        , 1, 1, 0, 10, FALSE )

        If iRc% > 0 Then
            HE.CurrentHost.Keys "mypassword\r"
            MsgBox "Login successful!"
        End If
    End If
End Sub
```

**WaitIdle****Method, Host Object 3270 5250 VT**

You can use this method to wait until the screen is idle for 'IdleTime' milliseconds. An idle screen is one that does not change in any fashion for the elapsed time. Valid return codes are:

- 0—Idletime has elapsed.
- 1—Session is closed during the waiting period.

**Syntax**

```
iRc% = WaitIdle( iIdleTime& )
```

where *iIdleTime&* is the timeout for the method (in milliseconds).

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.RunCmd( "PF1" )
    Result = HE.CurrentHost.WaitIdle( 1500 )
    if (Result = 0) then
        MsgBox "Time has elapsed", , "Output"
    else
        MsgBox "The session was closed during the waiting period."
        , , "Output"
    end if
End Sub
```

**WaitPSUpdated**

**Method, Host Object 3270 5250 VT**

You can use this method to synchronize events with the Host system. This method will wait until the PS is updated, for up to 'iTimeout' seconds. For 3270 and 5250, you have the option to wait for the hostkeyboard to unlock. Valid return codes are:

- 0—PS was updated within the timeout period.
- 1—Timer expired. PS was not updated.

**Syntax**

```
iRc% = WaitPSUpdated( iTimeout& [, bSyncWithKeyboard] )
```

where:

- *iTimeout&*—The timeout for the method (in seconds).
- *bSyncWithKeyboard*—Optional for 3270 and 5250. Set to TRUE or FALSE. When TRUE, after waiting for the host to update the screen, the method will wait for the host to unlock the host keyboard.

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Press Enter key
    HE.CurrentHost.Keys "@E"

    ' Wait for host to update screen
    iRC% = HE.CurrentHost.WaitPSUpdated( 30, TRUE )

    If iRC% = 0 Then
        MsgBox "PS updated and keyboard unlocked within interval specified"
    End If
End Sub
```

**WaitXfer****Method, Host Object 3270**

You can use this method to wait until a file transfer is completed. This synchronizes the macro with the file transfer.

**Syntax****WaitXfer****Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SendFile "c:\config.sys", "CONFIG SYS A1", "
    ( ASCII CRLF"
    HE.CurrentHost.WaitXfer
End Sub
```

## XferCount

### Property, Host Object 3270 5250 VT

You can use this property to return the number of bytes transferred in the current or last file transfer.

#### Syntax

```
iBytes& = XferCount
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iBytes& = HE.CurrentHost.XferCount
End Sub
```

## XferRC

### Property, Host Object 3270

This read-only property returns the error code for the last file transfer. The return codes are consistent with those found in the EHLLAPI and DDE application program interfaces.

#### Syntax

```
iRc% = XferRC
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.SendFile "c:\config.sys", "CONFIG SYS A1"
    , "( ASCII CRLF"
    HE.CurrentHost.WaitXfer

    If HE.CurrentHost.XferRC <> 3 Then
        MsgBox "Some error occurred during file transfer. Rc: "
        + Str$(HE.CurrentHost.XferRC)
    End If
End Sub
```

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

### Application

#### Property, Area Object **3270 5250 VT**

You can use this read-only property to return the System object.

#### Syntax

`Area.Application`

**Example**

```
Sub Main
    Dim Sys As Object, AllSess As Object, Sess As Object
    Dim MyScreen As Object, MyArea As Object
    Dim TBars As Object, MyTBar As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")
    Set AllSess = Sys.Sessions   ' Assumes an open session

    Set Sess = Sys.ActiveSession

    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1, 1, 10, 10, ,)
    Set TBars = Sess.Toolbars
    Set MyTBar = TBars(1)

    ' All these return the same result.

    MsgBox "System.Application = " + Sys.Application.Name
    MsgBox "Sessions.Application = " + AllSess.Application.Name
    MsgBox "Session.Application = " + Sess.Application.Name
    MsgBox "Screen.Application = " + MyScreen.Application.Name
    MsgBox "Area.Application = " + MyArea.Application.Name
    MsgBox "Toolbars.Application = " + TBars.Application.Name
    MsgBox "ToolBar.Application = " + MyTBar.Application.Name

End Sub
```

## Bottom

### Property, Area Object 3270 5250 VT

This property returns or sets the last row of the Area object.

#### Syntax

Area.Bottom

#### Example

```
Sub Main
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")

    ' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This illustrates the Top and Bottom properties for an Area
    ' object. For demonstration purposes, the Select method is used,
    ' but it is not required for setting Top and Bottom properties.
    MsgBox "Press to demonstrate the Top and Bottom properties for
    Area objects."

    Set MyArea = MyScreen.Area(1, 1, MyScreen.Rows, MyScreen.Cols,,3)

    MyArea.Select

    For i = 1 to Int(MyScreen.Rows/2)
        MyArea.Top = MyArea.Top + 1
        MyArea.Select
        MyArea.Bottom = MyArea.Bottom - 1
        MyArea.Select
    Next

    ' This demonstrates the Top property for a Session object.
    MsgBox "Press to demonstrate the Top property for Session objects."
    Sess.Top = 50
    MsgBox "The session top is now at 50. Press to move session top to 1"
    Sess.Top = 1
    MsgBox "The session top is now at 1. Press to move session top to 100"
    Sess.Top = 100

End Sub
```

## Copy

**Method, Area Object 3270 5250 VT**

You can use the Copy method to copy data from the screen to the clipboard.

### Syntax

Area.Copy

### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    ' Invoke the clipboard viewer
    Shell("clipbrd.exe")

    ' Assumes an open session
    Set Sys = CreateObject("HBOLEAUT.HostExSystem")

    Set Sess = Sys.ActiveSession
    ' This example demonstrates the Copy method for Screen objects.
    Set MyScreen = Sess.Screen
    MyScreen.SelectAll
    MyScreen.Copy

    MsgBox "The screen was copied to the clipboard. Press to copy a smaller
area."

    ' This example demonstrates the Copy method for Area objects.
    Set MyArea = MyScreen.Area(1,1,10,10, , )
    MyArea.Select
    MyArea.Copy

End Sub
```

## Cut

### Method, Area Object 3270 5250

You can use this method to cut the current selection of the screen to the clipboard.

#### Syntax

Area.Cut

#### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    ' Invoke the clipboard viewer
    Shell("clipbrd.exe")

    ' Assumes an open session
    Set Sys = CreateObject("HEOLEAUT.HostExSystem")
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This example demonstrates the Cut method for Area objects.

    Set MyArea = MyScreen.Area(1,1,10,10, , )
    MyArea.Select
    MyArea.Cut
    MsgBox "Press to Cut the entire Screen."

    ' This example demonstrates the Cut method for Screen objects.
    MyScreen.SelectAll
    MyScreen.Cut

End Sub
```

## Delete

### Method, Area Object 3270 5250

You can use this method to remove the current selection from the screen.

#### Syntax

Area.**Delete**

#### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    ' Assumes an open session
    Set Sys = CreateObject("HEOLEAUT.HostExSystem")

    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This example demonstrates the Delete method for Area objects.
    Set MyArea = MyScreen.Area(1,1,5,5 , ,3)
    MyArea.Select
    MyArea.Delete
    MsgBox "Press to Delete a larger section of the screen."

    ' This example demonstrates the Delete method for Screen objects.
    MyScreen.Select 10,10,20,20
    MyScreen.Delete
End Sub
```

## Left

### Property, Area Object 3270 5250 VT

This property returns or sets the column of the screen where the area begins.

#### Syntax

Area.Left

#### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This moves the session to the left of the screen.
    Sess.Left = 1

    Set MyArea = MyScreen.Area(1,1,MyScreen.Rows,1,,3)
    MsgBox "The text in column 1 is: " + MyArea.Value

    MyArea.Left = 2
    MyArea.Right = 2

    MsgBox "The test in column 2 is: " + MyArea.Value
End Sub
```

## Parent

### Property, Area Object 3270 5250 VT

This read-only property returns the parent of the specified object.

#### Syntax

Area.Parent

**Example**

```
Sub Main()
    Dim Sys As Object, AllSess As Object, Sess As Object
    Dim MyScreen As Object, MyArea As Object
    Dim TBars As Object, MyTBar As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")
    Set AllSess = Sys.Sessions

        ' Assumes an open session. Retrieve objects.
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1, 1, 10, 10, , 3)
    Set TBars = Sess.Toolbars
    Set MyTBar = TBars(1)

        ' The System object is its own parent.
    MsgBox "System.Parent = " + Sys.Parent.Name

    'The parent of a Sessions object is also the System object.
    MsgBox "Sessions.Parent = " + AllSess.Parent.Name

        ' The parent of a Session object is a Sessions object,
        ' whose parent is ...
    MsgBox Sess.Parent.Parent.Name

        ' The parent of a Screen object is a Session object,
        ' whose parent is ...
    MsgBox MyScreen.Parent.Parent.Parent.Name

        ' The parent of a Toolbars object is also a
        ' Session, whose parent is ...
    MsgBox TBars.Parent.Parent.Parent.Name

        ' The parent of an Area object is a Screen object,
        ' whose parent is ...
    MsgBox MyArea.Parent.Parent.Parent.Parent.Name

        ' The parent of a Toolbar object is a Toolbars object,
        ' whose parent is ...
    MsgBox MyTBar.Parent.Parent.Parent.Parent.Name

End Sub
```

## Paste

### Property, Area Object 3270 5250 VT

This method pastes text residing on the clipboard into the Area object.

#### Syntax

Area.Paste

#### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")

    ' Assumes an open session
    Set Sess = Sys.ActiveSession

    ' This example demonstrates the Paste method for Screen objects.
    Set MyScreen = Sess.Screen

    MyScreen.Select 7,10,7,10
    MyScreen.Copy
    MyScreen.Select 23,6,23,6
    MyScreen.Paste

    ' This example demonstrates the Paste method for Area objects.

    Set MyArea = MyScreen.Area(9,10,9,10,,3)
    MyArea.Select
    MyArea.Copy
    Set MyArea = MyScreen.Area(23,6,23,6,,3)
    MyArea.Select
    MyArea.Paste

End Sub
```

## Right

### Property, Area Object 3270 5250 VT

This property returns or sets the column of the screen where the area ends.

#### Syntax

Area.Right

#### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1,1,MyScreen.Rows,1,,3)

    MsgBox "The text in column 1 is: " + MyArea.Value

    ' This moves the session to the left of the screen.
    MyArea.Left = 2
    MyArea.Right = 2

    MsgBox "The test in column 2 is: " + MyArea.Value
End Sub
```

## Select

**Method, Area Object 3270 5250 VT**

This method selects the Area object.

### Syntax

Area.**Select**

### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")

    ' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    Set MyArea = MyScreen.Area(5, 5, 10, 10, , 3)
    MyArea.Select
    MsgBox "Select the Area object ..."

    Set MyArea = MyScreen.Select(11, 11, 20, 20)
    MsgBox "Select the Screen object ..."

End Sub
```

**Top****Property, Area Object 3270 5250 VT**

This property returns or sets the row where the area starts.

**Syntax**

Area.**Top**

**Example**

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")

    ' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This illustrates the Top and Bottom properties for an Area
    ' object. For demonstration purposes, the Select method is used,
    ' but it is not required for setting Top and Bottom properties.
    MsgBox "Press to demonstrate the Top and Bottom properties for Area
objects."

    Set MyArea = MyScreen.Area(1, 1, MyScreen.Rows, MyScreen.Cols,,3)
    MyArea.Select

    For i = 1 to Int(MyScreen.Rows/2)
        MyArea.Top = MyArea.Top + 1
        MyArea.Select
        MyArea.Bottom = MyArea.Bottom - 1
        MyArea.Select
    Next

    ' This demonstrates the Top property for a Session object.
    MsgBox "Press to demonstrate the Top property for Session objects."

    Sess.Top = 50
    MsgBox "The session top is now at 50. Press to move session top to 1"

    Sess.Top = 1
    MsgBox "The session top is now at 1. Press to move session top to 100"

    Sess.Top = 100

End Sub
```

**Type****Property, Area Object 3270 5250 VT**

You can use this property to determine how the area is selected (for example, as a rectangular range of characters or a continuous stream of characters).

**Syntax**`Area.Type`**Example**

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")

    ' Assumes an open session
    Set Sess = Sys.ActiveSession

    'Using Type with Session object
    Select Case Sess.Type
        Case 1
            MsgBox "This is a 3270 session."
        Case 2
            MsgBox "This is a 5250 session."
        Case 3
            MsgBox "This is a VT session."
    End Select

    'Using Type with Area object
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(5, 5, 10, 10, , 3)

    Select Case MyArea.Type
        Case 0
            MsgBox "The Area type is xNONE."
        Case 1
            MsgBox "The Area type is xPOINT."
        Case 2
            MsgBox "The Area type is xSTREAM."
        Case 3
            MsgBox "The Area type is xBLOCK."
    End Select
End Sub
```

## Value

### Property, Area Object 3270 5250 VT

You can use this property to return or set the text in the Area or Operator Information Area (OIA).

#### Syntax

Area.Value

#### Example

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As Object

    Set Sys = CreateObject("HEOLEAUT.HostExSystem")
    Set Sess = Sys.ActiveSession

    ' Assumes an open session
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(23, 7, 23, 7, , 3)

    MyArea.Value = "a"
    MyScreen.SendKeys ("<Pf1>")

    Set MyArea = MyScreen.Area(3, 2, 3, 7, , 3)

    nCounter = 0
    maxCounter = 500
    Wait$ = "USERID"

    While ((Watch$ <> Wait$) And (nCounter < maxCounter))
        Watch$ = MyArea.Value
        nCounter = nCounter + 1
    Wend

    If nCounter = maxCounter Then
        MsgBox Wait$ + " not found in time."
    Else
        Set MyArea = MyScreen.Area(20, 16, 20, 23, , 3)
        MyArea = "user1"
        ' This is identical to MyArea.Value = "user1"
        MyScreen.SendKeys ("<Enter>")
    End If
End Sub
```

## 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
- Attr
- IsBold
- IsModified
- IsPenSelectable
- Length
- Text
- Fields Collection
- ExtAttr
- IsHidden
- IsNumeric
- IsProtected
- Pos

## **Hosts.Fields Collection Object**

### **Object, Field Object [3270](#) [5250](#)**

The Hosts.Fields Collection Object is a collection of Field objects. It represents all fields on the current host screen in a logical fashion. Although it is not an array, you may think of it as a dynamic array. Although the object contains its own set of properties, the particularly useful ones are the Fields(n), and Fields.Count properties. The Fields(n) return a pointer to the *n*th field available. The Field index starts at 0. Fields.Count returns the total number of fields.

In 3270 mode, fields can be protected or unprotected. In a screen that contains at least one attribute, you can typically represent the entire screen by a collection of fields.

In 5250 mode, fields can only be unprotected. They represent the entries in the format table, areas in which you can type. You cannot use a field object to get data from a protected portion of the screen.

**Syntax**

```
Dim Fld as Object  
Set Fld = CurrentHost.Fields(n)  
iNumFields = CurrentHost.Fields.Count
```

**Example**

```
Sub Main  
    Dim Fld as Object  
    Dim HE as Object  
    Set HE = CreateObject( "HostExplorer" )  
    Set Fld = HE.CurrentHost.Fields(2)  
  
    MsgBox "Value of 2'nd field is: " + Fld.Text  
End Sub
```

**Attr****Property, Field Object [3270 5250](#)**

Within TN3270, this read-only property returns the 3270 attribute of the field selected by the Field object. You can use the following literals to test the bit options:

- ATTR\_MODIFIED
- ATTR\_BOLD
- ATTR\_NUMERIC
- ATTR\_PROTECTED
- ATTR\_NODISPLAY

Within TN5250, this read-only property returns the Field Format Word of the field selected by the Field object.

You can use the following literals to test the bit options:

- FFW\_BYPASS
- FFW\_ALLOWDUP
- FFW\_MDT
- FFW\_SHIFTEDITMASK
- FFW\_ALPHASHIFT
- FFW\_ALPHAONLY
- FFW\_NUMERICSHIFT
- FFW\_NUMERICONLY
- FFW\_KATASHIFT
- FFW\_DIGITSONLY
- FFW\_IO
- FFW\_SIGNEDNUMERIC
- FFW\_AUTOENTERONEXIT
- FFW\_FIELDEXITREQUIRED
- FFW\_MONOCASE
- FFW\_MANDATORYENTRY
- FFW\_RIGHTFILLMASK
- FFW\_NOADJUST
- FFW\_RIGHTADJUSTZEROFILL
- FFW\_RIGHTADJUSTBLANKFILL
- FFW\_MANDATORYFILL

## Syntax

iAttr% = **Attr**

## Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field 2 has an attribute of: " + Str$(Fld.Attr)
End Sub
```

## ExtAttr

### Property, Field Object 3270 5250

Within TN3270, this read-only property returns the extended 3270 attribute of the field selected by the Field object. The extended attribute defines the color and highlighting for the field. You can use the following literals to test the bit options:

- ATTR\_REVERSE
- ATTR\_BLINK
- ATTR\_UNDERLINE

Within TN5250, this read-only property returns the Field Control Word of the field selected by the Field object.

You can use the following literals to test the bit options:

- FCW\_LIGHTPEN
- FCW\_STRIPEANDLIGHTPEN

## Syntax

```
iExtAttr% = ExtAttr
```

## Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field 2 has an extended attribute of: " + Str$(Fld.ExtAttr)
End Sub
```

## IsBold

### Property, Field Object 3270

This read-only property returns TRUE (-1) if the field is bold or FALSE (0) if it is not.

#### Syntax

```
bVal = IsBold
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field is Bold " + Fld.IsBold
End Sub
```

## IsHidden

### Property, Field Object 3270 5250

This read-only property returns TRUE (-1) if the field is hidden (non-visible) or FALSE (0) if it is not (visible).

#### Syntax

```
bVal = IsHidden
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field is Hidden " + Fld.isHidden
End Sub
```

## IsModified

### Property, Field Object 3270

This read-only property returns TRUE (-1) if the field is modified or FALSE (0) if it is not.

#### Syntax

```
bVal = IsModified
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field is Modified " + Fld.IsModified
End Sub
```

## IsNumeric

### Property, Field Object 3270

This read-only property returns TRUE (-1) if the field is numeric or FALSE (0) if it is not.

#### Syntax

```
bVal = IsNumeric
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field is Numeric " + Fld.IsNumeric
End Sub
```

## **IsPenSelectable**

### **Property, Field Object 3270**

This read-only property returns TRUE (-1) if the field is pen selectable or FALSE (0) if it is not.

#### **Syntax**

```
bVal = IsPenSelectable
```

#### **Example**

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field is PenSelectable " + Fld.IsPenSelectable
End Sub
```

## **IsProtected**

### **Property, Field Object 3270**

This read-only property returns TRUE (-1) if the field is protected or FALSE (0) if it is not.

#### **Syntax**

```
bVal = IsProtected
```

#### **Example**

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field is Protected " + Fld.IsProtected
End Sub
```

## Length

### Property, Field Object 3270 5250

This read-only property returns the length of the field selected by the Field object.

#### Syntax

```
iLen% = Length
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field 2 has length: " + Str$(Fld.Length)
End Sub
```

## Pos

### Property, Field Object 3270 5250

This read-only property returns the position of the field selected by the Field object. The first position on the screen is 1.

#### Syntax

```
iPos% = Pos
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field 2 has position: " + Str$(Fld.Pos)
End Sub
```

## Text

### Property, Field Object 3270 5250

You can use this property to get or set the contents of the field as a string. The system converts all Nulls to Blanks and removes all trailing blanks.

#### Syntax

```
szText$ = Text
```

#### Example

```
Sub Main
    Dim Fld as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    ' Get second field object
    Set Fld = HE.CurrentHost.Fields(2)

    MsgBox "Field 2 has value: " + Fld.Text
End Sub
```

## Methods and Properties of the 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:

- Cfg3270 Objects List
- Cfg5250 Objects List
- CfgVT Objects List

The following methods and properties are available:

- ActionOnExist
- AlwaysAutoSkip
- Answerback
- AreaCode
- AutoClearXferMonitor
- AutoCopySelectedText
- AutoMacro
- AutoUnlockKeyboard
- AutoWrap
- BellMargin
- BitMode
- BlinkToltalic
- BSIsDel
- ClearAllTabStops
- ColorDisplay
- CompressBlankLinesInScrollbar
- ConcealAnswerback
- ConnectTimeout
- ConvertNulls
- Country<sub>=</sub>
- CRTToCRLF
- CursorKeyMode
- CursorMode
- CursorType
- DefaultRecvDir
- DefaultHeight
- OptimizedDisplayMode
- Password
- PrintBorder
- PrintDocumentName
- PrinterDeInit
- PrinterInit
- PrintFooter
- PrintHeader
- PrintLocation
- PrintOIA
- Profile
- ProportionalFonts
- RawAddFormFeed
- RawCaptureMode
- ReplyOEM
- ReRunAutoMacro
- RespectNumeric
- RightMargin
- SaveAppend
- SaveAttrsInScrollbar
- SaveConfirm
- SaveFile
- SaveFontOnExit
- SaveMode
- SaveProfile
- SaveProfileOnClose

- DefaultWidth
- Dev7171Terminal
- DirectToModem
- Display3DBorder
- DisplayAttr
- DisplayControlCodes
- DisplayRowCol
- DisplayUpperCase
- EntryAssist
- FileXferProtocol
- ForceAltSize
- ForceExactSize
- Host
- HostBGColor
- HostColor
- HostFGColor
- KermitBinPrefix
- KermitCompression
- KermitTextMode
- KermitUseFullPath
- KeyboardProfileName
- KeypadMode
- LeftMargin
- LinesInScrollbar
- Linemode
- LocalEcho
- LongName
- ShowDialupDlg
- ShowHotspots
- ShowNulls
- ShowRecvDialog
- SmoothScrolling
- SmoothScrollSpeed
- Sound
- StatusLineMode
- TabStop
- TCPPort
- TelnetEcho
- TelnetName
- TPRINTDestination
- TypeAhead
- UPSSet
- UseDialProperties
- WindowTitle
- WordWrap
- XferBlockSize
- XferHostSystem
- XferProgramName
- XferStartAction
- XModem16BitCrc
- XModemPkt1024
- XModemSendTimeout
- YModemSendTimeout
- YModemUseFullPath

- LUName
- Modem
- MultiLineDelete
- MultiLineInsert
- Notify
- NRCSet
- OnDisconnect
- Online
- ZModemAutoDownload
- ZModemCrashRecovery
- ZModemMaxErr
- ZModemOverwrite
- ZModemSlidingBytes
- ZModemSlidingWin
- ZModemUseFullPath

## 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.

### **Example**

```
'$include:"-E\hebasic.ebh"

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

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    With HE.CurrentHost.Cfg5250.CursorType = CURSOR_BLOCK
        End With
    End Sub
'$include:"-E\hebasic.ebh"

Sub Main
    Dim Cfg as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    Set Cfg = HE.CurrentHost.Cfg5250

    Cfg.CursorType = CURSOR_BLOCK
End Sub
```

## 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( "HostExplorer" )

    HE.CurrentHost.CfgVT.CursorType = CURSOR_BLOCK
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    With HE.CurrentHost.CfgVT.CursorType = CURSOR_BLOCK
        End With
    End Sub

'$include:"-E\hebasic.ebh"

Sub Main
    Dim Cfg as Object
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    Set Cfg = HE.CurrentHost.CfgVT

    Cfg.CursorType = CURSOR_BLOCK
End Sub
```

## ActionOnExist

### Property, CfgVT Object

You can use this property to get or set the Action On Exist setting. Possible values are:

- XFER\_OVERWRITE
- XFER\_RENAME
- XFER\_SKIP

#### Syntax

```
iAction% = ActionOnExist%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue% = HE.CurrentHost.CfgVT.ActionOnExist
    HE.CurrentHost.CfgVT.ActionOnExist = XFER_OVERWRITE
End Sub
```

## AlwaysAutoSkip

### Property, Cfg3270 Object

You can use this property to get or set the Always Auto Skip option.

#### Syntax

```
AlwaysAutoSkip = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.AlwaysAutoSkip
    HE.CurrentHost.Cfg3270.AlwaysAutoSkip = TRUE
End Sub
```

## Answerback

### Property, CfgVT Object

You can use this property to get or set the Answerback string.

#### Syntax

**Answerback** = szNewAnswerbackMsg\$

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue$ = HE.CurrentHost.CfgVT.Answerback
    HE.CurrentHost.CfgVT.Answerback = "Hello World"
End Sub
```

## AreaCode

### Property, CfgVT Object

You can use this property to get or set the Area Code when using a Modem connection.

#### Syntax

**AreaCode** = iNewAreaCode%

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iAreaCode% = HE.CurrentHost.CfgVT.AreaCode
    HE.CurrentHost.CfgVT.AreaCode = 514
End Sub
```

## **AutoClearXferMonitor**

### **Property, Cfg3270 and CfgVT Objects**

You can use this property to get or set the Auto Clear File Transfer Monitor option.

#### **Syntax**

```
AutoClearXferMonitor = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.AutoClearXferMonitor
    HE.CurrentHost.Cfg3270.AutoClearXferMonitor = TRUE
End Sub
```

## **AutoCopySelectedText**

### **Property, Cfg3270 , Cfg5250 , and CfgVT Objects**

You can use this property to get or set the Auto Copy Selected Text option.

#### **Syntax**

```
AutoCopySelectedText = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.AutoCopySelectedText
    HE.CurrentHost.Cfg3270.AutoCopySelectedText = TRUE
End Sub
```

## AutoMacro

### Property, Cfg3270 , Cfg5250 , and CfgVT Objects

You can use this property to get or set the Auto Quick-Key/Macro/Quick Script Name option. The value can be the name of a Quick-Key (\*.qk3, \*.qk5, \*.qkv), macro (\*.ebs or .ebx), or Quick Script (\*.qs3, \*.qs5, or \*.qsv) file.

**Note:** If you specify only the file name, make sure that the file is in its default directory. Otherwise, specify the full directory path.

#### Syntax

```
AutoMacro = AutoMacroName$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szAutoMacro$ = HE.CurrentHost.Cfg3270.AutoMacro
    HE.CurrentHost.Cfg3270.AutoMacro = "RunMeFirst.ebs"
End Sub
```

## AutoUnlockKeyboard

### Property, Cfg3270 Object

You can use this property to get or set the Automatic Keyboard Unlock option.

#### Syntax

```
AutoUnlockKeyboard = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.AutoUnlockKeyboard
    HE.CurrentHost.Cfg3270.AutoUnlockKeyboard = TRUE
End Sub
```

## AutoWrap

### Property, CfgVT Object

You can use this property to get or set the Auto Wrap option. When set (TRUE), HostExplorer automatically wraps lines that extend past the last column on the screen. When reset (FALSE), HostExplorer discards data that extends past the end of the screen.

#### Syntax

```
AutoWrap = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.AutoWrap
    HE.CurrentHost.CfgVT.AutoWrap = TRUE
End Sub
```

## BellMargin

### Property, Cfg3270 and Cfg5250 Object

You can use this property to get or set the Bell Margin column. The column value must be between 1 and the total number of columns.

#### Syntax

```
BellMargin = iNewColumn%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue% = HE.CurrentHost.Cfg3270.BellMargin
    HE.CurrentHost.Cfg3270.BellMargin = 72
End Sub
```

## **BitMode**

### **Property, CfgVT Object**

You can use this property to get or set the 8-Bit Transmission Mode option.

#### **Syntax**

**BitMode** = TRUE/FALSE

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.BitMode
    HE.CurrentHost.CfgVT.BitMode = TRUE
End Sub
```

## **BlinkToItalic**

### **Property, Cfg3270 , Cfg5250 , and CfgVT Object**

You can use this property to get or set the Convert Blink to Italic option.

#### **Syntax**

**BlinkToItalic** = TRUE/FALSE

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.BlinkToItalic
    HE.CurrentHost.Cfg3270.BlinkToItalic = TRUE
End Sub
```

**BSIsDel****Property, CfgVT Object**

You can use this property to get or set the Backspace Key option. When set (TRUE), the Backspace key will send a Del (127) code. When reset (FALSE), the Backspace key sends a BS (8) code.

**Syntax****BSIsDel = TRUE/FALSE****Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.BSIsDel
    HE.CurrentHost.CfgVT.BSIsDel = TRUE
End Sub
```

**ClearAllTabStops****Method, Cfg3270 , Cfg5250 , and CfgVT Objects**

You can use this method to clear all tab stops.

**Syntax****ClearAllTabStops****Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.ClearAllTabStops
End Sub
```

## **ColorDisplay**

### **Property, Cfg5250 Object**

You can use this property to get or set the Color Display option for a 5250 terminal. This determines whether HostExplorer will use the values in the color attributes as a color map (TRUE) or as highlighting options (FALSE).

#### **Syntax**

```
ColorDisplay = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg5250.ColorDisplay
    HE.CurrentHost.Cfg5250.ColorDisplay = TRUE
End Sub
```

## **CompressBlankLinesInScrollbar**

### **Property, CfgVT Object**

You can use this property to get or set the Compress Blank Lines In Scrollback option. When enabled, HostExplorer does not add blank lines to the scrollback buffer. Changing this option does not affect the current contents of the scrollback buffer.

#### **Syntax**

```
CompressBlankLinesInScrollbar = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.CompressBlankLinesInScrollbar
    HE.CurrentHost.CfgVT.CompressBlankLinesInScrollbar = TRUE
End Sub
```

## ConcealAnswerback

### Property, CfgVT Object

You can use this property to get or set the Conceal Answerback option. If you set this property to FALSE and it was previously set to TRUE, the Answerback message will be automatically cleared to nulls.

#### Syntax

```
ConcealAnswerback = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.ConcealAnswerback
    HE.CurrentHost.CfgVT.ConcealAnswerback = TRUE
End Sub
```

## ConnectTimeout

### Property, Cfg3270 , Cfg5250 , and CfgVT Objects

You can use this property to get or set the connect-timeout option. Use this option when establishing a new session.

#### Syntax

```
ConnectTimeout = iNewTimeout%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.ConnectTimeout
    HE.CurrentHost.Cfg3270.ConnectTimeout = 60
End Sub
```

## ConvertNulls

### Property, Cfg3270 Object

You can use this property to get or set the Convert Nulls to Blanks option.

#### Syntax

```
ConvertNulls = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ConvertNulls
    HE.CurrentHost.Cfg3270.ConvertNulls = TRUE
End Sub
```

## Country

### Property, CfgVT Object

You can use this property to get or set the country from which you are dialing.

#### Syntax

```
Country = szCountry$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szCountry$ = HE.CurrentHost.CfgVT.Country
    HE.CurrentHost.CfgVT.Country = "Canada"
End Sub
```

## CRTToCRLF

### Property, CfgVT Object

You can use this property to get or set the CR to CR/LF option. When set to TRUE, the return key will send a CR/LF sequence. When reset to FALSE, the return key sends CR.

#### Syntax

```
CRTToCRLF = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.CRTToCRLF
    HE.CurrentHost.CfgVT.CRTToCRLF = TRUE
End Sub
```

## CursorKeyMode

### Property, CfgVT Object

You can use this property to get or set the Cursor Key Mode. When set to TRUE, the cursor key is in Application Mode. When reset to FALSE, the cursor keys operate in normal mode.

#### Syntax

```
CursorKeyMode = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.CursorKeyMode
    HE.CurrentHost.CfgVT.CursorKeyMode = TRUE
End Sub
```

## **CursorMode**

### **Property, Cfg3270 , Cfg5250 , and CfgVT Objects**

You can use this property to get or set the Cursor Mode option. The values can be CURSOR\_BLINK or CURSOR\_SOLID.

#### **Syntax**

```
CursorMode = iNewMode%
```

#### **Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.CursorMode
    HE.CurrentHost.Cfg3270.CursorMode = CURSOR_BLINK
End Sub
```

## **CursorType**

### **Property, Cfg3270, Cfg5250, and CfgVT Objects**

You can use this property to get or set the Cursor Type option. The possible values are:

- CURSOR\_UNDERSCORE
- CURSOR\_BLOCK
- CURSOR\_LINE

#### **Syntax**

```
CursorType = iNewType%
```

#### **Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )
    iValue = HE.CurrentHost.Cfg3270.CursorType
    HE.CurrentHost.Cfg3270.CursorType = CURSOR_BLOCK
End Sub
```

## **DefaultHeight**

### **Property, CfgVT Object**

You can use this property to get or set the Default Height of the session. The possible values are between the range of 24 and 72.

#### **Syntax**

```
DefaultHeight = inewValue%
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.DefaultHeight
    HE.CurrentHost.CfgVT.DefaultHeight = 24
End Sub
```

## **DefaultRecvDir**

### **Property, CfgVT Object**

You can use this property to get or set the default receive path for file downloads.

#### **Syntax**

```
szPath$ = DefaultRecvDir
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szPath = HE.CurrentHost.CfgVT.DefaultRecvDir
    HE.CurrentHost.CfgVT.DefaultRecvDir = "c:\temp"
End Sub
```

## **DefaultWidth**

### **Property, CfgVT Object**

You can use this property to get or set the Default Width of the session. The possible values are between the range of 80 and 200.

#### **Syntax**

```
DefaultWidth = iNewValue%
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    Value = HE.CurrentHost.CfgVT.DefaultWidth
    HE.CurrentHost.CfgVT.DefaultWidth = 80
End Sub
```

## **Dev7171Terminal**

### **Property, Cfg3270 Object**

You can use this property to get or set the 7171 Passthru Printing terminal type option. The possible values are:

- DEV7171\_VT100
- DEV7171\_IBM3164

#### **Syntax**

```
Dev7171Terminal = iNewTermType%
```

#### **Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.Dev7171Terminal
    HE.CurrentHost.Cfg3270.Dev7171Terminal = DEV7171_VT100
End Sub
```

## **DirectToModem** **Property, CfgVT Object**

You can use this property to get or set the DirectToModem flag. When set, this flag causes the emulator to connect to the modem immediately and bypass the dialing stage. This lets you use a modem link as if it were a COM port.

### **Syntax**

**DirectToModem** = TRUE/FALSE

### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.DirectToModem
    HE.CurrentHost.CfgVT.DirectToModem = TRUE
End Sub
```

## **Display3DBorder** **Property, Cfg3270, Cfg5250, and CfgVT Objects**

You can use this property to get or set the Frame Terminal as Workspace's Background option. This option determines whether HostExplorer frames the terminal as the background of the workspace.

### **Syntax**

**Display3DBorder** = TRUE/FALSE

### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.Display3DBorder
    HE.CurrentHost.Cfg3270.Display3DBorder = TRUE

End Sub
```

## DisplayAttr Property, Cfg3270 Object

You can use this property to get or set the Display Attributes option.

### Syntax

```
DisplayAttr = TRUE/FALSE
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.DisplayAttr
    HE.CurrentHost.Cfg3270.DisplayAttr = TRUE

End Sub
```

## DisplayControlCodes Property, CfgVT Object

You can use this property to get or set the Display Control Codes option. When set to TRUE, the system displays control codes, which are received. When reset to FALSE, HostExplorer acts upon the control codes it receives.

### Syntax

```
DisplayControlCodes = TRUE/FALSE
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.DisplayControlCodes
    HE.CurrentHost.CfgVT.DisplayControlCodes = TRUE

End Sub
```

## DisplayRowCol

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Display Row/Col Indicator option.

#### Syntax

**DisplayRowCol** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.DisplayRowCol
    HE.CurrentHost.Cfg3270.DisplayRowCol = TRUE

End Sub
```

## DisplayUpperCase

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the All Upper Case option.

#### Syntax

**DisplayUpperCase** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.DisplayUpperCase
    HE.CurrentHost.Cfg3270.DisplayUpperCase = TRUE

End Sub
```

## EntryAssist

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Entry Assist option.

#### Syntax

```
EntryAssist = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.EntryAssist
    HE.CurrentHost.Cfg3270.EntryAssist = TRUE

End Sub
```

## FileXferProtocol

### Property, CfgVT Object

You can use this property to get or set the default file transfer protocol used in VT mode. The possible values are:

- XFER\_TYPE\_XMODEM
- XFER\_TYPE\_YMODEM
- XFER\_TYPE\_ZMODEM
- XFER\_TYPE\_KERMIT

#### Syntax

```
FileXferProtocol = iValue%
```

#### Example

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.FileXferProtocol
    HE.CurrentHost.CfgVT.FileXferProtocol = XFER_TYPE_ZMODEM

End Sub
```

## ForceAltSize

### Property, Cfg3270 Object

You can use this property to get or set the Force Alternate Size option.

#### Syntax

**ForceAltSize** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ForceAltSize
    HE.CurrentHost.Cfg3270.ForceAltSize = TRUE

End Sub
```

## ForceExactSize

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Force Exact Size Font option.

#### Syntax

**ForceExactSize** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ForceExactSize
    HE.CurrentHost.Cfg3270.ForceExactSize = TRUE

End Sub
```

## Host

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the IP Host/Gateway option (for Telnet connections) or the phone number to dial (for Modem connections).

#### Syntax

```
Host = szNewHost$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue$ = HE.CurrentHost.Cfg3270.Host
    HE.CurrentHost.Cfg3270.Host = "myhost.mydomain"
End Sub
```

## HostBGColor

### Method, Cfg3270, Cfg5250, and CfgVT Objects

You can use this method to retrieve the background color for a given 3270, 5250, or Telnet system color.

#### Syntax

```
iBG% = .HostBGColor( iColor% )
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iBG% = HE.CurrentHost.Cfg3270.HostBGColor( 2 )
    MsgBox "The value for that specified background system color is " & iBG
    % , , "Color Information"
End Sub
```

## HostColor

### Method, Cfg3270, Cfg5250, and CfgVT Objects

You can use this method to set the foreground and background colors for 3270, 5250, or Telnet system color.

#### Syntax

```
HostColor iColor%, iFG%, iBG%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.HostColor 2, 5, 1
End Sub
```

## HostFGColor

### Method, Cfg3270, Cfg5250, and CfgVT Objects

You can use this method to retrieve the foreground color for a given 3270, 5250, or Telnet system color.

#### Syntax

```
iFG% = .HostFGColor( iColor% )
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iFG% = HE.CurrentHost.Cfg3270.HostFGColor( 2 )
    MsgBox "The value for that specified foreground system color is "
    & iFG% & ". Color Information"
End Sub
```

## **KermitBinPrefix**

### **Property, CfgVT Object**

You can use this property to get or set the Binary Prefix flag.

#### **Syntax**

```
KermitBinPrefix = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.KermitBinPrefix
    HE.CurrentHost.CfgVT.KermitBinPrefix = FALSE
End Sub
```

## **KermitCompression**

### **Property, CfgVT Object**

You can use this property to get or set the RLE Compression flag.

#### **Syntax**

```
KermitCompression = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.KermitCompression
    HE.CurrentHost.CfgVT.KermitCompression = FALSE
End Sub
```

## KermitTextMode

### Property, CfgVT Object

You can use this property to get or set the Text Mode flag.

#### Syntax

**KermitTextMode** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.KermitTextMode
    HE.CurrentHost.CfgVT.KermitTextMode = FALSE
End Sub
```

## KermitUseFullPath

### Property, CfgVT Object

You can use this property to get or set the Use Full Path Name to/from Host flag.

#### Syntax

**KermitUseFullPath** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.KermitUseFullPath
    HE.CurrentHost.CfgVT.KermitUseFullPath = FALSE
End Sub
```

## KeyboardProfileName

### Method, Cfg3270, Cfg5250, and CfgVT Objects

You can use this method to change the current keyboard profile. This method loads the specified keyboard profile as the active session keyboard profile.

**Syntax**            **KeyboardProfileName** szKeyProfile\$

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.KeyboardProfileName = "MYKEYBOARD"
End Sub
```

## KeypadMode

### Property, CfgVT Object

You can use this property to get or set the Keypad mode. With a value of TRUE, the keypad operates in Application mode. When reset to FALSE, the keypad operates in Numeric mode.

**Syntax**            **KeypadMode** = TRUE/FALSE

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.KeypadMode
    HE.CurrentHost.CfgVT.KeypadMode = TRUE
End Sub
```

## LeftMargin

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Left Margin column. The column value must be between 1 and the number of columns.

#### Syntax

```
LeftMargin = iColumn%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.LeftMargin
    HE.CurrentHost.Cfg3270.LeftMargin = 5
End Sub
```

## LinesInScrollbar

### Property, CfgVT Object

You can use this property to get or set the Lines Attrs In Scrollback option. This is the number of lines available in the Scrollback buffer. Use a value of 0 to disable the scrollback buffer.

#### Syntax

```
LinesInScrollbar = iNewLines%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.LinesInScrollbar
    HE.CurrentHost.CfgVT.LinesInScrollbar= 120
End Sub
```

## **Linemode**

### **Property, CfgVT Object**

You can use this property to get or set the Telnet Linemode option. This option is used during telnet option negotiation with a host. The possible values are:

- 0—Don't do linemode.
- 1—Always do linemode.
- 2—During Local Echo.
- 3—When Not in SGA.
- 4—When Local Echo or Not SGA.

#### **Syntax**

```
Linesmode = iLineMode%
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.Linemode
    HE.CurrentHost.CfgVT.Linemode = 0
End Sub
```

## **LocalEcho**

### **Property, CfgVT Object**

You can use this property to get or set the Local Echo option. When set to TRUE, HostExplorer echoes all data entered. When reset to FALSE, HostExplorer does not echo any data entered.

#### **Syntax**

```
LocalEcho = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.LocalEcho
    HE.CurrentHost.CfgVT.LocalEcho = TRUE
End Sub
```

## LongName

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Session Long Name option. This property will accept only the first 8 characters.

#### Syntax

```
LongName = szNewName$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.LongName
    HE.CurrentHost.Cfg3270.LongName = "VM/CMS"
End Sub
```

## LUName

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Logical Units (LU) Name option.

#### Syntax

```
LUName = szNewLUName$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.LUName
    HE.CurrentHost.Cfg3270.LUName = "LU1234"
End Sub
```

## Modem

### Property, CfgVT Object

You can use this property to get or set the modem used for the current connection. You can supply the full string as located in the Modem applet in the Control Panel or any unique substring of the modem string.

#### Syntax

```
Modem = szNewModem$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szCurrentModem$ = HE.CurrentHost.CfgVT.Modem
    HE.CurrentHost.CfgVT.Modem = "MyModem"
End Sub
```

## MultiLineDelete

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Multiline Delete Mode option.

#### Syntax

```
MultiLineDelete = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.MultiLineDelete
    HE.CurrentHost.Cfg3270.MultiLineDelete = TRUE
End Sub
```

## MultilineInsert

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Multiline Insert Mode option.

#### Syntax

```
MultilineInsert = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.MultilineInsert
    HE.CurrentHost.Cfg3270.MultilineInsert = TRUE
End Sub
```

## Notify

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Notify (Update Alarm) option.

#### Syntax

```
Notify = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.Notify
    HE.CurrentHost.Cfg3270.Notify = TRUE
End Sub
```

## NRCSet

### Property, CfgVT Object

You can use this property to get or set the National Replacement Character (NRC) set. The following string values can be set:

- None
- ISO United Kingdom
- DEC Finnish
- ISO French
- DEC French Canadian
- ISO German
- ISO Italian
- ISO Norwegian/Danish
- DEC Norwegian/Danish
- DEC Portuguese
- ISO Spanish
- DEC Swedish
- DEC Swiss

### Syntax

**NRCSet** = iNewValue\$

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.CfgVT.NRCSet
    HE.CurrentHost.CfgVT.NRCSet = "None"
End Sub
```

## OnDisconnect

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Upon Disconnect option. You can use the following literals to read or change the value:

- ONDISC\_CLOSEWINDOW
- ONDISC\_KEEPWINDOWOPEN
- ONDISC\_RESTARTSESSION
- ONDISC\_SHOWNEWSSESSION

#### Syntax

**OnDisconnect** = iNewDisconnectMode%

#### Example

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.OnDisconnect
    HE.CurrentHost.Cfg3270.OnDisconnect= ONDISC_CLOSEWINDOW
End Sub
```

## Online Property, CfgVT Object

You can use this property to get or set the Online option. When set to TRUE, the terminal is online and sends data to the remote system. When reset to FALSE, the terminal is offline and does not send any data to the remote system.

### Syntax

```
Online = TRUE/FALSE
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.Online
    HE.CurrentHost.CfgVT.Online = TRUE
End Sub
```

## OptimizedDisplayMode Property, CfgVT Object

You can use this property to get or set the Display Mode option. When set TRUE, HostExplorer operates in Optimized Display mode. In this mode, the system updates the screen only at the end of telnet buffers received. When reset to FALSE, HostExplorer operates in Realistic Normal Display mode. In this mode, as HostExplorer receives data, it is displayed. Due to the overhead, Realistic Normal Display mode is much slower than Optimized mode.

### Syntax

```
OptimizedDisplayMode = TRUE/FALSE
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.OptimizedDisplayMode
    HE.CurrentHost.CfgVT.OptimizedDisplayMode = TRUE
End Sub
```

## Password

Method, Cfg3270, Cfg5250, and CfgVT Objects

You can use this method to set the contents of the Password variable used in the macro system.

### Syntax

```
Password = szNewPassword$
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.Password = "givemeaccess"
End Sub
```

## PrintBorder

Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Print Border setting.

### Syntax

```
PrintBorder = TRUE/FALSE
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.PrintBorder
    HE.CurrentHost.Cfg3270.PrintBorder = TRUE
End Sub
```

## PrintDocumentName

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to set or retrieve the document name. You can insert the document name in the print header or print footer using the &F variable.

#### Syntax

```
PrintDocumentName = szNewDocumentName$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue$ = HE.CurrentHost.Cfg3270.PrintDocumentName
    HE.CurrentHost.Cfg3270.PrintDocumentName = "Test Document"
End Sub
```

## PrinterDeInit

### Property, Cfg3270 Object

You can use this property to get or set the Printer Deinitialization String.

#### Syntax

```
PrinterDeInit = szNewDeInit$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.PrinterDeInit
    HE.CurrentHost.Cfg3270.PrinterDeInit = "\eE"
End Sub
```

## **PrinterInit**

### **Property, Cfg3270 Object**

You can use this property to get or set the Printer Initialization String.

#### **Syntax**

```
PrinterInit = szNewInit$
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.PrinterInit
    HE.CurrentHost.Cfg3270.PrinterInit = "\eE"
End Sub
```

## **PrintFooter**

### **Property, Cfg3270, Cfg5250, and CfgVT Objects**

You can use this property to set or retrieve the Print Screen footer string. This string can contain the following special variables:

- &C—Insert computer name.
- &D—Insert current date.
- &F—Insert document name string.
- &P—Insert page number.
- &T—Insert current line.
- &U—Insert user name.

#### **Syntax**

```
PrintFooter = szNewPrintFooter$
```

#### **Example**

```
Sub Main
    Dim He as Object
    Set He = CreateObject( "HostExplorer" )

    szValue$ = He.CurrentHost.Cfg3270.PrintFooter
    He.CurrentHost.Cfg3270.PrintFooter = "&U - &D"
End Sub
```

## PrintHeader

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to set or retrieve the Print Screen header string. This string can contain the following special variables:

- &C—Insert computer name.
- &D—Insert current date.
- &F—Insert document name string.
- &P—Insert page number.
- &T—Insert current line.
- &U—Insert user name.

#### Syntax

```
PrintHeader = szNewPrintHeader$
```

#### Example

```
Sub Main
    Dim He as Object
    Set He = CreateObject( "HostExplorer" )

    szValue$ = He.CurrentHost.Cfg3270.PrintHeader
    He.CurrentHost.Cfg3270.PrintHeader = "&U - &D"
End Sub
```

## PrintLocation

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Print Location setting. You can use the following literals to test the values returned:

- PRINT\_CENTERED
- PRINT\_UPPERLEFT
- PRINT\_FITTOPAGE

**Syntax** **PrintLocation**

**Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.PrintLocation
    HE.CurrentHost.Cfg3270.PrintLocation = PRINT_CENTERED
End Sub
```

## **PrintOIA**

### **Property, Cfg3270, Cfg5250, and CfgVT Objects**

You can use this property to get or set the Print Operator Information Area (OIA) setting.

**Syntax** **PrintOIA** = TRUE/FALSE

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.PrintOIA
    HE.CurrentHost.Cfg3270.PrintOIA = TRUE
End Sub
```

## Profile

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the profile name for the session. This method takes one parameter that is the name of the profile followed by the name of the folder separated by a period. For example, to set a profile called "VMCMS" in the "3270" folder, pass a string of "VMCMS.3270". You can also pass a complete path/file specification (including extension).

#### Syntax

```
Profile = szNewProfile$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.Profile
    HE.CurrentHost.Cfg3270.Profile = "ProfileName"
End Sub
```

## ProportionalFonts

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Proportional Fonts option. This property determines whether HostExplorer displays fonts for which you can alter the font width.

**Note:** You can set this property to TRUE only if the font is proportional.

#### Syntax

```
ProportionalFonts = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHostSetFont "Arial Black",0, 12
    bValue = HE.CurrentHost.Cfg3270.ProportionalFonts
    HE.CurrentHost.Cfg3270.ProportionalFonts = TRUE
End Sub
```

## RawAddFormFeed

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Add FormFeed after Raw Print Screen option.

#### Syntax

```
RawAddFormFeed = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.RawAddFormFeed
    HE.CurrentHost.Cfg3270.RawAddFormFeed = TRUE
End Sub
```

## RawCaptureMode

### Property, CfgVT Object

You can use this property to get or set the Capture Mode option. When set to **TRUE** and capturing is enabled, HostExplorer captures the raw data as it receives it. When reset to **FALSE** and capturing is enabled, HostExplorer captures only lines that are terminated by LF, VT, or FF.

#### Syntax

```
RawCaptureMode = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.RawCaptureMode
    HE.CurrentHost.CfgVT.RawCaptureMode = TRUE
End Sub
```

## ReplyOEM

### Property, Cfg3270 Object

You can use this property to get or set the Send OEM Reply to RPQ option.

#### Syntax

```
ReplyOEM = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ReplyOEM
    HE.CurrentHost.Cfg3270.ReplyOEM = TRUE
End Sub
```

## ReRunAutoMacro

### Property, Cfg3270, Cfg5250, and CfgVT Objects

By default, HostExplorer executes the Auto Macro/Quick-Key/Quick Script upon initiating a session. HostExplorer re-executes the Auto Macro/Quick-Key/Quick Script when you re-connect to a session. You can use this property to get or set the ReRun Auto Macro option.

#### Syntax

```
ReRunAutoMacro = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ReRunAutoMacro
    HE.CurrentHost.Cfg3270.ReRunAutoMacro = TRUE
End Sub
```

## RespectNumeric Property, Cfg3270 Object

You can use this property to get or set the Respect Numeric Fields option.

### Syntax

```
RespectNumeric = TRUE/FALSE
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.RespectNumeric
    HE.CurrentHost.Cfg3270.RespectNumeric = TRUE
End Sub
```

## RightMargin Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Right Margin column. The column value must be between 2 and the total number of columns.

### Syntax

```
RightMargin = iNewColumn%
```

### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.RightMargin
    HE.CurrentHost.Cfg3270.RightMargin = 16
End Sub
```

## **SaveAppend**

### **Property, Cfg3270, Cfg5250, and CfgVT Objects**

You can use this property to get or set the Save Mode Append option.

#### **Syntax**

```
SaveAppend = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.SaveAppend
    HE.CurrentHost.Cfg3270.SaveAppend = TRUE
End Sub
```

## **SaveAttrsInScrollbar**

### **Property, CfgVT Object**

You can use this property to get or set the Save Attrs In Scrollback option.

#### **Syntax**

```
SaveAttrsInScrollbar = TRUE/FALSE
```

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.SaveAttrsInScrollbar
    HE.CurrentHost.CfgVT.SaveAttrsInScrollbar = TRUE
End Sub
```

## SaveConfirm

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Confirm All Saves option.

#### Syntax

```
SaveConfirm = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.SaveConfirm
    HE.CurrentHost.Cfg3270.SaveConfirm = TRUE
End Sub
```

## SaveFile

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Save Filename option.

#### Syntax

```
SaveFile = szSaveFile$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.SaveFile
    HE.CurrentHost.Cfg3270.SaveFile = "c:\saved.txt"
End Sub
```

## SaveFontOnExit

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Save Font Information on Exit option.

#### Syntax

```
SaveFontOnExit = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.SaveFontOnExit
    HE.CurrentHost.Cfg3270.SaveFontOnExit = TRUE
End Sub
```

## SaveMode

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to set the Save As option. You can set the value as SAVE\_ASCII or SAVE\_ANSI.

#### Syntax

```
SaveMode = iNewMode%
```

#### Example

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.SaveMode
    HE.CurrentHost.Cfg3270.SaveMode = SAVE_ANSI
End Sub
```

## SaveProfile

### Method, Cfg3270, Cfg5250, and CfgVT Objects

You can use this method to save the current session settings to a profile on disk. This method takes one parameter that is the name of the profile followed by the name of the folder separated by a period. For example, to save a profile called "VMCMS" in the "3270" folder, pass a string of "VMCMS.3270". You can also pass a complete path/file specification (including extension).

#### Syntax

```
SaveProfile szProfileName$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.SaveProfile "ProfileName.FolderName"
End Sub
```

## SaveProfileOnClose

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Save Profile on Window Close option.

#### Syntax

```
SaveProfileOnClose = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.SaveProfileOnClose
    HE.CurrentHost.Cfg3270.SaveProfileOnClose = TRUE
End Sub
```

## ShowDialupDlg

### Property, CfgVT Object

You can use this property to get or set the Always Show Connect Dialog flag. This property has no effect for connections that are started from OLE Automation. This property is only used to get/set values so that you can load/save profiles.

#### Syntax

**ShowDialupDlg** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.ShowDialupDlg
    HE.CurrentHost.CfgVT.ShowDialupDlg = TRUE
End Sub
```

## ShowHotspots

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Show Hotspots option.

#### Syntax

**ShowHotspots** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ShowHotspots
    HE.CurrentHost.Cfg3270.ShowHotspots = TRUE
End Sub
```

## ShowNulls

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Show Nulls option.

#### Syntax

```
ShowNulls = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.ShowNulls
    HE.CurrentHost.Cfg3270.ShowNulls = TRUE
End Sub
```

## ShowRecvDialog

### Property, CfgVT Object

You can use this property to get or set the Show Receive Dialog flag.

#### Syntax

```
ShowRecvDialog = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.ShowRecvDialog
    HE.CurrentHost.CfgVT.ShowRecvDialog = TRUE
End Sub
```

## **SmoothScrolling** Property, CfgVT Object

You can use this property to get or set the Smooth Scrolling option. When set to TRUE, HostExplorer operates in Realistic Smooth mode; that is, HostExplorer scrolls data using a smooth scroll method. When reset to FALSE, HostExplorer operates in Realistic Normal mode; that is, HostExplorer scrolls data line-by-line.

### **Syntax**

```
SmoothScrolling = TRUE/FALSE
```

### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.SmoothScrolling
    HE.CurrentHost.CfgVT.SmoothScrolling = TRUE
End Sub
```

## **SmoothScrollSpeed** Property, CfgVT Object

You can use this property to get or set the Smooth Scrolling Speed option. The value can be from 1 to 30. If the value is greater than the number of vertical pixels per cell, then scrolling occurs in Linemode.

### **Syntax**

```
SmoothScrollSpeed = iNewValue%
```

### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.SmoothScrollSpeed
    HE.CurrentHost.CfgVT.SmoothScrollSpeed = 2
End Sub
```

## Sound

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Sound option.

#### Syntax

```
Sound = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.Sound
    HE.CurrentHost.Cfg3270.Sound = TRUE
End Sub
```

## StatusLineMode

### Property, CfgVT Object

You can use this property to get or set the Status Linemode option. You can use the following literals to test values:

- STATUS\_NONE—Displays neither the application status line nor the host writable status line.
- STATUS\_HOSTWRITABLE—Displays the host writable status line, but does not display the application status line.
- STATUS\_INDICATOR—Displays the application status line in Windows Style but does not display the host writable status line.
- STATUS\_INDICATORTERMINAL—Displays the application status line in Terminal Style but does not display the host writable status line.
- STATUS\_INDICATOR\_HOSTWRITABLE—Displays the host writable status line and displays the application status line in Windows Style.
- STATUS\_INDICATORTERMINAL\_HOSTWRITABLE—Displays the host writable status line and displays the application status line in Terminal Style.

**Syntax**

```
StatusLineMode = inewValue%
```

**Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.StatusLineMode
    HE.CurrentHost.CfgVT.StatusLineMode = STATUS_INDICATOR
End Sub
```

**TabStop****Method, Cfg3270, Cfg5250, and CfgVT Objects**

You can use this method to set a tabstop at a given column. The column value must be between 1 and the total number of columns.

**Syntax**

```
TabStop iColumn%
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    HE.CurrentHost.Cfg3270.TabStop 23
End Sub
```

## TCPPort

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the TCP Port value. The default value is 23 for Telnet.

#### Syntax

```
TCPPort = iNewPort%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.TCPPort
    HE.CurrentHost.Cfg3270.TCPPort = 23
End Sub
```

## TelnetEcho

### Property, CfgVT Object

You can use this property to get or set the Telnet Echo option. The possible values are:

- 0—No
- 1—Yes
- 2—Automatic

#### Syntax

```
TelnetEcho = iNewEchoMode%
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.CfgVT.TelnetEcho
    HE.CurrentHost.CfgVT.TelnetEcho = 2
End Sub
```

## TelnetName

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set the Telnet Name Override option.

#### Syntax

```
TelnetName = szNewName$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.TelnetName
    HE.CurrentHost.Cfg3270.TelnetName = "IBM-3278-2"
End Sub
```

## TPRINTDestination

### Property, Cfg3270 Object

You can use this property to get or set the Send PCPRINT/TPRINT Output To option. The possible values are:

- TPRINT\_DEFAULT
- TPRINT\_CLIPBOARD
- TPRINT\_LPT1
- TPRINT\_LPT2
- TPRINT\_LPT3

#### Syntax

```
TPRINTDestination = iNewMode%
```

#### Example

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.TPRINTDestination
    HE.CurrentHost.Cfg3270.TPRINTDestination = TPRINT_CLIPBOARD
End Sub
```

## TypeAhead

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to retrieve or set the Type Ahead option.

#### Syntax

**TypeAhead** = TRUE/FALSE

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.TypeAhead
    HE.CurrentHost.Cfg3270.TypeAhead = TRUE
End Sub
```

## UPSSet

### Property, CfgVT Object

You can use this property to get or set the User Preferred Supplemental Character Set. The following string values can be set:

- "DEC Supplemental"
- "Europa3 333"
- "ISO Latin-1 (8859-1)"
- "ISO Latin-4 (8859-4)"
- "ISO Greek (8859-7)"
- "ISO Latin-9 (8859-15)"
- "PC Icelandic (861)"
- "PC Modern Turksih (857)"
- "PC Nordic (865)"
- "PC Slavic (852)"
- "PC Cyrillic (855)"
- "PC Baltic (921)"
- "PC Greek (851)"
- "DEC Technical"
- "DEC Greek (373)"
- "ISO Latin-2 (8859-2)"
- "ISO Cyrillic (8859-5)"
- "ISO Latin 5 (8859-9)"
- "PC English (437)"
- "PC Modern Greek (869)"
- "PC Multilingual (850)"
- "PC Portuguese (860)"
- "PC Canadian-French (863)"
- "PC Cyrillic (866)"
- "PC Estonian (922)"
- "HP Roman 8 (1051)"

- "PC Greek (437G,210)"
- "ISO Latin-3 (8859-3)"
- "Windows Latin-2 (1251)"
- "Windows Latin-1 (1252)"
- "Windows Latin-5 Turkish (1254)"
- "ISO Latin-6 (8859-10)"
- "ISO Latin-8 (8859-14)"
- "Slovenian 7-Bit"
- "HP Roman 9"
- "PC Baltic (775)"
- "Windows Cyrillic (1251)"
- "Windows Greek (1253)"
- "Windows Baltic (1257)"
- "ISO Latin-7 (8859-13)"
- "IBM3151 Special Graphics"

**Syntax**

```
UPSSet = iNewValue$
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.CfgVT.UPSSet
    HE.CurrentHost.CfgVT.UPSSet = "DEC Supplemental"
End Sub
```

## UseDialProperties

### Property, CfgVT Object

You can use this property to get or set the Use Area Code and Country Code flag.

**Syntax**

```
UseDialProperties = TRUE/FALSE
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.UseDialProperties
    HE.CurrentHost.CfgVT.UseDialProperties = FALSE
End Sub
```

## WindowTitle

### Property, Cfg3270, Cfg5250, and CfgVT Objects

You can use this property to get or set a string indicating the name and/or description displayed in the top right-hand corner of the session window.

#### Syntax

```
WindowTitle = newWindowTitle$
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.CfgVT.WindowTitle
    HE.CurrentHost.CfgVT.WindowTitle = "%l - %p(%h)"
End Sub
```

## WordWrap

### Property, Cfg3270 and Cfg5250 Objects

You can use this property to get or set the Word Wrap option.

#### Syntax

```
WordWrap = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bValue = HE.CurrentHost.Cfg3270.WordWrap
    HE.CurrentHost.Cfg3270.WordWrap = TRUE
End Sub
```

## XferBlockSize

### Property, Cfg3270 Object

You can use this property to get or set the File Transfer Block Size option. Possible values for this setting are:

- 256
- 512
- 1024
- 2048
- 4096
- 8192
- 16384
- 32768

#### Syntax

**XferBlockSize** = iNewBlockSize%

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.XferBlockSize
    HE.CurrentHost.Cfg3270.XferBlockSize = 8192
End Sub
```

## XferHostSystem

### Property, Cfg3270 Object

You can use this property to get or set the 3270 Host System option. The string value can be:

- CMS
- TSO
- CICS

**Syntax**

```
XferHostSystem = szNewSystem$
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.XferHostSystem
    HE.CurrentHost.Cfg3270.XferHostSystem = "CMS"
End Sub
```

## **XferProgramName**

### **Property, Cfg3270 Object**

You can use this property to get or set the File Transfer Program Name option.

**Syntax**

```
XferProgramName = szNewXferName$
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    szValue = HE.CurrentHost.Cfg3270.XferProgramName
    HE.CurrentHost.Cfg3270.XferProgramName = "IND$FILE"
End Sub
```

## **XferStartAction**

### **Property, Cfg3270 Object**

You can use this property to get or set the File Transfer Initial Action option. The possible values are:

- XFER\_NOACTION
- XFER\_HOME
- XFER\_ENTER
- XFER\_CLEAR

**Syntax**

```
XferStartAction = iNewAction%
```

**Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iValue = HE.CurrentHost.Cfg3270.XferStartAction
    HE.CurrentHost.Cfg3270.XferStartAction = XFER_NOACTION
End Sub
```

**XModem16BitCrc****Property, CfgVT Object**

You can use this property to get or set the 16-bit CRC flag. This enables 16-bit Cyclic Redundancy Checksum for XModem file transfers.

**Syntax**

```
XModem16BitCrc = TRUE/FALSE
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.XModem16BitCrc
    HE.CurrentHost.CfgVT.XModem16BitCrc = TRUE
End Sub
```

**XModemPkt1024****Property, CfgVT Object**

You can use this property to get or set the 1024-byte packet flag. This enables XModem to use 1024-byte packets if the host software supports this option.

**Syntax**

```
XModemPkt1024 = TRUE/FALSE
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.XModemPkt1024
    HE.CurrentHost.CfgVT.XModemPkt1024 = TRUE
End Sub
```

**XModemSendTimeout****Property, CfgVT Object**

You can use this property to get or set the send timeout for XModem file transfers. The time set is in milliseconds.

**Syntax**

```
XModemSendTimeout = lTimeout&
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    lVal& = HE.CurrentHost.CfgVT.XModemSendTimeout
    HE.CurrentHost.CfgVT.XModemSendTimeout = 15000
End Sub
```

**YModemSendTimeout****Property, CfgVT Object**

You can use this property to get or set the send timeout for Y-Modem file transfers. The time set is in milliseconds.

**Syntax**

```
YModemSendTimeout = lTimeout&
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    lVal& = HE.CurrentHost.CfgVT.YModemSendTimeout
    HE.CurrentHost.CfgVT.YModemSendTimeout = 15000
End Sub
```

## **YModemUseFullPath**

### **Property, CfgVT Object**

You can use this property to get or set the Use Full Path Name to/from Host flag.

#### **Syntax**

**YModemUseFullPath** = TRUE/FALSE

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.YModemUseFullPath
    HE.CurrentHost.CfgVT.YModemUseFullPath = FALSE
End Sub
```

## **ZModemAutoDownload**

### **Property, CfgVT Object**

You can use this property to get or set the Auto Download flag.

#### **Syntax**

**ZModemAutoDownload** = TRUE/FALSE

#### **Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.ZModemAutoDownload
    HE.CurrentHost.CfgVT.ZModemAutoDownload = FALSE
End Sub
```

## **ZModemCrashRecovery**

### **Property, CfgVT Object**

You can use this property to get or set the Crash Recovery flag.

#### **Syntax**

**ZModemCrashRecovery** = TRUE/FALSE

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.ZModemCrashRecovery
    HE.CurrentHost.CfgVT.ZModemCrashRecovery = FALSE
End Sub
```

## **ZModemMaxErr**

### **Property, CfgVT Object**

You can use this property to get or set the Crash Recovery value.

**Syntax**

```
ZModemMaxErr = iVal
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iVal = HE.CurrentHost.CfgVT.ZModemMaxErr
    HE.CurrentHost.CfgVT.ZModemMaxErr = 15
End Sub
```

## **ZModemOverwrite**

### **Property, CfgVT Object**

You can use this property to get or set the Overwrite Management Option.  
The possible values are:

- `Z_ALWAYS`
- `Z_APPEND`
- `Z_NEWER_LONGER`
- `Z_NEWER`
- `Z_NEVER`
- `Z_SIZE_DATE_DIFFER`

**Syntax**

```
ZModemOverwrite = iVal%
```

**Example**

```
'$include:"-E\hebasic.ebh"

Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iVal% = HE.CurrentHost.CfgVT.ZModemOverwrite
    HE.CurrentHost.CfgVT.ZModemOverwrite = Z_ALWAYS
End Sub
```

**ZModemSlidingBytes****Property, CfgVT Object**

You can use this property to get or set the Sliding Window Size value.

**Syntax**

```
ZModemSlidingBytes = iVal
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    iVal = HE.CurrentHost.CfgVT.ZModemSlidingBytes
    HE.CurrentHost.CfgVT.ZModemSlidingBytes = 8192
End Sub
```

**ZModemSlidingWin****Property, CfgVT Object**

You can use this property to get or set the Sliding Window flag.

**Syntax**

```
ZModemSlidingWin = TRUE/FALSE
```

**Example**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.ZModemSlidingWin
    HE.CurrentHost.CfgVT.ZModemSlidingWin = FALSE
End Sub
```

## ZModemUseFullPath

### Property, CfgVT Object

You can use this property to get or set the Use Full Path Name to/from Host flag.

#### Syntax

```
ZModemUseFullPath = TRUE/FALSE
```

#### Example

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    bVal = HE.CurrentHost.CfgVT.ZModemUseFullPath
    HE.CurrentHost.CfgVT.ZModemUseFullPath = FALSE
End Sub
```

## 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

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

### ***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 user-defined 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.

## ***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.

## ***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.

## ***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.

## ***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.

## ***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 application-level properties and methods.

**Option Explicit**—Specifies that all variables in a module must be explicitly declared.

**Object\_Initialize**—An object module-initialization subroutine. Not supported by Hummingbird Basic.

**Object\_Terminate**—An object module-termination subroutine. Not supported by Hummingbird Basic.

**Private**—Creates arrays (or simple variables) that are available to an entire macro or module, but not other macros or modules. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

**Public**—Creates arrays (or simple variables) that are available to an entire macro or module, but not other macros or modules. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

**ReDim**—Changes the upper and lower bounds of a dynamic array's dimension.

**Static**—Used inside procedures to declare variables and allocate storage space.

**Sub**—Declares the name, arguments, and code that form the body of a subprocedure.

**Type**—Declares a user-defined type, which can then be used in the Dim statement to declare a record variable.

## ***DefType Statement***

### **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.

## ***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.

**DlgControlId**—Returns the numeric ID of a dialog box control.

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

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

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

**DlgEnd**—Closes the dialog box.

**DlgFocus**—Sets the focus to a dialog box control.

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

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

**DlgName**—Returns the field name. Not supported by Hummingbird Basic.

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

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

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

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

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

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

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

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

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

## ***Error Handling Group***

### **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.

## ***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.

## ***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.

## ***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 (date) containing the time.

## ***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.

## *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.

## *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.**

## *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.

## **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.

## ***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 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.

**DropDownBox**—Defines a drop-down list box item.

**GroupBox**—Defines and draws a box that encloses sets of dialog box items, such as option boxes and check boxes.

**ListBox**—Displays a list of items from which the user can select one or more.

**OKButton**—Defines an OK button dialog box control. See Dialog Box Definition.

**OptionButton**—Defines an option button item.

**OptionGroup**—Groups a series of option buttons under one heading in a dialog box.

**Picture**—Defines a picture control in a custom dialog box.

**PushButton**—Defines a custom push button.

**Text**—Places lines of text in a dialog box.

**TextBox**—Sometimes called an edit field or edit control. Displays information entered at design time, entered by a user, or assigned to the control in the code at run time.

## ***User Input Group***

**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.

## ***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.

## File Transfer Options

### 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"
```

### 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.

## 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.

## 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.

## 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.

## Special Sequences

### 3270/5250 Special Sequences

The format of the string is identical to the one used in the EHLLAPI, DDE, and VB interfaces. Listed below are special-character combinations. Keep in mind that they are case-sensitive. This method returns 0 if all keys were processed successfully. The Keys method is not the most efficient method of transferring large amounts of information to the screen buffer. For faster access, use the Fields.Text property.

@B	Backtab
@C	Clear
@D	Delete
@E	Enter
@F	Erase EOF
@H	5250 Help
@I	Insert
@J	Next-Session
@L	Cursor Left
@N	Newline
@P	5250Print
@R	Reset
@T	Tab
@U	Cursor Up
@V	Cursor Down
@Z	Cursor Right
@0	Home
@<	Backspace
@1	PF1

@2	PF2
@3	PF3
@4	PF4
@5	PF5
@6	PF6
@7	PF7
@8	PF8
@9	PF9
@a	PF10
@b	PF11
@c	PF12
@d	PF13
@e	PF14
@f	PF15
@g	PF16
@h	PF17
@i	PF18
@j	PF19
@k	PF20
@l	PF21
@m	PF22
@n	PF23
@o	PF24
@u	Roll Up
@v	Roll Down
@x	PA1
@y	PA2

@z	PA3
@A@E	Field Exit
@A@F	Erase Input
@A@H	Test Request
@A@J	Cursor Select
@A@L	Fast Left
@A@Q	Attention
@A@-	Field Minus
@A@+	Field Plus
@A@<	Record Backspace
@A@Z	Fast Right
@A@t	Print Screen
@A@y	Next Word
@A@z	Prev Word
@S@x	Duplicate
@S@y	Field Mark

## 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.

## ***Entering Control Sequences***

### **VT**

The system treats the following in-line spaces as part of a sequence:

- \a—Bell (alert)
- \b—Backspace
- \e—Escape
- \f—Formfeed
- \n—Newline
- \r—Carriage Return
- \t—Horizontal Tab
- \v—Vertical Tab
- \'—Single quotation mark
- ""—Double quotation mark
- \\—Backslash
- \xhh—ASCII character in hexadecimal notation

When you type control sequences, you can use caret format. For example, to type a Ctrl-A value, you would type ^A. To type a caret, type the caret character twice, for example, (^^).

### **Example:**

```
Sub Main
    Dim HE as Object
    Set HE = CreateObject( "HostExplorer" )

    ' Press tab key twice
    HE.CurrentHost.Keys "\t\tTab Twice"

    ' Press Ctrl-C
    HE.CurrentHost.Keys "^C"

End Sub
```

## 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

## ChooseTerminalFont

**Method, IHETerminal 3270 5250 VT**

This method generates the Font Selection dialog box.

**Basic Syntax**      `HETerminal.ChooseTerminalFont`

**C++ Syntax**      `HRESULT IHETerminal::ChooseTerminalFont();`

**Parameters**      This method has no parameters.

**Basic Example**

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\aix.hep"
Profile.Load
Terminal.Connected = True
Terminal.ChooseTerminalFont
```

**C++ Example**

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

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

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

## Connect

**Method, IHETerminal 3270 5250 VT**

This method establishes a connection to the host.

**Basic Syntax**      `HETerminal.Connect`

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

**Parameters**      This method has no parameters.

**Basic Example**

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

**C++ Example**

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

## Disconnect

**Method, IHETerminal 3270 5250 VT**

This method terminates the connection to the host.

**Basic Syntax**

```
HETerminal.Disconnect
```

**C++ Syntax**

```
HRESULT IHETerminal::Disconnect();
```

**Parameters**

This method has no parameters.

**Basic Example**

```
Dim bVal As Boolean  
bVal = Terminal.Disconnected  
if (bVal = True) Then  
    Terminal.Disconnect();  
End If
```

**C++ Example**

```
HRESULT Disconnect();  
/* check if connected*/  
pITerminal->get_Disconnected(&bVal);  
if (bVal==VARIANT_TRUE)  
{  
    /*if connected, then disconnect*/  
    pITerminal->Disconnect();  
}
```

## EditSessionProperties

Method, IHETerminal 3270 5250 VT

This method generates the Edit Session Properties dialog box.

### Basic Syntax

```
Boolean = HETerminal.EditSessionProperties
```

### C++ Syntax

```
HRESULT IHETerminal::EditSessionProperties(VARIANT_BOOL * retVal);
```

### Parameters

*retVal*—A returned value of 1 indicates the operation was successful. A returned value of 0 indicates the operation was unsuccessful.

### Basic Example

```
Dim Profile As HEProfile
Dim retval As Integer
Dim bval As Boolean
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\aix.hep"
Profile.Load
Terminal.Connected = False
bval = Terminal.EditSessionProperties
End Sub
```

### C++ Example

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

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

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

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

## Properties of the Terminal Objects

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

- Connected
- Host
- Session
- TCPPort
- Transport

### ConnectBy

#### Property, IHETerminal 3270 5250 VT

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

##### Basic Syntax

```
HOSTEX_CONNECT_BY = HETerminal.ConnectBy
```

```
HETerminal.ConnectBy = HOSTEX_CONNECT_BY
```

##### C++ Syntax

```
HRESULT IHETerminal::get_ConnectBy([out, retval] HOSTEX_CONNECT_BY  
*pVal);  
  
HRESULT IHETerminal::put_ConnectBy([in] HOSTEX_CONNECT_BY * newVal);
```

##### Parameters

*pVal*—The returned value, specifying the connection method.

*newVal*—The set value, specifying the connection method.

##### Basic Example

```
Dim ConnBy As HOSTEX_CONNECT_BY  
ConnBy = Terminal.ConnectBy  
If (ConnBy <> HOSTEX_CONNECT_BY_MSSNA) Then  
    Terminal.ConnectBy =  
        HOSTEX_CONNECT_BY_MSSNA  
End If
```

##### C++ Example

```
HOSTEX_CONNECT_BY ConnBy;  
Terminal.get_ConnectBy(&ConnBy);  
if (ConnBy != HOSTEX_CONNECT_BY_MSSNA)  
{  
    ConnBy = HOSTEX_CONNECT_BY_MSSNA;  
    Terminal.put_ConnectBy(ConnBy);  
}
```

## Connected

### Property, IHETerminal 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HETerminal.Connected
```

```
HETerminal.Connected = Boolean
```

#### C++ Syntax

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

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

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that you are currently connected to the host. A returned value of VARIANT\_FALSE indicates that you are currently disconnected from the host.

*newVal*—A value of VARIANT\_TRUE indicates that you are currently connected to the host. A value of VARIANT\_FALSE indicates that you are currently disconnected from the host.

#### Basic Example

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\aix.hep"
Profile.Load
'Connect to host
Terminal.Connected = True
If (Terminal.Connected = True) Then
    Terminal.Connected = False
End If
```

**C++ Example**

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

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

//connect
VARIANT_BOOL bConnected = TRUE;
pITerminal->put_Connected(bConnected);

//check if connected, if so then disconnect
pITerminal->get_Connected(&bConnected);
if (bConnected==TRUE)
{
    //disconnect
    bConnected = FALSE;
    pITerminal->put_Connected(bConnected);
}
```

**Host****Property, IHETerminal 3270 5250 VT**

This property returns or sets the address of the host.

**Basic Syntax**

```
String = HETerminal.Host  
HETerminal.Host = String
```

**C++ Syntax**

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

**Parameters**

*pVal*—The returned address of the current host.

*newVal*—The address that you set for the current host.

**Basic Example**

```
If (Session.HostName <> "aix " Then
    Terminal.Host = "aix"
End If

Terminal.Connected = True

'get the current Host name
Dim HostName As String
HostName = Terminal.Host

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

**C++ Example**

```
BSTR bstrHost = SysAllocString(OLESTR("aix"));
pITerminal->put_Host(bstrHost);
SysFreeString(bstrHost);

bstrHost= SysAllocString(OLESTR("sunset.cs.concordia.ca"));
pITerminal->put_Host(bstrHost);
```

## Parser

**Property, IHETerminal 3270 5250 VT**

This property returns the pointer to the currently loaded Parser.

**Basic Syntax**

```
Dispatch = HETerminal.Parser
```

**C++ Syntax**

```
HRESULT IHETerminal::get_Parser([out, retval] IDispatch ** pVal);
```

**Parameters**

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

**Basic Example**

```
Dim Parser As HEParse
Set Parser = Terminal.Parser
Parser.SendKeys "ls -alt"
```

**C++ Example**

```
IDispatch* pIDispatch;
IHEParser* pIParser;
PITerminal->get_Parser(&pIDispatch);
pIDispatch->QueryInterface(IID_IHEParser, (void**)&pIParser);
BSTR bstrKeys = SysAllocString(OLESTR("ls -l"));
pIParser->SendKeys(bstrKeys);
SysFreeString(bstrKeys);
```

## Session

### Property, IHETerminal 3270 5250 VT

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

#### Basic Syntax

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

#### C++ Syntax

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

#### Parameters

*pVal*—The returned address of the Profile object.  
*newVal*—The set address of the Profile object.

#### Basic Example

```
Dim Profile As HEProfile  
Set Profile = Terminal.Session  
Profile.ProfileFileName = "C:\\aix.hep"  
Profile.Load  
Terminal.Connected = True
```

#### C++ Example

```
IHEProfile* pIProfile;  
IDispatch* pIDispatch;  
pITerminal->get_Session(&pIDispatch);  
pIDispatch->QueryInterface( IID_IHESession, (void**) &pIProfile);  
  
BSTR bstrProfileName = SysAllocString(OLESTR("C:\\\\aix.hep"));  
pIProfile->put_ProfileName(bstrProfileName);  
pIProfile->Load();  
SysFreeString(bstrProfileName);
```

## SilentConnect

### Property, IHETerminal 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HETerminal.SilentConnect  
HETerminal.SilentConnect = Boolean
```

#### C++ Syntax

```
HRESULT IHETerminal::get_SilentConnect([out, retval] VARIANT_BOOL * pVal);  
HRESULT IHETerminal::put_SilentConnect([in] VARIANT_BOOL * newVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer connects your machine to the host without opening a progress dialog box. A returned value of VARIANT\_FALSE indicates that a progress dialog box opens when HostExplorer connects your machine to the host.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer connects your machine to the host without opening a progress dialog box. A value of VARIANT\_FALSE indicates that a progress dialog box opens when HostExplorer connects your machine to the host.

**Basic Example**

```
Dim bVal As Boolean  
bVal = Terminal.SilentConnect  
  
if (bVal = True) Then  
    Terminal.SilentConnect = False  
End If
```

**C++ Example**

```
VARIANT_BOOL bVal;  
Terminal->get_SilentConnect(&bVal);  
if (bVal==VARIANT_TRUE)  
{  
    bVal=VARIANT_FALSE;  
    Terminal->put_SilentConnect(&bVal);  
}
```

**TCPPort****Property, IHETerminal 3270 5250 VT**

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

**Basic Syntax**

```
Integer = HETerminal.TCPPort  
HETerminal.TCPPort = Integer
```

**C++ Syntax**

```
HRESULT IHETerminal::get_TCPPort([out, retval] short * pVal);  
HRESULT IHETerminal::put_TCPPort([in] short newVal);
```

**Parameters**

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

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

**Basic Example**

```
If (Session.Port <>23) Then  
    Terminal.TCPPort = 23  
End If
```

```
Terminal.Connected = True
```

```
Terminal.TCPPort = 17
```

**C++ Example**

```
short iPort;  
pISession->get_Port(&iPort);  
if (iPort != 23)  
{  
    iPort=23;  
    pITerminal->put_TCPPort(iPort);  
}  
  
IPort = 17;  
pITerminal->put_TCPPort (iPort);
```

## Transport

### Property, IHETerminal [3270 5250 VT](#)

This property returns the address to the Transport object.

**Basic Syntax**

```
Dispatch = HETerminal.Transport
```

**C++ Syntax**

```
HRESULT IHETerminal.get_Transport([out, retval] IDispatch ** pVal);
```

**Parameters**

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

**Basic Example**

```
Dim Transport As HETransport  
Set Transport = Terminal.Transport  
Transport.Connected = False
```

**C++ Example**

```
IDispatch* pIDispatch;  
IHETransport* pITransport;  
pITerminal->get_Transport(&pITransport);  
pIDispatch->QueryInterface(IID_IHETransport, (void**) &pITransport);  
pITransport->put_Connected(FALSE);
```

## UserDir

### Property, IHETerminal 3270 5250 VT

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

#### Basic Syntax

```
String = HETerminal.UserDir  
HETerminal.UserDir = String
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim str As String  
str = Terminal.UserDir  
  
if (Len(str)=0) Then  
    Terminal.UserDir =  
        "C:\\Program Files\\Hummingbird"  
End If
```

#### C++ Example

```
BSTR bstr;  
Terminal.get_UserDir(&bstr);  
if (strlen( OLE2A(bstr)) == 0 )  
{  
    if (bstr!=NULL)  
        SysFreeString(bstr);  
    bstr =  
        SysAllocString(OLESTR(  
            "C:\\Program Files\\Hummingbird"))  
    Terminal.put_UserDir(bstr);  
    SysFreeString(bstr);  
}
```

## 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.

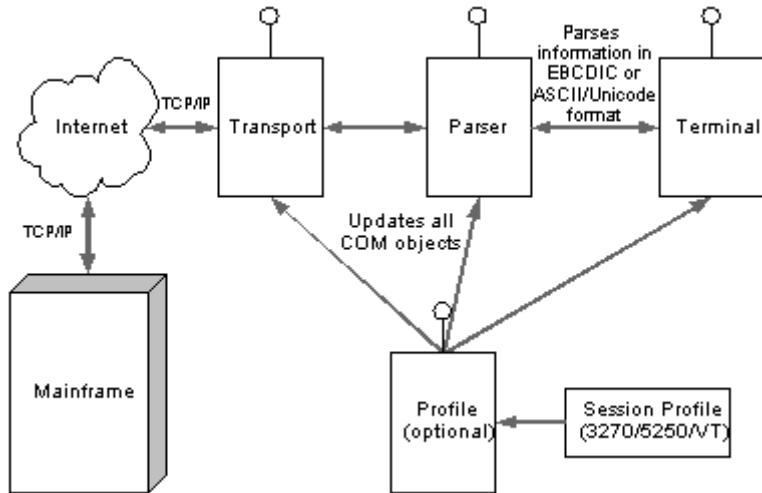
Sample files of COM objects are available in the directory where the program files are stored on your machine:

HostExplorer\SDK\Samples\COMObjects

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

## Relationship Between COM Objects

The following diagram illustrates the relationships between the Terminal, Profile, Parser, and Transport 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
- CharSet
- DetectChainedIO
- EnableAutoInsertToNextField
- EnableDisplayHostAddressOnOIA
- EnableOEMReply
- InsertKeyStyle
- LeftMargin
- Password
- Rows
- VTNRC

## Renamed or Moved COM Methods, Properties, and Interfaces

For version 8.0 and later, 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 HEPARSER_FEATURE 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	CellCopyMode
Feature	GetFeature and SetFeature methods
NextFieldKey	OnPasteFieldModeTabCharacter
NRC	NRCID
ReplaceFieldAttributeWith	OnCopyReplaceFieldAttributeWith
VTUPSS	UPSS
WrapLines	AutoWrap
XferMode	TransferMode
XferErrorCode	TransferErrorCode

## ***Renamed Interfaces of the Profile Object***

The following interfaces of the Profile object have been renamed:

<b>Former Name</b>	<b>New Name</b>
ProfileEditing	ProfileEdit
ProfilePrintExplorer	ProfilePrintSession
ProfileSaveFile	ProfileCapture
ProfileVTPrint	ProfileHostPrinting

## ***Renamed Properties of the Profile Object***

The following properties of the Profile object have been renamed:

<b>Former Name</b>	<b>New Name</b>	<b>Interface</b>
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

Former Name	New Name	Interface
PELUType	LUType	ProfilePrintSession
PEProfileName	ProfileName	ProfilePrintSession
PEStartPrinter	StartPrinter	ProfilePrintSession
PEStopPrinter	StopPrinter	ProfilePrintSession
ShowRecvDir	ShowRecvDlg	ProfileFileTransfer
7171PrintMode	PrintMode7171	ProfilePCPrint

### ***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

<b>Property</b>	<b>Former Interface</b>	<b>New Interface</b>
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
VTDoHostWindowSize	Profile	ProfileConnection
VTInitiateTelnetNegotiation	Profile	ProfileConnection
VTLineMode	Profile	ProfileConnection
Sound	Profile	ProfileSound
WindowTitle	Profile	ProfileSessionWindow

<b>Property</b>	<b>Former Interface</b>	<b>New Interface</b>
AutoRunMacro	Profile	ProfileConnection
DelayTime		
ColumnSeparators	ProfileTerminal	ProfileDisplay
VTDefColsPerScreen	ProfileDisplay	ProfileTerminal
VTDefLinesPerScreen	ProfileDisplay	ProfileTerminal
NoLockKeyb	ProfileDisplay	ProfileEdit
MultiLineDelete	ProfileDisplay	ProfileEdit
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

## **TN3270 Language-Conversion Table**

The following list consists of the available languages and the associated language index for TN3270 terminals. This list is a subset of the contents in the language-conversion table. Refer to `Hex3270.hxl`, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the `Hex3270.hxl` file is available.

<b>3270 Languages</b>	<b>Language Index</b>
Austrian ECECP (1141)	0x0200
Austrian CECP (273)	0x0201
Austrian (273)	0x0202
Belgian ECECP (1148)	0x0203
Belgian CECP (500)	0x0204
Belgian (500)	0x0205
Brazilian (275)	0x0206
Canadian Bilingual ECECP (1140)	0x0207
Canadian Bilingual CECP (037)	0x0208
Canadian Bilingual (037)	0x0209
Croatian	0x020A
Cyrillic (880)	0x0241
Cyrillic (1025)	0x020B
Czech (870)	0x020C
Danish ECECP (1142)	0x020D
Danish CECP (277)	0x020E
Danish (277)	0x020F
Dutch CECP (037)	0x0245
Dutch ECECP (1140)	0x024F
English-UK ECECP (1146)	0x0210

<b>3270 Languages</b>	<b>Language Index</b>
English-UK CECP (285)	0x0211
English-UK (285)	0x0212
English-US C370 CECP V2 (1047)	0x0213
English-US C370 CECP (037)	0x0214
English-US CECP (037)	0x0215
English-US (037)	0x0216
Estonian (1122)	0x024B
Finnish ECECP (1143)	0x0217
Finnish CECP (278)	0x0218
Finnish (278)	0x0219
French ECECP (1147)	0x021A
French CECP (297)	0x021B
French (297)	0x021C
German ECECP (1141)	0x0248
German CECP (273)	0x0249
German (273)	0x024A
Greek (423)	0x0240
Greek New (875)	0x021D
Hungarian (870)	0x021E
Icelandic ECECP (1149)	0x023E
Icelandic CECP (871)	0x023F
Italian ECECP (1144)	0x0220
Italian (280)	0x0221
Latin-2 (870)	0x0222
Latin-9 (924)	0x0250
Latvian (1112)	0x024C

<b>3270 Languages</b>	<b>Language Index</b>
Lithuanian (1112)	0x024D
Netherlands ECECP (1140)	0x0224
Netherlands CECP (037)	0x0225
Netherlands (037)	0x0226
Norwegian CECP (1142)	0x0227
Norwegian CECP (277)	0x0228
Norwegian (277)	0x0229
Polish (870)	0x022A
Portuguese ECECP (1140)	0x022B
Portuguese CECP (037)	0x022C
Portuguese (037)	0x022D
ROECE Latin/Yugoslav (870)	0x022E
Romanian (870)	0x022F
Russian (1025)	0x0247
Serbian (870)	0x0243
Slovenian (870)	0x0244
Spanish Speaking ECECP (1145)	0x0231
Spanish Speaking CECP (284)	0x0232
Spanish Speaking (284)	0x0233
Spanish CECP (284)	0x0234
Swedish ECECP (1143)	0x0235
Swedish CECP (278)	0x0236
Swedish (278)	0x0237
Swiss French ECECP (1148)	0x0238
Swiss French CECP (500)	0x0239
Swiss French (500)	0x023A

<b>3270 Languages</b>	<b>Language Index</b>
Swiss German ECECP (1148)	0x023B
Swiss German CECP (500)	0x023C
Swiss German (500)	0x023D
Turkish (905)	0x0242
Turkish New (1026)	0x0223
Ukrainian (1123)	0x024E

### ***Sample of Hex3270.hxl***

The following is a sample of `Hex3270.hxl`, the language-conversion table for TN3270 terminals:

[English-US C370 CECP V2 (1047) ]	The language string as it appears in the Translation Table Options dialog box under the Terminal interface.
HostCodepage=037	The host code page.
AnsiCodePage=1252	The ANSI code page.
acCECP0=0x00	The first CECP byte ID.
acCECP1=0x65	The second CECP byte ID.
acCECP2=0x00	The third CECP byte ID.
acCECP3=0x25	The fourth CECP byte ID.
INIFILE=037.HXL	The <code>.hx1</code> file containing the EBCDIC-to-ASCII and EBCDIC-to-Unicode conversion tables.
IndexCharset=0x0216	The language index for the conversion tables.

## **TN5250 Language-Conversion Table**

The following list consists of the available languages and the associated language index for TN5250 terminals. This list is a subset of the contents in the language-conversion table. Refer to `Hex5250.hxl`, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the `Hex5250.hxl` file is available.

<b>5250 Languages</b>	<b>Language Index</b>
Albania-MNCS	0x0142
Austrian/German-273	0x0100
Austrian/German-1141	0x0101
Austrian/German-MNCS	0x0102
Belgian-MNCS	0x0103
Belgian-1148	0x0103
Bosnian/Herzegovinian-870	0x014C
Brazilian Portuguese-037	0x0105
Brazilian Portuguese-1140	0x0141
Bulgarian-1025	0x010A
Canadian French-MNCS	0x0106
Canadian French-037	0x0107
Canadian French-1140	0x0108
Croatian-870	0x0109
Cyrillic-880	0x010B
Czech-870	0x010C
Danish-277	0x010D
Danish-1142	0x010E
Danish-MNCS	0x0143
Dutch-037	0x010F

<b>5250 Languages</b>	<b>Language Index</b>
Dutch-1140	0x0144
Dutch-MNCS	0x0110
UK English-285	0x0111
UK English-1146	0x0112
UK English-MNCS	0x0145
US English-037	0x0113
US English-1140	0x014D
US English-MNCS	0x0146
Estonian-1122	0x0115
Finnish-278	0x0116
Finnish-MNCS	0x0114
Finnish-1143	0x0117
French (Azerty)-297	0x0118
French (Azerty)-1147	0x0119
French (Azerty)-MNCS	0x011A
French (Qwerty)-297	0x0147
French (Qwerty)-1147	0x0148
French (Qwerty)-MNCS	0x0149
Greek New-875	0x011F
Hungarian-870	0x0120
Icelandic-1149	0x0121
Icelandic-871	0x0122
Icelandic-MNCS	0x011B
Italian-280	0x0123
Italian-1144	0x0124
Italian-MNCS	0x0125

<b>5250 Languages</b>	<b>Language Index</b>
Latin-2-870	0x0126
Latvian-1112	0x0127
Lithuanian-1112	0x0128
FYR Macedonia	0x011C
Norwegian-277	0x012B
Norwegian-1142	0x012C
Norwegian-MNCS	0x011D
Polish-870	0x012D
Portuguese-037	0x012E
Portuguese-MNCS	0x012F
Portuguese-1140	0x0130
Romanian-870	0x0131
Russian-1025	0x0133
Serbian Cyrillic-1025	0x0134
Serbian/Montenegro Latin-870	0x0129
Slovenian-870	0x0135
Slovakian-870	0x012A
Spanish-284	0x0136
Spanish-MNCS	0x012A
Spanish-1145	0x0132
Spanish Speaking-284	0x0137
Spanish Speaking-1145	0x0138
Spanish Speaking-MNCS	0x014A
Swedish-278	0x0139
Swedish-1143	0x013A
Swedish-MNCS	0x014B

5250 Languages	Language Index
Swiss French-MNCS	0x013B
Swiss French-1148	0x013C
Swiss German-MNCS	0x013D
Swiss German-1148	0x013E
Turkish-1026	0x013F
Ukrainian-1123	0x0140

### ***Sample of Hex5250.hxl***

The following is a sample of `Hex5250.hxl`, the language-conversion table for TN5250 terminals:

[US English-037]	The language string as it appears in the Translation Table Options dialog box under the Terminal interface.
HostCodepage=037	The host code page.
AnsiCodePage=1252	The ANSI code page.
KBDID5250=USB	The host keyboard ID.
CSID5250=00697	The host character set ID.
CPID5250=00037	The host code page ID.
INIFILE=037.HXL	The <code>.hxl</code> file containing the EBCDIC-to-ASCII and EBCDIC-to-Unicode conversion tables.
IndexCharset=0x0113	The language index for the conversion tables.

## TNVT UPSS Language-Conversion Table

The following list consists of the available UPSS (User Preferred Supplemental Character Set) and the associated language index for TNVT terminals. This list is a subset of the contents in the language-conversion table. Refer to `VT_UPS.hx1`, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the `VT_UPS.hx1` file is available.

VT UPSS Languages	Language Index
DEC Special	0x0030
DEC Supplemental	0x005B
DEC Technical	0x003E
Europa3 (333)	0x0007
DEC Greek (373)	0x0008
ISO Latin-1 (8859-1)	0x007B
ISO Latin-2 (8859-2)	0x0002
ISO Latin-4 (8859-4)	0x000B
ISO Cyrillic (8859-5)	0x000C
ISO Greek (8859-7)	0x000D
ISO Latin-5 (8859-9)	0x0006
ISO Latin 9 (8859-15)	0x0009
PC English (437)	0x0001
PC Icelandic (861)	0x000E
PC Modern Greek (869)	0x000F
PC Modern Turkish (857)	0x0010
PC Multilingual (850)	0x0011
PC Nordic (865)	0x0012
PC Portuguese (860)	0x0013

VT UPSS Languages	Language Index
PC Slavic (852)	0x0014
PC Canadian-French (863)	0x0015
PC Cyrillic (855)	0x0016
PC Cyrillic (866)	0x0017
PC Baltic (921)	0x0018
PC Estonian (922)	0x0019
PC Greek (851)	0x001A
HP Roman 8 (1051)	0x001B
PC Greek (437G, 210)	0x001C
HP Roman 9	0x001D
ISO Latin-3 (8859-3)	0x001E
PC Baltic (775)	0x0021
Windows Latin-2 (1250)	0x0022
Windows Cyrillic (1251)	0x0023
Windows Latin-1 (1252)	0x0024
Windows Greek (1253)	0x0025
Windows Latin-5 Turkish (1254)	0x0026
Windows Baltic (1257)	0x0027
ISO Latin-6 (8859-10)	0x0028
ISO Latin-7 (8859-13)	0x0029
ISO Latin-8 (8859-14)	0x002A
IBM3151 (Special Graphics)	0x002B
Slovenian 7-Bit	0x002C

### ***Sample of VT\_UPS.hx1***

The following is a sample of VT\_UPS.hx1, the UPSS language-conversion table for TNVT terminals:

[PC English (437)]	The language string as it appears in the VT Character Set dialog box under the Terminal interface.
DEFINE=VTCS_PCENGLISH437	An internal string.
INIFILE=437.HXL	The .hx1 file that contains the host-to-ASCII and host-to-Unicode conversion tables.
IndexCharset=0x0001	The language index for the conversion tables.

### ***TNVT NRC Language-Conversion Table***

The following list consists of the available NRC (National Replacement Character) support languages and the associated language index for TNVT terminals. This list is a subset of the contents in the language-conversion table. Refer to VT\_NRC.hx1, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the VT\_NRC.hx1 file is available.

<b>VT NRC Support Languages</b>	<b>Language Index</b>
None	0x0000
DEC Swedish	0x0037
DEC French Canadian	0x0039
DEC Swiss	0x003D
ISO United Kingdom	0x0041
DEC Finnish	0x0043
DEC Norwegian/Danish	0x0045
ISO German	0x004B
ISO French	0x0052

VT NRC Support Languages	Language Index
ISO Italian	0x0059
DEC Portuguese	0x005D
ISO Spanish	0x005A
ISO Norwegian/Danish	0x0060

### ***Sample of VT\_NRC.hxl***

The following is a sample of `VT_NRC.hxl`, the NRC language-conversion table for TNVT terminals:

[ISO German]	The language string as it appears in the VT Character Set dialog box under the Terminal interface.
IndexCharset=0x004B	The language index for the conversion tables.
Hex23=0x23	The new byte character that replaces the original value at the same character position.
Hex23U=0x0023	The new Unicode character that replaces the original value at the same character position.

### ***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
<code>HOSTEX_8BIT_MODE</code>	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.
<code>HOSTEX_ALLOW_AID_KEY_REPEAT</code>	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.

<b>Value</b>	<b>Definition</b>
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.
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.

Value	Definition
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.

<b>Value</b>	<b>Definition</b>
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.

Value	Definition
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.

## ***HEPARSER\_VALUE Data Type***

### **3270 5250 VT**

The `HEPARSER_VALUE` data type enumerates the configurable properties of the Parser objects. It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_BELL_MARGIN</code>	Specifies the bell margin. HostExplorer creates a beeping sound when your cursor reaches the bell margin—a specified column of an input field. By default, this property is set to 0.
<code>HOSTEX_LEFT_MARGIN</code>	Specifies the left margin of the screen. By default, this property is set to 0.
<code>HOSTEX_RIGHT_MARGIN</code>	Specifies the right margin of the screen. By default, this property is set to 0.
<code>HOSTEX_DISPLAY_IN_OIA</code>	Specifies whether HostExplorer displays the Host Response time ( <code>HOSTEX_OIA_DISPLAY_HOST_RESPONSE_TIME</code> ) or the host IP address ( <code>HOSTEX_OIA_DISPLAY_IP_ADDRESS</code> ) in the operation information area (OIA).

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_E_MODE</code>	Enables/disables 3270 E or 5250 E mode.
<code>HOSTEX_EAB</code>	Enables/disables the Extended Attribute feature (3270 only).
<code>HOSTEX_INITIATE_TELNET_NEGOTIATION</code>	Enables/disables Telnet negotiation (VT only).
<code>HOSTEX_ENABLEKERBEROSTICKETFORWARDING</code>	Enables/disables Kerberos ticket forwarding (3270 and VT only).

Value	Definition
HOSTEX_ENABLEKERBEROSAUTHENTICATION	Enables/disables Kerberos authentication (3270 and VT only).
HOSTEX_ENABLEKERBEROSECRIPTION	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_ENABLESSLTLSNEGOTIATEVIATETELNET	Enables/disables SSL negotiation over Telnet.
HOSTEX_ENABLESSLTLSSECURITY	Enables/disables the SSL security feature.

## ***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.

## ***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.

<b>Value</b>	<b>Definition</b>
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.

## ***HOSTEX\_CAPTURE\_MODE Data Type***

### **VT**

The HOSTEX\_CAPTURE\_MODE data type specifies how to capture selected text.

It has the following values:

<b>Value</b>	<b>Definition</b>
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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_CELL_DELIMITED_WORD</code>	Indicates that HostExplorer parses screen data at words.
<code>HOSTEX_CELL_DELIMITED_FIELD</code>	Indicates that HostExplorer parses screen data at field attributes.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_CONNECT_BY_TELNET</code>	Indicates that HostExplorer connects to the host using TCP/IP.
<code>HOSTEX_CONNECT_BY_MODEM</code>	Indicates that HostExplorer connects to the host using a modem. This data type applies only to TNVT terminals.
<code>HOSTEX_CONNECT_BY_MSSNA</code>	Indicates that HostExplorer connects to the host using a Microsoft SNA server gateway. This data type applies only to TN3270 terminals.

Value	Definition
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.

## ***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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_CUT_MODE_REPLACE_WITH_SPACES</code>	Indicates that the cut text is replaced with blank characters.
<code>HOSTEX_CUT_MODE_REPLACE_WITH_NULLS</code>	Indicates that the cut text is replaced with nulls (or zeros).
<code>HOSTEX_CUT_MODE_DELETE_TEXT</code>	Indicates that the cut text is deleted and not replaced by any characters.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_DEVICE_TYPE_DISPLAY</code>	Indicates that the Transport objects are used with a display terminal.
<code>HOSTEX_DEVICE_TYPE_PRINTER</code>	Indicates that the Transport objects are used with a printer device.

## ***HOSTEX\_ENCRYPTED Data Type***

### **3270 5250 VT**

The HOSTEX\_ENCRYPTED data type specifies the encryption level of the session.

It has the following values:

<b>Value</b>	<b>Definition</b>
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.

## ***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.

<b>Value</b>	<b>Definition</b>
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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_FIELD_ATTR_REPLACEMENT_NONE</code>	Indicates that HostExplorer does not replace the field attribute with anything.
<code>HOSTEX_FIELD_ATTR_REPLACEMENT_TAB</code>	Indicates that HostExplorer replaces the field attribute with a tab stop on the screen.
<code>HOSTEX_FIELD_ATTR_REPLACEMENT_COMMA</code>	Indicates that HostExplorer replaces the field attribute with a comma on the screen.
<code>HOSTEX_FIELD_ATTR_REPLACEMENT_PARAGRAPH</code>	Indicates that HostExplorer replaces the field attribute with a paragraph mark on the screen.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_FUNCTION_KEY_SYSTEM_REQUEST</code>	Executes a system-request command to the host.
<code>HOSTEX_FUNCTION_KEY_SEND_ATTENTION</code>	Executes an attention command to the host.
<code>HOSTEX_FUNCTION_KEY_SEND_ABORT_OUTPUT</code>	Executes an abort-output command to the host; this command stops the process.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC</code>	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.
<code>HOSTEX_GRAPHICS_CELLSIZE_NINE_BY_TWELVE</code>	Indicates that the cell size of the character is 9 x 12 pixels.

Value	Definition
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.

## ***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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_GRAPHICS_MODEL_NOGRAPHICS</code>	Indicates that HostExplorer displays only text.
<code>HOSTEX_GRAPHICS_MODEL_3179G</code>	Indicates that HostExplorer displays the IBM 3179G graphics terminal model.
<code>HOSTEX_GRAPHICS_MODEL_3472G</code>	Indicates that HostExplorer displays the IBM 3472G graphics terminal model.
<code>HOSTEX_GRAPHICS_MODEL_3270PCG</code>	Indicates that HostExplorer displays the IBM 3270G graphics terminal model.

## ***HOSTEX\_HOTSPOT\_DISPLAY Data Type***

### **3270 5250 VT**

The `HOSTEX_HOTSPOT_DISPLAY` data type specifies how HostExplorer displays hotspots.

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_HOTSPOT_DISPLAY_INVISIBLE</code>	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.
<code>HOSTEX_HOTSPOT_DISPLAY_RAISED_BUTTON</code>	Specifies that each hotspot text or region appears highlighted.

***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
<code>HOSTEX_HOTSPOT_MOUSE_ACTIVATION_SINGLE_CLICK</code>	Specifies that hotspots activate when you click them once with the left mouse button.
<code>HOSTEX_HOTSPOT_MOUSE_ACTIVATION_DOUBLE_CLICK</code>	Specifies that hotspots activate when you click them twice with the left mouse button.

***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
<code>HOSTEX_INSERT_KEY_STYLE_RESET</code>	Indicates that the Insert-key option is on until you press the Reset key.
<code>HOSTEX_INSERT_KEY_STYLE_ACTION</code>	Indicates that the Insert-key option is on until you press an action key, such as Enter or Clear.

## ***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:

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

## ***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

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_KEYBOARD_TYPE_PC_84</code>	Indicates that the PC keyboard has 84 keys.
<code>HOSTEX_KEYBOARD_TYPE_PC_101</code>	Indicates that the PC keyboard has 101 keys.
<code>HOSTEX_KEYBOARD_TYPE_PC_102</code>	Indicates that the PC keyboard has 102 keys.

Value	Definition
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.

## ***HOSTEX\_LINEMODE Data Type***

### **VT**

The HOSTEX\_LINEMODE data type specifies how HostExplorer stores characters in a buffer until you send a carriage return to the host. When enabled, Line mode forces HostExplorer to send characters one line at a time rather than as individual characters. Using line mode is useful when you are trying to reduce costs on networks that charge per packet or when you are experiencing long network delays. By default, this data type is set to HOSTEX\_LINEMODE\_DONTDOLINEMODE.

It has the following values:

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.

## ***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:

<b>Value</b>	<b>Definition</b>
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.

## ***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:

<b>Value</b>	<b>Definition</b>
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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_PASTE_MODE_PASTE_BLOCK</code>	Indicates that HostExplorer stops pasting text when it reaches a protected field on the screen.
<code>HOSTEX_PASTE_MODE_PASTE_OVERLAY</code>	Indicates that HostExplorer ignores pasted characters that overlay protected fields.
<code>HOSTEX_PASTE_MODE_PASTE_STREAM</code>	Indicates that HostExplorer pastes text one character at a time and stops when it reaches a protected field.
<code>HOSTEX_PASTE_MODE_PASTE_</code>	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.
<code>HOSTEX_PASTE_MODE_PASTE_FIELD</code>	Indicates that HostExplorer pastes text in a stream-like fashion, and moves to the next field when a <code>HOSTEX_NEXT_FIELD_KEY</code> character is encountered.

## ***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:

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

## ***HOSTEX\_PRINT\_TARGET Data Type***

### **VT**

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

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

## ***HOSTEX\_RESIZE\_BEHAVIOR Data Type***

### **VT**

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

Value	Definition
<code>HOSTEX_RESIZE_BEHAVIOR_CHANGE_FONT</code>	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.
<code>HOSTEX_RESIZE_BEHAVIOR_NEGOTIATE_WIN_SIZE</code>	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.
<code>HOSTEX_RESIZE_BEHAVIOR_DO NOTHING</code>	Specifies that HostExplorer does not do anything when you resize the window.

## ***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
<code>HOSTEX_SAVE_ALL</code>	Specifies that HostExplorer saves the values of all components in the Profile object.
<code>HOSTEX_SAVE_FONTS</code>	Specifies that HostExplorer saves any changes made to the session font.
<code>HOSTEX_SAVE_EVENT_SCHEME</code>	Specifies that HostExplorer saves the event scheme for the current session.

## ***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:

<b>Value</b>	<b>Definition</b>
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.

## ***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:

<b>Value</b>	<b>Definition</b>
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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_STATUS_LINE_MODE_NOSTATUSLINE</code>	Indicates that no status line is displayed.
<code>HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE</code>	Indicates that the status line appears at the bottom of the terminal screen.
<code>HOSTEX_STATUS_LINE_MODE_WINDOWSTATUSBAR</code>	Indicates that the status line appears at the bottom of the window.
<code>HOSTEX_STATUS_LINE_MODE_5250TERMINALSTATUSBAR</code>	Indicates that HostExplorer displays a 5250-terminal-style status bar.

## ***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_KEEP SIZE`.

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.

## ***HOSTEX\_TELNETECHO Data Type***

### **VT**

The HOSTEX\_TELNETECHO data type specifies how HostExplorer responds to remote echo negotiation with a Telnet host. By default, this data type is set to HOSTEX\_TELNETECHO\_AUTOMATIC.

It has the following values:

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.

## ***HOSTEX\_TERMINAL\_ID Data Type***

### **VT**

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

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_TERMINAL_ID_VT100</code>	Indicates that the terminal ID response is VT100.
<code>HOSTEX_TERMINAL_ID_VT101</code>	Indicates that the terminal ID response is VT101.
<code>HOSTEX_TERMINAL_ID_VT102</code>	Indicates that the terminal ID response is VT102.
<code>HOSTEX_TERMINAL_ID_VT220</code>	Indicates that the terminal ID response is VT220.
<code>HOSTEX_TERMINAL_ID_VT320</code>	Indicates that the terminal ID response is VT320.
<code>HOSTEX_TERMINAL_ID_VT420</code>	Indicates that the terminal ID response is VT420.
<code>HOSTEX_TERMINAL_ID_VT80</code>	Indicates that the terminal ID response is VT80.
<code>HOSTEX_TERMINAL_ID_VT100J</code>	Indicates that the terminal ID response is VT100J.
<code>HOSTEX_TERMINAL_ID_VT102J</code>	Indicates that the terminal ID response is VT102J.
<code>HOSTEX_TERMINAL_ID_VT220J</code>	Indicates that the terminal ID response is VT220J.
<code>HOSTEX_TERMINAL_ID_VT282</code>	Indicates that the terminal ID response is VT282.
<code>HOSTEX_TERMINAL_ID_VT382</code>	Indicates that the terminal ID response is VT382.

## ***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.

Value	Definition
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.

## ***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.

## ***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:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_TPRINT_OUTPUT_DEFAULT_WIN_PRINTER</code>	Indicates that the print job is being sent to the default printer specified on your machine.
<code>HOSTEX_TPRINT_OUTPUT_LPT1</code>	Indicates that the print job is being sent to the LPT1 port.
<code>HOSTEX_TPRINT_OUTPUT_LPT2</code>	Indicates that the print job is being sent to the LPT2 port.
<code>HOSTEX_TPRINT_OUTPUT_LPT3</code>	Indicates that the print job is being sent to the LPT3 port.
<code>HOSTEX_TPRINT_OUTPUT_CLIPBOARD</code>	Indicates that the print job is being sent to the Clipboard.

## ***HOSTEX\_TRANSFER Data Type***

### **3270**

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

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_TRANSFER_DOWNLOAD</code>	Indicates that HostExplorer downloads files.
<code>HOSTEX_TRANSFER_UPLOAD</code>	Indicates that HostExplorer uploads files.

***HOSTEX\_TRANSFER\_HOSTTYPE Data Type*****3270**

The `HOSTEX_TRANSFER_HOSTTYPE` data type specifies the operating system run by the host. By default, this data type is set to `HOSTEX_TRANSFER_HOSTTYPE_CMS`.

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_TRANSFER_HOSTTYPE_CMS</code>	Indicates that you are uploading files to a host that is running CMS.
<code>HOSTEX_TRANSFER_HOSTTYPE_TSO_MUSIC</code>	Indicates that you are uploading files to a host that is running TSO/MUSIC.
<code>HOSTEX_TRANSFER_HOSTTYPE_CICS</code>	Indicates that you are uploading files to a host that is running CICS.

## ***HOSTEX\_TRANSFER\_INITIALACTION Data Type***

### **3270**

The `HOSTEX_TRANSFER_INITIALACTION` data type specifies the action that HostExplorer performs before uploading or downloading files. By default, this data type is set to `HOSTEX_TRANSFER_INITIALACTION_NONE`.

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_TRANSFER_INITIALACTION_NONE</code>	Indicates that HostExplorer is not to perform an action before transferring a file.
<code>HOSTEX_TRANSFER_INITIALACTION_HOMEKEY</code>	Indicates that HostExplorer is to send a Home-key command to the host before transferring a file.
<code>HOSTEX_TRANSFER_INITIALACTION_ENTERKEY</code>	Indicates that HostExplorer is to send an Enter key or carriage return (CR) to the host before transferring a file.

## ***HOSTEX\_TRANSFER\_RECORDFORMAT Data Type***

### **3270**

The `HOSTEX_TRANSFER_RECORDFORMAT` data type specifies general record formats.

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>HOSTEX_TRANSFER_RECORDFORMAT_DEFAULT</code>	Indicates that the host portion of the file transfer must use the default record format.
<code>HOSTEX_TRANSFER_RECORDFORMAT_FIXED</code>	Indicates that HostExplorer must upload the file to a fixed-record-format file.

Value	Definition
HOSTEX_TRANSFER_RECORDFORMAT_VARIABLE	Indicates that HostExplorer must upload the file to a variable-record-format file.
HOSTEX_TRANSFER_RECORDFORMAT_UNDEFINED	Indicates that the record format is undefined; it applies only to Multiple Virtual Storage (MVS).

## ***HOSTEX\_TRANSFER\_TARGET Data Type***

### **3270 VT**

The `HOSTEX_TRANSFER_TARGET` data type specifies where the file transfer occurs.

It has the following values:

Value	Definition
HOSTEX_TRANSFER_PC_FILESYSTEM	Indicates that the file transfer occurs on the file system.
HOSTEX_TRANSFER_PC_CLIPBOARD	Indicates that the file transfer occurs on the Clipboard.

## ***HOSTEX\_TRANSFER\_TYPE Data Type***

### **3270 VT**

The `HOSTEX_TRANSFER_TYPE` data type specifies the name of the host file transfer program to use when uploading and/or downloading files.

It has the following values:

Value	Definition
HOSTEX_TRANSFER_TYPE_INDFILE	Indicates that the name of the file-transfer program for TN3270 terminals is IND\$File. You can use IND\$File to send and receive TN3270 files.

Value	Definition
HOSTEX_TRANSFER_TYPE_ZMODEM	Indicates that the name of the file-transfer program is Zmodem. This VT protocol provides end-to-end data security between program applications while significantly eliminating file-transfer errors.
HOSTEX_TRANSFER_TYPE_YMODEM	Indicates that the name of the file-transfer program is Ymodem. This VT protocol supports batch file transfers and can send the file name and file size before the actual data.
HOSTEX_TRANSFER_TYPE_XMODEM	Indicates that the name of the file-transfer program is Xmodem. This VT protocol is one of the original protocols written for asynchronous communications.
HOSTEX_TRANSFER_TYPE_XMODEM1K	Indicates that the name of the file-transfer program is Xmodem-1K. This VT protocol is part of Xmodem.
HOSTEX_TRANSFER_TYPE_KERMIT	Indicates that the name of the file-transfer program is Kermit. This VT protocol sends batch files with the name and time stamp of each file in small packet sizes. These packets contain fields that mark the beginning, length, type, and sequence number of the packet.

## ***HOSTOVERWRITE\_BEHAVIOUR Data Type***

### **VT**

The `HOSTOVERWRITE_BEHAVIOUR` data type specifies how files are written to the host.

It has the following values:

<b>Value</b>	<b>Definition</b>
<code>OVERWRITE_BEHAVIOUR_NEWERORLONGER</code>	Indicates that HostExplorer overwrites the existing file only if the file you are sending is newer or longer.
<code>OVERWRITE_BEHAVIOUR_APPEND</code>	Indicates that HostExplorer adds the file you are sending to the end of the existing file.
<code>OVERWRITE_BEHAVIOUR_ALWAYS</code>	Indicates that HostExplorer always overwrites the existing file.
<code>OVERWRITE_BEHAVIOUR_FILESIZEDATEDIFFER</code>	Indicates that HostExplorer overwrites the existing file only if the file you are sending is a different size and/or is newer.
<code>OVERWRITE_BEHAVIOUR_NEVER</code>	Indicates that HostExplorer never overwrites the existing file.
<code>OVERWRITE_BEHAVIOUR_NEWER</code>	Indicates that HostExplorer overwrites the existing file only if the file you are sending is newer.

## ***OLE\_COLOR Data Type***

### **3270 5250 VT**

The `OLE_COLOR` data type is used for properties that return colors. When a property is declared as `OLE_COLOR`, the Properties window displays a color palette, which you can use to select the color for the property rather than using the numeric equivalent.

This data type consists of a long value that is converted from a hexadecimal value. The hexadecimal value returns the corresponding values for BGR (Blue, Green, Red). The first two hexadecimal numbers refer to Blue; the middle two numbers refer to Green, and the last two numbers refer to Red. For example:

- Blue—hexadecimal FF0000; long 16711680
- Green—hexadecimal FF00; long 65280
- Red—hexadecimal FF; long 255

**Note:** A hexadecimal value must be converted to a long value so that you can use it in a property.

## ***PC\_OVERWRITE\_BEHAVIOUR Data Type***

### **3270 VT**

The PC\_OVERWRITE\_BEHAVIOUR data type specifies how files are written to the PC.

It has the following values:

<b>Value</b>	<b>Definition</b>
OVERWRITE_BEHAVIOUR_OVERWRITE	Indicates that HostExplorer overwrites the existing file with the file you are sending.
OVERWRITE_BEHAVIOUR_RENAME	Indicates that HostExplorer renames the file you are sending and leaves the existing file unchanged.
OVERWRITE_BEHAVIOUR_SKIP	Indicates that HostExplorer transfers all of the files except for those that already exist.

## 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
- Profile Security 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

## Properties

The Profile interface consists of the following properties:

- AddOIAToCapture
- AllowEmuTracing
- AllowErrorRestart
- AttnFormat
- Capture
- ClearPassword
- Color
- Connection
- 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

## Load

**Method, IHEProfile 3270 5250 VT**

This method loads the specified profiles.

**Basic Syntax**

`HEProfile.Load`

**C++ Syntax**

`HRESULT IHEProfile::Load();`

**Parameters**

This method has no parameters.

**Basic Example**

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\\aix.hep"
Profile.Load
```

**C++ Example**

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
BSTR bstr;
bstr = SysAllocString(OLESTR("C:\\aix.hep"));
pSess->put_ProfileName(bstr);
pSess->Load();
SysFreeString(bstr);
```

## LoadColorScheme

Method, IHEProfile 3270 5250 VT

This method loads the specified color scheme for the session. A scheme is a collection of settings.

### Basic Syntax

```
HEProfile.LoadColorScheme(bstrScheme As String)
```

### C++ Syntax

```
HRESULT IHEProfile::LoadColorScheme([in] BSTR bstrScheme);
```

### Parameters

*bstrScheme*—The name of the color scheme that you want to load for the current session.

### Basic Example

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\\aix.hep"
Profile.Load
Profile.LoadColorScheme("ATM-Saint_Louis_Blues")
```

### C++ Example

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
BSTR bstr;
bstr = SysAllocString(OLESTR("C:\\aix.hep"));
pSess->put_ProfileName(bstr);
SysFreeString(bstr);
pSess->Load();
bstr = SysAllocString(OLESTR("ATM-Saint_Louis_Blues"));
pSess->LoadColorScheme(bstr);
SysFreeString(bstr);
```

## LoadFileTransferScheme

Method, IHEProfile 3270 5250 VT

This method loads the specified file transfer scheme for the current session.

### Basic Syntax

```
HEProfile.LoadFileTransferScheme(bstrScheme As String)
```

### C++ Syntax

```
HRESULT IHEProfile::LoadFileTransferScheme([in] BSTR bstrScheme);
```

### Parameters

*bstrScheme*—The name of the file transfer scheme that you want to load for the current session.

**Basic Example**

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\\aix.hep"
Profile.Load
Profile.LoadFileTransferScheme("CMS_Text")
```

**C++ Example**

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
BSTR bstr;
bstr = SysAllocString(OLESTR("C:\\aix.hep"));
pSess->put_ProfileName(bstr);
SysFreeString(bstr);
pSess->Load();
bstr = SysAllocString(OLESTR("CMS_Text"));
pSess->LoadFileTransferScheme(bstr);
SysFreeString(bstr);
```

**LoadQuickKeyFile**

**Method, IHEProfile 3270 5250 VT**

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

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

**Basic Syntax**

```
HEProfile.LoadQuickKeyFile(FileName As String) As Integer
```

**C++ Syntax**

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

**Parameters**

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

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

**Basic Example**

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\\aix.hep"
Profile.Load
Profile.LoadQuickKeyFile "C:\\Login.QKV"
```

**C++ Example**

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
BSTR bstr;
bstr = SysAllocString(OLESTR("C:\\aix.hep"));
pSess->put_ProfileName(bstr);
SysFreeString(bstr);
pSess->Load();
bstr = SysAllocString(OLESTR("C:\\Login.QKV"));
pSess->LoadQuickKeyFile(bstr);
SysFreeString(bstr);
```

## Save

**Method, IHEProfile 3270 5250 VT**

This method saves and updates the profile file with the specified values in the associated Profile object. You can save one of the following types of values:

- All—Saves all values in the Profile object.
- Fonts—Saves all changes to the session font.
- Event Scheme—Saves the event scheme for the current session.

**Basic Syntax**

```
HEProfile.Save(newVal As HOSTEX_SAVE_OPTIONS)
```

**C++ Syntax**

```
HRESULT IHEProfile::Save([in] HOSTEX_SAVE_OPTIONS newVal);
```

**Parameters**

*newVal*—The type of values you want to save.

**Basic Example**

```
Dim Profile As HEProfile
Set Profile = Terminal.Session
Profile.ProfileFileName = "C:\\aix.hep"
Profile.Load
Terminal.Connected = True
Profile.HostName = "sunset"
Profile.Save (HOSTEX_SAVE_ALL)
```

**C++ Example**

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
BSTR bstr ;
bstr = SysAllocString(OLESTR("C:\\aix.hep"));
pSess->put_ProfileName(bstr);
SysFreeString(bstr);
pSess->Load();
VARIANT_BOOL vbVal = VARIANT_TRUE;
pSess->put_Connected(vbVal);
// Add code ...
pSess->Save(HOSTEX_SAVE_ALL);
```

## AddOIAToCapture

**Property, IHEProfile 3270 5250 VT**

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

**Basic Syntax**

```
Boolean = HEProfile.AddOIAToCapture
HEProfile.AddOIAToCapture = Boolean
```

**C++ Syntax**

```
HRESULT IHEProfile::get_AddOIAToCapture([out, retval] VARIANT_BOOL
*pVal);
HRESULT IHEProfile::put_AddOIAToCapture([in] VARIANT_BOOL newVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer captures the OIA. A returned value of VARIANT\_FALSE indicates that HostExplorer does not capture the OIA.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer captures the OIA. A value of VARIANT\_FALSE indicates that HostExplorer does not capture the OIA.

**Basic Example**

```
Dim Profile As HEProfile
Dim bVal As Boolean
    ' Set the appropriate type
Terminal.TerminalType=HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
    ' Get value
bVal = Profile.AddOIAToCapture
If (bVal = False) Then
    ' Set value
    Profile.AddOIAToCapture = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal*pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
//Get AddOIAToCapture property
VARIANT_BOOL bVal;
bVal = pSess->get_AddOIAToCapture(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pSess->put_AddOIAToCapture(bVal);
}
...
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AllowErrorRestart

Property, IHEProfile 3270 5250 VT

This property returns or sets the Allow Error Restart system setting. When this setting is enabled, HostExplorer automatically attempts to reconnect a terminated session, regardless of the reason. The default value of this property is VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfile.AllowErrorRestart
```

```
HEProfile.AllowErrorRestart = Boolean
```

### C++ Syntax

```
HRESULT IHEProfile::get_AllowErrorRestart([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfile::put_AllowErrorRestart([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer attempts to reconnect a terminated session. If *pVal* equals VARIANT\_FALSE, HostExplorer does not attempt to reconnect.

*newVal*—The value you want to set. If *newVal* equals VARIANT\_TRUE, HostExplorer attempts to reconnect a terminated session. If *newVal* equals VARIANT\_FALSE, HostExplorer does not try to reconnect.

### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
' Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.AllowErrorRestart  
If (bVal = False) Then  
    ' Set value  
    Profile.AllowErrorRestart = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get AllowErrorRestart property
    VARIANT_BOOL bVal;
    pSess->get_AllowErrorRestart(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_AllowErrorRestart(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AttnFormat

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
HOSTEX_ATN_FORMAT = HEProfile.AttnFormat  
HEProfile.AttnFormat = HOSTEX_ATN_FORMAT
```

#### C++ Syntax

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

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim AttnFormat As HOSTEX_ATN_FORMAT  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
AttnFormat = Profile.AttnFormat  
If (AttnFormat=HOSTEX_ATN_FORMAT_IBM) Then  
    Profile.AttnFormat = HOSTEX_ATN_FORMAT_ATTACHMATE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get AttnFormat property
    HOSTEX_ATN_FORMAT AttnFormat;
    pSess->get_AttnFormat(&AttnFormat);
    If (AttnFormat == HOSTEX_ATN_FORMAT_IBM)
    {
        AttnFormat = HOSTEX_ATN_FORMAT_ATTACHMATE;
        pSess->put_AttnFormat(AttnFormat);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClearPassword

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEProfile.ClearPassword
```

```
HEProfile.ClearPassword = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEProfile::put_ClearPassword([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer saves the password in non-encrypted format. A returned value of VARIANT\_FALSE indicates that HostExplorer saves the password in encrypted format.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer saves the password in non-encrypted format. A value of VARIANT\_FALSE indicates that HostExplorer saves the password in encrypted format.

#### Basic Example

```
Dim Profile As HEProfile
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
' Get value
bVal = Profile.ClearPassword
If (bVal = False) Then
    ' Set value
    Profile.ClearPassword = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_<3270>);

    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Add OIATOCapture property
    VARIANT_BOOL bVal;
    bVal = pSess->get_ClearPassword(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_ClearPassword(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Color

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileColor interface and its associated properties in HostExplorer.

**Basic Syntax**

```
HEProfileConnection = HEProfile.Color
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Color([out, retval] IHEDProfileColor **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEDProfileColor interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim ptr As HEProfileColor
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set ptr = Profile.Color
ptr.Schemes = "ATM-Philadelphia"
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPOVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
//Get ClearPassword property
BSTR bstr;
bstr = SysAllocString(OLESTR("ATM-Philadelphia"));
pColor->put_Schemes(bstr);
SysFreeString(bstr);
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Connection

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileConnection interface and its associated properties in HostExplorer.

#### Basic Syntax

```
HEProfileConnection = HEProfile.Connection
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Connection([out, retval] IHEProfileConnection **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileConnection interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim Conn As HEProfileConnection
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Conn = Profile.Connection
Conn.HostName = "aix"
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Connection property
    BSTR bstr;
    bstr = SysAllocString(OLESTR("aix"));
    pConn->put_HostName(bstr);
    SysFreeString(bstr);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

**Cursor****Property, IHEProfile 3270 5250 VT**

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

**Basic Syntax**

```
HEProfileCursor = HEProfile.Cursor
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Cursor([out, retval] IHEProfileCursor **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileCursor interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim ProfileCursor As HEProfileCursor
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set ProfileCursor = Profile.Cursor
ProfileCursor.MoveCursorOnMouseClicked = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Connection property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pCursor->put_MoveCursorOnMouseClicked(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DDEServerName

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
String = HEProfile.DDEServerName
```

```
HEProfile.DDEServerName = String
```

#### C++ Syntax

```
HRESULT IHEProfile::get_DDEServerName([out, retval] BSTR *pVal);  
HRESULT IHEProfile::put_DDEServerName([in] BSTR newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
strVal = Profile.DDEServerName  
If (Len(strVal) = 0) Then  
    ' Set Value  
    Profile.DDEServerName = "HOSTEX"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get DDEServerName property
    BSTR bstr;
    pSess->get_DDEServerName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("HOSTEX"));
        pSess->put_DDEServerName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Display

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
HEProfileDisplay = HEProfile.Display
```

### C++ Syntax

```
HRESULT IHEProfile::get_Display([out, retval] IHEProfileDisplay **pVal);
```

### Parameters

*pVal*—The returned pointer to the IHEProfileDisplay interface.

### Basic Example

```
Dim Profile As HEProfile
Dim ProfileDisplay As HEProfileDisplay
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set ProfileDisplay = Profile.Display
ProfileDisplay.AlwaysAutoskip = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_<3270>);

    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Display property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pSessDisplay->put_AlwaysAutoskip(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

**Edit****Property, IHEProfile 3270 5250 VT**

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

**Basic Syntax**

```
HEProfileEdit = HEProfile.Edit
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Edit([out, retval] IHEProfileEdit **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileEdit interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim ProfileEdit As HEProfileEdit
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set ProfileEdit = Profile.Edit
ProfileEdit.AutoCopy = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Edit property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pSessEdit->put_AutoCopy(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EmuTraceFilename

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
String = HEProfile.EmuTraceFilename  
HEProfile.EmuTraceFilename = String
```

#### C++ Syntax

```
HRESULT IHEProfile::get_EmuTraceFilename([out, retval] BSTR *pVal);  
HRESULT IHEProfile::put_EmuTraceFilename([in] BSTR newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
strVal = Profile.EmuTraceFilename  
If (Len(strVal) = 0) Then  
    ' Set value  
    Profile.EmuTraceFilename = "HETrace.trc"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get EmuTraceFilename property
    BSTR bstr ;
    pSess->get_EmuTraceFilename(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("HETrace.trc"));
        pSess->put_EmuTraceFilename(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AllowEmuTracing

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
Boolean = HEProfile.AllowEmuTracing
```

```
HEProfile.AllowEmuTracing = Boolean
```

### C++ Syntax

```
HRESULT IHEProfile::get_AllowEmuTracing([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfile::put_AllowEmuTracing([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer is running the emulator-tracing function. A returned value of VARIANT\_FALSE indicates that HostExplorer is not running the emulator-tracing function.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer is running the emulator-tracing function. A value of VARIANT\_FALSE indicates that HostExplorer is not running the emulator-tracing function.

### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.AllowEmuTracing  
If (bVal = False) Then  
    ' Set value  
    Profile.AllowEmuTracing = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get AllowEmuTracing property
    VARIANT_BOOL bVal;
    bVal = pSess->get_AllowEmuTracing(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_AllowEmuTracing(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnableHLLAPITracing

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
Boolean = HEProfile.EnableHLLAPITracing  
HEProfile.EnableHLLAPITracing = Boolean
```

### C++ Syntax

```
HRESULT IHEProfile::get_EnableHLLAPITracing([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfile::put_EnableHLLAPITracing([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE enables tracing. A returned value of VARIANT\_FALSE disables tracing.

*newVal*—A value of VARIANT\_TRUE enables tracing. A value of VARIANT\_FALSE disables tracing.

### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.EnableHLLAPITracing  
If (bVal = False) Then  
' Set value  
Profile.EnableHLLAPITracing = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Enable HLLAPITracing property
    VARIANT_BOOL bVal;
    bVal = pSess->get_EnableHLLAPITracing(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_EnableHLLAPITracing(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnableTracing

Property, IHEProfile 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer enables tracing and creates a trace file.

### Basic Syntax

```
Boolean = HEProfile.EnableTracing
```

```
HEProfile.EnableTracing = Boolean
```

### C++ Syntax

```
HRESULT IHEProfile::get_EnableTracing([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfile::put_EnableTracing([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE enables tracing. A returned value of VARIANT\_FALSE disables tracing.

*newVal*—A value of VARIANT\_TRUE enables tracing. A value of VARIANT\_FALSE disables tracing.

### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.EnableTracing  
If (bVal = False) Then  
    ' Set value  
    Profile.EnableTracing = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get EnableTracing property
    VARIANT_BOOL bVal;
    bVal = pSess->get_EnableTracing(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_EnableTracing(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Events

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileEvents interface and its associated properties in HostExplorer.

**Basic Syntax**

```
HEProfileEvents = HEProfile.Events
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Events([out, retval] IHEProfileEvents **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileEvents interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim Event As HEProfileEvents
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Event = Profile.Events
Event.EnableEvents = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
//Get Events property
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
IHEProfileEvents *pEvent;
pSess->get_Events(&pEvent);
pEvent->put_EnableEvents(VARIANT_TRUE);
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## FileTransfer

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
HEProfileFileTransfer = HEProfile.FileTransfer
```

#### C++ Syntax

```
HRESULT IHEProfile::get_FileTransfer([out, retval] IHEProfileFileTransfer
**pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileFileTransfer interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessFileXfer As HEProfileFileTransfer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessFileXfer = Profile.FileTransfer
ProfileFileXfer.Append = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get FileTransfer property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pxfer->put_Append(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Fonts

### Property, IHEProfile 3270 5250 VT

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

**Basic Syntax**

```
HEProfileFonts = HEProfile.Fonts
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Fonts([out, retval] IHEProfileFonts **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileFonts interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessFonts As HEProfileFonts
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessFonts = Profile.Fonts
SessFonts.SaveFontOnExit = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Fonts property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pFonts->put_SaveFontOnExit(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## GlobalSettingsPath

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
String = HEProfile.GlobalSettingsPath  
HEProfile.GlobalSettingsPath = String
```

### C++ Syntax

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

### Parameters

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

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

### Basic Example

```
Dim Profile As HEProfile  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get path  
strVal = Profile.GlobalSettingsPath  
If (Len(strVal) = 0) Then  
    ' Set path  
    Profile.GlobalSettingsPath = "C:\\HostEx.ini"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get GlobalSettingsPath property
    BSTR bstr ;
    pSess->get_GlobalSettingsPath(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("C:\\\\HostEx.ini"));
        pSess->put_GlobalSettingsPath(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Graphics

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
HEProfileGraphics = HEProfile.Graphics
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Graphics([out, retval] IHEProfileGraphics
**pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileGraphics interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessGraphics As HEProfileGraphics
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessGraphics = Profile.Graphics
SessGraphics.APL = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Graphics property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pGraphics->put_APL(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Hotspots

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileHotspots interface and its associated properties in HostExplorer.

**Basic Syntax**

```
HEProfileHotspots = HEProfile.Hotspots
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Hotspots([out, retval] IHEProfileHotspots **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileHotspots interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim HotSpot As HEProfileHotspots
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set HotSpot = Profile.Hotspots
HotSpot.EnableHotspots = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Hotspots property
    HEProfileHotspots HotSpot;
    pSess->get_Hotspots(&pHotSpot);
    pHotSpot->put_EnableHotspots(VARIANT_TRUE);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Keyboard

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
ProfileKeyboard = HEProfile.Keyboard
```

### C++ Syntax

```
HRESULT IHEProfile::get_Keyboard([out, retval] IHEProfileKeyboard**  
    *pVal);
```

### Parameters

*pVal*—The returned pointer to the IHEProfileKeyboard interface.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessKeyboard As HEProfileKeyboard  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessKeyboard = Profile.Keyboard  
SessKeyboard.AllowDiac = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Keyboard property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pKeyboard->put_AllowDiac(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KillMacrosOnSessionExit

Property, IHEProfile 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer kills any running macros when it terminates the current session.

### Basic Syntax

```
Boolean = HEProfile.KillMacrosOnSessionExit  
HEProfile.KillMacrosOnSessionExit = Boolean
```

### C++ Syntax

```
HRESULT IHEProfile::get_KillMacrosOnSessionExit([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfile::put_KillMacrosOnSessionExit([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer kills any macros that are still running when the session ends. If *pVal* equals VARIANT\_FALSE, HostExplorer does not kill any active macros when the session ends.

*newVal*—The value you want to set. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer kills all active macros before it ends the session. Set *newVal* to VARIANT\_FALSE to leave running macros active after the end of the session.

### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.KillMacrosOnSessionExit  
If (bVal = False) Then  
' Set value  
Profile.KillMacrosOnSessionExit = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get KillMacrosOnSessionExit property
    VARIANT_BOOL bVal;
    pSess->get_KillMacrosOnSessionExit(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_KillMacrosOnSessionExit(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Mouse

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
HEProfileMouse = HEProfile.Mouse
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Mouse([out, retval] IHEProfileMouse **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileMouse interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessMouse As HEProfileMouse
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessMouse = Profile.Mouse
SessMouse.BlockSelect = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Mouse property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pMouse->put_BlockSelect(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

**PCPrint****Property, IHEProfile 3270 5250 VT**

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

**Basic Syntax**

```
HEProfilePCPrint = HEProfile.PCPrint
```

**C++ Syntax**

```
HRESULT IHEProfile::get_PCPrint([out, retval] IHEProfilePCPrint **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfilePCPrint interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessPCPrint As HEProfilePCPrint
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessPCPrint = Profile.PCPrint
SessPCPrint.TprintMode = HOSTEX_TPRINT_OUTPUT_LPT1
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get PCPrint property
    HEProfilePCPrint PCPrint;
    pSess->get_PCPrint(&pPCPrint);
    pPCPrint->put_TprintMode(HOSTEX_TPRINT_OUTPUT_LPT1);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintSession

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfilePrintSession interface to retrieve options related to launching a Print session.

#### Basic Syntax

```
HEProfilePrintSession = HEProfile.PrintSession
```

#### C++ Syntax

```
HRESULT IHEProfile::get_PrintSession([out, retval] IHEProfilePrintSession **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfilePrintSession interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessPrint As HEProfilePrintSession
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessPrint = Profile.PrintSession
SessPrint.HostName = "sunset.cs.ca"
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get PrintSession property
    BSTR bstr = SysAllocString(OLESTR("sunset.cs.ca"));
    pPrintSess->put_HostName(bstr);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintScreen

### Property, IHEProfile 3270 5250 VT

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

**Basic Syntax**

```
HEProfilePrintScreen = HEProfile.PrintScreen
```

**C++ Syntax**

```
HRESULT IHEProfile::get_PrintScreen([out, retval] IHEProfilePrintScreen **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfilePrintScreen interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessPrintScrn As HEProfilePrintScreen
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessPrintScrn = Profile.PrintScreen
SessPrintScrn.PrintOIA = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get PrintScreen property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pPrintScrn->put_PrintOIA(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

**ProfileName**

**Property, IHEProfile 3270 5250 VT**

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

**Basic Syntax**

```
String = HEProfile.ProfileName
HEProfile.ProfileName = String
```

**C++ Syntax**

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

**Parameters**

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

**Basic Example**

```
Dim Profile As HEProfile
Dim strVal As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
' Get string
strVal = Profile.ProfileName
If (Len(strVal) = 0) Then
    ' Set string
    Profile.ProfileName = "C:\\aix.hep"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEPProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEPProfile, (void**) &pIProfile );
    //Get ProfileName property
    BSTR bstr ;
    pSess->get_ProfileName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("C:\\\\aix.hep"));
        pSess->put_ProfileName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## QueryShutdown

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
Boolean = HEProfile.QueryShutdown
```

```
HEProfile.QueryShutdown = Boolean
```

### C++ Syntax

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

```
HRESULT IHEProfile::put_QueryShutdown([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer forces Windows to postpone shutting down until HostExplorer fully shuts down. A returned value of VARIANT\_FALSE indicates that HostExplorer shuts down automatically when Windows shuts down.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer forces Windows to postpone shutting down until HostExplorer fully shuts down. A value of VARIANT\_FALSE indicates that HostExplorer shuts down automatically when Windows shuts down.

### Basic Example

```
Dim Profile As HEProfile
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
' Get value
bVal = Profile.QueryShutdown
If (bVal = False) Then
    ' Set value
    Profile.QueryShutdown = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get QueryShutdown property
    VARIANT_BOOL bVal;
    bVal = pSess->get_QueryShutdown(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_QueryShutdown(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Capture

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileCapture interface to retrieve the Save File options.

#### Basic Syntax

```
HEProfileCapture = HEProfile.Capture
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Capture([out, retval] IHEProfileCapture **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileCapture interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessSave As HEProfileCapture
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessSave = Profile.Capture
SessSave.SaveConfirm = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Capture property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pCapture->put_SaveConfirm(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## RecordPortableMacros Property, IHEProfile 3270 5250 VT

This property returns a value that indicates the current Record Portable Macros system setting. This setting enables or disables the Macro Recorder.

**Basic Syntax**

```
Boolean = HEProfile.RecordPortableMacros
```

**C++ Syntax**

```
HRESULT IHEProfile::get_RecordPortableMacros([out, retval] VARIANT_BOOL *pVal);
```

**Parameters**

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, the setting is ON and the Macro Recorder is enabled. If *pVal* equals VARIANT\_FALSE, the setting is OFF and the Macro Recorder is disabled.

**Basic Example**

```
Dim Profile As HEProfile
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
' Get value
bVal = Profile.RecordPortableMacros
If (bVal = False) Then
    ' Add code
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
//Get RecordPortableMacros property
VARIANT_BOOL bVal;
pSess->get_RecordPortableMacros(&bVal);
if (bVal == VARIANT_FALSE)
{
    // Add code
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ReRunAutoMacro

### Property, IHEProfile 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer re-executes the Auto Macro when you re-create the session using the Auto Reconnect feature.

#### Basic Syntax

```
Boolean = HEProfile.ReRunAutoMacro
```

```
HEProfile.ReRunAutoMacro = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfile::get_ReRunAutoMacro([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfile::put_ReRunAutoMacro([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer re-executes the Auto Macro when you re-create the session. If *pVal* equals VARIANT\_FALSE, HostExplorer does not re-execute the Auto Macro.

*newVal*—The value you want to set. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer re-executes the Auto Macro when you re-create the session. Set *newVal* to VARIANT\_FALSE to ensure that HostExplorer does not re-execute the Auto Macro.

#### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.ReRunAutoMacro  
If (bVal = False) Then  
    ' Set value  
    Profile.ReRunAutoMacro = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get ReRunAutoMacro property
    VARIANT_BOOL bVal;
    pSess->get_ReRunAutoMacro(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_ReRunAutoMacro(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Security

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
HEProfileSecurity = HEProfile.Security
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Security([out, retval] IHEProfileSecurity
**pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileSecurity interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessSec As HEProfileSecurity
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessSec = Profile.Security
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Security property
    HEProfileSecurity SessSec;
    pSess->get_Security(&pSessSec);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SessionWindow

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileSessionWindow interface and its associated properties in HostExplorer.

**Basic Syntax**

```
HEProfileSessionWindow = HEProfile.SessionWindow
```

**C++ Syntax**

```
HRESULT IHEProfile::get_SessionWindow([out, retval]
IHEProfileSessionWindow **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileSessionWindow interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessWin As HEProfileSessionWindow
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessWin = Profile.SessionWindow
SessWin.WindowTitle = "Host Explorer"
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPOVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
//Get SessionWindow property
BSTR bstr;
bstr = SysAllocString(OLESTR("Host Explorer"));
pWin->put_WindowTitle(bstr);
SysFreeString(bstr);
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Sound

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileSound interface and its associated properties in HostExplorer.

#### Basic Syntax

```
HEProfileSound = HEProfile.Sound
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Sound([out, retval] IHEProfileSound **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileSound interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim Sound As HEProfileSound
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Sound = Profile.Sound
Sound.Notify = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Sound property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pSound->put_Notify(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Terminal

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileTerminal interface and its associated properties in HostExplorer.

**Basic Syntax**

```
HEProfileTerminal = HEProfile.Terminal
```

**C++ Syntax**

```
HRESULT IHEProfile::get_Terminal([out, retval] IHEProfileTerminal
**pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileTerminal interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessTerm As HEProfileTerminal
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessTerm = Profile.Terminal
SessTerm.ReplyOEM = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Terminal property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pTerm->put_ReplyOEM(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TerminalType

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
Long = HEProfile.TerminalType  
HEProfile.TerminalType = Long
```

#### C++ Syntax

```
HRESULT IHEProfile::get_TerminalType([out, retval] long *pVal);  
HRESULT IHEProfile::put_TerminalType([in] long newVal);
```

#### Parameters

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

- 1—TN3270 terminals
- 2—TNVT terminals
- 3—TN5250 terminals

*newVal*—The set value, which indicates the terminal type for the session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim lVal As Long  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
lVal = Profile.TerminalType  
If (lVal <> 2) Then  
    ' Set value  
    Profile.TerminalType = 2  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get TerminalType property
    long lVal;
    pSess->get_TerminalType(&lVal);
    if (lVal != 2)
    {
        lVal = 2;
        pSess->put_TerminalType(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Toolbar

### Property, IHEProfile 3270 5250 VT

This property returns a pointer to the ProfileToolbar interface and its associated properties in HostExplorer.

#### Basic Syntax

```
HEProfileToolbar = HEProfile.Toolbar
```

#### C++ Syntax

```
HRESULT IHEProfile::get_Toolbar([out, retval] IHEProfileToolbar **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileToolbar interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim Toolbar As HEProfileToolbar
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Toolbar = Profile.Toolbar
Toolbar.ShowTips = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get Toolbar property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pTool->put_ShowTips(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## HLLAPITraceFilename

Property, IHEProfile 3270 5250 VT

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

### Basic Syntax

```
String = HEProfile.HLLAPITraceFilename  
HEProfile.HLLAPITraceFilename = String
```

### C++ Syntax

```
HRESULT IHEProfile::get_HLLAPITraceFilename([out, retval] BSTR *pVal);  
HRESULT IHEProfile::put_HLLAPITraceFilename([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the file name.

*newVal*—The set string, specifying the file name.

### Basic Example

```
Dim Profile As HEProfile  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get string  
strVal = Profile.HLLAPITraceFilename  
If (Len(strVal) = 0) Then  
    ' Set string  
    Profile.HLLAPITraceFilename = "C:\\\\trace.trc"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get HLLAPITraceFilename property
    BSTR bstr ;
    pSess->get_HLLAPITraceFilename(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("C:\\\\trace.trc"));
        pSess->put_HLLAPITraceFilename(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TrackMenu

### Property, IHEProfile 3270 5250 VT

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

#### Basic Syntax

```
HEProfileTrackMenu = HEProfile.TrackMenu
```

#### C++ Syntax

```
HRESULT IHEProfile::get_TrackMenu([out, retval] IHEProfileTrackMenu **pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileTrackMenu interface.

#### Basic Example

```
Dim Profile As HEProfile
Dim SessTrack As HEProfileTrackMenu
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessTrack = Profile.TrackMenu
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get TrackMenu property
    HEProfileTrackMenu SessTrack;
    pSess->get_TrackMenu(&pTrackMenu);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

**TranslationTable****IHEProfile 3270 5250 VT**

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

**Basic Syntax**

```
HEProfileTranslationTable = HEProfile.TranslationTable
```

**C++ Syntax**

```
HRESULT IHEProfile::get_TranslationTable([out, retval]
    IHEProfileTranslationTable **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileTranslationTable interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessTrans As HEProfileTranslationTable
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessTrans = Profile.TranslationTable
SessTrans.UsePrivateTranslate = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get TranslationTable property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pTrans->put_UsePrivateTranslate(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UserDirectory

Property, IHEProfile 3270 5250 VT

This property returns or sets a string specifying a user directory where objects such as Quick-Keys, key-map file, schemes, macros, and profiles are stored.

### Basic Syntax

```
String = HEProfile.UserDirectory
```

```
HEProfile.UserDirectory = String
```

### C++ Syntax

```
HRESULT IHEProfile::get_UserDirectory([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfile::put_UserDirectory([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the user directory.

*newVal*—The set string, specifying the user directory.

### Basic Example

```
Dim Profile As HEProfile
Dim strVal As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
' Get string
strVal = Profile.UserDirectory
If (Len(strVal) = 0) Then
    ' Set string
    Profile.UserDirectory = "C:\\Winnt\\Hummingbird\\10.00"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get UserDirectory property
    BSTR bstr ;
    pSess->get_UserDirectory(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("C:\\Winnt\\Hummingbird\\10.00"));
        pSess->put_UserDirectory(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTCharset

### Property, IHEProfile VT

This property returns a pointer to the ProfileVTCharset interface to retrieve general options related to the character set.

#### Basic Syntax

```
HEProfileVTCharset = HEProfile.VTCharset
```

#### C++ Syntax

```
HRESULT IHEProfile::get_VTCharset([out, retval] IHEProfileVTCharset  
**pVal);
```

#### Parameters

*pVal*—The returned pointer to the IHEProfileVTCharset interface.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessVTChSet As HEProfileVTCharset  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set SessVTChSet = Profile.VTCharset  
SessVTChSet.VTNRCMode = True
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get VTCharset property
    VARIANT_BOOL bVal = VARIANT_TRUE;
    pVTCharset->put_VTCharset(bVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## HostPrinting Property, IHEProfile VT

This property returns a pointer to the ProfileHostPrinting interface to retrieve the VT Printing options from the Output Group.

**Basic Syntax**

```
HEProfileHostPrinting = HEProfile.HostPrinting
```

**C++ Syntax**

```
HRESULT IHEProfile::get_HostPrinting([out, retval] IHEProfileHostPrinting **pVal);
```

**Parameters**

*pVal*—The returned pointer to the IHEProfileHostPrinting interface.

**Basic Example**

```
Dim Profile As HEProfile
Dim HPrint As HEProfileHostPrinting
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set HPrint = Profile.HostPrinting
HPrint.VTPrinterTimeout = 100
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get HostPrinting property
    short sVal = 100;
    pHPrint->put_VTPrinterTimeout(&sVal);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WinDDEEnabled

Property, IHEProfile 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables or disables DDE (Dynamic Data Exchange). DDE allows programs to communicate with the emulator. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfile.WinDDEEnabled
```

```
HEProfile.WinDDEEnabled = Boolean
```

### C++ Syntax

```
HRESULT IHEProfile::get_WinDDEEnabled([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfile::put_WinDDEEnabled([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables DDE. A returned value of VARIANT\_FALSE indicates that HostExplorer disables DDE.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables DDE. A value of VARIANT\_FALSE indicates that HostExplorer disables DDE.

### Basic Example

```
Dim Profile As HEProfile  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
' Get value  
bVal = Profile.WinDDEEnabled  
If (bVal = False) Then  
    ' Set value  
    Profile.WinDDEEnabled = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    //Get WinDDEEnabled property
    VARIANT_BOOL bVal;
    bVal = pSess->get_WinDDEEnabled &bVal);
    if (bVal = VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSess->put_WinDDEEnabled(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileTerminal Interface

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

## Properties

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

### AlternateScreen

#### Property, IHEProfileTerminal 3270

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

#### Basic Syntax

```
Boolean = HEProfileTerminal.AlternateScreen  
HEProfileTerminal.AlternateScreen = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_AlternateScreen([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileTerminal::put_AlternateScreen([in] VARIANT_BOOL  
newVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that the screen always opens in the alternate (larger) screen mode. A returned value of VARIANT\_FALSE indicates that the default screen opens.

*newVal*—A value of VARIANT\_TRUE indicates that the screen always opens in the alternate (larger) screen mode. A value of VARIANT\_FALSE indicates that the default screen opens.

**Basic Example**

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
bVal = SessTerm.AlternateScreen
If (bVal = False) Then
    ' Set value
    SessTerm.AlternateScreen = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get AlternateScreen property
    VARIANT_BOOL bVal;
    bVal = pTerm->get_AlternateScreen(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_AlternateScreen(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CharacterSet

### Property, IHEProfileTerminal 5250

This property returns or sets a value that specifies character-set information for the host.

#### Basic Syntax

```
Integer = HEProfileTerminal.CharacterSet
```

```
HEProfileTerminal.CharacterSet = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_CharacterSet([out, retval] short *pVal);  
HRESULT IHEProfileTerminal::put_CharacterSet([in] short newVal);
```

#### Parameters

*pVal*—The returned string, which indicates the character-set information.

*newVal*—The set string, which indicates the character-set information.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_5250  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
iVal = SessTerm.CharacterSet  
If (iVal = 0) Then  
    ' Set value  
    SessTerm.CharacterSet = 3  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get CharacterSet property
    IHEProfileTerminal *pTerm;
    pSess->get_Terminal(&pTerm);
    short sVal;
    pTerm->get_CharacterSet(&sVal);
    if (sVal == 0)
    {
        sVal = 3;
        pTerm->put_CharacterSet(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DetectChainedIO

### Property, IHEProfileTerminal 3270

This property returns or sets a value indicating whether HostExplorer enables the automatic detection of chained Write/Read commands. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.DetectChainedIO
```

```
HEProfileTerminal.DetectChainedIO = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEProfileTerminal::put_DetectChainedIO([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer enables the automatic detection of chained Write/Read commands. If *pVal* equals VARIANT\_FALSE, HostExplorer does not enable automatic detection.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to enable automatic detection of chained Write/Read commands. Set *newVal* to VARIANT\_FALSE to disable automatic detection.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.DetectChainedIO  
If (bVal = False) Then  
    ' Set value  
    SessTerm.DetectChainedIO = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get DetectChainedIO property
    VARIANT_BOOL bVal;
    pTerm->get_DetectChainedIO(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_DetectChainedIO(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ForceAltSize

### Property, IHEProfileTerminal [3270](#)

This property returns or sets a value indicating whether HostExplorer changes the window to the alternate size when the host receives an Erase Write command. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.ForceAltSize
```

```
HEProfileTerminal.ForceAltSize = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_ForceAltSize([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_ForceAltSize([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer changes the window to the alternate size. A returned value of VARIANT\_FALSE indicates that HostExplorer does not change the window to the alternate size.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer changes the window to the alternate size. A value of VARIANT\_FALSE indicates that HostExplorer does not change the window to the alternate size.

#### Basic Example

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
bVal = SessTerm.ForceAltSize
If (bVal = False) Then
    ' Set value
    SessTerm.ForceAltSize = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( & pTerminal );
    //Get ForceAltSize property
    VARIANT_BOOL bVal;
    pTerm->get_ForceAltSize(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_ForceAltSize(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## New3270EAB

### Property, IHEProfileTerminal [3270](#)

This property returns or sets a value indicating whether HostExplorer enables Extended Attributes when you select 3279 as the 3270 type.

Extended Attributes are the mainframe application codes used to display various colors, highlighting, reverse images, and blinking; they let your computer fully emulate the mainframe screen. The change in this value takes effect only after you disconnect and reconnect the session. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.New3270EAB
```

```
HEProfileTerminal.New3270EAB = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_New3270EAB([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_New3270EAB([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables Extended Attributes. A returned value of VARIANT\_FALSE indicates that HostExplorer disables Extended Attributes.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables Extended Attributes. A value of VARIANT\_FALSE indicates that HostExplorer disables Extended Attributes.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.New3270EAB  
If (bVal = False) Then  
    ' Set value  
    SessTerm.New3270EAB = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get New3270EAB property
    VARIANT_BOOL bVal;
    pTerm->get_New3270EAB(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_New3270EAB(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## NewModel3279

### Property, IHEProfileTerminal 3270

This property returns or sets a value indicating whether HostExplorer supports Extended Attributes mode. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
HEProfileTerminal.NewModel3279 = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_NewModel3279([out, retval] VARIANT_BOOL  
*pVal);  
  
HRESULT IHEProfileTerminal::put_NewModel3279([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE (3279) indicates that HostExplorer supports Extended Attributes mode. A returned value of VARIANT\_FALSE (3278) indicates that HostExplorer does not support Extended Attributes mode.

*newVal*—A value of VARIANT\_TRUE (3279) indicates that HostExplorer supports Extended Attributes mode. A value of VARIANT\_FALSE (3278) indicates that HostExplorer does not support Extended Attributes mode.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.NewModel3279  
If (bVal = False) Then  
    ' Set value  
    SessTerm.NewModel3279 = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get NewModel3279 property
    VARIANT_BOOL bVal;
    pTerm->get_NewModel3279(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_NewModel3279(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## NewModelType

Property, IHEProfileTerminal **3270** **5250** **VT**

This property returns or sets a value (Short integer) specifying the model type for the session. After the terminal model is changed, you must disconnect from and reconnect to the host. Otherwise, HostExplorer does not update the change to the current session. Changing this value takes effect only after you disconnect and reconnect the session. VT220 is the most commonly supported terminal on most UNIX systems. IBM 3151 is used to connect to AIX systems.

**Note:** Use the VT320 or VT420 terminal type only if you have proper termcap entries on the host. Use SCO ANSI when connecting to SCO UNIX systems.

### Basic Syntax

```
Integer = HEProfileTerminal.NewModelType  
HEProfileTerminal.NewModelType = Integer
```

### C++ Syntax

```
HRESULT IHEProfileTerminal::get_NewModelType([out, retval] short *pVal);  
HRESULT IHEProfileTerminal::put_NewModelType([in] short newVal);
```

**Parameters**

*pVal*—For TN3270 terminals, the following returned values specify the model type for the session.

- 2—Model 2 (24x80)
- 3—Model 3 (32x80)
- 4—Model 4 (43x80)
- 5—Model 5 (27x132)

For TN5250 terminals, the following returned values specify the model type for the session:

- 2—Model 2 (24x80)
- 5—Model 5 (27x132)

For TNVT terminals, the following returned values specify the model type for the session:

- HOSTEX\_TERM\_MODEL\_VT52
- HOSTEX\_TERM\_MODEL\_VT100
- HOSTEX\_TERM\_MODEL\_VT101
- HOSTEX\_TERM\_MODEL\_VT102
- HOSTEX\_TERM\_MODEL\_VT220
- HOSTEX\_TERM\_MODEL\_VT320
- HOSTEX\_TERM\_MODEL\_VT420
- HOSTEX\_TERM\_MODEL\_ANSI
- HOSTEX\_TERM\_MODEL\_SCOANSI
- HOSTEX\_TERM\_MODEL\_TERMBM3151

*newVal*—The set value, specifying the model type for the session.

**Basic Example**

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
iVal = SessTerm.NewModelType
If (iVal = 0) Then
    ' Set value
    SessTerm.NewModelType = 6
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get NewModelType property
    short sVal;
    pTerm->get_NewModelType(&sVal);
    if (sVal == 0)
    {
        sVal = 6;
        pTerm->put_NewModelType(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ReplyOEM

### Property, IHEProfileTerminal 3270

This property returns or sets a value indicating whether HostExplorer sends an OEM reply field back to the host in response to receiving a Read Partition Query. If sent, the OEM reply contains information about the terminal session and features available for the host to use. Clear this option if you experience difficulty starting GDDM, SAS, or other mainframe applications. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.ReplyOEM  
HEProfileTerminal.ReplyOEM = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_ReplyOEM([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileTerminal::put_ReplyOEM([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer sends an OEM reply field back to the host. A returned value of VARIANT\_FALSE indicates that HostExplorer sends an OEM reply field back to the host.  
*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer sends an OEM reply field back to the host. A value of VARIANT\_FALSE indicates that HostExplorer does not send an OEM reply field back to the host.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.ReplyOEM  
If (bVal = False) Then  
    ' Set value  
    SessTerm.ReplyOEM = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get ReplyOEM property
    VARIANT_BOOL bVal;
    pTerm->get_ReplyOEM(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_ReplyOEM(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ShortName

Property, IHEProfileTerminal **3270 5250 VT**

This property returns or sets an identifier string that HLLAPI applications use to access a particular session.

### Basic Syntax

```
String = HEProfileTerminal.ShortName  
HEProfileTerminal.ShortName = String
```

### C++ Syntax

```
HRESULT IHEProfileTerminal::get_ShortName ([out, retval] BSTR *pVal);  
HRESULT IHEProfileTerminal::put_ShortName ([in] BSTR newVal);
```

### Parameters

*pVal*—The return value that specifies the identifier string.  
*newVal*—The string value you want to set as the identifier for HLLAPI applications.

### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim HLLAPI_ID As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
HLLAPI_ID = SessTerm.ShortName  
' Add code . . .  
' Set value  
HLLAPI_ID = "B"  
SessTerm.ShortName = HLLAPI_ID
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get ShortName property
    BSTR bstr;
    // Get the ID
    pTerm->get_ShortName(&bstr);
    // Add code . . .
    // Set the ID
    if (bstr != NULL)
        SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("B"));
        pTerm->put_ShortName(bstr);
        SysFreeString(bstr);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CustomModel

### Property, IHEProfileTerminal 3270

This property returns or sets a value indicating whether HostExplorer is running a special model of the mainframe terminal to emulate the session.

#### Basic Syntax

```
Boolean = HEProfileTerminal.CustomModel  
HEProfileTerminal.CustomModel = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_CustomModel ([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileTerminal::put_CustomModel ([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer is running a special model of the mainframe. A returned value of VARIANT\_FALSE indicates that HostExplorer is not running a special model of the mainframe.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer is running a special model of the mainframe. A value of VARIANT\_FALSE indicates that HostExplorer is not running a special model of the mainframe.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.CustomModel  
If (bVal = False) Then  
    ' Set value  
    SessTerm.CustomModel = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get CustomModel property
    VARIANT_BOOL bVal;
    pTerm->get_CustomModel(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pTerm->put_CustomModel(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CustomModelCols

### Property, IHEProfileTerminal 3270 5250 VT

This property returns or sets a value (Short integer) specifying the special model number of columns for the session.

#### Basic Syntax

```
Integer = HEProfileTerminal.CustomModelCols
```

```
HEProfileTerminal.CustomModelCols = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_CustomModelCols([out, retval] short  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_CustomModelCols([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the special model number of columns for the session.

*newVal*—The set value, specifying the special model number of columns for the session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
iVal = SessTerm.CustomModelCols  
If (iVal = 0) Then  
    ' Set value  
    SessTerm.CustomModelCols = 5  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get CustomModelCols property
    short sVal;
    pTerm->get_CustomModelCols(&sVal);
    if (sVal == 0)
    {
        sVal = 5;
        pTerm->put_CustomModelCols(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CustomModelRows

### Property, IHEProfileTerminal 3270 5250 VT

This property returns or sets a value specifying the special model number of rows for the session.

#### Basic Syntax

```
Integer = HEProfileTerminal.CustomModelRows
```

```
HEProfileTerminal.CustomModelRows = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_CustomModelRows([out, retval] short  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_CustomModelRows([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the special model number of rows for the session.

*newVal*—The set value, specifying the special model number of rows for the session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
iVal = SessTerm.CustomModelRows  
If (iVal = 0) Then  
    ' Set value  
    SessTerm.CustomModelRows = 5  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get CustomModelRows property
    short sVal;
    pTerm->get_CustomModelRows(&sVal);
    if (sVal == 0)
    {
        sVal = 5;
        pTerm->put_CustomModelRows(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VT8BitMode

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer uses the 8-bit transmission mode to connect to the host. This 8-bit mode supports 7-bit and 8-bit data formats.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VT8BitMode
```

```
HEProfileTerminal.VT8BitMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VT8BitMode([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_VT8BitMode([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer uses the 8-bit transmission mode. A returned value of VARIANT\_FALSE indicates that HostExplorer does not use the 8-bit transmission mode.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer uses the 8-bit transmission mode. A value of VARIANT\_FALSE indicates that HostExplorer does not use the 8-bit transmission mode.

#### Basic Example

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
bVal = SessTerm.VT8BitMode
If (bVal = False) Then
    ' Set value
    SessTerm.VT8BitMode = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VT8BitMode property
VARIANT_BOOL bVal;
pTerm->get_VT8BitMode(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VT8BitMode(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTAnswerback

### Property, IHEProfileTerminal VT

This property returns or sets a string specifying an Answerback message. You can enter special-character sequences in this field.

#### Basic Syntax

```
String = HEProfileTerminal.VTAnswerback  
HEProfileTerminal.VTAnswerback = String
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTAnswerback([out, retval] BSTR *pVal);  
HRESULT IHEProfileTerminal::put_VTAnswerback([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the Answerback message.

*newVal*—The set string, specifying the Answerback message.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get string  
strVal = SessTerm.VTAnswerback  
If (Len(strVal) = 0) Then  
    ' Set string  
    SessTerm.VTAnswerback = "ls -l"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_<VT>);

    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileTerminal interface
    IHEProfileTerminal* pTerminal;
    pIProfile->get_Terminal ( &pTerminal );
    //Get VTAnswerback property
    BSTR bstr;
    pTerm->get_VTAnswerback(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("ls -l"));
        pTerm->put_VTAnswerback(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTAutoResize

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer forces the screen size to conform to the page size.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTAutoResize  
HEProfileTerminal.VTAutoResize = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTAutoResize([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileTerminal::put_VTAutoResize([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer forces the screen size to conform to the page size. If *pVal* equals VARIANT\_FALSE, the screen size does not automatically conform to the page size.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that the screen size conforms to the page size. Set *newVal* to VARIANT\_FALSE if you do not want the screen size set by the page size.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTAutoResize  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTAutoResize = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEPProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEPProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTAutoResize property
VARIANT_BOOL bVal;
pTerm->get_VTAutoResize(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTAutoResize(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTBSIsDel

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating the type of code that the Backspace key sends.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTBSIsDel
```

```
HEProfileTerminal.VTBSIsDel = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTBSIsDel([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_VTBSIsDel([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Backspace key sends a Del (127) code. A returned value of VARIANT\_FALSE indicates that the Backspace key sends a BS (8) code.

*newVal*—A value of VARIANT\_TRUE indicates that the Backspace key sends a Del (127) code. A value of VARIANT\_FALSE indicates that the Backspace key sends a BS (8) code.

#### Basic Example

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
bVal = SessTerm.VTBSIsDel
If (bVal = False) Then
    ' Set value
    SessTerm.VTBSIsDel = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTBSIsDel property
VARIANT_BOOL bVal;
pTerm->get_VTBSIsDel(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTBSIsDel(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTConcealAnswerback

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer clears the Answerback message field. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTConcealAnswerback  
HEProfileTerminal.VTConcealAnswerback = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTConcealAnswerback([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileTerminal::put_VTConcealAnswerback([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer clears the Answerback message field. A returned value of VARIANT\_FALSE indicates that HostExplorer does not clear the Answerback message field.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer clears the Answerback message field. A value of VARIANT\_FALSE indicates that HostExplorer does not clear the Answerback message field.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTConcealAnswerback  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTConcealAnswerback = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTConcealAnswer property
VARIANT_BOOL bVal;
pTerm->get_VTConcealAnswerback(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTConcealAnswerback(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTDefColsPerScreen

### Property, IHEProfileTerminal VT

This property returns or sets a value specifying the default screen width that HostExplorer uses when it launches a new session. The available screen widths are:

- 80 columns
- 132 columns
- Custom—between 80 and 200 columns

#### Basic Syntax

```
Integer = HEProfileTerminal.VTDefColsPerScreen  
HEProfileTerminal.VTDefColsPerScreen = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTDefColsPerScreen([out, retval] short  
*pVal);  
HRESULT IHEProfileTerminal::put_VTDefColsPerScreen([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the default screen width that HostExplorer uses when it launches a new session.

*newVal*—The set value, specifying the default screen width that HostExplorer uses when it launches a new session.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Terminal As HEProfileTerminal  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
iVal = SessTerm.VTDefColsPerScreen  
If (iVal = 80) Then  
    SessTerm.VTDefColsPerScreen = 120  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTDefColsPerScreen property
short sVal;
pTerm->get_VTDefColsPerScreen(&sVal);
if (sVal == 80)
{
    sVal = 120;
    pTerm->put_VTDefColsPerScreen(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTDefLinesPerScreen

### Property, IHEProfileTerminal VT

This property returns or sets a value specifying the default screen height that HostExplorer uses when it launches a new session. By default, this property is set to 24.

#### Basic Syntax

```
Integer = HEProfileTerminal.VTDefLinesPerScreen  
HEProfileTerminal.VTDefLinesPerScreen = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTDefLinesPerScreen([out, retval] short  
*pVal);  
HRESULT IHEProfileTerminal::put_VTDefLinesPerScreen([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the default screen height.

*newVal*—The set value, specifying the default screen height.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Terminal As HEProfileTerminal  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
iVal = SessTerm.VTDefLinesPerScreen  
If (iVal = 24) Then  
    SessTerm.VTDefLinesPerScreen = 40  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTDefLinesPerScreen property
IHEProfileTerminal *pTerm;
pSess->get_Terminal(&pTerm);
short sVal;
pTerm->get_VTDefLinesPerScreen(&sVal);
if (sVal == 24)
{
    sVal = 40;
    pTerm->put_VTDefLinesPerScreen(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTDisplayMode

### Property, IHEProfileTerminal VT

This property returns or sets a value specifying the display mode. By default, this property is set to Optimized mode.

#### Basic Syntax

```
Integer = HEProfileTerminal.VTDisplayMode
```

```
HEProfileTerminal.VTDisplayMode = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTDisplayMode([out, retval] short *pVal);
```

```
HRESULT IHEProfileTerminal::put_VTDisplayMode([in] short newVal);
```

#### Parameters

*pVal*—A returned value of Optimized mode indicates that HostExplorer performs “bulk” updates to the screen. Typically, the emulator performs bulk updates at the end of a data stream. A returned value of Realistic mode indicates that HostExplorer updates the screen as it receives new characters. Although this is a much slower option, it allows for smoother scrolling.

*newVal*—A value of Optimized mode indicates that HostExplorer performs “bulk” updates to the screen. Typically, the emulator performs bulk updates at the end of a data stream. A value of Realistic mode indicates that HostExplorer updates the screen as it receives new characters.

#### Basic Example

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
iVal = SessTerm.VTDisplayMode
If (iVal = 0) Then
    ' Set value
    SessTerm.VTDisplayMode = 1
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( & pTerminal );
//Get VTDisplayMode property
short sVal;
pTerm->get_VTDisplayMode(&sVal);
if (sVal == 0)
{
    sVal = 1;
    pTerm->put_VTDisplayMode (sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTEnableSSH

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer uses SSH (Secure Shell) encryption between the server and client.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTEnableSSH  
HEProfileTerminal.VTEnableSSH = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTEnableSSH([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileTerminal::put_VTEnableSSH([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer uses SSH encryption. If *pVal* equals VARIANT\_FALSE, HostExplorer does not use SSH encryption.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to enable SSH encryption. Set *newVal* to VARIANT\_FALSE to disable SSH.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTEnableSSH  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTEnableSSH = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTEnableSSH property
VARIANT_BOOL bVal;
pTerm->get_VTEnableSSH(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTEnableSSH(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTForce8Bit

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer supports 8-bit data transfers even when NRC support is enabled. Usually the high-order bit of incoming data is stripped when NRC is enabled. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTForce8Bit
```

```
HEProfileTerminal.VTForce8Bit = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTForce8Bit([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_VTForce8Bit([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer supports 8-bit data transfers. A returned value of VARIANT\_FALSE indicates that HostExplorer does not support 8-bit data transfers.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer supports 8-bit data transfers. A value of VARIANT\_FALSE indicates that HostExplorer does not support 8-bit data transfers.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTForce8Bit  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTForce8Bit = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTForce8Bit property
VARIANT_BOOL bVal;
pTerm->get_VTForce8Bit(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTForce8Bit(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTLocalEcho

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer enables local echo of characters that you type in the emulator. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTLocalEcho
```

```
HEProfileTerminal.VTLocalEcho = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTLocalEcho([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_VTLocalEcho([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer echoes all the data that you type. A returned value of VARIANT\_FALSE indicates that HostExplorer does not echo any data that you type.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer echoes all the data that you type. A value of VARIANT\_FALSE indicates that HostExplorer does not echo any data that you type.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTLocalEcho  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTLocalEcho = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTLocalEcho property
VARIANT_BOOL bVal;
pTerm->get_VTLocalEcho(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTLocalEcho(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTNewTerminalType

### Property, IHEProfileTerminal VT

This property returns or sets a value specifying the VT terminal type to use for the current session. After changing the terminal type, you must disconnect and reconnect to the host. Otherwise, HostExplorer does not update the current session. VT220 is the most commonly supported terminal on most UNIX systems.

**Note:** Use VT320 or VT420 only if you have proper termcap entries on the host. Use SCO ANSI when connecting to SCO UNIX systems.

#### Basic Syntax

```
Integer = HEProfileTerminal.VTNewTerminalType  
HEProfileTerminal.VTNewTerminalType = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTNewTerminalType([out, retval] short  
*pVal);  
HRESULT IHEProfileTerminal::put_VTNewTerminalType([in] short newVal);
```

**Parameters**

*pVal*—The following returned values specify the VT terminal type to use for the current session:

- 2—MODEL 2
- 3—MODEL 3
- 4—MODEL 4
- 5—MODEL 5
- 6—VT52
- 7—VT100
- 8—VT101
- 9—VT102
- 10—VT220
- 11—VT320
- 12—VT420
- 13—ANSI
- 14—SCOANSI
- 15—TERM\_IBM3151
- 16—WYSE50
- 17—WYSE60

*newVal*—The set value, specifying the VT terminal type to use for the current session.

**Basic Example**

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
iVal = SessTerm.VTNTerminalType
If (iVal = 5) Then
    ' Set value
    SessTerm.VTNTerminalType = 9
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTNewTerminalType property
short sVal;
pTerm->get_VTNewTerminalType(&sVal);
if (sVal == 0)
{
    sVal = 9;
    pTerm->put_VTNewTerminalType(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTOnLine

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer sends data to the host. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTOnLine
```

```
HEProfileTerminal.VTOnLine = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTOnLine([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_VTOnLine([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the terminal is online and sends data to the host. A returned value of VARIANT\_FALSE indicates that the terminal is offline, so you can type and move the cursor around the screen without sending data to the host.

*newVal*—A value of VARIANT\_TRUE indicates that the terminal is online and sends data to the host. A value of VARIANT\_FALSE indicates that the terminal is offline, so you can type and move the cursor around the screen without sending data to the host.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTOnLine  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTOnLine = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTOnLine property
VARIANT_BOOL bVal;
pTerm->get_VTOnLine(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTOnLine(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTScrollSpeed

### Property, IHEProfileTerminal VT

This property returns or sets a value specifying the Smooth Scrolling speed. The value can range from 1 to 30. If the value is greater than the number of vertical pixels per cell, scrolling occurs in Line mode.

#### Basic Syntax

```
Integer = HEProfileTerminal.VTScrollSpeed
```

```
HEProfileTerminal.VTScrollSpeed = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTScrollSpeed([out, retval] short *pVal);
```

```
HRESULT IHEProfileTerminal::put_VTScrollSpeed([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the Smooth Scrolling speed. The value can range from 2 to the value of the font height.

*newVal*—The set value, specifying the Smooth Scrolling speed. The value can range from 2 to the value of the font height.

#### Basic Example

```
Dim Profile As HEPProfile
Dim Terminal As HEProfileTerminal
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
iVal = SessTerm.VTScrollSpeed
If (iVal = 0) Then
    ' Set value
    SessTerm.VTScrollSpeed = 4
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( & pTerminal );
//Get VTScrollSpeed property
short sVal;
pTerm->get_VTScrollSpeed(&sVal);
if (sVal == 0)
{
    sVal = 4;
    pTerm->put_VTScrollSpeed(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTSmoothScroll

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer uses the Smooth Scrolling feature. Before setting this property, you have to set the VT Display mode to Realistic.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTSmoothScroll
```

```
HEProfileTerminal.VTSmoothScroll = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTSmoothScroll([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileTerminal::put_VTSmoothScroll([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer scrolls data using a smooth scroll method. A returned value of VARIANT\_FALSE indicates that HostExplorer does not scroll data line by line.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer scrolls data using a smooth scroll method. A value of VARIANT\_FALSE indicates that HostExplorer does not scroll data line by line.

#### Basic Example

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
bVal = SessTerm.VTSmoothScroll
If (bVal = False) Then
    ' Set value
    SessTerm.VTSmoothScroll = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTSmoothScroll property
VARIANT_BOOL bVal;
pTerm->get_VTSmoothScroll(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTSmoothScroll(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTTerminalID

### Property, IHEProfileTerminal VT

This property returns or sets a value specifying the terminal ID or Device Attribute (DA) response that HostExplorer sends to the host. The DA contains the control sequences that define the terminal and its configuration and identifies the particular type of terminal to the host.

#### Basic Syntax

```
Integer = HEProfileTerminal.VTTerminalID
```

```
HEProfileTerminal.VTTerminalID = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTTerminalID([out, retval] short *pVal);  
HRESULT IHEProfileTerminal::put_VTTerminalID([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify the terminal ID or DA response that HostExplorer sends to the host:

- 0—VT100
- 1—VT101
- 2—VT102
- 3—VT220
- 4—VT320
- 5—VT420
- 6—VT80
- 7—VT100J
- 8—VT102J
- 9—VT220J
- 10—VT282
- 11—VT382

*newVal*—The set value, specifying the terminal ID or DA response that HostExplorer sends to the host.

**Basic Example**

```
Dim Profile As HEProfile
Dim Terminal As HEProfileTerminal
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Terminal = Profile.Terminal
' Get value
iVal = SessTerm.VTTerminalID
If (iVal = 0) Then
    ' Set value
    SessTerm.VTTerminalID = 6
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTTerminalID property
short sVal;
pTerm->get_VTTerminalID(&sVal);
if (sVal == 0)
{
    sVal = 6;
    pTerm->put_VTTerminalID(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTWrapLine

### Property, IHEProfileTerminal VT

This property returns or sets a value indicating whether HostExplorer automatically wraps lines that extend past the last column on the screen. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileTerminal.VTWrapLine
```

```
HEProfileTerminal.VTWrapLine = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileTerminal::get_VTWrapLine([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileTerminal::put_VTWrapLine([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically wraps lines. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically wrap lines.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically wraps lines. A value of VARIANT\_FALSE indicates that HostExplorer does not automatically wrap lines.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Terminal As HEProfileTerminal  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Terminal = Profile.Terminal  
' Get value  
bVal = SessTerm.VTWrapLine  
If (bVal = False) Then  
    ' Set value  
    SessTerm.VTWrapLine = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTerminal interface
IHEProfileTerminal* pTerminal;
pIProfile->get_Terminal ( &pTerminal );
//Get VTWrapLine property
VARIANT_BOOL bVal;
pTerm->get_VTWrapLine(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pTerm->put_VTWrapLine(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileGraphics Interface

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

### Properties

The ProfileGraphics interface consists of the following properties:

- APL
- GraphicsCursorType
- GraphicsModel
- LightPen
- ProgramSymbols
- PSCellSize

### APL

#### Property, IHEProfileGraphics 3270

This property returns or sets a value that indicates whether HostExplorer supports APL (A Program Language). By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileGraphics.APL  
HEProfileGraphics.APL = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileGraphics::get_APL([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfileGraphics::put_APL([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer supports APL. A returned value of VARIANT\_FALSE indicates that HostExplorer does not support APL.  
*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer supports APL. A value of VARIANT\_FALSE indicates that HostExplorer does not support APL.

**Basic Example**

```
Dim Profile As HEProfile
Dim Graphics As HEProfileGraphics
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Graphics = Profile.Graphics
' Get value
bVal = SessGraph.APL
If (bVal = False) Then
    ' Set value
    SessGraph.APL = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileGraphics interface
    IHEProfileGraphics* pGraphics;
    pIProfile->get_Graphics ( &pGraphics );
    //Get APL property
    VARIANT_BOOL bVal;
    pGraphics->get_APL(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pGraphics->put_APL(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## GraphicsCursorType

### Property, IHEProfileGraphics 3270

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

#### Basic Syntax

```
HOSTEX_GRAPHICS_CURSOR_TYPE = HEProfileGraphics.GraphicsCursorType  
HEProfileGraphics.GraphicsCursorType = HOSTEX_GRAPHICS_CURSOR_TYPE
```

#### C++ Syntax

```
HRESULT IHEProfileGraphics::get_GraphicsCursorType([out, retval]  
HOSTEX_GRAPHICS_CURSOR_TYPE *pVal);  
HRESULT IHEProfileGraphics::put_GraphicsCursorType([in]  
HOSTEX_GRAPHICS_CURSOR_TYPE newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim Graphics As HEProfileGraphics  
Dim CurType As HOSTEX_GRAPHICS_CURSOR_TYPE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Graphics = Profile.Graphics  
CurType = SessGraph.GraphicsCursorType  
If (CurType <> HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE) Then  
    SessGraph.GraphicsCursorType = HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_  
WHITE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileGraphics interface
    IHEProfileGraphics* pGraphics;
    pIProfile->get_Graphics ( &pGraphics );
    //Get GraphicsCursorType property
    HOSTEX_GRAPHICS_CURSOR_TYPE CurType;
    pGraphics->get_GraphicsCursorType(&CurType);
    if (CurType!= HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE)
    {
        CurType = HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE;
        pGraphics->put_GraphicsCursorType(CurType);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## GraphicsModel

### Property, IHEProfileGraphics 3270

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

#### Basic Syntax

```
HOSTEX_GRAPHICS_MODEL = HEProfileGraphics.GraphicsModel
```

```
HEProfileGraphics.GraphicsModel = HOSTEX_GRAPHICS_MODEL
```

#### C++ Syntax

```
HRESULT IHEProfileGraphics::get_GraphicsModel([out, retval]  
HOSTEX_GRAPHICS_MODEL *pVal);  
  
HRESULT IHEProfileGraphics::put_GraphicsModel([in] HOSTEX_GRAPHICS_MODEL  
newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim Graphics As HEProfileGraphics  
Dim GraphMod As HOSTEX_GRAPHICS_MODEL  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Graphics = Profile.Graphics  
GraphMod = SessGraph.GraphicsModel  
If (GraphMod <> HOSTEX_GRAPHICS_MODEL_3179G) Then  
    SessGraph.GraphicsModel = HOSTEX_GRAPHICS_MODEL_3179G  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileGraphics interface
    IHEProfileGraphics* pGraphics;
    pIProfile->get_Graphics ( & pGraphics );
    //Get GraphicsModel property
    HOSTEX_GRAPHICS_MODEL GraphMod;
    pGraphics->get_GraphicsModel(&GraphMod);
    if (GraphMod!=HOSTEX_GRAPHICS_MODEL_3179G)
    {
        GraphMod = HOSTEX_GRAPHICS_MODEL_3179G;
        pGraphics->put_GraphicsModel(GraphMod);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LightPen

### Property, IHEProfileGraphics 3270

This property returns or sets a value indicating whether HostExplorer allows you to use the keyboard and/or mouse to emulate light-pen functionality. This functionality allows you to use a light-sensitive device to select screen fields. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileGraphics.LightPen
```

```
HEProfileGraphics.LightPen = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileGraphics::get_LightPen([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileGraphics::put_LightPen([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer allows you to use the keyboard and/or mouse to emulate light-pen functionality. A returned value of VARIANT\_FALSE indicates that HostExplorer does not allow you to use the keyboard and/or mouse to emulate light-pen functionality.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer allows you to use the keyboard and/or mouse to emulate light-pen functionality. A value of VARIANT\_FALSE indicates that HostExplorer does not allow you to use the keyboard and/or mouse to emulate light-pen functionality.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Graphics As HEProfileGraphics  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Graphics = Profile.Graphics  
' Get value  
bVal = SessGraph.LightPen  
If (bVal = False) Then  
    ' Set value  
    SessGraph.LightPen = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileGraphics interface
    IHEProfileGraphics* pGraphics;
    pIProfile->get_Graphics ( &pGraphics );
    //Get LightPen property
    VARIANT_BOOL bVal;
    pGraphics->get_LightPen(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pGraphics->put_LightPen(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProgramSymbols

### Property, IHEProfileGraphics 3270

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

#### Basic Syntax

```
Boolean = HEProfileGraphics.ProgramSymbols
```

```
HEProfileGraphics.ProgramSymbols = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEProfileGraphics::put_ProgramSymbols([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer supports program symbols. A returned value of VARIANT\_FALSE indicates that HostExplorer does not support program symbols.

*newVal*—A set value of VARIANT\_TRUE indicates that HostExplorer supports program symbols. A set value of VARIANT\_FALSE indicates that HostExplorer does not support program symbols.

#### Basic Example

```
Dim Profile As HEProfile
Dim Graphics As HEProfileGraphics
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Graphics = Profile.Graphics
' Get value
bVal = SessGraph.ProgramSymbols
If (bVal = False) Then
    ' Set value
    SessGraph.ProgramSymbols = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileGraphics interface
    IHEProfileGraphics* pGraphics;
    pIProfile->get_Graphics ( & pGraphics );
    //Get ProgramSymbols property
    VARIANT_BOOL bVal;
    pGraphics->get_ProgramSymbols(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pGraphics->put_ProgramSymbols(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PSCellSize

### Property, IHEProfileGraphics 3270

This property returns or sets a value that determines the cell size when HostExplorer reports a fixed-coordinate space to the host. Because several host systems are sensitive to cell size, you need to select the appropriate size. By default, this property is set to HOSTEX\_GRAPHICS\_CELLSIZE\_AUTOMATIC.

#### Basic Syntax

```
HOSTEX_GRAPHICS_CELL_SIZE = HEProfileGraphics.PSCellSize
```

```
HEProfileGraphics.PSCellSize = HOSTEX_GRAPHICS_CELLSIZE
```

#### C++ Syntax

```
HRESULT IHEProfileGraphics::get_PSCellSize([out, retval]  
HOSTEX_GRAPHICS_CELLSIZE *pVal);
```

```
HRESULT IHEProfileGraphics::put_PSCellSize([in] HOSTEX_GRAPHICS_CELLSIZE  
newVal);
```

#### Parameters

*pVal*—The returned value, which determines the cell size when HostExplorer reports a fixed-coordinate space to the host.

*newVal*—The set value, which determines the cell size when HostExplorer reports a fixed-coordinate space to the host.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Graphics As HEProfileGraphics  
Dim CellSize As HOSTEX_GRAPHICS_CELL_SIZE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Graphics = Profile.Graphics  
CellSize = SessGraph.PSCellSize  
If (CellSize <> HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC) Then  
    SessGraph.PSCellSize = HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileGraphics interface
    IHEProfileGraphics* pGraphics;
    pIProfile->get_Graphics ( &pGraphics );
    //Get PSCellSize property
    HOSTEX_GRAPHICS_CELL_SIZE CellSize;
    pGraphics->get_PSCellSize(&CellSize);
    if (CellSize!= HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC)
    {
        CellSize = HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC;
        pGraphics->put_PSCellSize(CellSize);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileKeyboard Interface

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

## Properties

The ProfileKeyboard interface consists of the following properties:

- AllowAIDKeyRepeat
- CurrentKeyboard
- LockOnAttention
- RemapKeypad
- TypeAheadTimeout
- VTEnableBreak
- VTNewLineMode
- AllowDiac
- KeyboardType
- MapNumLock
- TypeAhead
- VTCursorKeyApplMode
- VTKeypadApplMode

### AllowAIDKeyRepeat

**Property, IHEProfileKeyboard 3270**

This property returns or sets a value indicating whether HostExplorer can send multiple function key commands to the host without requiring you to lift and press a key again. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileKeyboard.AllowAIDKeyRepeat  
HEProfileKeyboard.AllowAIDKeyRepeat = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_AllowAIDKeyRepeat([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileKeyboard::put_AllowAIDKeyRepeat([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer can send multiple function-key commands to the host. A returned value of VARIANT\_FALSE indicates that HostExplorer does not send multiple function-key commands to the host.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer can send multiple function-key commands to the host. A value of VARIANT\_FALSE indicates that HostExplorer does not send multiple function-key commands to the host.

### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
bVal = SessKeyb.AllowAIDKeyRepeat
If (bVal = False) Then
    ' Set value
    SessKeyb.AllowAIDKeyRepeat = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get AllowAIDKeyRepeat property
    VARIANT_BOOL bVal;
    pKeyb->get_AllowAIDKeyRepeat(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pKeyb->put_AllowAIDKeyRepeat(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AllowDiac

### Property, IHEProfileKeyboard 3270 5250

This property returns or sets a value indicating whether HostExplorer supports accented and/or special characters. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileKeyboard.AllowDiac
```

```
HEProfileKeyboard.AllowDiac = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_AllowDiac([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileKeyboard::put_AllowDiac([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer supports accented and/or special characters. A returned value of VARIANT\_FALSE indicates that HostExplorer does not support accented and/or special characters.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer supports accented and/or special characters. A value of VARIANT\_FALSE indicates that HostExplorer does not support accented and/or special characters.

#### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
bVal = SessKeyb.AllowDiac
If (bVal = False) Then
    ' Set value
    SessKeyb.AllowDiac = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get AllowDiac property
    VARIANT_BOOL bVal;
    pKeyb->get_AllowDiac(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pKeyb->put_AllowDiac(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CurrentKeyboard

### Property, IHEProfileKeyboard 3270 5250 VT

This property returns or sets a string specifying a keyboard map to use for the current session.

#### Basic Syntax

```
String = HEProfileKeyboard.CurrentKeyboard
```

```
HEProfileKeyboard.CurrentKeyboard = String
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_CurrentKeyboard([out, retval] BSTR  
*pVal);
```

```
HRESULT IHEProfileKeyboard::put_CurrentKeyboard([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the keyboard map to use for the current session.

*newVal*—The set string, specifying the keyboard map to use for the current session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Keyboard As HEProfileKeyboard  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Keyboard = Profile.Keyboard  
' Get string  
strVal = SessKeyb.CurrentKeyboard  
If (Len(strVal) = 0) Then  
    ' Set string  
    SessKeyb.CurrentKeyboard = "Default"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get CurrentKeyboard property
    BSTR bstr ;
    pKeyb->get_CurrentKeyboard(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("Default"));
        pKeyb->put_CurrentKeyboard(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KeyboardType

Property, IHEProfileKeyboard 3270 5250 VT

This property returns or sets a value that specifies the type of keyboard to use for the current session.

### Basic Syntax

```
HOSTEX_KEYBOARD_TYPE = HEProfileKeyboard.KeyboardType
```

```
HEProfileKeyboard.KeyboardType = HOSTEX_KEYBOARD_TYPE
```

### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_KeyboardType([out, retval]  
HOSTEX_KEYBOARD_TYPE *pVal);
```

```
HRESULT IHEProfileKeyboard::put_KeyboardType([in] HOSTEX_KEYBOARD_TYPE  
newVal);
```

### Parameters

*pVal*—The returned value, specifying the type of keyboard to use for the current session.

*newVal*—The set value, specifying the type of keyboard to use for the current session.

### Basic Example

```
Dim Profile As HEProfile  
Dim Keyboard As HEProfileKeyboard  
Dim kbType As HOSTEX_KEYBOARD_TYPE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Keyboard = Profile.Keyboard  
kbType = SessKeyb.KeyboardType  
If (kbType <> HOSTEX_KEYBOARD_TYPE_PC_101) Then  
    SessKeyb.KeyboardType = HOSTEX_KEYBOARD_TYPE_PC_101  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get KeyboardType property
    HOSTEX_KEYBOARD_TYPE kbType;
    pKeyb->get_KeyboardType(&kbType);
    if (kbType != HOSTEX_KEYBOARD_TYPE_PC_101)
    {
        kbType = HOSTEX_KEYBOARD_TYPE_PC_101;
        pKeyb->put_KeyboardType(kbType);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LockOnAttention

Property, IHEProfileKeyboard 3270 5250 VT

This property returns or sets a value that indicates whether the keyboard is locked after you send an Attention-key command. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfileKeyboard.LockOnAttention
```

```
HEProfileKeyboard.LockOnAttention = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_LockOnAttention([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileKeyboard::put_LockOnAttention([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the keyboard is locked. A returned value of VARIANT\_FALSE indicates that the keyboard is not locked.

*newVal*—A value of VARIANT\_TRUE indicates that the keyboard is locked. A value of VARIANT\_FALSE indicates that the keyboard is not locked.

### Basic Example

```
Dim Profile As HEProfile  
Dim Keyboard As HEProfileKeyboard  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Keyboard = Profile.Keyboard  
' Get value  
bVal = SessKeyb.LockOnAttention  
If (bVal = False) Then  
    ' Set value  
    SessKeyb.LockOnAttention = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get LockOnAttention property
    VARIANT_BOOL bVal;
    pKeyb->get_LockOnAttention(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pKeyb->put_LockOnAttention(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MapNumLock

Property, IHEProfileKeyboard 3270 5250 VT

This property returns or sets a value indicating whether you can use the numeric keypad, regardless of the NumLock mode. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileKeyboard.MapNumLock
```

```
HEProfileKeyboard.MapNumLock = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_MapNumLock([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileKeyboard::put_MapNumLock([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that you can use the numeric keypad. A returned value of VARIANT\_FALSE indicates that you cannot use the numeric keypad.

*newVal*—A value of VARIANT\_TRUE indicates that you can use the numeric keypad. A value of VARIANT\_FALSE indicates that you cannot use the numeric keypad.

### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
bVal = SessKeyb.MapNumLock
If (bVal = False) Then
    ' Set value
    SessKeyb.MapNumLock = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get MapNumLock property
    VARIANT_BOOL bVal;
    pKeyb->get_MapNumLock(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pKeyb->put_MapNumLock(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## RemapKeypad

Property, IHEProfileKeyboard 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer allows you to map the /, ", -, and + keys while in NumLock mode. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfileKeyboard.RemapKeypad
```

```
HEProfileKeyboard.RemapKeypad = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_RemapKeypad([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileKeyboard::put_RemapKeypad([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer allows you to map the /, ", -, and + keys while in NumLock mode. A returned value of VARIANT\_FALSE indicates that HostExplorer does not allow you to map those keys while in NumLock mode.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer allows you to map the /, ", -, and + keys while in NumLock mode. A value of VARIANT\_FALSE indicates that HostExplorer does not allow you to map those keys while in NumLock mode.

### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
bVal = SessKeyb.RemapKeypad
If (bVal = False) Then
    ' Set value
    SessKeyb.RemapKeypad = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get RemapKeypad property
    VARIANT_BOOL bVal;
    pKeyb->get_RemapKeypad(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pKeyb->put_RemapKeypad(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TypeAhead

### Property, IHEProfileKeyboard 3270 5250

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

#### Basic Syntax

```
Boolean = HEProfileKeyboard.TypeAhead
```

```
HEProfileKeyboard.TypeAhead = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_TypeAhead([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileKeyboard::put_TypeAhead([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer buffers typed characters. A returned value of VARIANT\_FALSE indicates that HostExplorer is not buffering typed characters.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer buffers typed characters. A value of VARIANT\_FALSE indicates that HostExplorer is not buffering typed characters.

#### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
bVal = SessKeyb.TypeAhead
If (bVal = False) Then
    ' Set value
    SessKeyb.TypeAhead = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( & pKeyboard );
    //Get TypeAhead property
    VARIANT_BOOL bVal;
    bVal = pKeyb->get_TypeAhead(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pKeyb->put_TypeAhead(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TypeAheadTimeout

### Property, IHEProfileKeyboard 3270 5250

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

#### Basic Syntax

```
Long = HEProfileKeyboard.TypeAheadTimeout
```

```
HEProfileKeyboard.TypeAheadTimeout = Long
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_TypeAheadTimeout([out, retval] long *pVal);
```

```
HRESULT IHEProfileKeyboard::put_TypeAheadTimeout([in] long newVal);
```

#### Parameters

*pVal*—The returned value, which indicates the time delay.

*newVal*—The set value, which indicates the time delay.

#### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim lVal As Long
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
lVal = SessKeyb.TypeAheadTimeout
If (lVal = 0) Then
    ' Set value
    SessKeyb.TypeAheadTimeout = 100
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileKeyboard interface
    IHEProfileKeyboard* pKeyboard;
    pIProfile->get_Keyboard ( &pKeyboard );
    //Get TypeAheadTimeout property
    long lVal;
    pKeyb->get_TypeAheadTimeout(&lVal);
    if (lVal == 0)
    {
        lVal =100;
        pKeyb->put_TypeAheadTimeout(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTCursorKeyApplMode

### Property, IHEProfileKeyboard VT

This property returns or sets a variable indicating the cursor-key mode. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileKeyboard.VTCursorKeyApplMode  
HEProfileKeyboard.VTCursorKeyApplMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_VTCursorKeyApplMode([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileKeyboard::put_VTCursorKeyApplMode([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the cursor key is in Application mode. A returned value of VARIANT\_FALSE indicates that the cursor key operates in Normal mode.

*newVal*—A value of VARIANT\_TRUE indicates that the cursor key is in Application mode. A value of VARIANT\_FALSE indicates that the cursor key operates in Normal mode.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Keyboard As HEProfileKeyboard  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Keyboard = Profile.Keyboard  
' Get value  
bVal = SessKeyb.VTCursorKeyApplMode  
If (bVal = False) Then  
    ' Set value  
    SessKeyb.VTCursorKeyApplMode = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileKeyboard interface
IHEProfileKeyboard* pKeyboard;
pIProfile->get_Keyboard ( &pKeyboard );
//Get VTCursorKeyApplMode property
VARIANT_BOOL bVal;
pKeyb->get_VTCursorKeyApplMode(&bVal);
if (bVal==VARIANT_FALSE)
{
    bVal=VARIANT_TRUE;
    pKeyb->put_VTCursorKeyApplMode (bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTEnableBreak

### Property, IHEProfileKeyboard VT

This property returns or sets a variable indicating whether HostExplorer enables the Break key to send a break signal to the host.

**Basic Syntax**

```
Boolean = HEProfileKeyboard.VTEnableBreak
```

```
HEProfileKeyboard.VTEnableBreak = Boolean
```

**C++ Syntax**

```
HRESULT IHEProfileKeyboard::get_VTEnableBreak([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileKeyboard::put_VTEnableBreak([in] VARIANT_BOOL newVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables the Break key. A returned value of VARIANT\_FALSE indicates that HostExplorer does not enable the Break key.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables the Break key. A value of VARIANT\_FALSE indicates that HostExplorer does not enable the Break key.

**Basic Example**

```
Dim Profile As HEProfile  
Dim Keyboard As HEProfileKeyboard  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Keyboard = Profile.Keyboard  
' Get value  
bVal = SessKeyb.VTEnableBreak  
If (bVal = False) Then  
    ' Set value  
    SessKeyb.VTEnableBreak = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileKeyboard interface
IHEProfileKeyboard* pKeyboard;
pIProfile->get_Keyboard ( & pKeyboard );
//Get VTEnableBreak property
VARIANT_BOOL bVal;
pKeyb->get_VTEnableBreak(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pKeyb->put_VTEnableBreak(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTKeypadApplMode

### Property, IHEProfileKeyboard VT

This property returns or sets a variable indicating the Keypad mode. When set to VARIANT\_TRUE, the keypad operates in Application mode. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileKeyboard.VTKeypadApplMode  
HEProfileKeyboard.VTKeypadApplMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_VTKeypadApplMode ([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileKeyboard::put_VTKeypadApplMode ([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the keypad operates in Application mode. A returned value of VARIANT\_FALSE indicates that the keypad operates in Numeric mode.

*newVal*—A value of VARIANT\_TRUE indicates that the keypad operates in Application mode. A value of VARIANT\_FALSE indicates that the keypad operates in Numeric mode.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Keyboard As HEProfileKeyboard  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Keyboard = Profile.Keyboard  
' Get value  
bVal = SessKeyb.VTKeypadApplMode  
If (bVal = False) Then  
    ' Set value  
    SessKeyb.VTKeypadApplMode = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileKeyboard interface
IHEProfileKeyboard* pKeyboard;
pIProfile->get_Keyboard ( &pKeyboard );
//Get VTKeypadApplMode property
VARIANT_BOOL bVal;
pKeyb->get_VTKeypadApplMode (&bVal);
if (bVal==VARIANT_FALSE)
{
    bVal=VARIANT_TRUE;
    pKeyb->put_VTKeypadApplMode (bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTNewLineMode

### Property, IHEProfileKeyboard VT

This property returns or sets a variable that determines whether pressing Enter sends a carriage return (CR) or carriage return/line feed (CR/LF) to the host. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileKeyboard.VTNewLineMode
```

```
HEProfileKeyboard.VTNewLineMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileKeyboard::get_VTNewLineMode([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileKeyboard::put_VTNewLineMode([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that pressing Enter sends a CR to the host. A returned value of VARIANT\_FALSE indicates that pressing Enter sends a CR/LF to the host.

*newVal*—A value of VARIANT\_TRUE indicates that pressing Enter sends a CR to the host. A value of VARIANT\_FALSE indicates that pressing Enter sends a CR/LF to the host.

#### Basic Example

```
Dim Profile As HEProfile
Dim Keyboard As HEProfileKeyboard
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Keyboard = Profile.Keyboard
' Get value
bVal = SessKeyb.VTNewLineMode
If (bVal = False) Then
    ' Set value
    SessKeyb.VTNewLineMode = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileKeyboard interface
IHEProfileKeyboard* pKeyboard;
pIProfile->get_Keyboard ( & pKeyboard );
//Get VTNewLineMode property
VARIANT_BOOL bVal;
pKeyb->get_VTNewLineMode(&bVal);
if (bVal==VARIANT_FALSE)
{
    bVal=VARIANT_TRUE;
    pKeyb->put_VTNewLineMode (bVal) ;
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileMouse Interface

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

## Properties

The ProfileMouse interface consists of the following properties:

- BlockSelect
- SelectHilight

### BlockSelect

**Property, IHEProfileMouse VT**

This property returns or sets a value indicating whether HostExplorer selects a rectangular region of the screen when selecting text. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileMouse.BlockSelect
```

```
HEProfileMouse.BlockSelect = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileMouse::get_BlockSelect([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileMouse::put_BlockSelect([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer selects a rectangular region of the screen when selecting text. A returned value of VARIANT\_FALSE indicates that HostExplorer selects text in a stream-like fashion.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer selects a rectangular region of the screen when selecting text. A value of VARIANT\_FALSE indicates that HostExplorer selects text in a stream-like fashion.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Mouse As HEProfileMouse  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Mouse = Profile.Mouse  
' Get value  
bVal = SessMouse.BlockSelect  
If (bVal = False) Then  
    ' Set value  
    SessMouse.BlockSelect = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileMouse interface
IHEProfileMouse* pMouse;
pIProfile->get_Mouse ( &pMouse );
//Get BlockSelect property
VARIANT_BOOL bVal;
pMouse->get_BlockSelect(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pMouse->put_BlockSelect(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SelectHilight

Property, IHEProfileMouse 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer highlights selected text in reverse video. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileMouse.SelectHilight  
HEProfileMouse.SelectHilight = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileMouse::get_SelectHilight([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileMouse::put_SelectHilight([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer highlights selected text in reverse video. If *pVal* equals VARIANT\_FALSE, HostExplorer does not highlight selected text in reverse video.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to highlight selected text in reverse video; otherwise, set *newVal* to VARIANT\_FALSE.

### Basic Example

```
Dim Profile As HEProfile  
Dim Mouse As HEProfileMouse  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Mouse = Profile.Mouse  
' Get value  
bVal = SessMouse.SelectHilight  
If (bVal = False) Then  
    ' Set value  
    SessMouse.SelectHilight = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileMouse interface
IHEProfileMouse* pMouse;
pIProfile->get_Mouse ( &pMouse );
//Get SelectHilight property
VARIANT_BOOL bVal;
pMouse->get_SelectHilight(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pMouse->put_SelectHilight(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileToolbar Interface

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

## Properties

The ProfileToolbar interface consists of the following properties:

- BigToolbar
- MaxBitmaps
- ShowTips
- ToolbarDockType
- Button
- NumTools
- TBUserBitmaps
- ToolbarFilename

### **BigToolbar**

**Property, IHEProfileToolbar 3270 5250 VT**

This property returns or sets a value that specifies whether HostExplorer displays the icons in a toolbar as large icons.

#### **Basic Syntax**

```
Boolean = HEProfileToolbar.BigToolbar
```

```
HEProfileToolbar.BigToolbar = Boolean
```

#### **C++ Syntax**

```
HRESULT IHEProfileToolbar::get_BigToolbar([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileToolbar::put_BigToolbar([in] VARIANT_BOOL newVal);
```

#### **Parameters**

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer displays toolbar icons as large icons. If *pVal* equals VARIANT\_FALSE, HostExplorer displays toolbar icons at their regular size.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to display toolbar icons as large icons; otherwise, set *newVal* to VARIANT\_FALSE.

#### **Basic Example**

```
Dim Profile As HEProfile  
Dim Toolbar As HEProfileToolbar  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Toolbar = Profile.Toolbar  
' Get value  
bVal = SessToolbar.BigToolbar  
If (bVal = False) Then  
    ' Set value  
    SessToolbar.BigToolbar = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileToolbar interface
IHEProfileToolbar* pToolbar;
pIProfile->get_Toolbar ( & pToolbar );
//Get BigToolbar property
VARIANT_BOOL bVal;
pToolbar->get_BigToolbar(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pToolbar->put_BigToolbar(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Button

### Property, IHEProfileToolbar 3270 5250 VT

This property returns or sets the number of buttons on a customized toolbar.

#### Basic Syntax

```
Long = HEProfileToolbar.Button
```

```
HEProfileToolbar.Button = Long
```

#### C++ Syntax

```
HRESULT IHEProfileToolbar::get_Button([out, retval] long *pVal);  
HRESULT IHEProfileToolbar::put_Button([in] long newVal);
```

#### Parameters

*pVal*—The returned value, specifying the number of buttons on a customized toolbar.

*newVal*—The set value, specifying the number of buttons that you want on a customized toolbar.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Toolbar As HEProfileToolbar  
Dim lVal As Long  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Toolbar = Profile.Toolbar  
' Get value  
lVal = Toolbar.Button  
If (lVal = 3) Then  
    ' Set value  
    Toolbar.Button = 4  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileToolbar interface
    IHEProfileToolbar* pToolbar;
    pIProfile->get_Toolbar( & pToolbar );
    //Get Button property
    long lVal;
    pToolbar->get_Button(&lVal);
    if (lVal == 3)
    {
        bVal = 4;
        pToolbar->put_Button(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MaxBitmaps

Property, IHEProfileToolbar 3270 5250 VT

This property returns or sets the maximum number of bitmaps available for buttons on any toolbar.

### Basic Syntax

```
Integer = HEProfileToolbar.MaxBitmaps
```

```
HEProfileToolbar.MaxBitmaps = Integer
```

### C++ Syntax

```
HRESULT IHEProfileToolbar::get_MaxBitmaps([out, retval] short *pVal);
```

```
HRESULT IHEProfileToolbar::put_MaxBitmaps([in] short newVal);
```

### Parameters

*pVal*—The returned value, specifying the maximum number of bitmaps available for toolbar buttons.

*newVal*—The set value, specifying the maximum number of bitmaps that you want for toolbar buttons.

### Basic Example

```
Dim Profile As HEProfile
Dim Toolbar As HEProfileToolbar
Dim maxBmps As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Toolbar = Profile.Toolbar
' Get value
maxBmps = Toolbar.MaxBitmaps
If (maxBmps <= 3) Then
    ' Set value
    Toolbar.MaxBitmaps = 4
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileToolbar interface
    IHEProfileToolbar* pToolbar;
    pIProfile->get_Toolbar ( & pToolbar );
    //Get MaxBitmaps property
    short maxBmps;
    pToolbar->get_MaxBitmaps(&maxBmps);
    if (maxBmps <= 3)
    {
        maxBmps = 4;
        pToolbar->put_MaxBitmaps(maxBmps);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## NumTools

### Property, IHEProfileToolbar 3270 5250 VT

This property returns or sets the maximum number of tools allowed on any toolbar.

#### Basic Syntax

```
Integer = HEProfileToolbar.NumTools  
HEProfileToolbar.NumTools = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileToolbar::get_NumTools([out, retval] short *pVal);  
HRESULT IHEProfileToolbar::put_NumTools([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the maximum number of tools allowed for any toolbar.

*newVal*—The set value, specifying the maximum number of tools that you want for toolbars.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Toolbar As HEProfileToolbar  
Dim nTools As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Toolbar = Profile.Toolbar  
' Get value  
nTools = Toolbar.NumTools  
If (nTools <= 3) Then  
    ' Set value  
    Toolbar.NumTools = 4  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileToolbar interface
    IHEProfileToolbar* pToolbar;
    pIProfile->get_Toolbar( & pToolbar );
    //Get NumTools property
    short nTools;
    pToolbar->get_NumTools(&nTools);
    if (nTools <= 3)
    {
        nTools = 4;
        pToolbar->put_NumTools(nTools);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ShowTips

### Property, IHEProfileToolbar 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer displays a tool tip when you move the cursor over a toolbar button.

#### Basic Syntax

```
Boolean = HEProfileToolbar.ShowTips  
HEProfileToolbar.ShowTips = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileToolbar::get_ShowTips ([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileToolbar::put_ShowTips ([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer displays tool tips when you move the cursor over toolbar buttons. If *pVal* equals VARIANT\_FALSE, HostExplorer does not display tool tips.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to display tool tips. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to display tool tips.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Toolbar As HEProfileToolbar  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Toolbar = Profile.Toolbar  
' Get value  
bVal = Toolbar.ShowTips  
If (bVal = False) Then  
    ' Set value  
    Toolbar.ShowTips = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileToolbar interface
    IHEProfileToolbar* pToolbar;
    pIProfile->get_Toolbar ( & pToolbar );
    //Get ShowTips property
    VARIANT_BOOL bVal;
    pToolbar->get_ShowTips(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pToolbar->put_ShowTips(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TBUserBitmaps

Property, IHEProfileToolbar 3270 5250 VT

This property returns or sets the name of a user-customized image file that can be used as the image for a button in any toolbar.

### Basic Syntax

```
String = HEProfileToolbar.TBUserBitmaps  
HEProfileToolbar.TBUserBitmaps = String
```

### C++ Syntax

```
HRESULT IHEProfileToolbar::get_TBUserBitmaps([out, retval] BSTR *pVal);  
HRESULT IHEProfileToolbar::put_TBUserBitmaps([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the name of the user-customized image file.  
*newVal*—The set string, specifying the name of the image file that you want to use.

### Basic Example

```
Dim Profile As HEProfile  
Dim Toolbar As HEProfileToolbar  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Toolbar = Profile.Toolbar  
' Get value  
str = Toolbar.TBUserBitmaps  
If (Len(str) = 0) Then  
    ' Set value  
    Toolbar.TBUserBitmaps = "file.bmp"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileToolbar interface
    IHEProfileToolbar* pToolbar;
    pIProfile->get_Toolbar ( & pToolbar );
    //Get TBUserBitmaps property
    BSTR bstr;
    pToolbar->get_TBUserBitmaps(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("file.bmp"));
        pToolbar->put_TBUserBitmaps(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ToolbarDockType

Property, IHEProfileToolbar 3270 5250 VT

This property returns or sets a value that indicates the manner in which the toolbar docks to the session window.

### Basic Syntax

```
Integer = HEProfileToolbar.ToolbarDockType
```

```
HEProfileToolbar.ToolbarDockType = Integer
```

### C++ Syntax

```
HRESULT IHEProfileToolbar::get_ToolbarDockType([out, retval] short  
*pVal);
```

```
HRESULT IHEProfileToolbar::put_ToolbarDockType([in] short newVal);
```

### Parameters

*pVal*—The returned value, specifying the docking method for the toolbar.

*newVal*—The set value, specifying the docking method you want to use for the toolbar.

### Basic Example

```
Dim Profile As HEProfile  
Dim Toolbar As HEProfileToolbar  
Dim dockType As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Toolbar = Profile.Toolbar  
' Get value  
dockType = Toolbar.ToolbarDockType  
If (dockType = 0) Then  
    ' Set value  
    Toolbar.ToolbarDockType = 1  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileToolbar interface
    IHEProfileToolbar* pToolbar;
    pIProfile->get_Toolbar ( & pToolbar );
    //Get ToolbarDockType property
    short dockType;
    pToolbar->get_ToolbarDockType(&dockType);
    if (dockType == 0)
    {
        docktype = 1;
        pToolbar->put_ToolbarDockType(dockType);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ToolbarFilename

**Property, IHEProfileToolbar 3270 5250 VT**

This property returns or sets the file name of the default toolbar for the session window.

### Basic Syntax

```
String = HEProfileToolbar.ToolbarFilename
```

```
HEProfileToolbar.ToolbarFilename = String
```

### C++ Syntax

```
HRESULT IHEProfileToolbar::get_ToolbarFilename([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileToolbar::put_ToolbarFilename([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the file name of the default toolbar for the session window.

*newVal*—The file name of the toolbar that you want to set as the default for the session window.

### Basic Example

```
Dim Profile As HEProfile
Dim Toolbar As HEProfileToolbar
Dim str As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Toolbar = Profile.Toolbar
' Get value
str = Toolbar.ToolbarFilename
If (Len(str) = 0) Then
    ' Set value
    Toolbar.ToolbarFilename = "file"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileToolbar interface
IHEProfileToolbar* pToolbar;
pIProfile->get_Toolbar ( & pToolbar );
//Get ToolbarFilename property
BSTR bstr;
pToolbar->get_ToolbarFilename(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("file"));
    pToolbar->put_ToolbarFilename(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileTrackMenu Interface

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

## Properties

The ProfileTrackMenu interface consists of the following properties:

- TrackCommands
- TrackLabels

### TrackCommands

**Property, IHEProfileTrackMenu 3270 5250 VT**

This property returns or sets a string specifying the Track Menu functions, which are represented as predefined strings. To determine these strings, open a HostExplorer session, then open the Edit Session Properties dialog box. In the Track Menu section, the labels correspond to the functions. A function and label usually have the same string value.

#### Basic Syntax

```
String = HEProfileTrackMenu.TrackCommands  
HEProfileTrackMenu.TrackCommands = String
```

#### C++ Syntax

```
HRESULT IHEProfileTrackMenu::get_TrackCommands([out, retval] BSTR *pVal);  
HRESULT IHEProfileTrackMenu::put_TrackCommands([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the Track menu functions.

*newVal*—The set string, specifying the Track menu functions. These functions must correspond to the labels that you chose in the "ProfileTrackMenu.TrackLabels" property.

#### Basic Example

```
Dim Profile As HEProfile  
Dim TrackMenu As HEProfileTrackMenu  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set TrackMenu = Profile.TrackMenu  
' Get string  
strVal = SessTrk.TrackCommands  
If (Len(strVal) = 0) Then  
    SessTrk.TrackCommands = "Edit-Copy, Edit-Paste,Select-Line"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTrackMenu interface
IHEProfileTrackMenu* pTrackMenu;
pIProfile->get_TrackMenu ( &pTrackMenu );
//Get TrackCommands property
BSTR bstr ;
pTrack->get_TrackCommands(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Edit-Copy, Edit-Paste,Select-Line"));
    pTrack->put_TrackCommands(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TrackLabels

Property, IHEProfileTrackMenu 3270 5250 VT

This property returns or sets a string specifying the Track menu labels.

### Basic Syntax

```
String = HEProfileTrackMenu.TrackLabels  
HEProfileTrackMenu.TrackLabels = String
```

### C++ Syntax

```
HRESULT IHEProfileTrackMenu::get_TrackLabels([out, retval] BSTR *pVal);  
HRESULT IHEProfileTrackMenu::put_TrackLabels([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the Track menu labels.

*newVal*—The set string, specifying the Track menu labels. These labels must correspond to the commands that you chose in the "ProfileTrackMenu.TrackCommands" property.

### Basic Example

```
Dim Profile As HEProfile  
Dim TrackMenu As HEProfileTrackMenu  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set TrackMenu = Profile.TrackMenu  
' Get string  
strVal = SessTrk.TrackLabels  
If (Len(strVal) <> 0) Then  
    SessTrk.TrackLabels = "Copy,Paste,Select"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTrackMenu interface
IHEProfileTrackMenu* pTrackMenu;
pIProfile->get_TrackMenu ( & pTrackMenu );
//Get TrackLabels property
BSTR bstr ;
pTrack->get_TrackLabels(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Copy,Paste,Select"));
    pTrack->put_TrackLabels(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## 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.

### Properties

The ProfileTranslationTable interface consists of the following property:

CurrentLanguage

#### CurrentLanguage

**Property, IHEProfileTranslationTable 3270 5250 VT**

This property returns or sets a value specifying the host code page.

#### Basic Syntax

```
HEProfileTranslationTable.CurrentLanguage = String
```

#### C++ Syntax

```
HRESULT IHEProfileTranslationTable::get_CurrentLanguage([out, retval]
BSTR *pVal);

HRESULT IHEProfileTranslationTable::put_CurrentLanguage([in] BSTR
newVal);
```

#### Parameters

*pVal*—The returned value, specifying the host code page.

*newVal*—The set value, specifying the host code page.

#### Basic Example

```
Dim Profile As HEProfile
Dim TranslationTable As HEProfileTranslationTable
Dim strVal As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set TranslationTable = Profile.TranslationTable
strVal = SessTrans.CurrentLanguage
if (Len(strVal)=0) Then
    SessTrans.CurrentLanguage = "Default"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileTranslationTable interface
IHEProfileTranslationTable* pTranslationTable;
pIProfile->get_TranslationTable ( &pTranslationTable );
//Get CurrentLanguage property
BSTR bstr ;
pTrans->get_CurrentLanguage(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Default"));
    pTrans->put_CurrentLanguage(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileVTCharset Interface

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

### Properties

The ProfileVTCharset interface consists of the following properties:

- VTForceNRC
- VTNRC
- VTNRCMode
- VTUPSS

#### VTForceNRC

**Property, IHEProfileVTCharSet VT**

This property returns or sets a value indicating whether HostExplorer forces the current VT session to use the National Replacement Character (NRC) set.

#### Basic Syntax

```
Boolean = HEProfileVTCharset.VTForceNRC
```

```
HEProfileVTCharset.VTForceNRC = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileVTCharset::get_VTForceNRC([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileVTCharset::put_VTForceNRC([in] VARIANT_BOOL newVal);
```

**Parameters**

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer forces the current VT session to use the NRC set. If *pVal* equals VARIANT\_FALSE, the character set of the session is not restricted to NRC.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want the current VT session to use the NRC set. Otherwise, set *newVal* to VARIANT\_FALSE.

**Basic Example**

```
Dim Profile As HEProfile
Dim VTCharset As HEProfileVTCharset
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set VTCharset = Profile.VTCharset
bVal = VTCharset.VTForceNRC
If (bVal = False) Then
    VTCharset.VTForceNRC = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileVTCharSet interface
IHEProfileVTCharSet* pVTCharSet;
pIProfile->get_VTCharSet ( &pVTCharSet );
//Get VTForceNRC property
VARIANT_BOOL bVal;
pVtChSet->get_VTForceNRC(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pVtChSet->put_VTForceNRC(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTNRC

### Property, IHEProfileVTCharset VT

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

#### Basic Syntax

```
Integer = HEProfileVTCharset.VTNRC
```

```
HEProfileVTCharset.VTNRC = Integer
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned NRC set.

*newVal*—The NRC set that you specify.

#### Basic Example

```
Dim Profile As HEProfile  
Dim VTCharset As HEProfileVTCharset  
iVal = VTCharSet.VTNRC  
' Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set VTCharset = Profile.VTCharset  
' ISO French is hex 0x0052, which corresponds  
' to 82 in base ten.  
If (iVal = 82) Then  
    ' DEC Finnish is hex 0x0043, which  
    ' corresponds to 67 in base ten  
    VTCharSet.VTNRC = 67  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileVTCharSet interface
IHEProfileVTCharSet* pVTCharSet;
pIProfile->get_VTCharSet ( &pVTCharSet );
//Get VTNRC property
short sVal;
pVtChSet->get_VTNRC(&sVal);
//ISO French is hex 0x0052, which
//corresponds to 82 in base ten.
if (sVal == 82)
{
    //DEC Finnish is hex 0x0043, which
    //corresponds to 67 in base ten
    sVal = 67;
    pVtChSet->put_VTNRC(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTNRCMode

### Property, IHEProfileVTCharset VT

This property returns or sets a value indicating whether NRC (National Replacement Character) 7-bit mode is set for the current VT session.

#### Basic Syntax

```
Boolean = HEProfileVTCharset.VTNRCMode
```

```
HEProfileVTCharset.VTNRCMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileVTCharset::get_VTNRCMode([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileVTCharset::put_VTNRCMode([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that NRC 7-bit mode is set for the current session. A returned value of VARIANT\_FALSE indicates that NRC 7-bit mode is not set for the current session.

*newVal*—A value of VARIANT\_TRUE indicates that NRC 7-bit mode is set for the current session. A value of VARIANT\_FALSE indicates that NRC 7-bit mode is not set for the current session.

#### Basic Example

```
Dim Profile As HEProfile
Dim VTCharset As HEProfileVTCharset
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set VTCharset = Profile.VTCharset
bVal = VTCharset.VTNRCMode
If (bVal = False) Then
    VTCharset.VTNRCMode = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileVTCharSet interface
IHEProfileVTCharSet* pVTCharSet;
pIProfile->get_VTCharSet ( &pVTCharSet );
//Get VTNRCMode property
VARIANT_BOOL bVal;
pVtChSet->get_VTNRCMode(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pVtChSet->put_VTNRCMode(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTUPSS

### Property, IHEProfileVTCharset VT

This property returns or sets the UPSS (User Preferred Supplemental Character Set).

#### Basic Syntax

```
Integer = HEProfileVTCharset.VTUPSS
```

```
HEProfileVTCharset.VTUPSS = Integer
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned UPSS in base ten.

*newVal*—The set UPSS in base ten.

#### Basic Example

```
Dim Profile As HEProfile  
Dim VTCharset As HEProfileVTCharset  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set VTCharset = Profile.VTCharset  
iVal = VTCharSet.VTNRC  
' ISO Cyrillic is hex 0x000C, which  
' corresponds to 12 in base ten.  
If (iVal = 12) Then  
    ' PC Estonian is hex 0x0019, which  
    ' corresponds to 25 in base ten  
    VTCharSet.VTUPSS = 25  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileVTCharSet interface
IHEProfileVTCharSet* pVTCharSet;
pIProfile->get_VTCharSet ( &pVTCharSet );
//Get VTUPSS property
short sVal;
pVtChSet->get_VTUPSS(&sVal);
//PC Estonian is hex 0x0019, which
//corresponds to 25 in base ten
if (sVal == 12)
{
    //PC Estonian is hex 0x0019, which
    //corresponds to 25 in base ten
    sVal = 25;
    pVtChSet->put_VTUPSS(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileEdit Interface

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

### Methods

The ProfileEdit interface contains the following methods:

- ClearAllTabStops
- TabStop

### Properties

The ProfileEdit interface consists of the following properties:

- AlwaysAutoskip
- AutoCopyKeepSelection
- CellDelimited
- ClipFormatCSV
- ClipFormatPasteLink
- ClipFormatText
- CSVCopyEmptyFields
- CutChar
- EntryAssist
- LeftMargin
- MultiLineDelete
- NoLockKeyb
- PasteChar
- RemoveTrailingBlankOnCopy
- RespectNumeric
- SmartInsert
- WordWrapWithNewLineReturn
- AutoCopy
- BellMargin
- ClipFormatBitmap
- ClipFormatHE
- ClipFormatRTF
- ConvertNulls
- CSVTrimFields
- CutMode
- InsertResetByAttn
- MoveCursorAfterPaste
- MultiLineInsert
- NumericCharacters
- PasteMode
- ResetMDTOnEraseInput
- RightMargin
- WordWrap

## ClearAllTabStops

Method, IHEProfileEdit VT

This method clears all tab stops for the current session.

**Basic Syntax**      `HEProfileEdit.ClearAllTabStops`

**C++ Syntax**      `HRESULT IHEProfileEdit::ClearAllTabStops();`

**Parameters**      This method has no parameters.

**Basic Example**

```
Dim SessEd As HEProfileEdit
Set SessEd = Profile.Edit
' Clear all tab stops
SessEd.ClearAllTabStops
```

**C++ Example**

```
IHEProfileEdit *pEdit;
pSess->get_Edit(&pEdit);
// Clear all tab stops
pEdit->ClearAllTabStops;
```

## TabStop

Method, IHEProfileEdit VT

This method lets you specify a tab stop. You can use tab stops to control where the cursor moves when you press the Tab key. You specify the position of the tab stop as a column number.

**Basic Syntax**      `HEProfileEdit.TabStop(iTabStop As Integer)`

**C++ Syntax**      `HRESULT IHEProfileEdit::TabStop([in] short iTabStop);`

**Parameters**      *iTabStop*—The column number of the tab stop you want to add. The first possible column number is 1.

**Basic Example**

```
Dim SessEd As HEProfileEdit
Set SessEd = Profile.Edit
' Set a tab stop in the 10th column
SessEd.TabStop(10)
```

**C++ Example**

```
IHEProfileEdit *pEdit;  
pSess->get_Edit(&pEdit);  
// Set a tab stop in the 10th column  
pEdit->TabStop(10);
```

## AlwaysAutoskip

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating whether HostExplorer defines the field attribute that follows an unprotected field on the screen. By default, this property is set to VARIANT\_FALSE.

**Basic Syntax**

```
Boolean = HEProfileEdit.AlwaysAutoskip  
HEProfileEdit.AlwaysAutoskip = Boolean
```

**C++ Syntax**

```
HRESULT IHEProfileEdit::get_AlwaysAutoskip([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_AlwaysAutoskip([in] VARIANT_BOOL newVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer defines the field attribute that follows an unprotected field on the screen. A returned value of VARIANT\_FALSE indicates that HostExplorer does not define the field attribute that follows an unprotected field on the screen.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer defines the field attribute that follows an unprotected field on the screen. A value of VARIANT\_FALSE indicates that HostExplorer does not define the field attribute that follows an unprotected field on the screen.

**Basic Example**

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
bVal = SessEd.AlwaysAutoskip  
If (bVal = False) Then  
    SessEd.AlwaysAutoskip = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get AlwaysAutoskip property
    VARIANT_BOOL bVal;
    pEdit->get_AlwaysAutoskip(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_AlwaysAutoskip(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoCopy

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value that indicates whether HostExplorer automatically copies all selected text to the clipboard. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfileEdit.AutoCopy
```

```
HEProfileEdit.AutoCopy = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_AutoCopy([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileEdit::put_AutoCopy([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically copies all selected text to the clipboard. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically copy all selected text to the clipboard.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically copies all selected text to the clipboard. A value of VARIANT\_FALSE indicates that HostExplorer does not automatically copy all selected text to the clipboard.

### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
' Get value
bVal = SessEd.AutoCopy
If (bVal = False) Then
    ' Set value
    SessEd.AutoCopy = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get AutoCopy property
    VARIANT_BOOL bVal;
    pEdit->get_AutoCopy(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_AutoCopy(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoCopyKeepSelection

**Property, IHEProfileEdit 3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer maintains the selection once you have copied or cut text. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfileEdit.AutoCopyKeepSelection  
HEProfileEdit.AutoCopyKeepSelection = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_AutoCopyKeepSelection([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileEdit::put_AutoCopyKeepSelection([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer maintains the selection. A returned value of VARIANT\_FALSE indicates that HostExplorer does not maintain the selection.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer maintains the selection. A value of VARIANT\_FALSE indicates that HostExplorer does not maintain the selection.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.AutoCopyKeepSelection  
If (bVal = False) Then  
    ' Set value  
    SessEd.AutoCopyKeepSelection = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get AutoCopyKeepSelection property
    VARIANT_BOOL bVal;
    pEdit->get_AutoCopyKeepSelection(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_AutoCopyKeepSelection(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## BellMargin

### Property, IHEProfileEdit 3270 5250 VT

This property creates a beeping sound when the cursor reaches the last column of an input field.

#### Basic Syntax

```
Integer = HEProfileEdit.BellMargin
```

```
HEProfileEdit.BellMargin = Integer
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value, indicating the last column.

*newVal*—The set value, indicating the last column.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
iVal = SessEd.BellMargin  
If (iVal = 0) Then  
    ' Set value  
    SessEd.BellMargin = 7  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get BellMargin property
    short sVal;
    pEdit->get_BellMargin(&sVal);
    if (sVal == 0)
    {
        sVal = 7;
        pEdit->put_BellMargin(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CellDelimited

### Property, IHEProfileEdit 3270 5250 VT

This property enables CSV (Comma Separated Value) and BIF (Binary Interchange File) formats when copying data to the clipboard and pasting data from other applications. When copying data to the clipboard in Cell Delimited format, HostExplorer can parse screen data at words or at field attributes. This lets you determine how data appears in cells in your spreadsheet application.

#### Basic Syntax

```
HOSTEX_CELL_DELIMITED = HEProfileEdit.CellDelimited  
HEProfileEdit.CellDelimited = HOSTEX_CELL_DELIMITED
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value, which indicates how data appears in cells.

*newVal*—The set value, which indicates how data appears in cells.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim CellDelim As HOSTEX_CELL_DELIMITED  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
CellDelim = SessEd.CellDelimited  
If (CellDelim <> HOSTEX_CELL_DELIMITED_FIELD) Then  
    SessEd.CellDelimited = HOSTEX_CELL_DELIMITED_FIELD  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get CellDelimited property
    HOSTEX_CELL_DELIMITED CellDelim;
    pEdit->get_CellDelimited(&CellDelim);
    if (CellDelim != HOSTEX_CELL_DELIMITED_FIELD)
    {
        CellDelim = HOSTEX_CELL_DELIMITED_FIELD;
        pEdit->put_CellDelimited(CellDelim);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClipFormatBitmap

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables bitmap format when copying data to the clipboard. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileEdit.ClipFormatBitmap  
HEProfileEdit.ClipFormatBitmap = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_ClipFormatBitmap([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_ClipFormatBitmap([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables bitmap format. A returned value of VARIANT\_FALSE indicates that HostExplorer disables the Bitmap format.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables bitmap format. A value of VARIANT\_FALSE indicates that HostExplorer disables the Bitmap format.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.ClipFormatBitmap  
If (bVal = False) Then  
    ' Set value  
    SessEd.ClipFormatBitmap = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ClipFormatBitmap property
    VARIANT_BOOL bVal;
    pEdit->get_ClipFormatBitmap(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ClipFormatBitmap(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClipFormatCSV

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables CSV (Comma Separated Value) and BIF (Binary Interchange File) formats when copying data to the clipboard and pasting data from other applications. CSV and BI are common formats used by spreadsheet applications. When copying data to the clipboard in Cell Delimited format, HostExplorer can parse screen data at words or at field attributes. This lets you determine how data appears in cells in your spreadsheet application. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileEdit.ClipFormatCSV
```

```
HEProfileEdit.ClipFormatCSV = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_ClipFormatCSV([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileEdit::put_ClipFormatCSV([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables CSV and BIF formats. A returned value of VARIANT\_FALSE indicates that HostExplorer disables CSV and BIFF formats.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables CSV and BIF formats. A value of VARIANT\_FALSE indicates that HostExplorer disables CSV and BIFF formats.

### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
' Get value
bVal = SessEd.ClipFormatCSV
If (bVal = False) Then
    ' Set value
    SessEd.ClipFormatCSV = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ClipFormatCSV property
    VARIANT_BOOL bVal;
    pEdit->get_ClipFormatCSV(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ClipFormatCSV(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClipFormatHE

### Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables its proprietary format when copying data to the clipboard. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.ClipFormatHE
```

```
HEProfileEdit.ClipFormatHE = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_ClipFormatHE([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileEdit::put_ClipFormatHE([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables its proprietary format when copying data to the clipboard. A returned value of VARIANT\_FALSE indicates that HostExplorer disables its proprietary format when copying data to the clipboard.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables its proprietary format when copying data to the clipboard. A value of VARIANT\_FALSE indicates that HostExplorer disables its proprietary format when copying data to the clipboard.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.ClipFormatHE  
If (bVal = False) Then  
    ' Set value  
    SessEd.ClipFormatHE = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ClipFormatHE property
    VARIANT_BOOL bVal;
    pEdit->get_ClipFormatHE(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ClipFormatHE(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClipFormatPasteLink

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables Paste Link format when copying data to the clipboard. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileEdit.ClipFormatPasteLink
```

```
HEProfileEdit.ClipFormatPasteLink = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_ClipFormatPasteLink([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileEdit::put_ClipFormatPasteLink([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables Paste Link format when copying data to the clipboard. A returned value of VARIANT\_FALSE indicates that HostExplorer disables Paste Link format.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables the Paste Link format when copying data to the clipboard. A value of VARIANT\_FALSE indicates that HostExplorer disables Paste Link format.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.ClipFormatPasteLink  
If (bVal = False) Then  
    ' Set value  
    SessEd.ClipFormatPasteLink = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ClipFormatPasteLink property
    VARIANT_BOOL bVal;
    pEdit->get_ClipFormatPasteLink(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ClipFormatPasteLink(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClipFormatRTF

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables RTF (Rich Text Format) when copying data to the clipboard. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileEdit.ClipFormatRTF  
HEProfileEdit.ClipFormatRTF = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_ClipFormatRTF([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_ClipFormatRTF([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables RTF when copying data to the clipboard. A returned value of VARIANT\_FALSE indicates that HostExplorer disables RTF.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables RTF when copying data to the clipboard. A value of VARIANT\_FALSE indicates that HostExplorer disables RTF.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.ClipFormatRTF  
If (bVal = False) Then  
    ' Set value  
    SessEd.ClipFormatRTF = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ClipFormatRTF property
    VARIANT_BOOL bVal;
    pEdit->get_ClipFormatRTF(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ClipFormatRTF(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ClipFormatText

### Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables standard text format for clipboard use. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.ClipFormatText  
HEProfileEdit.ClipFormatText = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_ClipFormatText([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_ClipFormatText([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables standard text format for clipboard use. A returned value of VARIANT\_FALSE indicates that HostExplorer disables standard text format for clipboard use.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables standard text format for clipboard use. A value of VARIANT\_FALSE indicates that HostExplorer disables standard text format for clipboard use.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.ClipFormatText  
If (bVal = False) Then  
    ' Set value  
    SessEd.ClipFormatText = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ClipFormatText property
    VARIANT_BOOL bVal;
    pEdit->get_ClipFormatText(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ClipFormatText (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ConvertNulls

### Property, IHEProfileEdit 3270 5250

This property converts zeros (nulls) to blank characters in input fields when copying and pasting text. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileEdit.ConvertNulls  
HEProfileEdit.ConvertNulls = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_ConvertNulls([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_ConvertNulls([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the computer converts nulls to blank characters in input fields. A returned value of VARIANT\_FALSE indicates that the computer does not convert nulls to blank characters in input fields.

*newVal*—A value of VARIANT\_TRUE indicates that the computer converts nulls to blank characters in input fields. A value of VARIANT\_FALSE indicates that the computer does not convert nulls to blank characters in input fields.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
bVal = SessEd.ConvertNulls  
If (bVal = False) Then  
    SessEd.ConvertNulls = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_
5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileEdit interface
IHEProfileEdit* pEdit;
pIProfile->get_Edit ( &pEdit);
//Get ConvertNulls property
VARIANT_BOOL bVal;
pEdit->get_ConvertNulls(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pEdit->put_ConvertNulls(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CSVCopyEmptyFields

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer includes empty fields when copying data in cell-delimited format.

### Basic Syntax

```
Boolean = HEProfileEdit.CSVCopyEmptyFields
```

```
HEProfileEdit.CSVCopyEmptyFields = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_CSVCopyEmptyFields([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileEdit::put_CSVCopyEmptyFields([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer includes empty fields when copying cell-delimited data. If *pVal* equals VARIANT\_FALSE, HostExplorer does not copy empty fields.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer includes empty fields when copying cell-delimited data. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to copy empty fields.

### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
' Get value
bVal = SessEd.CSVCopyEmptyFields
If (bVal = False) Then
    ' Set value
    SessEd.CSVCopyEmptyFields = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get CSVCopyEmptyFields property
    VARIANT_BOOL bVal;
    pEdit->get_CSVCopyEmptyFields(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_CSVCopyEmptyFields(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CSVTrimFields

### Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer removes leading and trailing blanks when copying or pasting data in cell-delimited format.

#### Basic Syntax

```
Boolean = HEProfileEdit.CSVTrimFields
```

```
HEProfileEdit.CSVTrimFields = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_CSVTrimFields([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileEdit::put_CSVTrimFields([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer removes leading and trailing blanks when copying or pasting cell-delimited data. If *pVal* equals VARIANT\_FALSE, HostExplorer does not remove blanks.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer removes leading and trailing blanks when copying or pasting cell-delimited data. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to remove blanks.

#### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
' Get value
bVal = SessEd.CSVTrimFields
If (bVal = False) Then
    ' Set value
    SessEd.CSVTrimFields = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get CSVTrimFields property
    VARIANT_BOOL bVal;
    pEdit->get_CSVTrimFields(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_CSVTrimFields (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CutChar

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating how HostExplorer replaces the field-attribute character when you use the Cut application.

#### Basic Syntax

```
Integer = HEProfileEdit.CutChar
```

```
HEProfileEdit.CutChar = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_CutChar([out, retval] short *pVal);
```

```
HRESULT IHEProfileEdit::put_CutChar([in] short newVal);
```

#### Parameters

*pVal*—One of the following returned values, which indicate how HostExplorer replaces the field-attribute character:

0—None

1—Tab

2—Comma

3—Paragraph

*newVal*—The set value, which indicates how HostExplorer replaces the field-attribute character.

#### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
iVal = SessEd.CutChar
If (iVal = 0) Then
    SessEd.CutChar = 2
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( & pEdit);
    //Get CutChar property
    short sVal;
    pEdit->get_CutChar(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pEdit->put_CutChar(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CutMode

### Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating how HostExplorer removes selected text from unprotected areas of the screen.

#### Basic Syntax

```
Integer = HEProfileEdit.CutMode
```

```
HEProfileEdit.CutMode = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_CutMode([out, retval] short *pVal);  
HRESULT IHEProfileEdit::put_CutMode([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify how the selected text is removed:

0—Replace with Spaces

1—Replace with Nulls

2—Delete Text

*newVal*—The set value, which indicates how the selected text is removed.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
iVal = SessEd.CutMode  
If (iVal = 0) Then  
    SessEd.CutMode = 2  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( & pEdit);
    //Get CutMode property
    short sVal;
    pEdit->get_CutMode(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pEdit->put_CutMode(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EntryAssist

### Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables Entry Assist. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileEdit.EntryAssist  
HEProfileEdit.EntryAssist = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_EntryAssist([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_EntryAssist([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables Entry Assist. A returned value of VARIANT\_FALSE indicates that HostExplorer disables Entry Assist.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables Entry Assist. A value of VARIANT\_FALSE indicates that HostExplorer disables Entry Assist.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.EntryAssist  
If (bVal = False) Then  
    ' Set value  
    SessEd.EntryAssist = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get EntryAssist property
    VARIANT_BOOL bVal;
    pEdit->get_EntryAssist(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_EntryAssist(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## InsertResetByAttn

### Property, IHEProfileEdit 3270

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

**Note:** When this property is set to VARIANT\_TRUE, the Insert key remains on until you press an action key.

#### Basic Syntax

```
Boolean = HEProfileEdit.InsertResetByAttn
```

```
HEProfileEdit.InsertResetByAttn = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_InsertResetByAttn([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileEdit::put_InsertResetByAttn([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Insert key is toggled on. A returned value of VARIANT\_FALSE indicates that the Insert key remains toggled on until you press the Reset key.

*newVal*—A value of VARIANT\_TRUE indicates that the Insert key is toggled on. A value of VARIANT\_FALSE indicates that the Insert key remains toggled on until you press the Reset key.

#### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
bVal = SessEd.InsertResetByAttn
If (bVal = False) Then
    SessEd.InsertResetByAttn = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit );
    //Get InsertResetByAttn property
    VARIANT_BOOL bVal;
    pEdit->get_InsertResetByAttn(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_InsertResetByAttn(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LeftMargin

Property, IHEProfileEdit 3270 5250

This property returns or sets a value specifying the left margin of the screen.

### Basic Syntax

```
Integer = HEProfileEdit.LeftMargin  
HEProfileEdit.LeftMargin = Integer
```

### C++ Syntax

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

### Parameters

*pVal*—The returned value, specifying the left margin of the screen.

*newVal*—The set value, specifying the left margin of the screen.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
iVal = SessEd.LeftMargin  
If (iVal = 0) Then  
    ' Set value  
    SessEd.LeftMargin = 7  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get LeftMargin property
    short sVal;
    pEdit->get_LeftMargin(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pEdit->put_LeftMargin(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MoveCursorAfterPaste

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically repositions the cursor after pasting text. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileEdit.MoveCursorAfterPaste  
HEProfileEdit.MoveCursorAfterPaste = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_MoveCursorAfterPaste([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileEdit::put_MoveCursorAfterPaste([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically repositions the cursor after pasting text. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically reposition the cursor after pasting text.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically repositions the cursor after pasting text. A value of VARIANT\_FALSE indicates that HostExplorer does not automatically reposition the cursor after pasting text.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.MoveCursorAfterPaste  
If (bVal = False) Then  
    ' Set value  
    SessEd.MoveCursorAfterPaste = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get MoveCursorAfterPaste property
    VARIANT_BOOL bVal;
    pEdit->get_MoveCursorAfterPaste(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_MoveCursorAfterPaste(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MultilineDelete

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating whether the Delete and Backspace keys remove characters from the current and subsequent lines. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileEdit.MultilineDelete
```

```
HEProfileEdit.MultilineDelete = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_MultilineDelete([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileEdit::put_MultilineDelete([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Delete and Backspace keys remove characters from the current and subsequent lines. A returned value of VARIANT\_FALSE indicates that the Delete and Backspace keys remove characters only from the current line.

*newVal*—A value of VARIANT\_TRUE indicates that the Delete and Backspace keys remove characters from the current and subsequent lines. A value of VARIANT\_FALSE indicates that the Delete and Backspace keys remove characters only from the current line.

#### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
bVal = SessEd.MultilineDelete
If (bVal = False) Then
    SessEd.MultilineDelete = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get MultiLineDelete property
    VARIANT_BOOL bVal;
    pEdit->get_MultiLineDelete(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_MultiLineDelete(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MultilineInsert

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating whether the Insert key inserts characters on the current and subsequent lines. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.MultilineInsert
```

```
HEProfileEdit.MultilineInsert = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_MultilineInsert([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileEdit::put_MultilineInsert([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Insert key inserts characters on the current and subsequent lines. A returned value of VARIANT\_FALSE indicates that the Insert key inserts characters and stops when the cursor reaches the end of the current line.

*newVal*—A value of VARIANT\_TRUE indicates that the Insert key inserts characters on the current and subsequent lines. A value of VARIANT\_FALSE indicates that the Insert key inserts characters and stops when the cursor reaches the end of the current line.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
bVal = SessEd.MultilineInsert  
If (bVal = False) Then  
    SessEd.MultilineInsert = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit );
    //Get MultiLineInsert property
    VARIANT_BOOL bVal;
    pEdit->get_MultiLineInsert(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_MultiLineInsert(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## NoLockKeyb

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating whether HostExplorer sends a Never Lock the Keyboard command to the host. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.NoLockKeyb
```

```
HEProfileEdit.NoLockKeyb = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_NoLockKeyb([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfileEdit::put_NoLockKeyb([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer sends a Never Lock The Keyboard command to the host. A returned value of VARIANT\_FALSE indicates that HostExplorer does not send a Never Lock The Keyboard command to the host.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer sends a Never Lock The Keyboard command to the host. A value of VARIANT\_FALSE indicates that HostExplorer does not send a Never Lock The Keyboard command to the host.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
bVal = SessEd.NoLockKeyb  
If (bVal = False) Then  
    SessEd.NoLockKeyb = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get NoLockKeyb property
    VARIANT_BOOL bVal;
    pEdit->get_NoLockKeyb(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_NoLockKeyb(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## NumericCharacters

### Property, IHEProfileEdit 3270

This property returns or sets a string that specifies which characters are valid numeric characters.

#### Basic Syntax

```
String = HEProfileEdit.NumericCharacters  
HEProfileEdit.NumericCharacters = String
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_NumericCharacters([out, retval] BSTR *pVal);  
HRESULT IHEProfileEdit::put_NumericCharacters([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string that lists the valid numeric characters.

*newVal*—The string that specifies the numeric characters you want to set as valid.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim str As String  
Dim posn As Long  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
str = SessEd.NumericCharacters  
posn = InStr(1, str, ".", 1)  
If (posn = 0) Then  
    SessEd.NumericCharacters = str + "."  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get NumericCharacters property
    BSTR bstr;
    pEdit->get_NumericCharacters(&bstr);
    // Add code . . .
    // Set the list of valid characters
    if (bstr != NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("0123456789.-"));
    pTerm->put_ValidCharacters(bstr);
    SysFreeString(bstr);
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PasteChar

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating how HostExplorer replaces the field-attribute character when you use the Paste application.

#### Basic Syntax

```
Integer = HEProfileEdit.PasteChar
```

```
HEProfileEdit.PasteChar = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_PasteChar([out, retval] short *pVal);
```

```
HRESULT IHEProfileEdit::put_PasteChar([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify how HostExplorer replaces the field-attribute character:

0—None

1—Tab

2—Comma

3—Paragraph

*newVal*—The set value, which indicates how HostExplorer replaces the field-attribute character.

#### Basic Example

```
Dim Profile As HEPProfile
Dim Edit As HEProfileEdit
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
iVal = SessEd.PasteChar
If (iVal = 0) Then
    SessEd.PasteChar = 2
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get PasteChar property
    short sVal;
    pEdit->get_PasteChar(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pEdit->put_PasteChar(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PasteMode

### Property, IHEProfileEdit 3270 5250

This property lets you determine how HostExplorer pastes the contents of the clipboard to the current cursor location.

#### Basic Syntax

```
Integer = HEProfileEdit.PasteMode
```

```
HEProfileEdit.PasteMode = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_PasteMode([out, retval] short *pVal);
```

```
HRESULT IHEProfileEdit::put_PasteMode([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify how HostExplorer pastes the contents of the clipboard to the current cursor location:

0—Replace with Spaces

1—Replace with Nulls

2—Delete Text

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

#### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
iVal = SessEd.PasteMode
If (iVal = 0) Then
    SessEd.PasteMode = 2
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get PasteMode property
    short sVal;
    pEdit->get_PasteMode(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pEdit->put_PasteMode(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## RemoveTrailingBlankOnCopy

Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer removes all blank characters at the end of the text when copying text. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileEdit.RemoveTrailingBlankOnCopy
```

```
HEProfileEdit.RemoveTrailingBlankOnCopy = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_RemoveTrailingBlankOnCopy([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfileEdit::put_RemoveTrailingBlankOnCopy([in] VARIANT_BOOL
newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer removes trailing blanks when copying text. If *pVal* equals VARIANT\_FALSE, HostExplorer does not remove trailing blanks.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer removes trailing blanks when copying text. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to remove trailing blanks.

### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.RemoveTrailingBlankOnCopy  
If (bVal = False) Then  
    ' Set value  
    SessEd.RemoveTrailingBlankOnCopy = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit );
    //Get RemoveTrailingBlankOnCopy property
    VARIANT_BOOL bVal;
    pEdit->get_RemoveTrailingBlankOnCopy(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_RemoveTrailingBlankOnCopy(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ResetMDTOnEraseInput

### Property, IHEProfileEdit 5250

This property returns or sets a value that specifies whether HostExplorer resets the modify data tag (MDT). By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.ResetMDTOnEraseInput  
HEProfileEdit.ResetMDTOnEraseInput = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_ResetMDTOnEraseInput([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileEdit::put_ResetMDTOnEraseInput([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer resets the modify data tag. If *pVal* equals VARIANT\_FALSE, HostExplorer does not reset the modify data tag.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer resets the modify data tag; otherwise, set *newVal* to VARIANT\_FALSE.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.ResetMDTOnEraseInput  
If (bVal = False) Then  
    ' Set value  
    SessEd.ResetMDTOnEraseInput = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get ResetMDTOnEraseInput property
    VARIANT_BOOL bVal;
    pEdit->get_ResetMDTOnEraseInput(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_ResetMDTOnEraseInput(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## RespectNumeric

### Property, IHEProfileEdit 3270

This property returns or sets a value indicating whether HostExplorer forces data entry on numeric fields to be validated. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileEdit.RespectNumeric  
HEProfileEdit.RespectNumeric = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_RespectNumeric([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEdit::put_RespectNumeric([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer forces data entry on numeric fields to be validated. A returned value of VARIANT\_FALSE indicates that HostExplorer does not force data entry on numeric fields to be validated.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer forces data entry on numeric fields to be validated. A value of VARIANT\_FALSE indicates that HostExplorer does not force data entry on numeric fields to be validated.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
bVal = SessEd.RespectNumeric  
If (bVal = False) Then  
    SessEd.RespectNumeric = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit );
    //Get RespectNumeric property
    VARIANT_BOOL bVal;
    pEdit->get_RespectNumeric(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_RespectNumeric(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## RightMargin

Property, IHEProfileEdit [3270 5250](#)

This property returns or sets a value specifying the right margin of the screen.

### Basic Syntax

```
Integer = HEProfileEdit.RightMargin
```

```
HEProfileEdit.RightMargin = Integer
```

### C++ Syntax

```
HRESULT IHEProfileEdit::get_RightMargin([out, retval] short *pVal);
```

```
HRESULT IHEProfileEdit::put_RightMargin([in] short newVal);
```

### Parameters

*pVal*—The returned value specifying the right margin of the screen.

*newVal*—The set value specifying the right margin of the screen.

### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
' Get value
iVal = SessEd.RightMargin
If (iVal = 0) Then
    ' Set value
    SessEd.RightMargin = 7
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get RightMargin property
    short sVal;
    pEdit->get_RightMargin(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pEdit->put_RightMargin(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SmartInsert

### Property, IHEProfileEdit 3270 5250 VT

This property returns or sets a value that specifies whether you can insert characters in a field that contains nulls, spaces, or both at the end of the field. By default, this option is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.SmartInsert
```

```
HEProfileEdit.SmartInsert = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_SmartInsert([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileEdit::put_SmartInsert([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, you can insert characters in a field that contains nulls, spaces, or both at the end of the field. If *pVal* equals VARIANT\_FALSE, you can insert characters in a field that contains only nulls at the end.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to insert characters in a field that contains nulls, spaces, or both at the end of the field. Set *newVal* to insert characters in a field that contains only nulls at the end.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.SmartInsert  
If (bVal = False) Then  
    ' Set value  
    SessEd.SmartInsert = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get SmartInsert property
    VARIANT_BOOL bVal;
    pEdit->get_SmartInsert(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_SmartInsert(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WordWrap

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating whether HostExplorer cuts text upon reaching the end of a field or wraps text to the next available field. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileEdit.WordWrap
```

```
HEProfileEdit.WordWrap = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_WordWrap([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileEdit::put_WordWrap([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer wraps text to the next available field. A returned value of VARIANT\_FALSE indicates that HostExplorer cuts text when it reaches the end of a field.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer wraps text to the next available field. A value of VARIANT\_FALSE indicates that HostExplorer cuts text when it reaches the end of a field.

#### Basic Example

```
Dim Profile As HEProfile
Dim Edit As HEProfileEdit
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Edit = Profile.Edit
' Get value
bVal = SessEd.WordWrap
If (bVal = False) Then
    ' Set value
    SessEd.WordWrap = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get WordWrap property
    VARIANT_BOOL bVal;
    pEdit->get_WordWrap(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_WordWrap(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WordWrapWithNewLineReturn

### Property, IHEProfileEdit 3270 5250

This property returns or sets a value indicating whether HostExplorer—upon reaching the end of a field—truncates text or wraps text to the next available field and inserts a carriage return. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileEdit.WordWrapWithNewLineReturn  
HEProfileEdit.WordWrapWithNewLineReturn = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEdit::get_WordWrapWithNewLineReturn([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileEdit::put_WordWrapWithNewLineReturn([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer wraps text to the next available field and inserts a carriage return when it encounters the end of a field. If *pVal* equals VARIANT\_FALSE, HostExplorer truncates text when it encounters the end of a field.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer wraps text to the next available field and inserts a carriage return when it encounters the end of a field. Set *newVal* to VARIANT\_FALSE if you want HostExplorer to truncate text at the end of a field.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Edit As HEProfileEdit  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Edit = Profile.Edit  
' Get value  
bVal = SessEd.WordWrapWithNewLineReturn  
If (bVal = False) Then  
    ' Set value  
    SessEd.WordWrapWithNewLineReturn = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEdit interface
    IHEProfileEdit* pEdit;
    pIProfile->get_Edit ( &pEdit);
    //Get WordWrapWithNewLineReturn property
    VARIANT_BOOL bVal;
    pEdit->get_WordWrapWithNewLineReturn(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEdit->put_WordWrapWithNewLineReturn(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileColor Interface

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

### Methods

The ProfileColor interface contains the following method:

SystemColor

### Properties

The ProfileColor interface consists of the following property:

Schemes

### SystemColor

#### Method, IHEProfileColor 3270 5250 VT

This method lets you specify a customized color for a particular system color block. You can specify the color in terms of a percentage for each of its red, blue, and green components.

#### Basic Syntax

```
HEProfileColor.SystemColor(iColor As Integer, iRed As Integer, iGreen As Integer, iBlue As Integer)
```

#### C++ Syntax

```
HRESULT IHEProfileColor::SystemColor([in] short iColor, [in] short iRed, [in] short iGreen, [in] short iBlue);
```

#### Parameters

*iColor*—The index number of the color block you want to change. *iColor* can range from 1 to 16.

*iRed*—The percentage of red in the new color. *iRed* can range from 0 to 100.

*iGreen*—The percentage of green in the new color. *iGreen* can range from 0 to 100.

*iBlue*—The percentage of blue in the new color. *iBlue* can range from 0 to 100.

#### Basic Example

```
Dim color As HEProfileColor
Set color = Profile.Color
' Specify the color for color block #5
color.SystemColor(5, 60, 56, 80)
```

#### C++ Example

```
IHEProfileColor *pColor;
pSess->get_Color(&pColor);
// Specify the color for color block #5
pColor->SystemColor(5, 60, 56, 80);
```

## Schemes

### Property, IHEProfileColor 3270 5250 VT

This property returns or sets a string that specifies the name of the color scheme for the current session. A scheme is a collection of settings.

#### Basic Syntax

```
String = HEProfileColor.Schemes  
HEProfileColor.Schemes = String
```

#### C++ Syntax

```
HRESULT IHEProfileColor::get_Schemes([out, retval] BSTR *pVal);  
HRESULT IHEProfileColor::put_Schemes([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, which indicates the color scheme.  
*newVal*—The set value, which indicates the color scheme.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Color As HEProfileColor  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Color = Profile.Color  
' Get value  
str = color.Schemes  
If (Len(str) = 0) Then  
    ' Set value  
    color.Schemes = "ATM-Saint_Louis_Blues"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileColor interface
IHEProfileColor* pColor;
pIProfile->get_Color ( &pColor);
//Get Schemes property
BSTR bstr;
pColor->get_Schemes(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("ATM-Saint_Louis_Blues"));
    pColor->put_Schemes(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileFileTransfer Interface

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

## Properties

The ProfileFileTransfer interface consists of the following properties:

- Append
- ASCII
- AutoCC
- AutoClearMonitor
- BlkSize
- CRLF
- CustomTransferTable
- DefaultDownloadPath
- DefaultProtocol
- DefaultRecvDir
- DefaultUploadPath
- DownloadHostName
- DownloadPCFileName
- DownloadTranslate
- ExtraOptions
- FileExistAction
- Host
- INDFILEName
- KmBinaryPrefix
- KmRLE
- KmTextMode
- KmUseFullPath
- Lrec1
- QuickMode
- Recfm
- Schemes
- ShowRecvDlg
- UploadHostName
- UploadPCFileName
- UploadTranslate
- UserDefinedDownload
- UserDefinedUpload
- VTAutoClearMonitor
- XferBlockSize
- XferDest
- XferHostCodePage
- XferPCCodePage
- XferSource
- XferStartAction
- Xm1KPacket
- XmAckTimeout
- XmCRC
- YmAckTimeout
- YmUseFullPath
- ZmAutoDownload
- ZmCrashRecovery
- ZmMaxErrors
- ZmOverwriteMngmt
- ZmSlideWindow
- ZmUseFullPath
- ZmWindowSize

## Append

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer appends a file to an existing file on the host.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.Append
```

```
HEProfileFileTransfer.Append = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Append([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileFileTransfer::put_Append([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer appends the file. A returned value of VARIANT\_FALSE indicates that HostExplorer does not append the file.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer appends the file. A value of VARIANT\_FALSE indicates that HostExplorer does not append the file.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
bVal = SessXfr.Append
If (bVal = False) Then
    ' Set value
    SessXfr.Append = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get Append property
    VARIANT_BOOL bVal;
    pXfr->get_Append(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_Append(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ASCII

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer translates an ASCII (a character set used on a personal computer) file to an EBCDIC (an IBM host character set) file.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.Ascii
```

```
HEProfileFileTransfer.Ascii = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Ascii([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileFileTransfer::put_Ascii([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer translates an ASCII file to an EBCDIC file. A returned value of VARIANT\_FALSE indicates that HostExplorer does not translate an ASCII file to an EBCDIC file.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer translates an ASCII file to an EBCDIC file. A value of VARIANT\_FALSE indicates that HostExplorer does not translate an ASCII file to an EBCDIC file.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.Ascii  
If (bVal = False) Then  
    ' Set value  
    SessXfr.Ascii = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get ASCII property
    VARIANT_BOOL bVal;
    pXfr->get_Ascii(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_Ascii(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoCC

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer converts carriage control on the host to carriage control on your computer. This property applies only when you download files from the host.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.AutoCC
```

```
HEProfileFileTransfer.AutoCC = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_AutoCC([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileFileTransfer::put_AutoCC([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer converts carriage control on the host to carriage control on your computer. A returned value of VARIANT\_FALSE indicates that HostExplorer does not convert carriage control on the host to carriage control on your computer.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer converts carriage control on the host to carriage control on your computer. A value of VARIANT\_FALSE indicates that HostExplorer does not convert carriage control on the host to carriage control on your computer.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.AutoCC  
If (bVal = False) Then  
    ' Set value  
    SessXfr.AutoCC = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get AutoCC property
    VARIANT_BOOL bVal;
    pXfr->get_AutoCC(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_AutoCC(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoClearMonitor

Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically exits the File Transfer Monitor when it has finished transferring a file.

### Basic Syntax

```
Boolean = HEProfileFileTransfer.AutoClearMonitor
```

```
HEProfileFileTransfer.AutoClearMonitor = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_AutoClearMonitor([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_AutoClearMonitor([in] VARIANT_BOOL
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically exits the File Transfer Monitor. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically exit the File Transfer Monitor.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically exits the File Transfer Monitor. A value of VARIANT\_FALSE indicates that HostExplorer does not automatically exit the File Transfer Monitor.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.AutoClearMonitor  
If (bVal = False) Then  
    ' Set value  
    SessXfr.AutoClearMonitor = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer );
    //Get AutoClearMonitor property
    VARIANT_BOOL bVal;
    pXfr->get_AutoClearMonitor(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_AutoClearMonitor(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## BlkSize

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating the block size that HostExplorer uses when it transfers files. The size of the block ranges from 256 bytes to 32768 bytes. By default, this property is set to 2048 bytes.

#### Basic Syntax

```
Long = HEProfileFileTransfer.BlkSize
```

```
HEProfileFileTransfer.BlkSize = Long
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_BlkSize([out, retval] long *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_BlkSize([in] long newVal);
```

#### Parameters

*pVal*—The returned value, which indicates the block size of a file.

*newVal*—The set value, which indicates the block size of a file.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim lVal As Long
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
lVal = SessXfr.BlkSize
If (lVal = 256) Then
    ' Set value
    SessXfr.BlkSize = 1024
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEPProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEPProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get BlkSize property
long lVal;
pXfr->get_BlkSize(&lVal);
if (lVal == 256)
{
    lVal = 1024;
    pXfr->put_BlkSize(lVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CRLF

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating whether HostExplorer translates CR/LF (carriage return/line feed) characters to records on the host file system. This property is normally required when you transfer text files.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.CRLF
```

```
HEProfileFileTransfer.CRLF = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_CRLF([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_CRLF([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer translates CR/LF characters. A returned value of VARIANT\_FALSE indicates that HostExplorer does not translate CR/LF characters.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer translates CR/LF characters. A value of VARIANT\_FALSE indicates that HostExplorer does not translate CR/LF characters.

#### Basic Example

```
Dim Profile As HEPProfile
Dim FileTransfer As HEProfileFileTransfer
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
bVal = SessXfr.CRLF
If (bVal = False) Then
    ' Set value
    SessXfr.CRLF = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get CRLF property
    VARIANT_BOOL bVal;
    pXfr->get_CRLF(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_CRLF(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CustomTransferTable

### Property, IHEProfileFileTransfer 3270

This property returns or sets a string that specifies the full path and file name of the custom transfer table. The custom transfer table is an .ini file that lists the translation settings that HostExplorer uses when transferring data between the host and PC.

#### Basic Syntax

```
String = HEProfileFileTransfer.CustomTransferTable
```

```
HEProfileFileTransfer.CustomTransferTable = String
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_CustomTransferTable([out, retval]  
BSTR *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_CustomTransferTable([in] BSTR  
newVal);
```

#### Parameters

*pVal*—The returned string, specifying the path and file name of the custom transfer table.

*newVal*—The set string, specifying the path and file name of the custom transfer table that you want HostExplorer to use.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.CustomTransferTable  
If (Len(strVal) = 0) Then  
    SessXfr.CustomTransferTable = "C:\Mydata\cxfer.ini"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get CustomTransferTable property
BSTR bstr;
pXfr->get_CustomTransferTable(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr != NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\\\Mydata\\\\cxfer.ini"));
    pXfr->put_CustomTransferTable(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DefaultDownloadPath

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying the default directory for downloaded files.

### Basic Syntax

```
String = HEProfileFileTransfer.DefaultDownloadPath  
HEProfileFileTransfer.DefaultDownloadPath = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DefaultDownloadPath([out, retval]  
BSTR *pVal);  
HRESULT IHEProfileFileTransfer::put_DefaultDownloadPath([in] BSTR  
newVal);
```

### Parameters

*pVal*—The returned string, specifying the default download directory.

*newVal*—The set string, specifying the default download directory.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.DefaultDownloadPath  
If (Len(strVal) = 0) Then  
    SessXfr.DefaultDownloadPath = "C:\\"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get DefaultDownloadPath property
BSTR bstr;
pXfr->get_DefaultDownloadPath(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\\\"));
    pXfr->put_DefaultDownloadPath(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DefaultProtocol

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating the default file-transfer protocol.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.DefaultProtocol  
HEProfileFileTransfer.DefaultProtocol = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DefaultProtocol([out, retval] short  
*pVal);  
HRESULT IHEProfileFileTransfer::put_DefaultProtocol([in] short newVal);
```

#### Parameters

*pVal*—The following returned values indicate the default file-transfer protocol:

- 0—Xmodem
- 1—Ymodem
- 2—Kermit
- 3—Zmodem

*newVal*—The set value, which indicates the default file-transfer protocol.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
iVal = SessXfr.DefaultProtocol  
If (iVal=2) Then  
    ' Set value  
    SessXfr.DefaultProtocol = 3  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get DefaultProtocol property
    short sVal;
    pXfr->get_DefaultProtocol(&sVal);
    if (sVal == 2)
    {
        sVal = 3;
        pXfr->put_DefaultProtocol(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DefaultRecvDir

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string indicating the default directory for received files.

### Basic Syntax

```
String = HEProfileFileTransfer.DefaultRecvDir  
HEProfileFileTransfer.DefaultRecvDir = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DefaultRecvDir([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_DefaultRecvDir([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, which indicates the default receive directory.

*newVal*—The set string, which indicates the default receive directory.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.DefaultRecvDir  
If (Len(strVal) = 0) Then  
    SessXfr.DefaultRecvDir = "C:\\"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get DefaultRecvDir property
BSTR bstr;
pXfr->get_DefaultRecvDir(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\\\"));
    pXfr->put_DefaultRecvDir(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DefaultUploadPath

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying the default directory for uploaded files.

### Basic Syntax

```
String = HEProfileFileTransfer.DefaultUploadPath  
HEProfileFileTransfer.DefaultUploadPath = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DefaultUploadPath([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_DefaultUploadPath([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the default upload directory.

*newVal*—The set string, specifying the default upload directory.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.DefaultUploadPath  
If (Len(strVal) = 0) Then  
    SessXfr.DefaultUploadPath = "C:\\"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get DefaultUploadPath property
BSTR bstr;
pXfr->get_DefaultUploadPath(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\\\"));
    pXfr->put_DefaultUploadPath(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DownloadHostName

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying the name of a host file to download to your computer.

### Basic Syntax

```
String = HEProfileFileTransfer.DownloadHostName  
HEProfileFileTransfer.DownloadHostName = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DownloadHostName([out, retval]  
BSTR *pVal);  
HRESULT IHEProfileFileTransfer::put_DownloadHostName([in] BSTR  
newVal);
```

### Parameters

*pVal*—The returned string specifying the name of a host file to download.  
*newVal*—The set string specifying the name of a host file to download.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.DownloadHostName  
If (Len(strVal) = 0) Then  
    SessXfr.DownloadHostName = "C:\a.txt"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get DownloadHostName property
BSTR bstr;
pXfr->get_DownloadHostName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\a.txt"));
    pXfr->put_DownloadHostName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DownloadPCFileName

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying the file name to which HostExplorer saves a downloaded file on your computer.

### Basic Syntax

```
String = HEProfileFileTransfer.DownloadPCFileName  
HEProfileFileTransfer.DownloadPCFileName = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DownloadPCFileName([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_DownloadPCFileName([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the file name to which HostExplorer saves a downloaded file.

*newVal*—The set string, specifying the file name to which HostExplorer saves a downloaded file.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.DownloadPCFileName  
If (Len(strVal) = 0) Then  
    SessXfr.DownloadPCFileName = "C:\b.txt"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get DownloadPCFileName property
BSTR bstr;
pXfr->get_DownloadPCFileName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\b.txt"));
    pXfr->put_DownloadPCFileName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DownloadTranslate

### Property, IHEProfileFileTransfer 3270

This property returns or sets the Download section of the Custom Transfer Table. The Download section specifies the Host-to-PC translation settings in the following form:

*hx=px*

where *hx* is a host value in hexadecimal format and *px* is the same value on the PC in hexadecimal format.

The Download section must define translations for all values between 00 and FF inclusive; use a semicolon to separate successive translations.

#### Basic Syntax

```
String = HEProfileFileTransfer.DownloadTranslate
```

```
HEProfileFileTransfer.DownloadTranslate = String
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_DownloadTranslate([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_DownloadTranslate([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the Download section of the Custom Transfer Table.

*newVal*—The set string, specifying the Download translation settings that you want HostExplorer to use.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim strVal As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
strVal = SessXfr.DownloadTranslate
If (Len(strVal) = 0) Then
    SessXfr.DownloadTranslate = "F2=32"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get DownloadTranslate property
BSTR bstr;
pXfr->get_DownloadTranslate(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr != NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("F2=32"));
    pXfr->put_DownloadTranslate(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ExtraOptions

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying options that are specific to the operating system. You must specify these custom options in the appropriate format.

### Basic Syntax

```
String = HEProfileFileTransfer.ExtraOptions  
HEProfileFileTransfer.ExtraOptions = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ExtraOptions([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_ExtraOptions([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying options that are specific to the operating system.

*newVal*—The set string, specifying options that are specific to the operating system.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.ExtraOptions  
If (Len(strVal) = 0) Then  
    ' APND is a parameter that the host  
    ' application is customized to process  
    SessXfr.ExtraOptions = "APND"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get ExtraOptions property
    BSTR bstr;
    pXfr->get_ExtraOptions(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("APND"));
        pXfr->put_ExtraOptions(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## FileExistAction

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying what action HostExplorer performs if the name of a downloaded file already exists in the destination directory.

### Basic Syntax

```
Integer = HEProfileFileTransfer.FileExistAction  
HEProfileFileTransfer.FileExistAction = Integer
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_FileExistAction([out, retval] short  
*pVal);  
HRESULT IHEProfileFileTransfer::put_FileExistAction([in] short newVal);
```

### Parameters

*pVal*—One of the following returned values, which indicates the action to perform if the name of a downloaded file already exists in the destination directory:

0—Overwrite

1—Rename

2—Skip

*newVal*—The set value, which indicates the action to perform if the name of a downloaded file already exists in the destination directory.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
iVal = SessXfr.FileExistAction  
If (iVal=2) Then  
    ' Set value  
    SessXfr.FileExistAction = 0  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get FileExistAction property
    short sVal;
    pXfr->get_FileExistAction(&sVal);
    if (sVal == 2)
    {
        sVal = 0;
        pXfr->put_FileExistAction(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Host

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating the operating system run by the host.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.Host
```

```
HEProfileFileTransfer.Host = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Host([out, retval] short *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_Host([in] short newVal);
```

#### Parameters

*pVal*—One of the following returned values, indicating the address of the current host:

257—CMS

258—TSO/MUSIC

259—CICS

*newVal*—The address that you set for the current host.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.Host
If (iVal=257) Then
    ' Set value
    SessXfr.Host = 259
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get Host property
    short sVal;
    pXfr->get_Host(&sVal);
    if (sVal == 257)
    {
        sVal = 259;
        pXfr->put_Host(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## INDFILENAME

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a string specifying the name of the file-transfer program to use when uploading and/or downloading files.

#### Basic Syntax

```
String = HEProfileFileTransfer.INDFILENAME
```

```
HEProfileFileTransfer.INDFILENAME = String
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_INDFILENAME([out, retval] BSTR  
*pVal);  
  
HRESULT IHEProfileFileTransfer::put_INDFILENAME([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the name of the file-transfer program.

*newVal*—The set string, specifying the name of the file-transfer program.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
str = SessXfr.INDFILENAME  
If (Len(str) = 0) Then  
    ' Set value  
    SessXfr.INDFILENAME = "IND$FILE"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get INDFILEName property
BSTR bstr;
pXfr->get_INDFILEName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr=SysAllocString(OLESTR("IND$FILE"));
    pXfr->put_INDFILEName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KmBinaryPrefix

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer attempts to send 8-bit data characters over a 7-bit channel by prefixing non-printable characters (that is, using a binary prefix).

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.KmBinaryPrefix  
HEProfileFileTransfer.KmBinaryPrefix = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_KmBinaryPrefix([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_KmBinaryPrefix([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer attempts to send 8-bit data characters over a 7-bit channel. A returned value of VARIANT\_FALSE indicates that HostExplorer does not attempt to send 8-bit characters over a 7-bit channel.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer attempts to send 8-bit data characters over a 7-bit channel. A value of VARIANT\_FALSE indicates that HostExplorer does not attempt to send 8-bit characters over a 7-bit channel.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.KmBinaryPrefix  
If (bVal = False) Then  
    ' Set value  
    SessXfr.KmBinaryPrefix= True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get KmBinaryPrefix property
VARIANT_BOOL bVal;
pXfr->get_KmBinaryPrefix(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_KmBinaryPrefix(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KmRLE

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer uses run-length limited encoding (RLL) to compress data, thereby transferring files more efficiently. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.KmRLE
```

```
HEProfileFileTransfer.KmRLE = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_KmRLE([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileFileTransfer::put_KmRLE([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer uses RLL. A returned value of VARIANT\_FALSE indicates that HostExplorer does not use RLL.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer uses RLL. A value of VARIANT\_FALSE indicates that HostExplorer does not use RLL.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
bVal = SessXfr.KmRLE
If (bVal = False) Then
    ' Set value
    SessXfr.KmRLE = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get KmRLE property
    VARIANT_BOOL bVal;
    pXfr->get_KmRLE(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_KmRLE(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KmTextMode

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer strips the upper bit of each byte as it is received, thus preventing any non-ASCII characters from being saved.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.KmTextMode
```

```
HEProfileFileTransfer.KmTextMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_KmTextMode([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_KmTextMode([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer strips the upper bit of each byte and saves only ASCII characters. A returned value of VARIANT\_FALSE indicates that HostExplorer does not strip the upper bit of each byte and saves non-ASCII characters.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer strips the upper bit of each byte and saves only ASCII characters. A value of VARIANT\_FALSE indicates that HostExplorer does not strip the upper bit of each byte and saves non-ASCII characters.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
bVal = SessXfr.KmTextMode
If (bVal = False) Then
    ' Set value
    SessXfr.KmTextMode = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get KmTextMode property
VARIANT_BOOL bVal;
pXfr->get_KmTextMode(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_KmTextMode(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KmUseFullPath

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer saves a file to the current UNIX host directory. This property also lets you keep the entire path name as the file name.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.KmUseFullPath  
HEProfileFileTransfer.KmUseFullPath = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_KmUseFullPath([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_KmUseFullPath([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A returned value of VARIANT\_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory, and does not keep the entire pathname as the file name.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A value of VARIANT\_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory, and does not keep the entire pathname as the file name.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.KmUseFullPath  
If (bVal = False) Then  
    ' Set value  
    SessXfr.KmUseFullPath = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get KmUseFullPath property
VARIANT_BOOL bVal;
pXfr->get_KmUseFullPath(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_KmUseFullPath(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Lrecl

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating the logical record size of a file that you send to the host.

#### Basic Syntax

```
Long = HEProfileFileTransfer.Lrecl
```

```
HEProfileFileTransfer.Lrecl = Long
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Lrecl([out, retval] long *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_Lrecl([in] long newVal);
```

#### Parameters

*pVal*—The returned value, which indicates the logical record size of a file.

*newVal*—The set value, indicating the logical record size of a file.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.Lrecl
If (iVal=0) Then
    ' Set value
    SessXfr.Lrecl = 5
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get Lrecl property
    short sVal;
    pXfr->get_Lrecl(&sVal);
    if (sVal == 0)
    {
        sVal = 5;
        pXfr->put_Lrecl(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## QuickMode

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating the file-transfer mode (text, binary, or custom).

#### Basic Syntax

```
Integer = HEProfileFileTransfer.QuickMode
```

```
HEProfileFileTransfer.QuickMode = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_QuickMode([out, retval] short *pVal);  
HRESULT IHEProfileFileTransfer::put_QuickMode([in] short newVal);
```

#### Parameters

*pVal*—The following returned values indicate the file-transfer mode:

- 0—Text
- 1—Binary
- 2—Custom

*newVal*—The set value, which indicates the file-transfer mode.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
iVal = SessXfr.QuickMode  
If (iVal=0) Then  
    ' Set value  
    SessXfr.QuickMode = 2  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get QuickMode property
    short sVal;
    pXfr->get_QuickMode(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pXfr->put_QuickMode(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Recfm

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating the type of record format (default, fixed, or variable) to use when you transfer a file.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.Recfm
```

```
HEProfileFileTransfer.Recfm = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Recfm([out, retval] short *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_Recfm([in] short newVal);
```

#### Parameters

*pVal*—The following returned values indicate the type of record format:

- 253—Default
- 254—Fixed
- 255—Variable
- 256—Undefined

*newVal*—The set value, indicating the type of record format.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.Recfm
If (iVal=256) Then
    ' Set value
    SessXfr.Recfm = 253
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get Recfm property
    short sVal;
    pXfr->get_Recfm(&sVal);
    if (sVal == 256)
    {
        sVal = 253;
        pXfr->put_Recfm(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Schemes

### Property, **IHEProfileFileTransfer** [3270](#)

This property returns or sets a string that specifies the name of the file transfer scheme for the current session. A scheme is a collection of settings.

#### Basic Syntax

```
String = HEProfileFileTransfer.Schemes  
HEProfileFileTransfer.Schemes = String
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Schemes([out, retval] BSTR *pVal);  
HRESULT IHEProfileFileTransfer::put_Schemes([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, specifying the file transfer scheme.

*newVal*—The set value, specifying the file transfer scheme.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
str = SessXfr.Schemes  
If (Len(str) = 0) Then  
    ' Set value  
    SessXfr.Schemes = "CMS_Text"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get Schemes property
BSTR bstr;
pXfr->get_Schemes(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("CMS_Text"));
    pXfr->put_Schemes(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ShowRecvDlg

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a value indicating whether the Receive File dialog box opens each time you receive a file.

### Basic Syntax

```
Boolean = HEProfileFileTransfer.ShowRecvDlg
```

```
HEProfileFileTransfer.ShowRecvDlg = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ShowRecvDlg([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_ShowRecvDlg([in] VARIANT_BOOL
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Receive File dialog box opens each time you receive a file. A returned value of VARIANT\_FALSE indicates that a file begins to be transferred as soon as you select Receive File From Host from the menu. In this case, for the transfer to complete properly, you must have previously selected a default receive directory and a default protocol.

*newVal*—A value of VARIANT\_TRUE indicates that the Receive File dialog box opens each time you receive a file. A value of VARIANT\_FALSE indicates that a file begins to be transferred as soon as you select Receive File From Host from the menu.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.ShowRecvDlg  
If (bVal = False) Then  
    ' Set value  
    SessXfr.ShowRecvDlg = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get ShowRecvDlg property
    VARIANT_BOOL bVal;
    pXfr->get_ShowRecvDlg(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_ShowRecvDlg(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UploadHostName

Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a string specifying the name HostExplorer applies to a file uploaded on the host.

### Basic Syntax

```
String = HEProfileFileTransfer.UploadHostName  
HEProfileFileTransfer.UploadHostName = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_UploadHostName([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_UploadHostName([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string specifying the name HostExplorer applies to a file uploaded on the host.

*newVal*—The set string specifying the name HostExplorer applies to a file uploaded on the host.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.UploadHostName  
If (Len(strVal) = 0) Then  
    SessXfr.UploadHostName = "C:\b.txt"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get UploadHostName property
BSTR bstr;
pXfr->get_UploadHostName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\b.txt"));
    pXfr->put_UploadHostName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UploadPCFileName

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a string specifying the name HostExplorer applies to a file uploaded on your computer.

### Basic Syntax

```
String = HEProfileFileTransfer.UploadPCFileName  
HEProfileFileTransfer.UploadPCFileName = String
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_UploadPCFileName([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_UploadPCFileName([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the name HostExplorer applies to a file uploaded on your computer.

*newVal*—The set string, specifying the name HostExplorer applies to a file uploaded on your computer.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.UploadPCFileName  
If (Len(strVal) = 0) Then  
    SessXfr.UploadPCFileName = "C:\b.txt"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get UploadPCFileName property
BSTR bstr;
pXfr->get_UploadPCFileName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\b.txt"));
    pXfr->put_UploadPCFileName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UploadTranslate

### Property, IHEProfileFileTransfer 3270

This property returns or sets the Upload section of the Custom Transfer Table. The Upload section specifies the PC-to-Host translation settings in the following form:

*px=hx*

where *px* is a PC value in hexadecimal format and *hx* is the same value on the host in hexadecimal format.

The Upload section must define translations for all values between 00 and FF inclusive; use a semicolon to separate successive translations.

#### Basic Syntax

```
String = HEProfileFileTransfer.UploadTranslate  
HEProfileFileTransfer.UploadTranslate = String
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_UploadTranslate([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileFileTransfer::put_UploadTranslate([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the Upload section of the Custom Transfer Table.

*newVal*—The set string, specifying the Upload translation settings that you want HostExplorer to use.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
strVal = SessXfr.UploadTranslate  
If (Len(strVal) = 0) Then  
    SessXfr.UploadTranslate = "21=5A"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get UploadTranslate property
    BSTR bstr;
    pXfr->get_UploadTranslate(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr != NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("21=5A"));
        pXfr->put_UploadTranslate(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UserDefinedDownload

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer performs a user-defined download or a default download.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.UserDefinedDownload  
HEProfileFileTransfer.UserDefinedDownload = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_UserDefinedDownload([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_UserDefinedDownload([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates a user-defined download. A returned value of VARIANT\_FALSE indicates a default download.

*newVal*—A value of VARIANT\_TRUE indicates a user-defined download. A value of VARIANT\_FALSE indicates a default download.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.UserDefinedDownload  
If (bVal = False) Then  
    ' Set value  
    SessXfr.UserDefinedDownload = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer );
    //Get UserDefinedDownload property
    VARIANT_BOOL bVal;
    pXfr->get_UserDefinedDownload(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_UserDefinedDownload(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UserDefinedUpload

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer performs a user-defined upload or a default upload.

### Basic Syntax

```
Boolean = HEProfileFileTransfer.UserDefinedUpload  
HEProfileFileTransfer.UserDefinedUpload = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_UserDefinedUpload([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_UserDefinedUpload([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates a user-defined upload. A returned value of VARIANT\_FALSE indicates a default upload.

*newVal*—A value of VARIANT\_TRUE indicates a user-defined upload. A value of VARIANT\_FALSE indicates a default upload.

### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.UserDefinedUpload  
If (bVal = False) Then  
    ' Set value  
    SessXfr.UserDefinedUpload = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer );
    //Get UserDefinedUpload property
    VARIANT_BOOL bVal;
    pXfr->get_UserDefinedUpload(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_UserDefinedUpload(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTAutoClearMonitor

### Property, IHEProfileFileTransfer 3270 VT

This property returns or sets a value that specifies whether HostExplorer automatically exits the File Transfer Monitor upon completing a file transfer. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.VTAutoClearMonitor
```

```
HEProfileFileTransfer.VTAutoClearMonitor = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_VTAutoClearMonitor([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_VTAutoClearMonitor([in] VARIANT_BOOL
newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer automatically exits the File Transfer Monitor upon completing a file transfer. If *pVal* equals VARIANT\_FALSE, HostExplorer does not exit the File Transfer monitor.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer automatically exits the File Transfer Monitor upon completing a file transfer. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to exit the File Transfer Monitor upon completing a file transfer.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.VTAutoClearMonitor  
If (bVal = False) Then  
    ' Set value  
    SessXfr.VTAutoClearMonitor = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer );
    //Get VTAutoClearMonitor property
    VARIANT_BOOL bVal;
    pXfr->get_VTAutoClearMonitor(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pXfr->put_VTAutoClearMonitor(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XferBlockSize

### Property, IHEProfileFileTransfer 3270 5250 VT

This property returns or sets a value specifying the block size that HostExplorer uses when you transfer files. You can select a block size between 256 and 32768 bytes. By default, this property is set to 2048.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.XferBlockSize  
HEProfileFileTransfer.XferBlockSize = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XferBlockSize([out, retval] short  
*pVal);  
HRESULT IHEProfileFileTransfer::put_XferBlockSize([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the block size that HostExplorer uses when you transfer files.

*newVal*—The set value, specifying the block size that HostExplorer uses when you transfer files.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
iVal = SessXfr.XferBlockSize  
If (iVal = 256) Then  
    ' Set value  
    SessXfr.XferBlockSize = 1024  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get XferBlockSize property
    short sVal;
    pXfr->get_XferBlockSize(&sVal);
    if (sVal == 256)
    {
        sVal = 1024;
        pXfr->put_XferBlockSize(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XferDest

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating whether a file is being transferred to a file system or the clipboard.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.XferDest
```

```
HEProfileFileTransfer.XferDest = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XferDest([out, retval] short *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_XferDest([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify where a file is being transferred:

- 430—File system
- 431—Clipboard

*newVal*—The set value, specifying where a file is being transferred.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.XferDest
If (iVal=430) Then
    ' Set value
    SessXfr.XferDest = 431
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get XferDest property
    short sVal;
    pXfr->get_XferDest(&sVal);
    if (sVal == 430)
    {
        sVal = 431;
        pXfr->put_XferDest(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XferHostCodePage

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating the code page for the host data file.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.XferHostCodePage  
HEProfileFileTransfer.XferHostCodePage = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XferHostCodePage([out, retval] short  
*pVal);  
HRESULT IHEProfileFileTransfer::put_XferHostCodePage([in] short newVal);
```

#### Parameters

*pVal*—The following returned values indicate the code page for the host data file:

- 0—Austrian (273)
- 1—Belgian (037)
- 2—Brazilian (037)
- 3—Canadian French (037)
- 4—Danish (277)
- 5—English UK (285)
- 6—English US (1047)
- 7—English US (037)
- 8—Finnish (278)
- 9—French (297)
- 10—German (273)
- 11—Italian (280)
- 12—Multinational (500)
- 13—Dutch (037)
- 14—Norwegian (277)
- 15—Portuguese (037)
- 16—Spanish (Latin America)
- 17—Swedish (278)

*newVal*—The set value, specifying the code page for the host data file.

**Basic Example**

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.XferHostCodePage
If (iVal=0) Then
    ' Set value
    SessXfr.XferHostCodePage = 6
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get XferHostCodePage property
    short sVal;
    pXfr->get_XferHostCodePage(&sVal);
    if (sVal == 0)
    {
        sVal = 6;
        pXfr->put_XferHostCodePage(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XferPCCodePage

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value specifying the code page for a data file on your computer. This value determines the location of non-ASCII characters such as é. Windows uses the ANSI (American National Standards Institute) code page; DOS systems use other code pages.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.XferPCCodePage
```

```
HEProfileFileTransfer.XferPCCodePage = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XferPCCodePage([out, retval] short *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_XferPCCodePage([in] short newVal);
```

#### Parameters

*pVal*—One of the following returned values, indicating the code page for a data file on your computer:

- 0—ANSI (Latin-1) (1252/819)
- 1—English US (437)
- 2—Canadian French (863)
- 3—Western European/Latin 1 (850)

*newVal*—The set value, specifying the code page for a data file on your computer.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.XferPCCodePage
If (iVal=0) Then
    ' Set value
    SessXfr.XferPCCodePage = 2
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get XferPCCodePage property
    short sVal;
    pXfr->get_XferPCCodePage(&sVal);
    if (sVal == 0)
    {
        sVal = 2;
        pXfr->put_XferPCCodePage (sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XferSource

### Property, IHEProfileFileTransfer 3270

This property returns or sets a value indicating whether a file has been transferred from a file system or the clipboard.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.XferSource
```

```
HEProfileFileTransfer.XferSource = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XferSource([out, retval] short  
*pVal);  
HRESULT IHEProfileFileTransfer::put_XferSource([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify where a file has been transferred from:

- 430—File system
- 431—Clipboard

*newVal*—The set value, specifying where a file has been transferred from.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
iVal = SessXfr.XferSource  
If (iVal=430) Then  
    ' Set value  
    SessXfr.XferSource = 431  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get XferSource property
    short sVal;
    pXfr->get_XferSource(&sVal);
    if (sVal == 430)
    {
        sVal = 431;
        pXfr->put_XferSource(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XferStartAction

Property, IHEProfileFileTransfer **3270 5250 VT**

This property returns or sets a value indicating how HostExplorer performs before uploading or downloading a file.

### Basic Syntax

```
Integer = HEProfileFileTransfer.XferStartAction  
HEProfileFileTransfer.XferStartAction = Integer
```

### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XferStartAction([out, retval] short  
*pVal);  
HRESULT IHEProfileFileTransfer::put_XferStartAction([in] short newVal);
```

### Parameters

*pVal*—The following returned values indicate how HostExplorer performs before uploading or downloading a file:

- 200—No Action
- 201—Home
- 202—Enter
- 203—Clear

*newVal*—The set value, which indicates how HostExplorer performs before uploading or downloading a file.

### Basic Example

```
Dim Profile As HEPProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
iVal = SessXfr.XferStartAction  
If (iVal=200) Then  
    ' Set value  
    SessXfr.XferStartAction = 203  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFileTransfer interface
    IHEProfileFileTransfer* pFileTransfer;
    pIProfile->get_FileTransfer ( & pFileTransfer);
    //Get XferStartAction property
    short sVal;
    pXfr->get_XferStartAction(&sVal);
    if (sVal == 200)
    {
        sVal = 203;
        pXfr->put_XferStartAction(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Xm1KPacket

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer uses 1024 or 128 bytes as the packet size. Using 1024 bytes is good practice because it is faster. You may find the 128-byte setting useful if you have a communication problem such as a noisy telephone line.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.Xm1KPacket
```

```
HEProfileFileTransfer.Xm1KPacket = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_Xm1KPacket([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileFileTransfer::put_Xm1KPacket([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer uses 1024 bytes as the packet size. A returned value of VARIANT\_FALSE indicates that HostExplorer uses 128 bytes as the packet size.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer uses 1024 bytes as the packet size. A value of VARIANT\_FALSE indicates that HostExplorer uses 128 bytes as the packet size.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.Xm1Kpacket  
If (bVal = False) Then  
    ' Set value  
    SessXfr.Xm1Kpacket = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get XmlKPacket property
VARIANT_BOOL bVal;
pXfr->get_XmlKpacket(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_XmlKpacket(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XmAckTimeout

### Property, IHEProfileFileTransfer VT

This property returns or sets a value specifying the length of time in milliseconds before a file transfer times out. Transfer delays are common. If a delay becomes too long, HostExplorer cancels the transfer, allowing you to try again later.

#### Basic Syntax

```
Long = HEProfileFileTransfer.XmAckTimeout
```

```
HEProfileFileTransfer.XmAckTimeout = Long
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XmAckTimeout([out, retval] long *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_XmAckTimeout([in] long newVal);
```

#### Parameters

*pVal*—The returned value, specifying the length of time before a file transfer times out.

*newVal*—The set value, specifying the length of time before a file transfer times out.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim lVal As Long
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
lVal = SessXfr.XmAckTimeout
If (lVal = 0) Then
    ' Set value
    SessXfr.XmAckTimeout = 15000
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get XmAckTimeout property
long lVal;
pXfr->get_XmAckTimeout(&lVal);
if (lVal == 0)
{
    lVal = 15000;
    pXfr->put_XmAckTimeout(lVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## XmCRC

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer enables CRC (Cyclical Redundancy Check) to detect errors.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.XmCRC  
HEProfileFileTransfer.XmCRC = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_XmCRC ([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileFileTransfer::put_XmCRC ([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables CRC. A returned value of VARIANT\_FALSE indicates that HostExplorer uses Checksum to check errors. (Use Checksum only when communicating with older Xmodem products.)

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables CRC. A value of VARIANT\_FALSE indicates that HostExplorer uses Checksum to check errors.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.XmCRC  
If (bVal = False) Then  
    ' Set value  
    SessXfr.XmCRC = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get XmCRC property
VARIANT_BOOL bVal;
pXfr->get_XmCRC(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_XmCRC(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## **YmAckTimeout**

### **Property, IHEProfileFileTransfer VT**

This property returns or sets a value specifying the length of time in milliseconds before a file transfer times out. Transfer delays are common. If a delay becomes too long, HostExplorer cancels the transfer, allowing you to try again later.

#### **Basic Syntax**

```
Long = HEProfileFileTransfer.YmAckTimeout
```

```
HEProfileFileTransfer.YmAckTimeout = Long
```

#### **C++ Syntax**

```
HRESULT IHEProfileFileTransfer::get_YmAckTimeout([out, retval] long  
*pVal);
```

```
HRESULT IHEProfileFileTransfer::put_YmAckTimeout([in] long newVal);
```

#### **Parameters**

*pVal*—The returned value, specifying the length of time before a file transfer times out.

*newVal*—The set value, specifying the length of time before a file transfer times out.

#### **Basic Example**

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim lVal As Long  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
lVal = SessXfr.YmAckTimeout  
If (lVal=0) Then  
    ' Set value  
    SessXfr.YmAckTimeout = 15000  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get YmAckTimeout property
long lVal;
pXfr->get_YmAckTimeout(&lVal);
if (lVal == 0)
{
    lVal = 15000;
    pXfr->put_YmAckTimeout(lVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## **YmUseFullPath**

### **Property, IHEProfileFileTransfer VT**

This property returns or sets a value indicating whether HostExplorer saves a file to the current UNIX host directory and lets you keep the entire path name as the file name.

#### **Basic Syntax**

```
Boolean = HEProfileFileTransfer.YmUseFullPath  
HEProfileFileTransfer.YmUseFullPath = Boolean
```

#### **C++ Syntax**

```
HRESULT IHEProfileFileTransfer::get_YmUseFullPath([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_YmUseFullPath([in] VARIANT_BOOL  
newVal);
```

#### **Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer saves a file to the current UNIX host directory and keeps the entire pathname as the filename. A returned value of VARIANT\_FALSE indicates that HostExplorer excludes the directory path from the file name.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer saves a file to the current UNIX host directory and keeps the entire pathname as the filename. A value of VARIANT\_FALSE indicates that HostExplorer excludes the directory path from the file name.

#### **Basic Example**

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.YmUseFullPath  
If (bVal = False) Then  
    ' Set value  
    SessXfr.YmUseFullPath = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer );
//Get YmUseFullPath property
VARIANT_BOOL bVal;
pXfr->get_YmUseFullPath(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_YmUseFullPath(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmAutoDownload

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer detects the initial header of a request to receive files and automatically starts the receiving process.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.ZmAutoDownload
```

```
HEProfileFileTransfer.ZmAutoDownload = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ZmAutoDownload([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileFileTransfer::put_ZmAutoDownload([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer detects the initial header of a request to receive files and automatically starts the receiving process. A returned value of VARIANT\_FALSE indicates that HostExplorer does not detect the initial header of a request to receive files and does not automatically start the receiving process.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer detects the initial header of a request to receive files and automatically starts the receiving process. A value of VARIANT\_FALSE indicates that HostExplorer does not detect the initial header of a request to receive files and does not automatically start the receiving process.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.ZmAutoDownload  
If (bVal = False) Then  
    ' Set value  
    SessXfr.ZmAutoDownload = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmAutoDownload property
VARIANT_BOOL bVal;
pXfr->get_ZmAutoDownload(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_ZmAutoDownload(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmCrashRecovery

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer sets a computer to automatically resume receiving files if the computer crashed when the files were being transferred.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.ZmCrashRecovery
```

```
HEProfileFileTransfer.ZmCrashRecovery = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ZmCrashRecovery([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileFileTransfer::put_ZmCrashRecovery([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer sets a computer to automatically resume receiving files. A returned value of VARIANT\_FALSE indicates that a computer does not automatically resume receiving files.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer sets a computer to automatically resume receiving files. A value of VARIANT\_FALSE indicates that a computer does not automatically resume receiving files.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.ZmCrashRecovery  
If (bVal = False) Then  
    ' Set value  
    SessXfr.ZmCrashRecovery = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmCrashRecovery property
VARIANT_BOOL bVal;
pXfr->get_ZmCrashRecovery(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_ZmCrashRecovery(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmMaxErrors

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating how many times HostExplorer attempts to recover from errors before canceling a file transfer.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.ZmMaxErrors
```

```
HEProfileFileTransfer.ZmMaxErrors = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ZmMaxErrors([out, retval] short *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_ZmMaxErrors([in] short newVal);
```

#### Parameters

*pVal*—The returned value, which indicates how many times HostExplorer attempts to recover from errors before canceling a file transfer.

*newVal*—The set value, which indicates how many times HostExplorer attempts to recover from errors before canceling a file transfer.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.ZmMaxErrors
If (iVal= 0) Then
    ' Set value
    SessXfr.ZmMaxErrors = 10
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmMaxErrors property
short sVal;
pXfr->get_ZmMaxErrors(&sVal);
if (sVal == 0)
{
    sVal = 10;
    pXfr->put_ZmMaxErrors(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmOverwriteMngmt

### Property, IHEProfileFileTransfer VT

This property returns or sets a value specifying how files are written to the host. In order for this feature to function properly, the receiving host must also support this feature.

<b>Basic Syntax</b>	<pre>Integer = HEProfileFileTransfer.ZmOverwriteMngmt</pre> <pre>HEProfileFileTransfer.ZmOverwriteMngmt = Integer</pre>
<b>C++ Syntax</b>	<pre>HRESULT IHEProfileFileTransfer::get_ZmOverwriteMngmt([out, retval] short *pVal);</pre> <pre>HRESULT IHEProfileFileTransfer::put_ZmOverwriteMngmt([in] short newVal);</pre>
<b>Parameters</b>	<p><i>pVal</i>—One of the following returned values, specifying how files are written to the host:</p> <ul style="list-style-type: none"><li>• 0—Newer or longer</li><li>• 1—Append</li><li>• 2—Always overwrite</li><li>• 3—File size or date differ</li><li>• 4—Never overwrite</li><li>• 5—Newer</li></ul> <p><i>newVal</i>—The set value, specifying how files are written to the host.</p>
<b>Basic Example</b>	<pre>Dim Profile As HEProfile Dim FileTransfer As HEProfileFileTransfer Dim iVal As Integer 'Set the appropriate type Terminal.TerminalType = HOSTEX_TERMINAL_VT Set Profile = Terminal.Session Set FileTransfer = Profile.FileTransfer ' Get value iVal = SessXfr.ZmOverwriteMngmt If (iVal= 2) Then     ' Set value     SessXfr.ZmOverwriteMngmt = 4 End If</pre>

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmOverwriteMngmt property
short sVal;
pXfr->get_ZmOverwriteMngmt(&sVal);
if (sVal == 2)
{
    sVal = 4;
    pXfr->put_ZmOverwriteMngmt(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmSlideWindow

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer uses the Sliding Window option to send all files simultaneously. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.ZmSlideWindow  
HEProfileFileTransfer.ZmSlideWindow = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ZmSlideWindow([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_ZmSlideWindow([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer sends all files simultaneously using the Sliding Window option. A returned value of VARIANT\_FALSE indicates that HostExplorer sends files without using the Sliding Window option.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer sends all files simultaneously using the Sliding Window option. A value of VARIANT\_FALSE indicates that HostExplorer sends files without using the Sliding Window option.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.ZmSlideWindow  
If (bVal = False) Then  
    ' Set value  
    SessXfr.ZmSlideWindow = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmSlideWindow property
VARIANT_BOOL bVal;
pXfr->get_ZmSlideWindow(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_ZmSlideWindow(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmUseFullPath

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating whether HostExplorer saves a file to the current UNIX host directory and lets you keep the entire pathname as the file name.

#### Basic Syntax

```
Boolean = HEProfileFileTransfer.ZmUseFullPath  
HEProfileFileTransfer.ZmUseFullPath = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ZmUseFullPath([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileFileTransfer::put_ZmUseFullPath([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A returned value of VARIANT\_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A value of VARIANT\_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory.

#### Basic Example

```
Dim Profile As HEProfile  
Dim FileTransfer As HEProfileFileTransfer  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set FileTransfer = Profile.FileTransfer  
' Get value  
bVal = SessXfr.ZmUseFullPath  
If (bVal = False) Then  
    ' Set value  
    SessXfr.ZmUseFullPath = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmUseFullPath property
VARIANT_BOOL bVal;
pXfr->get_ZmUseFullPath(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pXfr->put_ZmUseFullPath(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ZmWindowSize

### Property, IHEProfileFileTransfer VT

This property returns or sets a value indicating the byte size of files transferred using the Sliding Window option. By default, HostExplorer sends 8192 bytes at a time, thereby allowing the receiver enough time to receive the data.

#### Basic Syntax

```
Integer = HEProfileFileTransfer.ZmWindowSize
```

```
HEProfileFileTransfer.ZmWindowSize = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileFileTransfer::get_ZmWindowSize([out, retval] short *pVal);
```

```
HRESULT IHEProfileFileTransfer::put_ZmWindowSize([in] short newVal);
```

#### Parameters

*pVal*—The returned value, which indicates the byte size of files transferred using the Sliding Window option.

*newVal*—The set value, which indicates the byte size of files transferred using the Sliding Window option.

#### Basic Example

```
Dim Profile As HEProfile
Dim FileTransfer As HEProfileFileTransfer
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set FileTransfer = Profile.FileTransfer
' Get value
iVal = SessXfr.ZmWindowSize
If (iVal= 0) Then
    ' Set value
    SessXfr.ZmWindowSize = 8192
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileFileTransfer interface
IHEProfileFileTransfer* pFileTransfer;
pIProfile->get_FileTransfer ( & pFileTransfer);
//Get ZmWindowSize property
short sVal;
pXfr->get_ZmWindowSize(&sVal);
if (sVal == 0)
{
    sVal = 8192;
    pXfr->put_ZmWindowSize(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Profile Security Interface

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

### Properties

The ProfileSecurity interface consists of the following properties:

- Kerberos
- KerberosAltName
- KerberosEncryption
- KerberosForwardTkt
- KerberosVersion
- SecurityOption

### Kerberos

#### Property, IHEProfileSecurity 3270 VT

This property returns or sets a value indicating whether HostExplorer uses Kerberos as a network authentication protocol.

#### Basic Syntax

```
Boolean = HEProfileSecurity.Kerberos  
HEProfileSecurity.Kerberos = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSecurity::get_Kerberos([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileSecurity::put_Kerberos([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer uses Kerberos as a network authentication protocol. A returned value of VARIANT\_FALSE indicates that HostExplorer does not use Kerberos as a network authentication protocol.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer uses Kerberos as a network authentication protocol. A value of VARIANT\_FALSE indicates that HostExplorer does not use Kerberos as a network authentication protocol.

**Basic Example**

```
Dim Profile As HEProfile
Dim Security As HEProfileSecurity
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Security = Profile.Security
' Get value
bVal = SessSec.Kerberos
If (bVal = False) Then
    ' Set value
    SessSec.Kerberos = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileSecurity interface
    IHEProfileSecurity* pSecurity;
    pIProfile->get_Security ( &pSecurity );
    //Get Kerberos property
    VARIANT_BOOL bVal;
    pSec->get_Kerberos(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSec->put_Kerberos(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KerberosAltName

### Property, IHEProfileSecurity 3270 VT

This property returns or sets a string that specifies the alternate user name when you are setting Kerberos authentication.

#### Basic Syntax

```
String = HEProfileSecurity.KerberosAltName
```

```
HEProfileSecurity.KerberosAltName = String
```

#### C++ Syntax

```
HRESULT IHEProfileSecurity::get_KerberosAltName([out, retval] BSTR  
*pVal);  
  
HRESULT IHEProfileSecurity::put_KerberosAltName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the alternate Kerberos user name.

*newVal*—The set string, specifying the alternate Kerberos user name.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Security As HEProfileSecurity  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Security = Profile.Security  
' Get value  
str = SessSec.KerberosAltName  
If (Len(str) = 0) Then  
    ' Set value  
    SessSec.KerberosAltName = "Camelot56"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSecurity interface
    IHEProfileSecurity* pSecurity;
    pIProfile->get_Security ( &pSecurity );
    //Get KerberosAltName property
    BSTR bstr;
    pSec->get_KerberosAltName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("Camelot56"));
        pSec->put_KerberosAltName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KerberosEncryption

### Property, IHEProfileSecurity 3270 VT

This property returns or sets a value that specifies whether you need to use Kerberos encryption for your current session.

#### Basic Syntax

```
Boolean = HEProfileSecurity.KerberosEncryption  
HEProfileSecurity.KerberosEncryption = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSecurity::get_KerberosEncryption([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileSecurity::put_KerberosEncryption([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that you need to use Kerberos encryption for your current session. A returned value of VARIANT\_FALSE indicates that you do not need to use Kerberos encryption for your current session.

*newVal*—A value of VARIANT\_TRUE indicates that you need to use Kerberos encryption for your current session. A value of VARIANT\_FALSE indicates that you do not need to use Kerberos encryption for your current session.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Security As HEProfileSecurity  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Security = Profile.Security  
' Get value  
bVal = SessSec.KerberosEncryption  
If (bVal = False) Then  
    ' Set value  
    SessSec.KerberosEncryption = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSecurity interface
    IHEProfileSecurity* pSecurity;
    pIProfile->get_Security ( &pSecurity );
    //Get KerberosEncryption property
    VARIANT_BOOL bVal;
    pSec->get_KerberosEncryption(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSec->put_KerberosEncryption(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KerberosForwardTkt

### Property, IHEProfileSecurity 3270 VT

This property returns or sets a value that enables or disables Kerberos ticket forwarding.

#### Basic Syntax

```
Boolean = HEProfileSecurity.KerberosForwardTkt  
HEProfileSecurity.KerberosForwardTkt = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSecurity::get_KerberosForwardTkt([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileSecurity::put_KerberosForwardTkt([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that Kerberos ticket forwarding is enabled. A returned value of VARIANT\_FALSE indicates that Kerberos ticket forwarding is disabled.

*newVal*—A value of VARIANT\_TRUE indicates that Kerberos ticket forwarding is enabled. A value of VARIANT\_FALSE indicates that Kerberos ticket forwarding is disabled.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Security As HEProfileSecurity  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Security = Profile.Security  
' Get value  
bVal = SessSec.KerberosForwardTkt  
If (bVal = False) Then  
    ' Set value  
    SessSec.KerberosForwardTkt = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSecurity interface
    IHEProfileSecurity* pSecurity;
    pIProfile->get_Security ( &pSecurity );
    //Get KerberosForwardTkt property
    VARIANT_BOOL bVal;
    pSec->get_KerberosForwardTkt(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSec->put_KerberosForwardTkt(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KerberosVersion

### Property, IHEProfileSecurity 3270 VT

This property returns or sets the Kerberos version for the current session.

#### Basic Syntax

```
Integer = HEProfileSecurity.KerberosVersion  
HEProfileSecurity.KerberosVersion = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileSecurity::get_KerberosVersion([out, retval] short  
*pVal);  
HRESULT IHEProfileSecurity::put_KerberosVersion([in] short newVal);
```

#### Parameters

*pVal*—The returned value, which indicates the Kerberos version of the current session.

*newVal*—The set value, which indicates the Kerberos version of the current session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Security As HEProfileSecurity  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Security = Profile.Security  
' Get value  
iVal = SessSec.KerberosVersion  
If (iVal=3) Then  
    ' Set value  
    SessSec.KerberosVersion = 4  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSecurity interface
    IHEProfileSecurity* pSecurity;
    pIProfile->get_Security ( &pSecurity );
    //Get KerberosVersion property
    short sVal;
    pSec->get_KerberosVersion(&sVal);
    if (sVal == 3)
    {
        sVal = 4;
        pSec->put_KerberosVersion(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SecurityOption

Property, IHEProfileSecurity 3270 5250 VT

This property returns or sets a value that specifies the method for securing the traffic between the server and the client.

### Basic Syntax

```
HOSTEX_SECURITY_OPTIONS = HEProfileSecurity.SecurityOption  
HEProfileSecurity.SecurityOption = HOSTEX_SECURITY_OPTIONS
```

### C++ Syntax

```
HRESULT IHEProfileSecurity::get_SecurityOption([out, retval]  
HOSTEX_SECURITY_OPTIONS *pVal);  
HRESULT IHEProfileSecurity::put_SecurityOption([in]  
HOSTEX_SECURITY_OPTIONS newVal);
```

### Parameters

*pVal*—The returned value, specifying the security option.

*newVal*—The set value, specifying the security method.

### Basic Example

```
Dim Profile As HEProfile  
Dim Security As HEProfileSecurity  
Dim SecurityVal As HOSTEX_SECURITY_OPTIONS  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Security = Profile.Security  
'Get value  
bval = HOSTEX_SECURITY_OPTIONS  
If bval = HOSTEX_SECURITY_NO_SECURITY  
MsgBox "No security available"
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileSecurity interface
    pIProfile->get_Security ( & pSecurity);
    //Get SecurityOption property
    HOSTEX_SECURITY_OPTIONS SecurityOption;
    HRESULT hr = HECoCreateInstance(CLSID_HEProfile, IID_IHEProfile, (LPVOID*)
    )&pProfile, "HEProfile");
    if (SUCCEEDED(hr))
    {
        pProfile->get_Security(&pSecurity);
        pSecurity->get_SecurityOption(&SecurityOption);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileDisplay Interface

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

### Properties

The ProfileDisplay interface consists of the following properties:

- BlinkToItalic
- DisplayInOIA
- DisplayUpperCase
- StatusLineMode
- VTHostWritableStatusLine
- VTMaxScrollBufferSize
- VTSaveAttribsInScrollbar
- VTScrollNoBlanks
- ColumnSeparators
- DisplayRowCol
- ShowNulls
- VTClearScreenOnSizeChange
- VTISOColors
- VTRestISOColors
- VTSaveEraseScreens

### BlinkToItalic

#### Property, IHEProfileDisplay VT

This property returns or sets a value indicating whether HostExplorer maps the Blink attribute to an italicized font. This property is independent of the cursor mode. By default, this property is set to VARIANT\_FALSE.

##### Basic Syntax

```
Boolean = HEPProfileDisplay.BlinkToItalic
```

```
HEPProfileDisplay.BlinkToItalic = Boolean
```

##### C++ Syntax

```
HRESULT IHEProfileDisplay::get_BlinkToItalic([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileDisplay::put_BlinkToItalic([in] VARIANT_BOOL newVal);
```

##### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer maps the Blink attribute to an italicized font. A returned value of VARIANT\_FALSE indicates that HostExplorer does not map the Blink attribute to an italicized font.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer maps the Blink attribute to an italicized font. A value of VARIANT\_FALSE indicates that HostExplorer does not map the Blink attribute to an italicized font.

**Basic Example**

```
Dim Profile As HEProfile
Dim Display As HEProfileDisplay
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Display = Profile.Display
bVal = SessDisp.BlinkToItalic
If (bVal = False) Then
    SessDisp.BlinkToItalic = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( & pDisplay );
//Get BlinkToItalic property
VARIANT_BOOL bVal;
pDisp->get_BlinkToItalic(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_BlinkToItalic(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ColumnSeparators

### Property, IHEProfileDisplay 5250

This property returns or sets a value specifying the column-separator style for the session. By default, this property is set to Dots.

#### Basic Syntax

```
Integer = HEProfileDisplay.ColumnSeparators
```

```
HEProfileDisplay.ColumnSeparators = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_ColumnSeparators([out, retval] short *pVal);
```

```
HRESULT IHEProfileDisplay::put_ColumnSeparators([in] short newVal);
```

#### Parameters

*pVal*—A returned value of:

- None—Indicates that HostExplorer does not display separators between columns.
- Dots—Indicates that HostExplorer displays column separators as dots on the terminal window.
- Lines—Indicates that HostExplorer displays column separators as lines on the terminal window.

*newVal*—A set value of:

- None—Indicates that HostExplorer does not display separators between columns.
- Dots—Indicates that HostExplorer displays column separators as dots on the terminal window.
- Lines—Indicates that HostExplorer displays column separators as lines on the terminal window.

#### Basic Example

```
Dim Profile As HEProfile
Dim Display As HEProfileDisplay
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_5250
Set Profile = Terminal.Session
Set Display = Profile.Display
' Get value
iVal = SessDisp.ColumnSeparators
If (iVal = 0) Then
    ' Set value
    SessDisp.ColumnSeparators = 3
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileDisplay interface
    IHEProfileDisplay* pDisplay;
    pIProfile->get_Display ( & pDisplay );
    //Get ColumnSeparators property
    short sVal;
    pDisp->get_ColumnSeparators(&sVal);
    if (sVal == 0)
    {
        sVal = 3;
        pDisp->put_ColumnSeparators(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DisplayInOIA

### Property, IHEProfileDisplay 3270

This property returns or sets a value that specifies whether Host Explorer displays the host IP address or the host response time in the OIA (Operator Information Area). By default, this property is set to

`HOSTEX_OIA_DISPLAY_IP_ADDRESS`.

#### Basic Syntax

```
HOSTEX_OIA_DISPLAY = HEProfileDisplay.DisplayInOIA
```

```
HEProfileDisplay.DisplayInOIA = HOSTEX_OIA_DISPLAY
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_DisplayInOIA([out, retval]  
HOSTEX_OIA_DISPLAY *pVal);
```

```
HRESULT IHEProfileDisplay::put_DisplayInOIA([in] HOSTEX_OIA_DISPLAY  
newVal);
```

#### Parameters

*pVal*—The returned value, specifying the OIA display mode.

*newVal*—The set value, specifying the OIA display mode you want.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim dVal As HOSTEX_OIA_DISPLAY  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
' Get value  
dVal = SessDisp.DisplayInOIA  
If (dVal = HOSTEX_OIA_DISPLAY_HOST_RESPONSE_TIME) Then  
    ' Set value  
    SessDisp.DisplayInOIA = HOSTEX_OIA_DISPLAY_IP_ADDRESS  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileDisplay interface
    IHEProfileDisplay* pDisplay;
    pIProfile->get_Display ( & pDisplay );
    //Get DisplayInOIA property
    HOSTEX_OIA_DISPLAY dVal;
    pDisp->get_DisplayInOIA(&dVal);
    if (dVal == HOSTEX_OIA_DISPLAY_HOST_RESPONSE_TIME)
    {
        dVal = HOSTEX_OIA_DISPLAY_IP_ADDRESS;
        pDisp->put_DisplayInOIA(dVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DisplayRowCol

### Property, IHEProfileDisplay VT

This property returns or sets a value indicating whether HostExplorer displays the row and column indicator in the right-hand corner of the OIA (Operator Information Area). By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.DisplayRowCol
```

```
HEProfileDisplay.DisplayRowCol = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_DisplayRowCol([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileDisplay::put_DisplayRowCol([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays the row and column indicator in the OIA. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display the row and column indicator in the OIA.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays the row and column indicator in the OIA. A value of VARIANT\_FALSE indicates that HostExplorer does not display the row and column indicator in the OIA.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.DisplayRowCol  
If (bVal = False) Then  
    SessDisp.DisplayRowCol = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( & pDisplay );
//Get DisplayRowCol property
VARIANT_BOOL bVal;
pDisp->get_DisplayRowCol(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_DisplayRowCol(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DisplayUpperCase

### Property, IHEProfileDisplay 3270 5250

This property returns or sets a value indicating whether HostExplorer displays all output in upper case. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.DisplayUpperCase
```

```
HEProfileDisplay.DisplayUpperCase = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_DisplayUpperCase([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileDisplay::put_DisplayUpperCase([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays all output in upper case. A returned value of VARIANT\_FALSE indicates that HostExplorer does not change the output.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays all output in upper case. A value of VARIANT\_FALSE indicates that HostExplorer does not change the output.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.DisplayUpperCase  
If (bVal = False) Then  
    SessDisp.DisplayUpperCase = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileDisplay interface
    IHEProfileDisplay* pDisplay;
    pIProfile->get_Display ( & pDisplay );
    //Get DisplayUpperCase property
    VARIANT_BOOL bVal;
    pDisp->get_DisplayUpperCase(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pDisp->put_DisplayUpperCase(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ShowNulls

### Property, IHEProfileDisplay 3270 5250

This property returns or sets a value indicating whether HostExplorer displays null characters in unprotected fields as centered dots. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.ShowNulls
```

```
HEProfileDisplay.ShowNulls = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_ShowNulls([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileDisplay::put_ShowNulls([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays null characters as centered dots. A returned value of VARIANT\_FALSE indicates that HostExplorer displays null characters as spaces.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays null characters as centered dots. A value of VARIANT\_FALSE indicates that HostExplorer displays null characters as spaces.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.ShowNulls  
If (bVal = False) Then  
    SessDisp.ShowNulls = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileDisplay interface
    IHEProfileDisplay* pDisplay;
    pIProfile->get_Display ( &pDisplay );
    //Get ShowNulls property
    VARIANT_BOOL bVal;
    pDisp->get_ShowNulls(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pDisp->put_ShowNulls(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## StatusLineMode

### Property, IHEProfileDisplay 3270 5250 VT

This property returns or sets the type of status line mode (either terminal or window).

#### Basic Syntax

```
HOSTEX_STATUS_LINE_MODE = HEProfileDisplay.StatusLineMode  
HEProfileDisplay.StatusLineMode = HOSTEX_STATUS_LINE_MODE
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value, which indicates the type of status-line mode.

*newVal*—The set value, which indicates the type of status-line mode.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim slMode As HOSTEX_STATUS_LINE_MODE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
slMode = SessDisp.StatusLineMode  
If (slMode = HOSTEX_STATUS_LINE_MODE_NOSTATUSLINE) Then  
    SessDisp.StatusLineMode = HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileDisplay interface
    IHEProfileDisplay* pDisplay;
    pIProfile->get_Display ( &pDisplay );
    //Get StatusLineMode property
    HOSTEX_STATUS_LINE_MODE slMode;
    pDisp->get_StatusLineMode(&slMode);
    if (slMode == HOSTEX_STATUS_LINE_MODE_NOSTATUSLINE)
    {
        slMode = HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE;
        pDisp->put_StatusLineMode(slMode);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTClearScreenOnSizeChange

### Property, IHEProfileDisplay VT

This property returns or sets a value that specifies whether HostExplorer clears the screen display whenever you resize the screen.

#### Basic Syntax

```
Boolean = HEProfileDisplay.VTClearScreenOnSizeChange  
HEProfileDisplay.VTClearScreenOnSizeChange = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTClearScreenOnSizeChange([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileDisplay::put_VTClearScreenOnSizeChange([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer clears the screen display whenever you resize the screen. If *pVal* equals VARIANT\_FALSE, HostExplorer does not clear the screen.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to clear the screen display whenever you resize the screen. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to clear the screen.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTClearScreenOnSizeChange  
If (bVal = False) Then  
    SessDisp.VTClearScreenOnSizeChange = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( &pDisplay );
//Get VTClearScreenOnSizeChange property
VARIANT_BOOL bVal;
pDisp->get_VTClearScreenOnSizeChange(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTClearScreenOnSizeChange(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTHostWritableStatusLine

### Property, IHEProfileDisplay VT

This property returns or sets a value indicating whether you want HostExplorer to display messages on the status line. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.VTHostWritableStatusLine  
HEProfileDisplay.VTHostWritableStatusLine = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTHostWritableStatusLine([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileDisplay::put_VTHostWritableStatusLine([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays messages on the status line. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display messages on the status line.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays messages on the status line. A value of VARIANT\_FALSE indicates that HostExplorer does not display messages on the status line.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTHostWritableStatusLine  
If (bVal = False) Then  
    SessDisp.VTHostWritableStatusLine = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( &pDisplay );
//Get VTHostWritableStatusLine property
VARIANT_BOOL bVal;
pDisp->get_VTHostWritableStatusLine(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTHostWritableStatusLine(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTISOColors

### Property, IHEProfileDisplay VT

This property returns or sets a value indicating whether HostExplorer enables support for ISO colors for ANSI (American National Standards Institute) color-escape sequences on VT100, VT101, VT220, VT320, and VT420 models. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.VTISOColors
```

```
HEProfileDisplay.VTISOColors = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTISOColors([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileDisplay::put_VTISOColors([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables support for ISO colors. A returned value of VARIANT\_FALSE indicates that HostExplorer disables support for ISO colors.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables support for ISO colors. A value of VARIANT\_FALSE indicates that HostExplorer disables support for ISO colors.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTISOColors  
If (bVal = False) Then  
    SessDisp.VTISOColors = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( &pDisplay );
//Get VTISOCOLORS property
VARIANT_BOOL bVal;
pDisp->get_VTISOCOLORS(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTISOCOLORS(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTMaxScrollBufferSize

### Property, IHEProfileDisplay VT

This property returns or sets the number of lines that HostExplorer maintains in the Scrollback buffer. This number ranges from 1 to 9,999. By default, this property is set to 100. To disable the Scrollback buffer, set the property to 0.

#### Basic Syntax

```
Integer = HEProfileDisplay.VTMaxScrollBufferSize  
HEProfileDisplay.VTMaxScrollBufferSize = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTMaxScrollBufferSize([out, retval] short  
*pVal);  
HRESULT IHEProfileDisplay::put_VTMaxScrollBufferSize([in] short newVal);
```

#### Parameters

*pVal*—The returned value, indicating the number of lines that HostExplorer maintains in the Scrollback buffer.

*newVal*—The set value, indicating the number of lines that HostExplorer maintains in the Scrollback buffer.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
iVal = SessDisp.VTMaxScrollBufferSize  
If (iVal = 0) Then  
    SessDisp.VTMaxScrollBufferSize = 2000  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( & pDisplay );
//Get VTMaxScrollBufferSize property
short sVal;
pDisp->get_VTMaxScrollBufferSize(&sVal);
if (sVal == 0)
{
    sVal = 2000;
    pDisp->put_VTMaxScrollBufferSize(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTResetISOColors

### Property, IHEProfileDisplay VT

This property returns or sets a value that specifies whether HostExplorer obeys a Reset Set Graphics Rendition (SGR 0) command when the host is sending color graphics commands. If you encounter screens that appeared properly with an older version of HostExplorer but no longer do, set this property to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.VTResetISOColors
```

```
HEProfileDisplay.VTResetISOColors = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTResetISOColors([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileDisplay::put_VTResetISOColors([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer resets ISO colors when prompted to do so by the host. If *pVal* equals VARIANT\_FALSE, HostExplorer does not reset ISO colors.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to reset ISO colors when it is prompted to do so by the host. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to reset ISO colors.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTResetISOColors  
If (bVal = False) Then  
    SessDisp.VTResetISOColors = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( & pDisplay );
//Get VTResetISOColors property
VARIANT_BOOL bVal;
pDisp->get_VTResetISOColors(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTResetISOColors(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTSaveAttribsInScrollback

### Property, IHEProfileDisplay VT

This property returns or sets a value indicating whether HostExplorer saves the Telnet screen attributes in the Scrollback buffer. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.VTSaveAttribsInScrollback  
HEProfileDisplay.VTSaveAttribsInScrollback = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTSaveAttribsInScrollback([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileDisplay::put_VTSaveAttribsInScrollback([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer saves the Telnet screen attributes in the Scrollback buffer. A returned value of VARIANT\_FALSE indicates that HostExplorer does not save the Telnet screen attributes in the Scrollback buffer.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer saves the Telnet screen attributes in the Scrollback buffer. A value of VARIANT\_FALSE indicates that HostExplorer does not save the Telnet screen attributes in the Scrollback buffer.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTSaveAttribsInScrollback  
If (bVal = False) Then  
    SessDisp.VTSaveAttribsInScrollback = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( &pDisplay );
//Get VTSaveAttribsInScrollbar property
VARIANT_BOOL bVal;
pDisp->get_VTSaveAttribsInScrollbar(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTSaveAttribsInScrollbar(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## **VTSaveEraseScreens**

### **Property, IHEProfileDisplay VT**

This property returns or sets a value indicating whether HostExplorer saves the screen to the Scrollback buffer before performing the Erase-Screen Host command. By default, this property is set to VARIANT\_FALSE.

#### **Basic Syntax**

```
Boolean = HEProfileDisplay.VTSaveEraseScreens  
HEProfileDisplay.VTSaveEraseScreens = Boolean
```

#### **C++ Syntax**

```
HRESULT IHEProfileDisplay::get_VTSaveEraseScreens ([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileDisplay::put_VTSaveEraseScreens ([in] VARIANT_BOOL  
newVal);
```

#### **Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer saves the screen to the Scrollback buffer before performing the Erase-Screen Host command. A returned value of VARIANT\_FALSE indicates that HostExplorer does not save the screen to the Scrollback buffer before performing the Erase-Screen Host command.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer saves the screen to the Scrollback buffer before performing the Erase-Screen Host command. A value of VARIANT\_FALSE indicates that HostExplorer does not save the screen to the Scrollback buffer before performing the Erase-Screen Host command.

#### **Basic Example**

```
Dim Profile As HEProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTSaveEraseScreens  
If (bVal = False) Then  
    SessDisp.VTSaveEraseScreens = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( & pDisplay );
//Get VTSaveEraseScreens property
VARIANT_BOOL bVal;
pDisp->get_VTSaveEraseScreens(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTSaveEraseScreens(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTScrollNoBlanks

### Property, IHEProfileDisplay VT

This property returns or sets a value indicating whether to prevent HostExplorer from adding blank lines to the Scrollback buffer. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileDisplay.VTScrollNoBlanks
```

```
HEProfileDisplay.VTScrollNoBlanks = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileDisplay::get_VTScrollNoBlanks([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileDisplay::put_VTScrollNoBlanks([in] BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer cannot add blank lines to the Scrollback buffer, so you can compress new lines to create more space in the buffer. A returned value of VARIANT\_FALSE indicates that HostExplorer can add blank lines to the Scrollback buffer.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer cannot add blank lines to the Scrollback buffer, so you can compress new lines to create more space in the buffer. A value of VARIANT\_FALSE indicates that HostExplorer can add blank lines to the Scrollback buffer.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Display As HEProfileDisplay  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Display = Profile.Display  
bVal = SessDisp.VTScrollNoBlanks  
If (bVal = False) Then  
    SessDisp.VTScrollNoBlanks = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileDisplay interface
IHEProfileDisplay* pDisplay;
pIProfile->get_Display ( & pDisplay );
//Get VTScrollNoBlanks property
VARIANT_BOOL bVal;
pDisp->get_VTScrollNoBlanks(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pDisp->put_VTScrollNoBlanks(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileCursor Interface

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

### Properties

The ProfileCursor interface consists of the following properties:

- CursorMode
- CursorType
- DisplayCrossHairCursor
- MoveCursorOnMouseClicked

#### CursorMode

**Property, IHEProfileCursor 3270 5250 VT**

This property returns or sets a value specifying the cursor mode (solid or blink) to use for the current Profile.

#### Basic Syntax

```
Integer = HEProfileCursor.CursorMode  
HEProfileCursor.CursorMode = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileCursor::get_CursorMode([out, retval] short *pVal);  
HRESULT IHEProfileCursor::put_CursorMode([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify the cursor mode: 0—Solid or 1—Blink.

*newVal*—The set value, specifying the cursor mode.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Cursor As HEProfileCursor  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Cursor = Profile.Cursor  
iVal = SessCur.CursorMode  
If (iVal = 0) Then  
    SessCur.CursorMode = 1  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileCursor interface
    IHEProfileCursor* pCursor;
    pIProfile->get_Cursor ( & pCursor );
    //Get CursorMode property
    short sVal;
    pCursor->get_CursorMode(&sVal);
    if (sVal == 0)
    {
        sVal = 1;
        pCursor->put_CursorMode(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CursorType

Property, IHEProfileCursor 3270 5250 VT

This property returns or sets a value specifying the cursor type (vertical bar, underscore, or block) to use for the current session.

### Basic Syntax

```
Integer = HEProfileCursor.CursorType
```

```
HEProfileCursor.CursorType = Integer
```

### C++ Syntax

```
HRESULT IHEProfileCursor::get_CursorType([out, retval] short *pVal);
```

```
HRESULT IHEProfileCursor::put_CursorType([in] short newVal);
```

### Parameters

*pVal*—The following returned values specify the cursor type:

- 0—Vertical bar
- 1—Underscore
- 2—Block

*newVal*—The set value, specifying the cursor type.

### Basic Example

```
Dim Profile As HEProfile
Dim Cursor As HEProfileCursor
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Cursor = Profile.Cursor
iVal = SessCur.CursorType
If (iVal = 0) Then
    SessCur.CursorType = 1
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileCursor interface
    IHEProfileCursor* pCursor;
    pIProfile->get_Cursor ( & pCursor );
    //Get CursorType property
    short sVal;
    pCursor->get_CursorType(&sVal);
    if (sVal == 0)
    {
        sVal = 1;
        pCursor->put_CursorType(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DisplayCrossHairCursor

Property, IHEProfileCursor 3270 5250 VT

This property returns or sets a value that specifies whether the cursor appears as a cross-hair cursor. A cross-hair cursor consists of two "cross-hair" lines that span across the screen and intersect at the cursor. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfileCursor.DisplayCrossHairCursor
```

```
HEProfileCursor.DisplayCrossHairCursor = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileCursor::get_DisplayCrossHairCursor([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileCursor::put_DisplayCrossHairCursor([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, the cursor appears as a cross-hair. If *pVal* equals VARIANT\_FALSE, the cursor appears in its regular form.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to make the cursor appear as a cross hair. Set *newVal* to VARIANT\_FALSE to make the cursor appear in its normal form.

### Basic Example

```
Dim Profile As HEPProfile  
Dim Cursor As HEProfileCursor  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Cursor = Profile.Cursor  
bVal = SessCur.DisplayCrossHairCursor  
If (bVal = False) Then  
    SessCur.DisplayCrossHairCursor = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileCursor interface
    IHEProfileCursor* pCursor;
    pIProfile->get_Cursor ( & pCursor );
    //Get DisplayCrossHairCursor property
    VARIANT_BOOL bVal;
    pCursor->get_DisplayCrossHairCursor(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pCursor->put_DisplayCrossHairCursor(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MoveCursorOnMouseClicked

### Property, IHEProfileCursor VT

This property lets you force HostExplorer to automatically move the cursor when you click the mouse. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileCursor.MoveCursorOnMouseClicked  
HEProfileCursor.MoveCursorOnMouseClicked = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileCursor::get_MoveCursorOnMouseClicked([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileCursor::put_MoveCursorOnMouseClicked([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically moves the cursor when you click the mouse. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically move the cursor when you click the mouse.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically moves the cursor when you click the mouse. A value of VARIANT\_FALSE indicates that HostExplorer does not automatically move the cursor when you click the mouse.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Cursor As HEProfileCursor  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Cursor = Profile.Cursor  
Dim bVal As Boolean  
bVal = SessCur.MoveCursorOnMouseClicked  
If (bVal = False) Then  
    SessCur.MoveCursorOnMouseClicked = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileCursor interface
IHEProfileCursor* pCursor;
pIProfile->get_Cursor ( & pCursor );
//Get MoveCursorOnMouseClicked property
VARIANT_BOOL bVal;
pCursor->get_MoveCursorOnMouseClicked(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pCursor->put_MoveCursorOnMouseClicked(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileFonts Interface

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

### Methods

The ProfileFonts interface has the following method:

SetFont

### Properties

The ProfileFonts interface consists of the following properties:

- CharacterSpacing
- VariableWidthFont

### SetFont

**Method, IHEProfileFonts 3270 5250 VT**

This method lets you specify the font and font size you want for the session window. You specify the font size in terms of a width and height. If the specified font is a TrueType font, HostExplorer uses the larger of the two values you supply (width and height) for the font size.

#### Basic Syntax

```
HEProfileFonts.SetFont(bstrFontName As String, nXSize As Integer, nYSize As Integer)
```

#### C++ Syntax

```
HRESULT IHEProfileFonts::SetFont(  
    [in] BSTR bstrFontName,  
    [in] short nXSize,  
    [in] short nYSize);
```

#### Parameters

*bstrFontName*—The name of the font you want for the session window.

*nXSize*—The width of the specified font, in points.

*nYSize*—The height of the specified font, in points.

#### Basic Example

```
Dim Profile As HEProfile  
Set Profile = Terminal.Session  
Dim SessFonts As HEProfileFonts  
Set SessFonts = Profile.Fonts  
' Set HE_Bitmap, 14x35, as the font and size  
SessFonts.SetFont("HE_Bitmap", 14, 35)
```

**C++ Example**

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHEProfile *pSess;
pIDispatch->QueryInterface(IID_IHEProfile, (void**) &pSess);
IHEProfileFonts *pFonts;
pSess->get_Fonts(&pFonts);
BSTR bstr;
bstr = SysAllocString(OLESTR("HE_Bitmap"));
// Set HE_Bitmap, 14x35, as the font and size
pFonts->SetFont(bstr, 14, 35);
SysFreeString(bstr);
```

## CharacterSpacing

### Property, IHEProfileFonts 3270 5250 VT

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

**Basic Syntax**

```
Integer = HEProfileFonts.CharacterSpacing
```

```
HEProfileFonts.CharacterSpacing = Integer
```

**C++ Syntax**

```
HRESULT IHEProfileFonts::get_CharacterSpacing([out, retval short *pVal);
```

```
HRESULT IHEProfileFonts::put_CharacterSpacing([in] short newVal);
```

**Parameters**

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

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

**Basic Example**

```
Dim Profile As HEPProfile
Dim Fonts As HEPProfileFonts
Dim SessFonts As HEPProfileFonts
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Fonts = Profile.Fonts
Set SessFonts = Profile.Fonts
Dim iVal As Integer
' Get value
iVal = SessFonts.CharacterSpacing
If (iVal = 0) Then
    ' Set value
    SessFonts.CharacterSpacing = 2
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileFonts interface
    IHEProfileFonts* pFonts;
    pIProfile->get_Fonts ( &pFonts );
    //Get CharacterSpacing property
    short sVal;
    pFonts->get_CharacterSpacing(&sVal);
    if (sVal == 0)
    {
        sVal=5;
        pFonts->put_CharacterSpacing(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VariableWidthFont

### Property, IHEProfileFonts 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer uses proportional fonts.

#### Basic Syntax

```
Boolean = HEProfileFonts.VariableWidthFont
```

```
HEProfileFonts.VariableWidthFont = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileFonts::get_VariableWidthFont([out, retval] VARIANT_BOOL  
*pVal);  
  
HRESULT IHEProfileFonts::put_VariableWidthFont([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer uses proportional fonts. A returned value of VARIANT\_FALSE indicates that HostExplorer does not use proportional fonts.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer uses proportional fonts. A value of VARIANT\_FALSE indicates that HostExplorer does not use proportional fonts.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Fonts As HEProfileFonts  
Dim SessFonts As HEProfileFonts  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Fonts = Profile.Fonts  
Set SessFonts = Profile.Fonts  
Dim bVal As Boolean  
' Get value  
bVal = SessFonts.VariableWidthFont  
If (bVal = False) Then  
    ' Set value  
    SessFonts.VariableWidthFont = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileFonts interface
    IHEProfileFonts* pFonts;
    pIProfile->get_Fonts ( &pFonts );
    //Get VariableWidthFont property
    VARIANT_BOOL bVal;
    bVal = pFonts->get_VariableWidthFont(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pFonts->put_VariableWidthFont(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfilePCPrint Interface

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

### Properties

The ProfilePCPrint interface consists of the following properties:

- PrintMode7171
- PrinterDeinit
- PrinterInit
- TprintMode

#### PrintMode7171

**Property, IHEProfilePCPrint 3270**

This property returns or sets a value specifying what sequences the emulator searches for in PassThru mode.

#### Basic Syntax

```
Integer = HEProfilePCPrint.PrintMode7171  
HEProfilePCPrint.PrintMode7171 = Integer
```

#### C++ Syntax

```
HRESULT IHEProfilePCPrint::get_PrintMode7171([out, retval] short *pVal) ;  
HRESULT IHEProfilePCPrint::put_PrintMode7171([in] short newVal) ;
```

#### Parameters

*pVal*—The following returned values specify the sequences that the emulator searches for in pass-through mode:

- 0—VT100
- 1—IBM 3164

*newVal*—The set value, specifying the sequences that the emulator searches for in pass-through mode.

### Basic Example

```
Dim Profile As HEProfile
Dim PCPrint As HEProfilePCPrint
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set PCPrint = Profile.PCPrint
iVal = SessFonts.PrintMode7171
If (iVal = 0) Then
    SessFonts.PrintMode7171 = 1
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfilePCPrint interface
    IHEProfilePCPrint* pPCPrint;
    pIProfile->get_PCPrint ( & pPCPrint );
    //Get PrintMode7171 property
    short sVal;
    pPCPrint->get_PrintMode7171(&sVal);
    if (sVal == 0)
    {
        sVal = 1;
        pPCPrint->put_PrintMode7171(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrinterDeinit

### Property, IHEProfilePCPrint 3270

This property returns or sets a string specifying the escape sequences that you can send to the printer at the end of a PCPRINT/TPRINT job. The string can contain up to 255 characters and is printer-specific.

#### Basic Syntax

```
String = HEProfilePCPrint.PrinterDeinit  
HEProfilePCPrint.PrinterDeinit = String
```

#### C++ Syntax

```
HRESULT IHEProfilePCPrint::get_PrinterDeinit([out, retval] BSTR *pVal);  
HRESULT IHEProfilePCPrint::put_PrinterDeinit([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned de-initialization string that is sent to the printer for pass-through printing.

*newVal*—The set de-initialization string that is sent to the printer for pass-through printing.

#### Basic Example

```
Dim Profile As HEProfile  
Dim PCPrint As HEProfilePCPrint  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PCPrint = Profile.PCPrint  
' Get value  
str = SessPCPrint.PrinterDeinit  
If (Len(str) = 0) Then  
    ' Set value  
    SessPCPrint.PrinterDeinit = "^M-AT&C1&D2Q0V1X4E1S0=0^M"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePCPrint interface
IHEProfilePCPrint* pPCPrint;
pIProfile->get_PCPrint ( & pPCPrint );
//Get PrinterDeinit property
BSTR bstr;
pPCPrint->get_PrinterDeinit(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("^\u004D-\u0041\u0054&\u0043\u0031&\u0044\u0032\u0030\u0030\u0031\u0034\u0031\u0030\u0030=0^\u004d"));
    pPCPrint->put_PrinterDeinit(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrinterInit

### Property, IHEProfilePCPrint 3270

This property returns or sets a string specifying the escape sequences that you can send to the printer at the beginning of a PCPRINT/TPRINT job. The string can contain up to 255 characters and is printer-specific.

#### Basic Syntax

```
String = HEProfilePCPrint.PrinterInit
```

```
HEProfilePCPrint.PrinterInit = String
```

#### C++ Syntax

```
HRESULT IHEProfilePCPrint::get_PrinterInit([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfilePCPrint::put_PrinterInit([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned initialization string that is sent to the printer for pass-through printing.

*newVal*—The set initialization string that is sent to the printer for pass-through printing.

#### Basic Example

```
Dim Profile As HEProfile
Dim PCPrint As HEProfilePCPrint
Dim str As String
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set PCPrint = Profile.PCPrint
' Get value
str = SessPCPrint.PrinterInit
If (Len(str) = 0) Then
    ' Set value
    SessPCPrint.PrinterInit = "^M-NT&C1&89ROCYX84X4E1S0=7^M"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePCPrint interface
IHEProfilePCPrint* pPCPrint;
pIProfile->get_PCPrint ( & pPCPrint );
//Get PrinterInit property
BSTR bstr;
pPCPrint->get_PrinterInit(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("^\u00d7M-NT&C1&89ROCYX84X4E1S0=\u00d7M"));
    pPCPrint->put_PrinterInit(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TprintMode

### Property, IHEProfilePCPrint 3270 5250 VT

This property returns or sets a value specifying one of the following output destinations for TPRINT:

- default Windows printer
- the printer connected to an LPT1, LPT2, or LPT3 parallel port
- Windows clipboard

#### Basic Syntax

```
HOSTEX_TPRINT_OUTPUT = HEProfilePCPrint.TprintMode  
HEProfilePCPrint.TprintMode = HOSTEX_TPRINT_OUTPUT
```

#### C++ Syntax

```
HRESULT IHEProfilePCPrint::get_TprintMode([out, retval]  
HOSTEX_TPRINT_OUTPUT *pVal);  
  
HRESULT IHEProfilePCPrint::put_TprintMode([in] HOSTEX_TPRINT_OUTPUT  
newVal);
```

#### Parameters

*pVal*—The returned value, specifying the output destination for TPRINT.

*newVal*—The set value, specifying the output destination for TPRINT.

#### Basic Example

```
Dim Profile As HEProfile  
Dim PCPrint As HEProfilePCPrint  
Dim iVal As HOSTEX_TPRINT_OUTPUT  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PCPrint = Profile.PCPrint  
iVal = SessFonts.TprintMode  
If (iVal = HOSTEX_TPRINT_OUTPUT_LPT1) Then  
    SessFonts.TprintMode = HOSTEX_TPRINT_OUTPUT_DEFAULT_WIN_PRINTER  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePCPrint interface
    IHEProfilePCPrint* pPCPrint;
    pIProfile->get_PCPrint ( &pPCPrint );
    //Get TprintMode property
    HOSTEX_TPRINT_OUTPUT sVal;
    pPCPrint->get_TprintMode(&sVal);
    if (sVal == HOSTEX_TPRINT_OUTPUT_LPT1)
    {
        sVal = HOSTEX_TPRINT_OUTPUT_DEFAULT_WIN_PRINTER;
        pPCPrint->put_TprintMode(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfilePrintScreen Interface

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

### Properties

The ProfilePrintScreen interface consists of the following properties:

- AddFormFeed
- DisplayPrintDlg
- PrintBlackAndWhite
- PrinterDocname
- PrinterHeader
- PrintOIA
- PrintScreenFontName
- PRTSCRUseSpecificPrinter
- DisplayAbortDlg
- HostScreenPerPage
- PrintBorder
- PrinterFooter
- PrintLocation
- PrintReversedColors
- PrintScreenFontSize

### AddFormFeed

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer disables automatic form feed after HostExplorer sends a “screen image” to the printer.

#### Basic Syntax

```
Boolean = HEProfilePrintScreen.AddFormFeed
```

```
HEProfilePrintScreen.AddFormFeed = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_AddFormFeed([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfilePrintScreen::put_AddFormFeed([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables automatic form feed. A returned value of VARIANT\_FALSE indicates that HostExplorer disables automatic form feed.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables automatic form feed. A value of VARIANT\_FALSE indicates that HostExplorer disables automatic form feed.

**Basic Example**

```
Dim Profile As HEProfile
Dim PrintScreen As HEProfilePrintScreen
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set PrintScreen = Profile.PrintScreen
bVal = PrintScrn.AddFormFeed
If (bVal = False) Then
    PrintScrn.AddFormFeed = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get AddFormFeed property
    VARIANT_BOOL bVal;
    pPrtScrn->get_AddFormFeed(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_AddFormFeed(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DisplayAbortDlg

Property, IHEProfilePrintScreen 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays the Abort dialog box while printing. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfilePrintScreen.DisplayAbortDlg
```

```
HEProfilePrintScreen.DisplayAbortDlg = Boolean
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_DisplayAbortDlg([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfilePrintScreen::put_DisplayAbortDlg([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays the Abort dialog box. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display the Abort dialog box.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays the Abort dialog box. A value of VARIANT\_FALSE indicates that HostExplorer does not display the Abort dialog box.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.DisplayAbortDlg  
If (bVal = False) Then  
    PrintScrn.DisplayAbortDlg = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( &pPrintScreen);
    //Get DisplayAbortDlg property
    VARIANT_BOOL bVal;
    pPrtScrn->get_DisplayAbortDlg(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_DisplayAbortDlg(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DisplayPrintDlg

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer displays the Print Screen dialog box when it begins printing. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfilePrintScreen.DisplayPrintDlg
```

```
HEProfilePrintScreen.DisplayPrintDlg = Boolean
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_DisplayPrintDlg([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfilePrintScreen::put_DisplayPrintDlg([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays the PrintScreen dialog box when it begins printing. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display the PrintScreen dialog box.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays the PrintScreen dialog box when it begins printing. A value of VARIANT\_FALSE indicates that HostExplorer does not display the PrintScreen dialog box.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.DisplayPrintDlg  
If (bVal = False) Then  
    PrintScrn.DisplayPrintDlg = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( &pPrintScreen);
    //Get DisplayPrintDlg property
    VARIANT_BOOL bVal;
    pPrtScrn->get_DisplayPrintDlg(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_DisplayPrintDlg(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## HostScreenPerPage

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets the number of host screens that you want to print on one page.

### Basic Syntax

```
Integer = HEProfilePrintScreen.HostScreenPerPage  
HEProfilePrintScreen.HostScreenPerPage = Integer
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_HostScreenPerPage([out, retval] short  
*pVal);  
HRESULT IHEProfilePrintScreen::put_HostScreenPerPage([in] short newVal);
```

### Parameters

*pVal*—The returned value, specifying the current number of host screens printed on a single page.

*newVal*—The set value, specifying the number of host screens that you want to print on a single page.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
iVal = PrintScrn.HostScreenPerPage  
If (iVal < 4) Then  
    PrintScrn.HostScreenPerPage = 4  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( &pPrintScreen);
    //Get HostScreenPerPage property
    short sVal;
    pPrtScrn->get_HostScreenPerPage(&sVal);
    if (sVal < 4)
    {
        sVal = 4;
        pPrtScrn->put_HostScreenPerPage(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintBlackAndWhite

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer forces black-and-white printing on color printers by automatically converting colors to gray-scale. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfilePrintScreen.PrintBlackAndWhite
```

```
HEProfilePrintScreen.PrintBlackAndWhite = Boolean
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintBlackAndWhite([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfilePrintScreen::put_PrintBlackAndWhite([in] VARIANT_BOOL
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE forces black-and-white printing. A returned value of VARIANT\_FALSE indicates color printing.

*newVal*—A value of VARIANT\_TRUE forces black-and-white printing. A value of VARIANT\_FALSE indicates color printing.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.PrintBlackAndWhite  
If (bVal = False) Then  
    PrintScrn.PrintBlackAndWhite = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PrintBlackAndWhite property
    VARIANT_BOOL bVal;
    pPrtScrn->get_PrintBlackAndWhite(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_PrintBlackAndWhite (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintBorder

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer prints a thick border around the printed screen image. You can set this property only if you specify Centered On Page in the PrintLocation property.

### Basic Syntax

```
Boolean = HEProfilePrintScreen.PrintBorder
```

```
HEProfilePrintScreen.PrintBorder = Boolean
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintBorder([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfilePrintScreen::put_PrintBorder([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer prints a thick border around the printed screen image. A returned value of VARIANT\_FALSE indicates that HostExplorer does not print a border.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer prints a thick border around the printed screen image. A value of VARIANT\_FALSE indicates that HostExplorer does not print a border.

### Basic Example

```
Dim Profile As HEPProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.PrintBorder  
If (bVal = False) Then  
    PrintScrn.PrintBorder = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PrintBorder property
    VARIANT_BOOL bVal;
    pPrtScrn->get_PrintBorder(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_PrintBorder(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrinterDocname

### Property, IHEProfilePrintScreen 3270 5250 VT

This property returns or sets a string specifying the name of the document to be printed.

#### Basic Syntax

```
String = HEProfilePrintScreen.PrinterDocname  
HEProfilePrintScreen.PrinterDocname = String
```

#### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrinterDocname([out, retval] BSTR  
*pVal);  
HRESULT IHEProfilePrintScreen::put_PrinterDocname([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the name of the document to be printed.

*newVal*—The set string, specifying the name of the document to be printed.

#### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
str = PrintScrn.PrinterDocname  
If (Len(str) = 0) Then  
    PrintScrn.PrinterDocname = "MyDoc.txt"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintScreen interface
IHEProfilePrintScreen* pPrintScreen;
pIProfile->get_PrintScreen ( & pPrintScreen);
//Get PrintDocname property
BSTR bstr;
pPrtScrn->get_PrinterDocname(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("MyDoc.txt"));
    pPrtScrn->put_PrinterDocname(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrinterFooter

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a string specifying the information to appear in the document footer.

### Basic Syntax

```
String = HEProfilePrintScreen.PrinterFooter
```

```
HEProfilePrintScreen.PrinterFooter = String
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrinterFooter([out, retval] BSTR  
*pVal);
```

```
HRESULT IHEProfilePrintScreen::put_PrinterFooter([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the information to appear in the document footer.

*newVal*—The set string, specifying the information to appear in the document footer.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
str = PrintScrn.PrinterFooter  
If (Len(str) = 0) Then  
    PrintScrn.PrinterFooter = "Hummingbird"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintScreen interface
IHEProfilePrintScreen* pPrintScreen;
pIProfile->get_PrintScreen ( & pPrintScreen);
//Get PrinterFooter property
BSTR bstr;
pPrtScrn->get_PrinterFooter(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Hummingbird"));
    pPrtScrn->put_PrinterFooter(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrinterHeader

Property, IHEProfilePrintScreen 3270 5250 VT

This property returns or sets a value specifying the information to appear in the document header.

### Basic Syntax

```
String = HEProfilePrintScreen.PrinterHeader
```

```
HEProfilePrintScreen.PrinterHeader = String
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrinterHeader([out, retval] BSTR  
*pVal);
```

```
HRESULT IHEProfilePrintScreen::put_PrinterHeader([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the information to appear in the document header.

*newVal*—The set string, specifying the information to appear in the document header.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
str = PrintScrn.PrinterHeader  
If (Len(str) = 0) Then  
    PrintScrn.PrinterHeader = "Hummingbird"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintScreen interface
IHEProfilePrintScreen* pPrintScreen;
pIProfile->get_PrintScreen ( & pPrintScreen);
//Get PrinterHeader property
BSTR bstr;
pPrtScrn->get_PrinterHeader(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Hummingbird"));
    pPrtScrn->put_PrinterHeader(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintLocation

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a value specifying how a screen appears on a printed page (either in the center or in the upper left-hand corner).

### Basic Syntax

```
Integer = HEProfilePrintScreen.PrintLocation  
HEProfilePrintScreen.PrintLocation = Integer
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintLocation([out, retval] short  
*pVal);  
HRESULT IHEProfilePrintScreen::put_PrintLocation([in] short newVal);
```

### Parameters

*pVal*—The following returned values specify how a screen appears on a printed page:

- 485—Centered
- 486—Upper left

*newVal*—The set value, specifying how a screen appears on a printed page.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
iVal = PrintScrn.PrintLocation  
If (iVal=486) Then  
    PrintScrn.PrintLocation = 485  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PrintLocation property
    short sVal;
    pPrtScrn->get_PrintLocation(&sVal);
    if (sVal == 486)
    {
        sVal = 485;
        pPrtScrn->put_PrintLocation(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintOIA

### Property, IHEProfilePrintScreen 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints the OIA (Operator Information Area) on the printed screen image.

#### Basic Syntax

```
Boolean = HEProfilePrintScreen.PrintOIA  
HEProfilePrintScreen.PrintOIA = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintOIA([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfilePrintScreen::put_PrintOIA([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer prints the OIA on the printed screen image. A returned value of VARIANT\_FALSE indicates that HostExplorer does not print the OIA on the printed screen image.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer prints the OIA on the printed screen image. A value of VARIANT\_FALSE indicates that HostExplorer does not print the OIA on the printed screen image.

#### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.PrintOIA  
If (bVal = False) Then  
    PrintScrn.PrintOIA = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PrintOIA property
    VARIANT_BOOL bVal;
    pPrtScrn->get_PrintOIA(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_PrintOIA(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintReversedColors

Property, IHEProfilePrintScreen 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints reverse-color modes. This property forces the print engine to swap black and white colors while printing images. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfilePrintScreen.PrintReversedColors
```

```
HEProfilePrintScreen.PrintReversedColors = Boolean
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintReversedColors([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfilePrintScreen::put_PrintReversedColors([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer prints reverse-color modes. A returned value of VARIANT\_FALSE indicates that HostExplorer does not print reverse-color modes.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer prints reverse-color modes. A value of VARIANT\_FALSE indicates that HostExplorer does not print reverse-color modes.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.PrintReversedColors  
If (bVal = False) Then  
    PrintScrn.PrintReversedColors = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PrintReversedColors property
    VARIANT_BOOL bVal;
    pPrtScrn->get_PrintReversedColors(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_PrintReversedColors (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintScreenFontName

Property, IHEProfilePrintScreen **3270 5250 VT**

This property lets you change the default printer font used to print screen images (or the keyboard template) from Times New Roman to any TrueType font of your choice.

### Basic Syntax

```
String = HEProfilePrintScreen.PrintScreenFontName  
HEProfilePrintScreen.PrintScreenFontName = String
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintScreenFontName([out, retval] BSTR  
*pVal);  
HRESULT IHEProfilePrintScreen::put_PrintScreenFontName([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, which indicates the TrueType font.

*newVal*—The set string, which indicates the TrueType font.

### Basic Example

```
Dim Profile As HEPProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
str = PrintScrn.PrinterHeader  
If (Len(str) = 0) Then  
    PrintScrn.PrintScreenFontName = "Arial"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintScreen interface
IHEProfilePrintScreen* pPrintScreen;
pIProfile->get_PrintScreen ( & pPrintScreen);
//Get PrintScreenFontName property
BSTR bstr;
pPrtScrn->get_PrintScreenFontName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Arial"));
    pPrtScrn->put_PrintScreenFontName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PrintScreenFontSize

Property, IHEProfilePrintScreen **3270 5250 VT**

This property returns or sets a value indicating the font size for printing.

### Basic Syntax

```
Integer = HEProfilePrintScreen.PrintScreenFontSize  
HEProfilePrintScreen.PrintScreenFontSize = Integer
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PrintScreenFontSize([out, retval]  
short *pVal);  
HRESULT IHEProfilePrintScreen::put_PrintScreenFontSize([in] short  
newVal);
```

### Parameters

*pVal*—The returned value, which indicates the font size for printing.

*newVal*—The set value, which indicates the font size for printing.

### Basic Example

```
Dim Profile As HEPProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
iVal = PrintScrn.PrintScreenFontSize  
If (iVal = 0) Then  
    PrintScrn.PrintScreenFontSize = 12  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PrintScreenFontSize property
    short sVal;
    pPrtScrn->get_PrintScreenFontSize(&sVal);
    if (sVal == 0)
    {
        sVal = 12;
        pPrtScrn->put_PrintScreenFontSize(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PRTSCRUseSpecificPrinter

Property, IHEProfilePrintScreen 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints using the printer displayed in the Selected Printer Info box or the default printer. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HEProfilePrintScreen.PRTSCRUseSpecificPrinter
```

```
HEProfilePrintScreen.PRTSCRUseSpecificPrinter = Boolean
```

### C++ Syntax

```
HRESULT IHEProfilePrintScreen::get_PRTSCRUseSpecificPrinter([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfilePrintScreen::put_PRTSCRUseSpecificPrinter([in]  
VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer prints using the specified printer. A returned value of VARIANT\_FALSE indicates that HostExplorer prints using the default printer.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer prints using the specified printer. A value of VARIANT\_FALSE indicates that HostExplorer prints using the default printer.

### Basic Example

```
Dim Profile As HEProfile  
Dim PrintScreen As HEProfilePrintScreen  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintScreen = Profile.PrintScreen  
bVal = PrintScrn.PRTSCRUseSpecificPrinter  
If (bVal = False) Then  
    PrintScrn.PRTSCRUseSpecificPrinter = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintScreen interface
    IHEProfilePrintScreen* pPrintScreen;
    pIProfile->get_PrintScreen ( & pPrintScreen);
    //Get PRTSCRUseSpecificPrinter property
    VARIANT_BOOL bVal;
    pPrtScrn->get_PRTSCRUseSpecificPrinter(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrtScrn->put_PRTSCRUseSpecificPrinter(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfilePrintSession Interface

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

### Properties

The ProfilePrintSession interface consists of the following properties:

- HostName
- LimitToSingleInstance
- LUName
- LUType
- ProfileName
- StartPrinter
- StopPrinter

### HostName

**Property, IHEProfilePrintSession 3270 5250**

This property returns or sets a string specifying the host name or IP address that PrintSession uses to establish a connection.

#### Basic Syntax

```
String = HEProfilePrintSession.HostName
```

```
HEProfilePrintSession.HostName = String
```

#### C++ Syntax

```
HRESULT IHEProfilePrintSession::get_HostName([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfilePrintSession::put_HostName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the host name or IP address.

*newVal*—The set string, specifying the host name or IP address.

**Basic Example**

```
Dim Profile As HEProfile
Dim PrintSession As HEProfilePrintSession
Dim str As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set PrintSession = Profile.PrintSession
' Get value
str = PrintSess.HostName
If (Len(str) = 0) Then
    ' Set value
    PrintSess.HostName = "vm1.mcgill.ca"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintSession interface
IHEProfilePrintSession* pPrintSession;
pIProfile->get_PrintSession ( & pPrintSession );
//Get HostName property
BSTR bstr;
pPrintSess->get_HostName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("vm1.mcgill.ca"));
    pPrintSess->put_HostName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## **LimitToSingleInstance**

**Property, IHEProfilePrintSession 3270 5250 VT**

This property returns or sets a value that specifies whether the printer session associated with a display session starts only once if you have launched more than one instance of the display session.

### **Basic Syntax**

```
Boolean = HEProfilePrintSession.LimitToSingleInstance  
HEProfilePrintSession.LimitToSingleInstance = Boolean
```

### **C++ Syntax**

```
HRESULT IHEProfilePrintSession::get_LimitToSingleInstance([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfilePrintSession::get_LimitToSingleInstance([in]  
VARIANT_BOOL newVal);
```

### **Parameters**

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, the printer session starts only once. If *pVal* equals VARIANT\_FALSE, a printer session starts for each instance of the display session.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that the printer session for a display session starts only once (even though there may be multiple instances of the display session). Set *newVal* to VARIANT\_FALSE if you do not want a single printer session.

### **Basic Example**

```
Dim Profile As HEProfile  
Dim PrintSession As HEProfilePrintSession  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintSession = Profile.PrintSession  
bVal = PrintSess.LimitToSingleInstance  
If (bVal = False) Then  
    PrintSess.LimitToSingleInstance = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintSession interface
    IHEProfilePrintSession* pPrintSession;
    pIProfile->get_PrintSession ( & pPrintSession );
    //Get LimitToSingleInstance property
    VARIANT_BOOL bVal;
    pPrintSess->get_LimitToSingleInstance(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrintSess->put_LimitToSingleInstance(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LUName

### Property, IHEProfilePrintSession 3270 5250

This property returns or sets a string specifying the LU (Logical Units) name for the current print Profile.

#### Basic Syntax

```
String = HEProfilePrintSession.LUName
```

```
HEProfilePrintSession.LUName = String
```

#### C++ Syntax

```
HRESULT IHEProfilePrintSession::get_LUName([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfilePrintSession::put_LUName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the LU name.

*newVal*—The set string, specifying the LU name.

#### Basic Example

```
Dim Profile As HEProfile
Dim PrintSession As HEProfilePrintSession
Dim str As String
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set PrintSession = Profile.PrintSession
' Get value
str = PrintSess.LUName
If (Len(str) = 0) Then
    ' Set value
    PrintSess.LUName = "LUName1"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintSession interface
IHEProfilePrintSession* pPrintSession;
pIProfile->get_PrintSession ( & pPrintSession );
//Get LUName property
BSTR bstr;
pPrintSess->get_LUName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("LUNAme1"));
    pPrintSess->put_LUName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LUType

### Property, IHEProfilePrintSession 3270 5250

This property returns or sets a value specifying how the LU (Logical Units) type is determined.

#### Basic Syntax

```
Integer = HEProfilePrintSession.LUType
```

```
HEProfilePrintSession.LUType = Integer
```

#### C++ Syntax

```
HRESULT IHEProfilePrintSession::get_LUType([out, retval] short *pVal);
```

```
HRESULT IHEProfilePrintSession::put_LUType([in] short newVal);
```

#### Parameters

*pVal*—The following returned values indicate how the LU type is determined:

- 460—ConnectLU
- 461—LUName
- 462—AssociateLU
- 463—ProfileLU

*newVal*—The set string, which indicates how the LU type is determined.

#### Basic Example

```
Dim Profile As HEPProfile
Dim PrintSession As HEProfilePrintSession
Dim iVal As Integer
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set PrintSession = Profile.PrintSession
iVal = PrintSess.LUType
If (iVal=462) Then
    PrintSess.LUType =464
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintSession interface
    IHEProfilePrintSession* pPrintSession;
    pIProfile->get_PrintSession ( & pPrintSession );
    //Get LUType property
    short sVal;
    pPrintSess->get_LUType(&sVal);
    if (sVal == 462)
    {
        sVal = 464;
        pPrintSess->put_LUType(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileName

### Property, IHEProfilePrintSession 3270 5250

This property returns or sets a string specifying the LU (Logical Units) name for the Base PrintSession profile.

#### Basic Syntax

```
String = HEProfilePrintSession.ProfileName
```

```
HEProfilePrintSession.ProfileName = String
```

#### C++ Syntax

```
HRESULT IHEProfilePrintSession::get_ProfileName([out, retval] BSTR  
*pVal);
```

```
HRESULT IHEProfilePrintSession::put_ProfileName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the LU name for the Base PrintSession profile.

*newVal*—The set string, specifying the LU name for the Base PrintSession profile.

#### Basic Example

```
Dim Profile As HEProfile  
Dim PrintSession As HEProfilePrintSession  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintSession = Profile.PrintSession  
' Get value  
str = PrintSess.Profile  
If (Len(str) = 0) Then  
    ' Set value  
    PrintSess.Profile = "test.HPR"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfilePrintSession interface
IHEProfilePrintSession* pPrintSession;
pIProfile->get_PrintSession ( & pPrintSession );
//Get ProfileName property
BSTR bstr;
pPrintSess->get_ProfileName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("test.HPR"));
    pPrintSess->put_ProfileName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## StartPrinter

### Property, IHEProfilePrintSession 3270 5250

This property returns or sets a value indicating whether HostExplorer starts the printer automatically.

#### Basic Syntax

```
Boolean = HEProfilePrintSession.StartPrinter  
HEProfilePrintSession.StartPrinter = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfilePrintSession::get_StartPrinter([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfilePrintSession::put_StartPrinter([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer starts the printer automatically. A returned value of VARIANT\_FALSE indicates that HostExplorer does not start the printer automatically.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer starts the printer automatically. A value of VARIANT\_FALSE indicates that HostExplorer does not start the printer automatically.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim PrintSession As HEProfilePrintSession  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintSession = Profile.PrintSession  
bVal = PrintSess.StartPrinter  
If (bVal = False) Then  
    PrintSess.StartPrinter = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfilePrintSession interface
    IHEProfilePrintSession* pPrintSession;
    pIProfile->get_PrintSession ( & pPrintSession );
    //Get StartPrinter property
    VARIANT_BOOL bVal;
    pPrintSess->get_StartPrinter(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrintSess->put_StartPrinter(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## StopPrinter

### Property, IHEProfilePrintSession 3270 5250

This property returns or sets a value indicating whether HostExplorer stops the printer.

#### Basic Syntax

```
Boolean = HEProfilePrintSession.StopPrinter
```

```
HEProfilePrintSession.StopPrinter = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfilePrintSession::get_StopPrinter([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfilePrintSession::put_StopPrinter([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer stops the printer. A returned value of VARIANT\_FALSE indicates that HostExplorer does not stop the printer.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer stops the printer. A value of VARIANT\_FALSE indicates that HostExplorer does not stop the printer.

#### Basic Example

```
Dim Profile As HEProfile  
Dim PrintSession As HEProfilePrintSession  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set PrintSession = Profile.PrintSession  
bVal = PrintSess.StopPrinter  
If (bVal = False) Then  
    PrintSess.StopPrinter = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfilePrintSession interface
    IHEProfilePrintSession* pPrintSession;
    pIProfile->get_PrintSession ( & pPrintSession );
    //Get StopPrinter property
    VARIANT_BOOL bVal;
    pPrintSess->get_StopPrinter(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pPrintSess->put_StopPrinter(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileCapture Interface

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

### Properties

The ProfileCapture interface consists of the following properties:

- SaveAppend
- SaveConfirm
- SaveFileName
- SaveMode
- VTCaptureMode

#### SaveAppend

**Property, IHEProfileCapture 3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer adds a new file to the end of an existing file. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileCapture.SaveAppend  
HEProfileCapture.SaveAppend = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileCapture::get_SaveAppend([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileCapture::put_SaveAppend([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer adds a new file to the end of an existing file. A returned value of VARIANT\_FALSE indicates that HostExplorer does not add a new file to the end of an existing file.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer adds a new file to the end of an existing file. A returned value of VARIANT\_FALSE indicates that HostExplorer does not add a new file to the end of an existing file.

**Basic Example**

```
Dim Profile As HEProfile
Dim Capture As HEProfileCapture
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Capture = Profile.Capture
bVal = SessSave.SaveAppend
If (bVal = False) Then
    SessSave.SaveAppend = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileCapture interface
    IHEProfileCapture* pCapture;
    pIProfile->get_Capture ( &pCapture );
    //Get SaveAppend property
    VARIANT_BOOL bVal;
    pCapture->get_SaveAppend(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pCapture->put_SaveAppend(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SaveConfirm

### Property, IHEProfileCapture 3270 5250 VT

This property returns or sets a value indicating whether to force the Save Screen command to prompt for a file name.

#### Basic Syntax

```
Boolean = HEProfileCapture.SaveConfirm
```

```
HEProfileCapture.SaveConfirm = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileCapture::get_SaveConfirm([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileCapture::put_SaveConfirm([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE forces the Save Screen command to prompt for a file name. A returned value of VARIANT\_FALSE does not force the Save Screen command to prompt for a file name.

*newVal*—A value of VARIANT\_TRUE forces the Save Screen command to prompt for a file name. A value of VARIANT\_FALSE does not force the Save Screen command to prompt for a file name.

#### Basic Example

```
Dim Profile As HEProfile
Dim Capture As HEProfileCapture
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Capture = Profile.Capture
bVal = SessSave.SaveConfirm
If (bVal = False) Then
    SessSave.SaveConfirm = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileCapture interface
    IHEProfileCapture* pCapture;
    pIProfile->get_Capture ( &pCapture );
    //Get SaveConfirm property
    VARIANT_BOOL bVal;
    pCapture->get_SaveConfirm(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pCapture->put_SaveConfirm(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SaveFileName

Property, IHEProfileCapture 3270 5250 VT

This property returns or sets a string specifying the default path and file name that HostExplorer uses for saved and captured files. Unless otherwise specified, HostExplorer uses the same file name for all captured and saved screens.

### Basic Syntax

```
String = HEProfileCapture.SaveFileName
```

```
HEProfileCapture.SaveFileName = String
```

### C++ Syntax

```
HRESULT IHEProfileCapture::get_SaveFileName([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileCapture::put_SaveFileName([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the default path and file name.

*newVal*—The set string, specifying the default path and file name.

### Basic Example

```
Dim Profile As HEProfile
Dim Capture As HEProfileCapture
Dim str As String
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Capture = Profile.Capture
' Get value
str = SessSave.SaveFileName
If (Len(str) = 0) Then
    ' Set value
    SessSave.SaveFileName = "test.SAV"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileCapture interface
IHEProfileCapture* pCapture;
pIProfile->get_Capture ( &pCapture );
//Get SaveFileName property
BSTR bstr;
pCapture->get_SaveFileName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("test.SAV"));
    pCapture->put_SaveFileName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SaveMode

Property, IHEProfileCapture 3270 5250 VT

This property returns or sets a value specifying the format in which HostExplorer saves the .SAV file.

### Basic Syntax

```
Integer = HEProfileCapture.SaveMode  
HEProfileCapture.SaveMode = Integer
```

### C++ Syntax

```
HRESULT IHEProfileCapture::get_SaveMode([out, retval] short *pVal);  
HRESULT IHEProfileCapture::put_SaveMode([in] short newVal);
```

### Parameters

*pVal*—The following returned values specify the format for saved files:

- 447—ASCII
- 448—ANSI

*newVal*—The set value, specifying the save mode.

### Basic Example

```
Dim Profile As HEProfile  
Dim Capture As HEProfileCapture  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Capture = Profile.Capture  
' Get value  
iVal = SessSave.Savemode  
If (iVal= 448) Then  
    ' Set value  
    SessSave.Savemode = 447  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileCapture interface
    IHEProfileCapture* pCapture;
    pIProfile->get_Capture ( & pCapture );
    //Get SaveMode property
    short sVal;
    pCapture->get_Savemode(&sVal);
    if sVal == 448)
    {
        sVal = 447;
        pCapture->put_Savemode(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTCaptureMode

### Property, IHProfileCapture VT

This property returns or sets a value specifying how to capture selected text.

#### Basic Syntax

```
Integer = HEProfileCapture.VTCaptureMode  
HEProfileCapture.VTCaptureMode = Integer
```

#### C++ Syntax

```
HRESULT IHProfileCapture::get_VTCaptureMode([out, retval] short *pVal);  
HRESULT IHProfileCapture::put_VTCaptureMode([in] short newVal);
```

#### Parameters

*pVal*—The following returned values specify how to capture selected text:

- 546—Raw
- 547—Text

*newVal*—The set value, specifying how to capture selected text.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Capture As HEProfileCapture  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Capture = Profile.Capture  
' Get value  
iVal = SessSave.VTCaptureMode  
If (iVal= 546) Then  
    ' Set value  
    SessSave.VTCaptureMode = 547  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileCapture interface
IHEProfileCapture* pCapture;
pIProfile->get_Capture ( & pCapture );
//Get VTCaptureMode property
short sVal;
pCapture->get_VTCaptureMode(&sVal);
if (sVal == 546)
{
    sVal = 547;
    pCapture->put_VTCaptureMode(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileHostPrinting Interface

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

### Properties

The ProfileHostPrinting interface consists of the following properties:

- VTAutoFormFeed
- VTIPrintMaxRows
- VTPrintByPassWindows
- VTPrintDisableTranslation
- VTPrinterEnableTimeout
- VTPrinterInit
- VTPrinterTimeoutValue
- VTPrint FileMode
- VTPrintLFtoCRLF
- VTUseSpecificPrinter
- VTIPrintMaxCols
- VTPassThruUPSS
- VTPrintDefaultFont
- VTPrinterDeinit
- VTPrinterFontName
- VTPrinterTimeout
- VTPrintFile
- VTPrintFitFontToPage
- VTPrintTarget

### VTAutoFormFeed

Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer automatically adds a form feed to the end of the print job.

#### Basic Syntax

```
Boolean = HEPProfileHostPrinting.VTAutoFormFeed
```

```
HEProfileHostPrinting.VTAutoFormFeed = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTAutoFormFeed([out, retval]
VARIANT_BOOL *pVal);

HRESULT IHEProfileHostPrinting::put_VTAutoFormFeed([in] VARIANT_BOOL
newVal);
```

**Parameters**

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer adds a form feed to the end of the print job. If *pVal* equals VARIANT\_FALSE, HostExplorer does not add a form feed.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer adds a form feed to the end of each print job. Set *newVal* to VARIANT\_FALSE to disable the automatic addition of a form feed to the end of the print job.

**Basic Example**

```
Dim Profile As HEProfile
Dim HostPrinting As HEProfileHostPrinting
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set HostPrinting = Profile.HostPrinting
' Get value
bVal = HostPrint.VTAutoFormFeed
If (bVal = False) Then
    ' Set value
    HostPrint.VTAutoFormFeed = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTAutoFormFeed property
VARIANT_BOOL bVal;
pHostPrint->get_VTAutoFormFeed(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint->put_VTAutoFormFeed(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTiPrintMaxCols

### Property, IHEProfileHostPrinting VT

This property returns or sets the maximum number of columns that fit on a print-output page.

#### Basic Syntax

```
Integer = HEProfileHostPrinting.VTiPrintMaxCols  
HEProfileHostPrinting.VTiPrintMaxCols = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTiPrintMaxCols([out, retval] short  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTiPrintMaxCols([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the maximum number of columns that fit on a print-output page.

*newVal*—The set value, specifying the maximum number of columns that you want to fit on a print-output page.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim maxCols As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
maxCols = hostPrint.VtiPrintMaxCols  
If (maxCols = 3) Then  
    ' Set value  
    hostPrint.VtiPrintMaxCols = 4  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VtiPrintMaxCols property
short maxCols;
hostPrint->get_VtiPrintMaxCols(&maxCols);
if (maxCols == 3)
{
    maxCols = 4;
    hostPrint->put_VtiPrintMaxCols(maxCols);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTiPrintMaxRows

### Property, IHEProfileHostPrinting VT

This property returns or sets the maximum number of rows that fit on a print-output page.

#### Basic Syntax

```
Integer = HEProfileHostPrinting.VTiPrintMaxRows  
HEProfileHostPrinting.VTiPrintMaxRows = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTiPrintMaxRows([out, retval] short  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTiPrintMaxRows([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the maximum number of rows that fit on a print-output page.

*newVal*—The set value, specifying the maximum number of rows that you want to fit on a print-output page.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim maxRows As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
maxRows = hostPrint.VtiPrintMaxRows  
If (maxRows = 3) Then  
    ' Set value  
    hostPrint.VtiPrintMaxRows = 4  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VtiPrintMaxRows property
short maxRows;
hostPrint->get_VtiPrintMaxRows (&maxRows);
if (maxRows == 3)
{
    maxRows = 4;
    hostPrint->put_VtiPrintMaxRows (maxRows);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPassThruUPSS

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that defines the host character set that you want to use for Passthru printing. The value of the property corresponds to an index in the `VT_UPS.hxl` file. HostExplorer loads this file into memory at start-up. The file acts as a table that relates numeric indexes to host character sets. The file is located in the `Xtable` subdirectory of the directory in which you installed HostExplorer.

#### Basic Syntax

```
Integer = HEProfileHostPrinting.VTPassThruUPSS  
HEProfileHostPrinting.VTPassThruUPSS = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPassThruUPSS([out, retval] short  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTPassThruUPSS([in] short newVal);
```

#### Parameters

`pVal`—The returned value, specifying the index of the host character set that is used during Passthru printing.

`newVal`—The set value, specifying the index of the host character set that you want to use for Passthru printing. `newVal` can be any value between 0 and 255 inclusive. For example, if the Slovenian 7-bit character set has the following index in the `VT_UPS.hxl` file,

`IndexCharset=0x002C`

and you want to use this character set, set `newVal` to decimal 44 (hexadecimal 2C).

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim indexVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
indexVal = HostPrint.VTPassThruUPSS  
If (indexVal <> 44) Then  
    ' Use the host character set that has  
    ' an index of 44 in the VT_UPS.hxl file  
    HostPrint.VTPassThruUPSS = 44  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPassThruUPSS property
short indexVal;
pHostPrint->get_VTPassThruUPSS(&indexVal);
if (indexVal != 44)
{
    // Use the host character set that has
    // an index of 44 in the VT_UPS.hxl file
    indexVal = 44;
    pHostPrint->put_VTPassThruUPSS(codeVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintByPassWindows

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer bypasses the Windows print driver.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTPrintByPassWindows  
HEProfileHostPrinting.VTPrintByPassWindows = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintByPassWindows([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrintByPassWindows([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer bypasses the Windows print driver. If *pVal* equals VARIANT\_FALSE, HostExplorer does not bypass the driver.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to bypass the Windows print driver. Set *newVal* to VARIANT\_FALSE to use the Windows print driver.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrintByPassWindows  
If (bVal = False) Then  
    ' Set value  
    HostPrint.VTPrintByPassWindows = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrintByPassWindows property
VARIANT_BOOL bVal;
pHostPrint->get_VTPrintByPassWindows (&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint->put_VTPrintByPassWindows (bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintDefaultFont

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer uses the default font when printing.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTPrintDefaultFont
```

```
HEProfileHostPrinting.VTPrintDefaultFont = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintDefaultFont([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrintDefaultFont([in] VARIANT_BOOL
newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer uses the default font when printing.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer uses the default font for printing.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrintDefaultFont  
If (bVal = False) Then  
    ' Set value  
    HostPrint.VTPrintDefaultFont = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileHostPrinting interface
    IHEProfileHostPrinting* pHostPrinting;
    pIProfile->get_HostPrinting ( & pHostPrinting );
    //Get VTPrintDefaultFont property
    VARIANT_BOOL bVal;
    pHostPrint->get_VTPrintDefaultFont(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pHostPrint->put_VTPrintDefaultFont(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintDisableTranslation

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer disables translation of host data from the host code page to the Windows code page.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTPrintDisableTranslation  
HEProfileHostPrinting.VTPrintDisableTranslation = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintDisableTranslation([out,  
retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrintDisableTranslation([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer disables translation of host data from the host code page to the Windows code page. If *pVal* equals VARIANT\_FALSE, HostExplorer does not disable translation of host data.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to disable translation of host data from the host code page to the Windows code page. Set *newVal* to VARIANT\_FALSE to enable translation of host data.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrintDisableTranslation  
If (bVal = False) Then  
    ' Set value  
    HostPrint.VTPrintDisableTranslation = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrintDisableTranslation property
VARIANT_BOOL bVal;
pHostPrint->get_VTPrintDisableTranslation(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint->put_VTPrintDisableTranslation(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrinterDeinit

### Property, IHEProfileHostPrinting VT

This property returns or sets a string specifying the escape sequences that you can send to the printer at the end of a print job. The string can contain up to 255 characters and is printer-specific. You can enter Escape and binary codes using the backslash character (\). HostExplorer treats Inline spaces as part of the sequence.

#### Basic Syntax

```
String = HEProfileHostPrinting.VTPrinterDeinit  
HEProfileHostPrinting.VTPrinterDeinit = String
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrinterDeinit([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrinterDeinit([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned de-initialization string that HostExplorer sends to the printer for pass-through printing.

*newVal*—The de-initialization string that you want sent to the printer for pass-through printing.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
str = HostPrint.VTPrinterDeinit  
If (Len(str) = 0) Then  
    ' Set value  
    HostPrint.VTPrinterDeinit = "^M-AT&C1&D2Q0V1X4E1S0=0^M"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrinterDeinit property
BSTR bstr;
pHostPrint->get_VTPrinterDeinit(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("^\u00c3M-\u00c3AT&\u00c3C1&\u00c3D2Q0V1X4E1S0=\u00c30^\u00c3M"));
    pHostPrint->put_VTPrinterDeinit(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrinterEnableTimeout

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer delays the printer output.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTPrinterEnableTimeout  
HEProfileHostPrinting.VTPrinterEnableTimeout = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrinterEnableTimeout([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrinterEnableTimeout([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer delays the printer output. If *pVal* equals VARIANT\_FALSE, HostExplorer does not delay printer output.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to enable the printer delay. Set *newVal* to VARIANT\_FALSE to disable the printer delay.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrinterEnableTimeout  
If (bVal = False) Then  
    ' Set value  
    HostPrint.VTPrinterEnableTimeout = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrinterEnableTimeout property
VARIANT_BOOL bVal;
pHostPrint->get_VTPrinterEnableTimeout(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint->put_VTPrinterEnableTimeout(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrinterFontName

### Property, IHEProfileHostPrinting VT

This property returns or sets a string that specifies the default printer font. The string can specify the name of any TrueType font. The default setting is "Times New Roman".

#### Basic Syntax

```
String = HEProfileHostPrinting.VTPrinterFontName
```

```
HEProfileHostPrinting.VTPrinterFontName = String
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrinterFontName([out, retval] BSTR  
*pVal);
```

```
HRESULT IHEProfileHostPrinting::put_VTPrinterFontName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string that specifies the current default printer font.

*newVal*—The string that specifies the name of the font that you want to set as the default printer font.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
str = HostPrint.VTPrinterFontName  
If (Len(str) = 0) Then  
    ' Set value  
    HostPrint.VTPrinterFontName = "Arial"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrinterFontName property
BSTR bstr;
pHostPrint->get_VTPrinterFontName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Arial"));
    pHostPrint->put_VTPrinterFontName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrinterInit

### Property, IHEProfileHostPrinting VT

This property returns or sets a string specifying the escape sequences that you can send to the printer at the beginning of a print job. The string can contain up to 255 characters and is printer-specific. You can enter Escape and binary codes using the backslash character (\). HostExplorer treats Inline spaces as part of the sequence.

#### Basic Syntax

```
String = HEProfileHostPrinting.VTPrinterInit  
HEProfileHostPrinting.VTPrinterInit = String
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrinterInit([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrinterInit([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned initialization string that is sent to the printer for pass-through printing.

*newVal*—The initialization string that you want sent to the printer for pass-through printing.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
str = HostPrint.VTPrinterInit  
If (Len(str) = 0) Then  
    ' Set value  
    HostPrint.VTPrinterInit = "^M-NT&C1&89ROCYX84X4E1S0=7^M"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrinterInit property
BSTR bstr;
pHostPrint->get_VTPrinterInit(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("^\u00d7M-NT&C1&89ROCYX84X4E1S0=\u00d7M"));
    pHostPrint->put_VTPrinterInit(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrinterTimeout

### Property, IHEProfileHostPrinting VT

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

#### Basic Syntax

```
Integer = HEProfileHostPrinting.VTPrinterTimeout  
HEProfileHostPrinting.VTPrinterTimeout = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrinterTimeout([out, retval] short  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrinterTimeout([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the printer delay.

*newVal*—The set value, specifying the printer delay.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
iVal = HostPrint.VTPrinterTimeout  
If (iVal = 0) Then  
    ' Set value  
    HostPrint.VTPrinterTimeout = 500  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrinterTimeout property
short sVal;
pHostPrint->get_VTPrinterTimeout(&sVal);
if (sVal == 0)
{
    sVal = 500;
    pHostPrint->put_VTPrinterTimeout(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrinterTimeoutValue

### Property, IHEProfileHostPrinting VT

This property returns or sets the timeout value for the DO\_PRNCTRL command.

#### Basic Syntax

```
Integer = HEProfileHostPrinting.VTPrinterTimeoutValue  
HEProfileHostPrinting.VTPrinterTimeoutValue = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrinterTimeoutValue([out, retval]  
short *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrinterTimeoutValue([in] short  
newVal);
```

#### Parameters

*pVal*—The returned value, specifying the timeout value for the command.  
*newVal*—The timeout value you want to set for the command.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
iVal = HostPrint.VTPrinterTimeoutValue  
If (iVal = 0) Then  
    ' Set value  
    HostPrint.VTPrinterTimeoutValue = 500  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrinterTimeoutValue property
short sVal;
pHostPrint->get_VTPrinterTimeoutValue(&sVal);
if (sVal == 0)
{
    sVal = 500;
    pHostPrint->put_VTPrinterTimeoutValue(sVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintFile

### Property, IHEProfileHostPrinting VT

This property returns or sets a string that specifies the path and name of the target file that you want HostExplorer to use when it prints to a file.

#### Basic Syntax

```
String = HEProfileHostPrinting.VTPrintFile  
HEProfileHostPrinting.VTPrintFile = String
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintFile([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrintFile([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the path and name of the target file for host printing.

*newVal*—The string that specifies the path and name of the target file for host printing.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
str = HostPrint.VTPrintFile  
If (Len(str) = 0) Then  
    ' Set value  
    HostPrint.VTPrintFile = "C:\MyData\print_output"  
    HostPrint.VTPrintTarget = HOSTEX_PRINT_TARGET_FILE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrintFile property
BSTR bstr;
pHostPrint->get_VTPrintFile(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("C:\\\\MyData\\\\print_output"));
    pHostPrint->put_VTPrintFile(bstr);
    SysFreeString(bstr);
    pHostPrint->put_VTPrintTarget(HOSTEX_PRINT_TARGET_FILE);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrint FileMode

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies which of the following methods HostExplorer uses to print to a file:

- Overwrite—HostExplorer overwrites the contents of the target file with the new material.
- Append—HostExplorer appends the new material to the existing contents of the target file.
- Auto number—HostExplorer prefixes a line number to each line it writes to the target file.

By default, the value for this property is HOSTEX\_PRINTFILE\_MODE\_OVERWRITE.

#### Basic Syntax

```
HOSTEX_PRINTFILE_MODE = HEProfileHostPrinting.VTPrint FileMode  
HEProfileHostPrinting.VTPrint FileMode = HOSTEX_PRINTFILE_MODE
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrint FileMode([out, retval]  
HOSTEX_PRINTFILE_MODE *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrint FileMode([in]  
HOSTEX_PRINTFILE_MODE newVal);
```

#### Parameters

*pVal*—The return value, specifying the method that HostExplorer uses to print to a file.

*newVal*—The set value, specifying the method you want HostExplorer to use when it prints to a file.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As HOSTEX_PRINTFILE_MODE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrint FileMode  
If (bVal <> HOSTEX_PRINTFILE_MODE_APPEND) Then  
    ' Set value  
    HostPrint.VTPrint FileMode = HOSTEX_PRINTFILE_MODE_APPEND  
    HostPrint.VTPrint Target = HOSTEX_PRINT_TARGET_FILE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrint FileMode property
HOSTEX_PRINTFILE_MODE bVal;
pHostPrint->get_VTPrint FileMode(&bVal);
if (bVal != HOSTEX_PRINTFILE_MODE_APPEND)
{
    bVal = HOSTEX_PRINTFILE_MODE_APPEND;
    pHostPrint->put_VTPrint FileMode(bVal);
    pHostPrint->put_VTPrint Target(HOSTEX_PRINT_TARGET_FILE);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintFitFontToPage

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer changes the point size of the font you have specified so that the font fits the print-output page.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTPrintFitFontToPage  
HEProfileHostPrinting.VTPrintFitFontToPage = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintFitFontToPage([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileHostPrinting::put_VTPrintFitFontToPage([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer changes the font size to fit the print-output page. If *pVal* equals VARIANT\_FALSE, HostExplorer does not change the font size.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to change the font size to fit the print-output page; otherwise, set *newVal* to VARIANT\_FALSE.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = Hostprint.VTPrintFitFontToPage  
If (bVal = False) Then  
    ' Set value  
    Hostprint.VTPrintFitFontToPage = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrintFitFontToPage property
VARIANT_BOOL bVal;
pHostPrint->get_VTPrintFitFontToPage(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint->put_VTPrintFitFontToPage(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintLFtoCRLF

### Property, IHEProfileHostPrinting VT

This property returns or sets a value that specifies whether HostExplorer expands linefeed (LF) characters to carriage return + linefeed (CR + LF) characters when printing.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTPrintLFtoCRLF
```

```
HEProfileHostPrinting.VTPrintLFtoCRLF = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintLFtoCRLF([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileHostPrinting::put_VTPrintLFtoCRLF([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer prints each linefeed character as a carriage return followed by a linefeed character. If *pVal* equals VARIANT\_FALSE, HostExplorer does not expand linefeed characters.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to expand linefeed characters during printing. Set *newVal* to VARIANT\_FALSE to print linefeed characters as is.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrintLFtoCRLF  
If (bVal = False) Then  
    ' Set value  
    HostPrint.VTPrintLFtoCRLF = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrintLFtoCRLF property
VARIANT_BOOL bVal;
pHostPrint->get_VTPrintLFtoCRLF(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint->put_VTPrintLFtoCRLF(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTPrintTarget

### Property, IHEProfileHostPrinting VT

This property specifies which of the following targets HostExplorer uses for host printing:

- Default printer
- Specified printer
- Specified file

The default value for this property is HOSTEX\_PRINT\_TARGET\_DEFAULT\_PRT.

#### Basic Syntax

```
HOSTEX_PRINT_TARGET = HEProfileHostPrinting.VTPrintTarget  
HEProfileHostPrinting.VTPrintTarget = HOSTEX_PRINT_TARGET
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTPrintTarget([out, retval]  
HOSTEX_PRINT_TARGET *pVal);  
  
HRESULT IHEProfileHostPrinting::put_VTPrintTarget([in]  
HOSTEX_PRINT_TARGET newVal);
```

#### Parameters

*pVal*—The return value, specifying the target for host printing.

*newVal*—The set value, specifying the target you want HostExplorer to use for printing.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As HOSTEX_PRINT_TARGET  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTPrintTarget  
If (bVal <> HOSTEX_PRINT_TARGET_FILE) Then  
    ' Set value  
    HostPrint.VTPrintFile = "C:\print_output"  
    HostPrint.VTPrintTarget = HOSTEX_PRINT_TARGET_FILE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTPrintTarget property
HOSTEX_PRINT_TARGET bVal;
BSTR bstr;
pHostPrint->get_VTPrintTarget(&bVal);
if (bVal != HOSTEX_PRINT_TARGET_FILE)
{
    bstr = SysAllocString(OLESTR("C:\\\\print_output"));
    pHostPrint->put_VTPrinterFile(bstr);
    SysFreeString(bstr);
    bVal = HOSTEX_PRINT_TARGET_FILE;
    pHostPrint->put_VTPrintTarget(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTUseSpecificPrinter

### Property, IHEProfileHostPrinting VT

This property determines whether HostExplorer prints using the printer displayed in the Selected Printer Info box. When this option is disabled, HostExplorer uses the default printer. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileHostPrinting.VTUseSpecificPrinter  
HEProfileHostPrinting.VTUseSpecificPrinter = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHostPrinting::get_VTUseSpecificPrinter([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileHostPrinting::put_VTUseSpecificPrinter([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer prints using the specified printer. A returned value of VARIANT\_FALSE indicates that HostExplorer prints using the default printer.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer prints using the specified printer. A value of VARIANT\_FALSE indicates that HostExplorer prints using the default printer.

#### Basic Example

```
Dim Profile As HEProfile  
Dim HostPrinting As HEProfileHostPrinting  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set HostPrinting = Profile.HostPrinting  
' Get value  
bVal = HostPrint.VTUseSpecificPrinter  
If (bVal= False) Then  
    ' Set value  
    HostPrint.VTUseSpecificPrinter = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHostPrinting interface
IHEProfileHostPrinting* pHostPrinting;
pIProfile->get_HostPrinting ( & pHostPrinting );
//Get VTUseSpecificPrinter property
VARIANT_BOOL bVal;
pHostPrint->get_VTUseSpecificPrinter(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pHostPrint-> put_VTUseSpecificPrinter(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileHotspots Interface

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

### Properties

The ProfileHotspots interface consists of the following properties:

- `DisplayStyle`
- `EnableHotspots`
- `MouseActivation`
- `Schemes`

### DisplayStyle

**Property, IHEProfileHotspots** **3270** **5250** **VT**

This property returns or sets a value that specifies how HostExplorer displays hotspots. The hotspot display style can be one of the following:

- Invisible—The hotspot region or text appears in its regular display style until you move the cursor over it, at which point the cursor changes into a hand.
- Raised Button—The hotspot region or text always appears highlighted.

By default, this property is set to `HOSTEX_HOTSPOT_DISPLAY_RAISED_BUTTON`.

#### Basic Syntax

```
HOSTEX_HOTSPOT_DISPLAY = HEProfileHotspots.DisplayStyle  
HEProfileHotspots.DisplayStyle = HOSTEX_HOTSPOT_DISPLAY
```

#### C++ Syntax

```
HRESULT IHEProfileHotspots::get_DisplayStyle([out, retval]  
HOSTEX_HOTSPOT_DISPLAY *pVal);  
  
HRESULT IHEProfileHotspots::put_DisplayStyle([in] HOSTEX_HOTSPOT_DISPLAY  
newVal);
```

#### Parameters

*pVal*—The return value, specifying the hotspot display style (either invisible or raised button).

*newVal*—The set value, specifying the hotspot display style that you want HostExplorer to use.

**Basic Example**

```
Dim Profile As HEProfile
Dim Hotspots As HEProfileHotspots
Dim dVal As HOSTEX_HOTSPOT_DISPLAY
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Hotspots = Profile.Hotspots
' Get value
dVal = HotSpot.DisplayStyle
If (dVal = HOSTEX_HOTSPOT_DISPLAY_INVISIBLE) Then
    ' Set value
    HotSpot.DisplayStyle = HOSTEX_HOTSPOT_DISPLAY_RAISED_BUTTON
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileHotspots Interface
    IHEProfileHotspots* pHotspos;
    pIProfile->get_Hotspos ( &pHotspots );
    //Get DisplayStyle Property
    HOSTEX_HOTSPOT_DISPLAY dVal;
    pHotspos->get_DisplayStyle(&dVal);
    if (dVal == HOSTEX_HOTSPOT_DISPLAY_INVISIBLE)
    {
        dVal = HOSTEX_HOTSPOT_DISPLAY_RAISED_BUTTON;
        pHotspos->put_DisplayStyle(mVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnableHotspots

### Property, IHEProfileHotspots 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer enables hotspots on the host screen.

#### Basic Syntax

```
Boolean = HEProfileHotspots.EnableHotspots
```

```
HEProfileHotspots.EnableHotspots = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileHotspots::get_EnableHotspots([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfileHotspots::put_EnableHotspots([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer enables hotspots on the host screen. If *pVal* equals VARIANT\_FALSE, HostExplorer disables hotspots.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to enable hotspots on the host screen. Set *newVal* to VARIANT\_FALSE to disable hotspots.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Hotspots As HEProfileHotspots  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Hotspots = Profile.Hotspots  
' Get value  
bVal = HotSpot.EnableHotspots  
If (bVal = False) Then  
    ' Set value  
    HotSpot.EnableHotspots = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileHotspots Interface
    IHEProfileHotspots* pHotspots;
    pIProfile->get_Hotspots ( & pHotspots );
    //Get EnableHotspots Property
    VARIANT_BOOL bVal;
    pHotSpot->get_EnableHotspots(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pHotSpot->put_EnableHotspots(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## MouseActivation

### Property, IHEProfileHotspots 3270 5250 VT

This property returns or sets a value that specifies how hotspots are activated using a mouse. By default, this property is set to HOSTEX\_HOTSPOT\_MOUSE\_ACTIVATION\_SINGLE\_CLICK.

#### Basic Syntax

```
HOSTEX_HOTSPOT_MOUSE_ACTIVATION = HEProfileHotspots.MouseActivation  
HEProfileHotspots.MouseActivation = HOSTEX_HOTSPOT_MOUSE_ACTIVATION
```

#### C++ Syntax

```
HRESULT IHEProfileHotspots::get_MouseActivation([out, retval]  
HOSTEX_HOTSPOT_MOUSE_ACTIVATION *pVal);  
HRESULT IHEProfileHotspots::put_MouseActivation([in]  
HOSTEX_HOTSPOT_MOUSE_ACTIVATION newVal);
```

#### Parameters

*pVal*—The return value, specifying the hotspot mouse-activation mode (either single-click or double-click).

*newVal*—The set value, specifying the hotspot mouse-activation mode that you want HostExplorer to use.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Hotspots As HEProfileHotspots  
Dim mVal As HOSTEX_HOTSPOT_MOUSE_ACTIVATION  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Hotspots = Profile.Hotspots  
' Get value  
mVal = HotSpot.MouseActivation  
If (mVal = HOSTEX_HOTSPOT_MOUSE_ACTIVATION_DOUBLE_CLICK) Then  
    ' Set value  
    HotSpot.MouseActivation = HOSTEX_HOTSPOT_MOUSE_ACTIVATION_SINGLE_CLICK  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileHotspots Interface
    IHEProfileHotspots* pHotspots;
    pIProfile->get_Hotspots ( &pHotspots );
    //Get MouseActivation Property
    HOSTEX_HOTSPOT_MOUSE_ACTIVATION mVal;
    pHotSpot->get_MouseActivation(&mVal);
    if (mVal == HOSTEX_HOTSPOT_MOUSE_ACTIVATION_DOUBLE_CLICK)
    {
        mVal = HOSTEX_HOTSPOT_MOUSE_ACTIVATION_SINGLE_CLICK;
        pHotSpot->put_MouseActivation(mVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Schemes

### Property, IHEProfileHotspots 3270 5250 VT

This property returns or sets a string that specifies the name of the hotspot scheme for the current session. A scheme is a collection of settings.

#### Basic Syntax

```
String = HEProfileHotspots.Schemes
```

```
HEProfileHotspots.Schemes = String
```

#### C++ Syntax

```
HRESULT IHEProfileHotspots::get_Schemes([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileHotspots::put_Schemes([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, specifying the hotspot scheme currently in effect.

*newVal*—The set value, specifying the hotspot scheme that you want to use for the current session.

#### Basic Example

```
Dim Profile As HEProfile
Dim Hotspots As HEProfileHotspots
Dim str As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Hotspots = Profile.Hotspots
' Get value
str = HotSpot.Schemes
If (Len(str) = 0) Then
    ' Set value
    HotSpot.Schemes = "Default"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileHotspots Interface
IHEProfileHotspots* pHotspos;
pIProfile->get_Hotspos ( &pHotspots );
//Get Schemes Property
BSTR bstr;
pHotSpot->get_Schemes(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Default"));
    pHotSpot->put_Schemes(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileEvents Interface

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

### Methods

The ProfileEvents interface has the following method:

RemoveEvent

### Properties

The ProfileEvents interface consists of the following properties:

- EnableEvents
- Schemes

### RemoveEvent

**Method, IHEProfileEvents 3270 5250 VT**

This method lets you delete a particular event from the current event scheme. You specify the event as a number that corresponds to its position in the event list.

#### Basic Syntax

```
HEProfileEvents.RemoveEvent (nIndex As Integer)
```

#### C++ Syntax

```
HRESULT IHEProfileEvents::RemoveEvent([in] short nIndex);
```

#### Parameters

*nIndex*—The position of the event in the event list. The first event in the list has position 1.

#### Basic Example

```
Dim Event As HEProfileEvents
Set Event = Profile.Events
' Remove the 3rd event from the scheme
Event.RemoveEvent(3)
```

#### C++ Example

```
IHEProfileEvents *pEvent;
pSess->get_Events(&pEvent);
// Remove the 3rd event from the scheme
pEvent->RemoveEvent(3);
```

## EnableEvents

### Property, IHEProfileEvents 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer enables the events that you have programmed.

#### Basic Syntax

```
Boolean = HEProfileEvents.EnableEvents  
HEProfileEvents.EnableEvents = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileEvents::get_EnableEvents([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileEvents::put_EnableEvents([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The return value. If *pVal* equals VARIANT\_TRUE, HostExplorer enables your programmed events. If *pVal* equals VARIANT\_FALSE, HostExplorer disables those events.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to enable the events you have programmed. Set *newVal* to VARIANT\_FALSE to disable those events.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Events As HEProfileEvents  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Events = Profile.Events  
' Get value  
bVal = Event.EnableEvents  
If (bVal = False) Then  
    ' Set value  
    Event.EnableEvents = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileEvents Interface
    IHEProfileEvents* pEvents;
    pIProfile->get_Events ( & pEvents );
    //Get EnableEvents Property
    VARIANT_BOOL bVal;
    pEvent->get_EnableEvents(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pEvent->put_EnableEvents (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Schemes

### Property, IHEProfileEvents 3270 5250 VT

This property returns or sets a string that specifies the name of the event scheme for the current session. A scheme is a collection of settings.

#### Basic Syntax

```
String = HEProfileEvents.Schemes  
HEProfileEvents.Schemes = String
```

#### C++ Syntax

```
HRESULT IHEProfileEvents::get_Schemes([out, retval] BSTR *pVal);  
HRESULT IHEProfileEvents::put_Schemes([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, specifying the event scheme currently in effect.  
*newVal*—The set value, specifying the event scheme that you want to use for the current session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Events As HEProfileEvents  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Events = Profile.Events  
' Get value  
str = Event.Schemes  
If (Len(str) = 0) Then  
    ' Set value  
    Event.Schemes = "event_scheme_1"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileEvents Interface
IHEProfileEvents* pEvents;
pIProfile->get_Events ( & pEvents );
//Get Schemes Property
BSTR bstr;
pEvent->get_Schemes(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("event_scheme_1"));
    pEvent->put_Schemes(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileConnection Interface

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

### Properties

The ProfileConnection interface consists of the following properties:

- acPassword
- AlwaysPromptForHostName
- AreaCode
- AutoEndMacroName
- AutoMacroName
- AutoRunMacroDelayTime
- AutoSignOn
- BackspaceKeyInterpretation
- ConnectBy
- Country
- CountryCode
- CountryID
- DeviceName
- DirectToModem
- EnableEMode
- EnableInfiniteRetries
- EnterKeyInterpretation
- HostName
- KeyboardBufferMode
- LUName
- Modem
- ModemID
- NumberOfRetries
- Password
- Port
- PortList
- RetryDelayTimeBetweenHosts
- ShowDialupDlg
- SilentConnect
- SYSREQasIACIP
- TelnetEcho
- TelnetName
- Timeout
- UponDisconnect
- UseDialProp
- UserName
- VTDoHostWindowSize
- VTInitiateTelnetNegotiation
- VTLineMode

## acPassword

### Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a string that specifies a password to use for creating Quick-Keys. The password can have up to 32 characters inclusive. HostExplorer stores and displays the password in encrypted form.

#### Basic Syntax

```
String = HEProfileConnection.acPassword  
HEProfileConnection.acPassword = String
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_acPassword([out, retval] BSTR *pVal);  
HRESULT IHEProfileConnection::put_acPassword([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the password for creating Quick-Keys.

*newVal*—The set string, specifying the password you want to use for creating Quick-Keys.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim str As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
str = connect.acPassword  
If (Len(str) = 0) Then  
    ' Set value  
    connect.acPassword = "word"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get acPassword Property
    BSTR bstr;
    pconnect->get_acPassword(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr != NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("word"));
        pconnect->put_acPassword(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AlwaysPromptForHostName

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer prompts you for a host name whenever you attempt to connect to a host using the current session profile.

### Basic Syntax

```
Boolean = HEProfileConnection.AlwaysPromptForHostName  
HEProfileConnection.AlwaysPromptForHostName = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_AlwaysPromptForHostName([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileConnection::put_AlwaysPromptForHostName([in]  
VARIANT_BOOL newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer prompts you for a host name whenever you attempt to connect to a host. If *pVal* equals VARIANT\_FALSE, HostExplorer uses the host name stored in the profile.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer prompts you for a host name whenever you attempt to connect to a host. Set *newVal* to VARIANT\_FALSE if you always want HostExplorer to use the host name stored in the profile.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.AlwaysPromptForHostName  
If (bVal = False) Then  
    ' Set value  
    Connection.AlwaysPromptForHostName = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection );
    //Get AlwaysPromptForHostName Property
    VARIANT_BOOL bVal;
    bVal = pConn->get_AlwaysPromptForHostName (&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pConn->put_AlwaysPromptForHostName (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AreaCode

**Property, IHEProfileConnection 3270 5250 VT**

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

### Basic Syntax

```
Long = HEProfileConnection.AreaCode  
HEProfileConnection.AreaCode = Long
```

### C++ Syntax

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

### Parameters

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

### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim lVal As Long  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
lVal = Connection.AreaCode  
If (lVal = 0) Then  
    ' Set value  
    Connection.AreaCode = 514  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get AreaCode Property
    long lVal;
    pConn->get_AreaCode(&lVal);
    if (lVal!=514)
    {
        lVal=514;
        pConn->put_AreaCode(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoEndMacroName

Property, IHEProfileConnection **3270 5250 VT**

This property returns or sets a string that specifies the name of the macro that you want to terminate each time you start a new session.

### Basic Syntax

```
String = HEProfileConnection.AutoEndMacroName  
HEProfileConnection.AutoEndMacroName = String
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_AutoEndMacroName([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileConnection::put_AutoEndMacroName([in] BSTR newVal);
```

### Parameters

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

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

### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get string  
strVal = Connection.AutoEndMacroName  
If (Len(strVal) = 0) Then  
    ' Set string  
    Connection.AutoEndMacroName = "C:\\a.ebs"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection);
    //Get AutoEndMacroName Property
    BSTR bstr ;
    pConn->get_AutoEndMacroName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("C:\\a.ebs"));
        pConn->put_AutoEndMacroName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoMacroName

Property, IHEProfileConnection 3270 5250 VT

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

### Basic Syntax

```
String = HEProfileConnection.AutoMacroName
```

```
HEProfileConnection.AutoMacroName = String
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_AutoMacroName([out, retval] BSTR  
*pVal);
```

```
HRESULT IHEProfileConnection::put_AutoMacroName([in] BSTR newVal);
```

### Parameters

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

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

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get string  
strVal = Connection.AutoMacroName  
If (Len(strVal) = 0) Then  
    ' Set string  
    Connection.AutoMacroName = "C:\\a.ebs"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get AutoMacroName Property
    BSTR bstr ;
    pConn->get_AutoMacroName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("C:\\a.ebs"));
        pConn->put_AutoMacroName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoRunMacroDelayTime

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets the time (in seconds) that HostExplorer waits before it launches the start-up macro whenever you start a new session. By default, the delay is set to zero.

### Basic Syntax

```
Long = HEProfileConnection.AutoRunMacroDelayTime  
HEProfileConnection.AutoRunMacroDelayTime = Long
```

### C++ Syntax

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

### Parameters

*pVal*—The returned value, specifying the time that HostExplorer waits before it launches the start-up macro.

*newVal*—The set value, specifying the time that you want HostExplorer to wait before it launches the start-up macro.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim lVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
lVal = Connection.AutoRunMacroDelayTime  
If (lVal < 20) Then  
' Set value  
Connection.AutoRunMacroDelayTime = 20  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection );
    //Get AutoRunMacroDelayTime Property
    long lVal;
    pConn->get_AutoRunMacroDelayTime(&lVal);
    if (lVal < 20)
    {
        lVal = 20;
        pConn->put_AutoRunMacroDelayTime(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## AutoSignOn

### Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer automatically logs onto the host (using the supplied AS/400 user ID and password) and bypasses the sign-on screen when a connection to the host is established. You must supply an AS/400 user ID and password prior to enabling this property.

#### Basic Syntax

```
Boolean = HEProfileConnection.AutoSignOn  
HEProfileConnection.AutoSignOn = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_AutoSignOn([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileConnection::put_AutoSignOn([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer automatically logs onto the host when a connection to the host is established. If *pVal* equals VARIANT\_FALSE, HostExplorer does not automatically log onto the host.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer logs onto the host when the connection is established. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to automatically log onto the host.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.AutoSignOn  
If (bVal = False) Then  
    ' Set value  
    Connection.AutoSignOn = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get AutoSignOn Property
    VARIANT_BOOL bVal;
    bVal = pConn->get_AutoSignOn(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pConn->put_AutoSignOn(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## BackspaceKeyInterpretation

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies the command that HostExplorer sends to the host when you press the Backspace key. By default, this property is set to HOSTEX\_BACKSPACE\_KEY\_AS\_DELETE.

### Basic Syntax

```
HOSTEX_BACKSPACE_KEY_INTERPRETATION =
IHEProfileConnection.BackspaceKeyInterpretation

IHEProfileConnection.BackspaceKeyInterpretation =
HOSTEX_BACKSPACE_KEY_INTERPRETATION
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_BackspaceKeyInterpretation([out,
retval] HOSTEX_BACKSPACE_KEY_INTERPRETATION *pVal);

HRESULT IHEProfileConnection::put_BackspaceKeyInterpretation([in]
HOSTEX_BACKSPACE_KEY_INTERPRETATION newVal);
```

### Parameters

*pVal*—The returned value, specifying the command that HostExplorer sends to the host when you press the Backspace key.

*newVal*—The set value, specifying the command you want HostExplorer to send to the host when you press the Backspace key.

### Basic Example

```
Dim Profile As HEProfile
Dim Connection As HEProfileConnection
Dim bVal As HOSTEX_BACKSPACE_KEY_INTERPRETATION
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get value
bVal = Connection.BackspaceKeyInterpretation
If (bVal = HOSTEX_BACKSPACE_KEY_AS_BACKSPACE) Then
    ' Set value
    Connection.BackspaceKeyInterpretation =
        HOSTEX_BACKSPACE_KEY_AS_DELETE
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection );
    //Get BackspaceKeyInterpretation Property
    HOSTEX_BACKSPACE_KEY_INTERPRETATION bVal;
    bVal = pConn->get_BackspaceKeyInterpretation(&bVal);
    if (bVal == HOSTEX_BACKSPACE_KEY_AS_BACKSPACE)
    {
        bVal = HOSTEX_BACKSPACE_KEY_AS_DELETE;
        pConn->put_BackspaceKeyInterpretation(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ConnectBy

**Property, IHEProfileConnection 3270 5250 VT**

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

### Basic Syntax

```
HOSTEX_CONNECT_BY = HEProfileConnection.ConnectBy
```

```
HEProfileConnection.ConnectBy = HOSTEX_CONNECT_BY
```

### C++ Syntax

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

```
HRESULT IHEProfileConnection::put_ConnectBy([in] HOSTEX_CONNECT_BY  
newVal);
```

### Parameters

*pVal*—The returned value, indicating the connection.

*newVal*—The set value, indicating the connection.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim ConnBy As HOSTEX_CONNECT_BY  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
ConnBy = Connection.ConnectBy  
If (ConnBy <> HOSTEX_CONNECT_BY_MSSNA) Then  
    Connection.ConnectBy = HOSTEX_CONNECT_BY_MSSNA  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get ConnectBy Property
    HOSTEX_CONNECT_BY ConnBy;
    pConn->get_ConnectBy(&ConnBy);
    if (ConnBy != HOSTEX_CONNECT_BY_MSSNA)
    {
        ConnBy = HOSTEX_CONNECT_BY_MSSNA;
        pConn->put_ConnectBy(ConnBy);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Country

### Property, IHEProfileConnection VT

This property returns or sets a string specifying the country you are calling when you use a modem connection.

#### Basic Syntax

```
String = HEProfileConnection.Country
```

```
HEProfileConnection.Country = String
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_Country([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileConnection::put_Country([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, which indicates the country you are calling.

*newVal*—The set string, which indicates the name of the country you are calling.

#### Basic Example

```
Dim Profile As HEPProfile
Dim Connection As HEProfileConnection
Dim strVal As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_VT
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get string
strVal = Connection.Country
If (Len(strVal) = 0) Then
    ' Set string
    Connection.Country = "Canada"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get Country Property
BSTR bstr;
pConn->get_Country(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Canada"));
    pConn->put_Country(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CountryCode

Property, IHEProfileConnection **3270 5250 VT**

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

### Basic Syntax

```
Long = HEProfileConnection.CountryCode  
HEProfileConnection.CountryCode = Long
```

### C++ Syntax

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

### Parameters

*pVal*—The returned value, specifying the country code.

*newVal*—The set value, specifying the country code.

### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim lVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
lVal = Connection.CountryCode  
If (lVal = 0) Then  
    ' Set value  
    Connection.CountryCode = 9133  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get CountryCode Property
    long lVal;
    pConn->get_CountryCode(&lVal);
    if (lVal == 0)
    {
        lVal =9133;
        pConn->put_CountryCode(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## CountryID

Property, IHEProfileConnection **3270 5250 VT**

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

### Basic Syntax

```
Long = HEProfileConnection.CountryID
```

```
HEProfileConnection.CountryID = Long
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_CountryID([out, retval] long *pVal);
```

```
HRESULT IHEProfileConnection::put_CountryID([in] long newVal);
```

### Parameters

*pVal*—The returned value specifying the country ID.

*newVal*—The set value specifying the country ID.

### Basic Example

```
Dim Profile As HEPProfile
Dim Connection As HEProfileConnection
Dim lVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get value
lVal = Connection.CountryID
If (lVal = 0) Then
    ' Set value
    Connection.CountryID = 47
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get CountryID Property
    long lVal;
    pConn->get_CountryID(&lVal);
    if (lVal == 0)
    {
        lVal = 47;
        pConn->put_CountryID(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DeviceName

### Property, IHEProfileConnection 5250

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

#### Basic Syntax

```
String = HEProfileConnection.DeviceName  
HEProfileConnection.DeviceName = String
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value, which indicates whether to connect to a default or specific device name.

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_5250  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
strVal = Connection.DeviceName  
If (Len(strVal) = 0) Then  
    ' Set Value  
    Connection.DeviceName = "TTS"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get <DeviceName Property
    BSTR bstr ;
    pConn->get_DeviceName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("TTS"));
        pConn->put_DeviceName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## DirectToModem

### Property, IHEProfileConnection VT

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

#### Basic Syntax

```
Boolean = HEProfileConnection.DirectToModem
```

```
HEProfileConnection.DirectToModem = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_DirectToModem([out, retval]  
VARIANT_BOOL *pVal);
```

```
HRESULT IHEProfileConnection::put_DirectToModem([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the DirectToModem flag is set. A returned value of VARIANT\_FALSE indicates that the DirectToModem flag is not set.

*newVal*—A value of VARIANT\_TRUE indicates that the DirectToModem flag is set. A value of VARIANT\_FALSE indicates that the DirectToModem flag is not set.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.DirectToModem  
If (bVal = False) Then  
    ' Set value  
    Connection.DirectToModem = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get DirectToModem Property
VARIANT_BOOL bVal;
bVal = pConn->get_DirectToModem(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pConn->put_DirectToModem(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnableEMode

### Property, IHEProfileConnection 3270

This property returns or sets a value that specifies whether HostExplorer connects to hosts using the TN3270E protocol. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileConnection.EnableEMode
```

```
HEProfileConnection.EnableEMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_EnableEMode([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHEProfileConnection::put_EnableEMode([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer connects to hosts using the TN3270E protocol. If *pVal* equals VARIANT\_FALSE, HostExplorer does not connect using this protocol.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer connects to hosts using the TN3270E protocol; otherwise, set *newVal* to VARIANT\_FALSE.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Sesscon.EnableEMode  
If (bVal = False) Then  
    ' Set value  
    Sesscon.EnableEMode = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get EnableEMode Property
VARIANT_BOOL bVal;
bVal = pConn->get_EnableEMode(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pConn->put_EnableEMode(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnableInfiniteRetries

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer continually attempts to connect to all the hosts listed in the Hosts pane until one becomes available.

### Basic Syntax

```
Boolean = HEProfileConnection.EnableInfiniteRetries  
HEProfileConnection.EnableInfiniteRetries = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_EnableInfiniteRetries([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileConnection::put_EnableInfiniteRetries([in] VARIANT_BOOL  
newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer attempts to connect to all hosts until one becomes available. If *pVal* equals VARIANT\_FALSE, HostExplorer does not attempt to connect continually.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer attempts to connect to all listed hosts until one becomes available. Set *newVal* to VARIANT\_FALSE if you do not want HostExplorer to continually attempt to connect.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.EnableInfiniteRetries  
If (bVal = False) Then  
    ' Set value  
    Connection.EnableInfiniteRetries = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection);
    //Get EnableInfiniteRetries Property
    VARIANT_BOOL bVal;
    bVal = pConn->get_EnableInfiniteRetries(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pConn->put_EnableInfiniteRetries (bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnterKeyInterpretation

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies the key sequence that HostExplorer sends to the host when you press the Enter key. By default, this property is set to HOSTEX\_ENTER\_KEY\_AS\_RETURN\_AND\_LINEFEED.

### Basic Syntax

```
HOSTEX_ENTER_KEY_INTERPRETATION =
HEProfileConnection.EnterKeyInterpretation

HEProfileConnection.EnterKeyInterpretation =
HOSTEX_ENTER_KEY_INTERPRETATION
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_EnterKeyInterpretation([out, retval]
HOSTEX_ENTER_KEY_INTERPRETATION *pVal);

HRESULT IHEProfileConnection::put_EnterKeyInterpretation([in]
HOSTEX_ENTER_KEY_INTERPRETATION newVal);
```

### Parameters

*pVal*—The returned value, specifying the key sequence that HostExplorer sends to the host when you press the Enter key.

*newVal*—The set value, specifying the key sequence you want HostExplorer to send to the host when you press Enter.

### Basic Example

```
Dim Profile As HEProfile
Dim Connection As HEProfileConnection
Dim bVal As HOSTEX_ENTER_KEY_INTERPRETATION
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get value
bVal = Connection.EnterKeyInterpretation
If (bVal = HOSTEX_ENTER_KEY_AS_CARRIAGE_RETURN) Then
    ' Set value
    Connection.EnterKeyInterpretation =
        HOSTEX_ENTER_KEY_AS_RETURN_AND_LINEFEED
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection );
    //Get EnterKeyInterpretation Property
    HOSTEX_ENTER_KEY_INTERPRETATION bVal;
    bVal = pConn->get_EnterKeyInterpretation(&bVal);
    if (bVal == HOSTEX_ENTER_KEY_AS_CARRIAGE_RETURN)
    {
        bVal = HOSTEX_ENTER_KEY_AS_RETURN_AND_LINEFEED;
        pConn->put_EnterKeyInterpretation(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## HostName

**Property, IHEProfileConnection 3270 5250 VT**

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

### Basic Syntax

```
String = HEProfileConnection.HostName  
HEProfileConnection.HostName = String
```

### C++ Syntax

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

### Parameters

*pVal*—The returned string, indicating the host name.

*newVal*—The set string, indicating the host name.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get string  
strVal = Connection.HostName  
If (Len(strVal) = 0) Then  
    ' Set string  
    Connection.HostName = "aix"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL, CLSCTX_INPRO
C_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_
5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection );
//Get HostName Property
BSTR bstr;
pConn->get_HostName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("aix"));
    pConn->put_HostName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KeyboardBufferMode

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies how HostExplorer stores characters in a buffer until they are sent to the host. The buffer mode can be one of the following:

- Character Mode—Forces HostExplorer to send each character immediately to the host.
- Line Mode—Forces HostExplorer to send characters one line at a time until you press the Enter key. Line mode is useful when you are trying to reduce costs on networks that charge per packet, or when you are dealing with long network delays.

By default, this property is set to

HOSTEX\_KEYBOARD\_BUFFER\_AS\_CHARACTER\_MODE.

### Basic Syntax

```
HOSTEX_KEYBOARD_BUFFER_MODE = HEProfileConnection.KeyboardBufferMode  
HEProfileConnection.KeyboardBufferMode = HOSTEX_KEYBOARD_BUFFER_MODE
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_KeyboardBufferMode ([out, retval]  
HOSTEX_KEYBOARD_BUFFER_MODE *pVal);  
  
HRESULT IHEProfileConnection::put_KeyboardBufferMode ([in]  
HOSTEX_KEYBOARD_BUFFER_MODE newVal);
```

### Parameters

pVal—The returned value, specifying the buffer mode for key sequences.

newVal—The set value, specifying the buffer mode you want HostExplorer to use for key sequences.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As HOSTEX_KEYBOARD_BUFFER_MODE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.KeyboardBufferMode  
If (bVal = HOSTEX_KEYBOARD_BUFFER_AS_LINE_MODE) Then  
    ' Set value  
    Connection.KeyboardBufferMode =  
    HOSTEX_KEYBOARD_BUFFER_AS_CHARACTER_MODE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection);
    //Get KeyboardBufferMode Property
    HOSTEX_KEYBOARD_BUFFER_MODE bVal;
    bVal = pConn->get_KeyboardBufferMode(&bVal);
    if (bVal == HOSTEX_KEYBOARD_BUFFER_AS_LINE_MODE)
    {
        bVal = HOSTEX_KEYBOARD_BUFFER_AS_CHARACTER_MODE;
        pConn->put_KeyboardBufferMode(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LUName

### Property, IHEProfileConnection 3270

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

#### Basic Syntax

```
String = HEProfileConnection.LUName
```

```
HEProfileConnection.LUName = String
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_LUName([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileConnection::put_LUName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the LU name.

*newVal*—The set string, specifying the LU name.

#### Basic Example

```
Dim Profile As HEProfile
Dim Connection As HEProfileConnection
Dim strVal As String
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get string
strVal = Connection.LUName
If (Len(strVal) = 0) Then
    ' Set string
    Connection.LUName = "XYZ"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get LUName Property
BSTR bstr ;
pConn->get_LUName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("XYZ"));
    pConn->put_LUName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Modem

### Property, IHEProfileConnection VT

This property returns or sets a string specifying the modem you are using for a modem connection.

#### Basic Syntax

```
String = HEProfileConnection.Modem
```

```
HEProfileConnection.Modem = String
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_Modem([out, retval] BSTR *pVal);  
HRESULT IHEProfileConnection::put_Modem([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, which indicates the modem you are using.

*newVal*—The set string, which indicates the modem you are using.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get string  
strVal = Connection.Modem  
If (Len(strVal) = 0) Then  
    ' Set string  
    Connection.Modem = "modemx"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get Modem Property
BSTR bstr;
pConn->get_Modem(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("modemx"));
    pConn->put_Modem(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ModemID

Property, IHEProfileConnection 3270 5250 VT

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

### Basic Syntax

```
Long = HEProfileConnection.ModemID
```

```
HEProfileConnection.ModemID = Long
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_ModemID([out, retval] long *pVal);
```

```
HRESULT IHEProfileConnection::put_ModemID([in] long newVal);
```

### Parameters

*pVal*—The returned value, specifying the modem ID.

*newVal*—The set value, specifying the modem ID.

### Basic Example

```
Dim Profile As HEPProfile
Dim Connection As HEProfileConnection
Dim lVal As Long
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
lVal = Connection.ModemID
If (lVal = 6734) Then
    ' Add code
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get ModemID Property
    long lVal;
    pConn->get_ModemID(&lVal);
    if (lVal == 0)
    {
        lVal = 6734;
        pConn->put_ModemID(lVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## NumberOfRetries

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value that specifies the number of times that HostExplorer attempts to connect to all the hosts listed in the Hosts pane until a host becomes available.

### Basic Syntax

```
Integer = HEProfileConnection.NumberOfRetries  
HEProfileConnection.NumberOfRetries = Integer
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_NumberOfRetries([out, retval] short  
*pVal);  
HRESULT IHEProfileConnection::put_NumberOfRetries([in] short newVal);
```

### Parameters

*pVal*—The returned value, specifying the current number of retries.

*newVal*—The set value, specifying the number of times that you want HostExplorer to attempt to connect.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
iVal = Connection.NumberOfRetries  
If (iVal < 20) Then  
    Connection.NumberOfRetries = 20  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get NumberOfRetries Property
    short sVal;
    pConn->get_NumberOfRetries(&sVal);
    if (sVal < 20)
    {
        sVal= 20;
        pConn->put_NumberOfRetries(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Password

### Property, IHEProfileConnection 5250

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

#### Basic Syntax

```
String = HEProfileConnection.Password  
HEProfileConnection.Password = String
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
strVal = Connection.Password  
If (Len(strVal) = 0) Then  
    Connection.Password = "as14rty"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get Password Property
    BSTR bstr ;
    pConn->get_Password(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("as14rty "));
        pConn->put_Password(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Port

### Property, IHEProfileConnection 3270 5250 VT

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

#### Basic Syntax

```
Integer = HEProfileConnection.Port
```

```
HEProfileConnection.Port = Integer
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_Port([out, retval] short *pVal);
```

```
HRESULT IHEProfileConnection::put_Port([in] short newVal);
```

#### Parameters

*pVal*—The returned value, specifying the Internet port.

*newVal*—The set value, specifying the Internet port.

#### Basic Example

```
Dim Profile As HEProfile
Dim Connection As HEProfileConnection
Dim iVal As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get value
iVal = Connection.Port
If (iVal <> 23) Then
    ' Set value
    Connection.Port = 23
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get Port Property
    short sVal;
    pConn->get_Port(&sVal);
    if (sVal == 0)
    {
        sVal=23;
        pConn->put_Port(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PortList

**Property, IHEProfileConnection 3270 5250 VT**

This property returns or sets a string that lists the port numbers you want to use. Use a semicolon (;) to separate successive port numbers in the string.

### Basic Syntax

```
String = HEProfileConnection.PortList
HEProfileConnection.PortList = String
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_PortList([out, retval] BSTR *pVal);
HRESULT IHEProfileConnection::put_PortList([in] BSTR newVal);
```

### Parameters

*pVal*—The returned string, specifying the list of ports.

*newVal*—The set string, specifying the list of ports that you want to use.

### Basic Example

```
Dim Profile As HEPProfile
Dim Connection As HEProfileConnection
Dim ports As String
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get port list
ports = Connection.PortList
If (Len(ports) = 0) Then
    ' Set port list
    Connection.PortList = "1;23;80;8080"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get PortList Property
BSTR ports;
pConn->get_PortList(&ports);
if (strlen(OLE2A(ports)) == 0)
{
    if (ports != NULL)
        SysFreeString(ports);
    ports = SysAllocString(OLESTR("1;23;80;8080"));
    pConn->put_PortList(ports);
    SysFreeString(ports);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## RetryDelayTimeBetweenHosts

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets the delay (in seconds) before HostExplorer attempts to connect to another host after attempting to connect to an unavailable one.

### Basic Syntax

```
Integer = HEProfileConnection.RetryDelayTimeBetweenHosts  
HEProfileConnection.RetryDelayTimeBetweenHosts = Integer
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_RetryDelayTimeBetweenHosts([out,  
retval] short *pVal);  
HRESULT IHEProfileConnection::put_RetryDelayTimeBetweenHosts([in] short  
newVal);
```

### Parameters

*pVal*—The returned value, specifying the current number of seconds that elapse before HostExplorer attempts to connect to another host.

*newVal*—The set value, specifying the number of seconds that you want HostExplorer to wait before it attempts to connect to another host.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
iVal = Connection.RetryDelayTimeBetweenHosts  
If (iVal > 5) Then  
    Connection.RetryDelayTimeBetweenHosts = 5  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( &pConnection );
    //Get RetryDelayTimeBetweenHosts Property
    short sVal;
    pConn->get_RetryDelayTimeBetweenHosts(&sVal);
    if (sVal > 5)
    {
        sVal= 5;
        pConn->put_RetryDelayTimeBetweenHosts (sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ShowDialupDlg

### Property, IHEProfileConnection VT

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

#### Basic Syntax

```
Boolean = HEProfileConnection.ShowDialupDlg  
HEProfileConnection.ShowDialupDlg = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_ShowDialupDlg([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileConnection::put_ShowDialupDlg([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays the Always Show Connect Dialog flag. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display the Always Show Connect Dialog flag.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays the Always Show Connect Dialog flag. A value of VARIANT\_FALSE indicates that HostExplorer does not display the Always Show Connect Dialog flag.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.ShowDialupDlg  
If (bVal = False) Then  
    ' Set value  
    Connection.ShowDialupDlg = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get ShowDialupDlg Property
VARIANT_BOOL bVal;
bVal = pConn->get_ShowDialupDlg(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pConn->put_ShowDialupDlg(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SilentConnect

Property, IHEProfileConnection 3270 5250 VT

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

### Basic Syntax

```
Boolean = HEProfileConnection.SilentConnect
```

```
HEProfileConnection.SilentConnect = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_SilentConnect([out, retval]
VARIANT_BOOL *pVal);  
HRESULT IHEProfileConnection::put_SilentConnect([in] VARIANT_BOOL
newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer displays a progress dialog box while connecting to the host. If *pVal* equals VARIANT\_FALSE, HostExplorer does not display a progress dialog box (a "silent connect").

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE if you want HostExplorer to display a progress dialog box while connecting to the host; otherwise, set *newVal* to VARIANT\_FALSE.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.SilentConnect  
If (bVal = False) Then  
    ' Set value  
    Connection.SilentConnect = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get SilentConnect Property
    VARIANT_BOOL bVal;
    bVal = pConn->get_SilentConnect(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pConn->put_SilentConnect(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SYSREQasIACIP

### Property, IHEProfileConnection 3270

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

#### Basic Syntax

```
Boolean = HEProfileConnection.SYSREQasIACIP
```

```
HEProfileConnection.SYSREQasIACIP = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEProfileConnection::put_SYSREQasIACIP([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the SYSREQ key as the Telnet IAC IP sequence is enabled. A returned value of VARIANT\_FALSE indicates that the sequence is disabled.

*newVal*—A value of VARIANT\_TRUE indicates that the SYSREQ key as the Telnet IAC IP sequence is enabled. A value of VARIANT\_FALSE indicates that the sequence is disabled.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.SYSREQasIACIP  
If (bVal = False) Then  
    ' Set value  
    Connection.SYSREQasIACIP = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get SYSREQasIACIP Property
    VARIANT_BOOL bVal;
    bVal = pConn->get_SYSREQasIACIP(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pConn->put_SYSREQasIACIP(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TelnetEcho

### Property, IHEProfileConnection VT

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

#### Basic Syntax

```
HOSTEX_TELNETECHO = HEProfileConnection.TelnetEcho
```

```
HEProfileConnection.TelnetEcho = HOSTEX_TELNETECHO
```

#### C++ Syntax

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

```
HRESULT IHEProfileConnection::put_TelnetEcho([in] HOSTEX_TELNETECHO  
newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim TEcho As HOSTEX_TELNETECHO  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
TEcho = Connection.TelnetEcho  
If (TEcho = HOSTEX_TELNETECHO_NO) Then  
    ' Set value  
    Connection.TEcho = HOSTEX_TELNETECHO_YES  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get TelnetEcho Property
HOSTEX_TELNETECHO TEcho;
pConn->get_TelnetEcho(&TEcho);
if (TEcho!= HOSTEX_TELNETECHO_NO)
{
    TEcho = HOSTEX_TELNETECHO_NO;
    pConn->put_TelnetEcho(TEcho);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## TelnetName

### Property, IHEProfileConnection VT

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

#### Basic Syntax

```
String = HEProfileConnection.TelnetName  
HEProfileConnection.TelnetName = String
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get string  
strVal = Connection.TelnetName  
If (Len(strVal) = 0) Then  
    ' Set string  
    Connection.TelnetName = "sunset"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get TelnetName Property
BSTR bstr ;
pConn->get_TelnetName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("sunset"));
    pConn->put_TelnetName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Timeout

### Property, IHEProfileConnection 3270 5250 VT

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

#### Basic Syntax

```
Long = HEProfileConnection.Timeout
```

```
HEProfileConnection.Timeout = Long
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_Timeout([out, retval] long *pVal);
```

```
HRESULT IHEProfileConnection::put_Timeout([in] long newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile
Dim Connection As HEProfileConnection
Dim lVal As Long
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Connection = Profile.Connection
' Get value
lVal = Connection.Timeout
If (lVal = 0) Then
    ' Set value
    Connection.Timeout = 100
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get Timeout Property
    long lVal;
    pConn->get_Timeout(&lVal);
    if (lVal == 0)
    {
        lVal =2;
        pConn->put_Timeout(100);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UponDisconnect

Property, IHEProfileConnection 3270 5250 VT

This property returns or sets a value specifying the action that HostExplorer performs if the host disconnects from the current session.

### Basic Syntax

```
Integer = HEProfileConnection.UponDisconnect  
HEProfileConnection.UponDisconnect = Integer
```

### C++ Syntax

```
HRESULT IHEProfileConnection::get_UponDisconnect([out, retval] short  
*pVal);  
HRESULT IHEProfileConnection::put_UponDisconnect([in] short newVal);
```

### Parameters

*pVal*—The following returned values specify the action that HostExplorer performs in the event that the host disconnects from the current session:

- UPONDISC\_CLOSEWINDOW
- UPONDISC\_KEEPWINDOWOPEN
- UPONDISC\_RESTARTPROFILE
- UPONDISC\_SHOWOPENNEWPROFILE

*newVal*—The set value, specifying the action that HostExplorer performs in the event that the host disconnects from the current session.

### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim iVal As Integer  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
iVal = Connection.UponDisconnect  
If (iVal <> UPONDISC_KEEPWINDOWOPEN) Then  
    Connection.UponDisconnect = UPONDISC_KEEPWINDOWOPEN  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get UponDisconnect Property
    short sVal;
    pConn->get_UponDisconnect(&sVal);
    if (sVal != UPONDISC_KEEPWINDOWOPEN)
    {
        sVal= UPONDISC_KEEPWINDOWOPEN;
        pConn->put_UponDisconnect(sVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UseDialProp

### Property, IHEProfileConnection VT

This property returns or sets a value indicating whether HostExplorer uses the settings provided in the Windows Dialing Properties dialog box while you are installing a modem. When this property is set to VARIANT\_TRUE, HostExplorer automatically adds the area code, country code, 8, 9, or calling-card numbers, if applicable. By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileConnection.UseDialProp  
HEProfileConnection.UseDialProp = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_UseDialProp([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEProfileConnection::put_UseDialProp([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer uses the settings provided in the Windows Dialing Properties dialog box. A returned value of VARIANT\_FALSE indicates that HostExplorer dials using the “raw” phone number.  
*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer uses the settings provided in the Windows Dialing Properties dialog box. A value of VARIANT\_FALSE indicates that HostExplorer dials using the “raw” phone number.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.UseDialProp  
If (bVal = False) Then  
    ' Set value  
    Connection.UseDialProp = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get UseDialProp Property
VARIANT_BOOL bVal;
pConn->get_UseDialProp(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pConn->put_UseDialProp(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## UserName

### Property, IHEProfileConnection 5250

This property returns or sets a string that specifies the user name that you use to connect to the host.

#### Basic Syntax

```
String = HEProfileConnection.UserName  
HEProfileConnection.UserName = String
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_UserName([out, retval] BSTR *pVal);  
HRESULT IHEProfileConnection::put_UserName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying your username.  
*newVal*—The set string, specifying your username.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim Connection As HEProfileConnection  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_5250  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get string  
strVal = Connection.UserName  
If (Len(strVal) = 0) Then  
    ' Set string  
    Connection.UserName = "Jack17"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_5250);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileConnection Interface
    IHEProfileConnection* pConnection;
    pIProfile->get_Connection ( & pConnection);
    //Get UserName Property
    BSTR bstr ;
    pConn->get_UserName(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("Jack17"));
        pConn->put_UserName(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTDoHostWindowSize

### Property, IHEProfileConnection VT

This property returns or sets a value indicating whether HostExplorer sends or negotiates a change in window size (that is, the number of rows and columns to the Telnet host). By default, this property is set to VARIANT\_FALSE.

#### Basic Syntax

```
Boolean = HEProfileConnection.VTDoHostWindowSize
```

```
HEProfileConnection.VTDoHostWindowSize = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_VTDoHostWindowSize([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileConnection::put_VTDoHostWindowSize([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer sends or negotiates a change in window size. A returned value of VARIANT\_FALSE indicates that HostExplorer does not send or negotiate a change in window size.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer sends or negotiates a change in window size. A value of VARIANT\_FALSE indicates that HostExplorer does not send or negotiate a change in window size.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.VTDoHostWindowSize  
If (bVal = False) Then  
    ' Set value  
    Connection.VTDoHostWindowSize = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( &pConnection);
//Get VTDoHostWindowSize Property
VARIANT_BOOL bVal;
bVal = pConn->get_VTDoHostWindowSize(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pConn->put_VTDoHostWindowSize(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTInitiateTelnetNegotiation

### Property, IHEProfileConnection VT

This property returns or sets a value indicating whether HostExplorer (VT emulator) negotiates connection options when it establishes a Telnet connection. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileConnection.VTInitiateTelnetNegotiation  
HEProfileConnection.VTInitiateTelnetNegotiation = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_VTInitiateTelnetNegotiation([out,  
retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfileConnection::put_VTInitiateTelnetNegotiation([in]  
VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer negotiates connection options when it establishes a Telnet connection. A returned value of VARIANT\_FALSE indicates that HostExplorer does not negotiate connection options when it establishes a Telnet connection.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer negotiates connection options when it establishes a Telnet connection. A value of VARIANT\_FALSE indicates that HostExplorer does not negotiate connection options when it establishes a Telnet connection.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
' Get value  
bVal = Connection.VTInitiateTelnetNegotiation  
If (bVal = False) Then  
    ' Set value  
    Connection.VTInitiateTelnetNegotiation = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( &pConnection );
//Get VTInitiateTelnetNegotiation Property
VARIANT_BOOL bVal;
bVal = pConn->get_VTInitiateTelnetNegotiation(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pConn->put_VTInitiateTelnetNegotiation(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## VTLineMode

### Property, IHEProfileConnection VT

This property returns or sets a value specifying how HostExplorer stores characters in a buffer until you send a carriage return to the host.

#### Basic Syntax

```
HOSTEX_LINEMODE = HEProfileConnection.VTLineMode
```

```
HEProfileConnection.VTLineMode = HOSTEX_LINEMODE
```

#### C++ Syntax

```
HRESULT IHEProfileConnection::get_VTLineMode([out, retval]  
HOSTEX_LINEMODE *pVal);
```

```
HRESULT IHEProfileConnection::put_VTLineMode([in] HOSTEX_LINEMODE  
newVal);
```

#### Parameters

*pVal*—The returned value specifying how HostExplorer stores characters in a buffer until you send a carriage return to the host.

*newVal*—The set value specifying how HostExplorer stores characters in a buffer until you send a carriage return to the host.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Connection As HEProfileConnection  
Dim LMode As HOSTEX_LINEMODE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_VT  
Set Profile = Terminal.Session  
Set Connection = Profile.Connection  
LMode = Connection.VTLineMode  
If (LMode <> HOSTEX_LINEMODE_ALWAYS) Then  
    Connection.VTLineMode = HOSTEX_LINEMODE_ALWAYS  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_VT);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileConnection Interface
IHEProfileConnection* pConnection;
pIProfile->get_Connection ( & pConnection);
//Get VTLineMode Property
HOSTEX_LINEMODE LMode;
pConn->get_VTLineMode (&LMode);
if (LMode!= HOSTEX_LINEMODE_ALWAYS)
{
    LMode = HOSTEX_LINEMODE_ALWAYS;
    pConn->put_VTLineMode (LMode);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileSessionWindow Interface

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

### Properties

The ProfileSessionWindow interface consists of the following properties:

- DisplayBorder
- EnableWorkspaceBackground Bitmap
- FullScreenMode
- KeepFontAspectRatio
- LongName
- PromptOnClose
- ResizeBehavior
- SaveProfOnClose
- SaveFontOnExit
- SnapFrameBack
- SwitchScreenType
- WindowState
- WindowTitle
- WorkSpaceBackgroundBitmap
- WorkspaceBackgroundColor
- WorkspaceForegroundColor

### DisplayBorder

**Property, IHEProfileSessionWindow 3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer fills the empty space between the terminal window and the window frame with a gray border. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileSessionWindow.DisplayBorder  
HEProfileSessionWindow.DisplayBorder = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_DisplayBorder([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_DisplayBorder([in] VARIANT_BOOL  
newVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer creates a gray border between the terminal window and the window frame. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display a border, so that the screen display closely matches any resized window.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer creates a gray border between the terminal window and the window frame. A value of VARIANT\_FALSE indicates that HostExplorer does not display a border, so that the screen display closely matches any resized window.

**Basic Example**

```
Dim Profile As HEProfile
Dim SessionWindow As HEProfileSessionWindow
Dim bVal As Boolean
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessionWindow = Profile.SessionWindow
bVal = SessWin.DisplayBorder
If (bVal = False) Then
    SessWin.DisplayBorder = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get DisplayBorder Property
    VARIANT_BOOL bVal;
    pWin->get_DisplayBorder(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pWin->put_DisplayBorder(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## EnableWorkspaceBackgroundBitmap

**Property, IHEProfileSessionWindow 3270 5250 VT**

This property returns or sets a value indicating whether HostExplorer enables the bitmap for the background of the terminal screen.

### Basic Syntax

```
Boolean = HEProfileSessionWindow.EnableWorkspaceBackgroundBitmap  
HEProfileSessionWindow.EnableWorkspaceBackgroundBitmap = Boolean
```

### C++ Syntax

```
HRESULT  
IHEProfileSessionWindow::get_EnableWorkspaceBackgroundBitmap([out,  
retval] VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_EnableWorkspaceBackgroundBitmap([in]  
VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables the bitmap for the background of the terminal screen. A returned value of VARIANT\_FALSE indicates that HostExplorer uses the default terminal screen.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer enables the bitmap for the background of the terminal screen. A value of VARIANT\_FALSE indicates that HostExplorer uses the default terminal screen.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
bVal = SessWin.EnableWorkspaceBackgroundBitmap  
If (bVal = False) Then  
    SessWin.EnableWorkspaceBackgroundBitmap = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get EnableWorkspaceBackgroundBitmap Property
    VARIANT_BOOL bVal;
    pWin->get_EnableWorkspaceBackgroundBitmap(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pWin->put_EnableWorkspaceBackgroundBitmap(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## FullScreenMode

### Property, IHEProfileSessionWindow 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEProfileSessionWindow.FullScreenMode
```

```
HEProfileSessionWindow.FullScreenMode = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_FullScreenMode([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_FullScreenMode([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the session will run in full-screen display mode. A returned value of VARIANT\_FALSE indicates that the session will run in regular-screen display mode.

*newVal*—A set value of VARIANT\_TRUE indicates that the session will run in full-screen display mode. A set value of VARIANT\_FALSE indicates that the session will run in regular-screen display mode.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
' Get value  
bVal = SessWin.FullScreenMode  
If (bVal = False) Then  
    ' Set value  
    SessWin.FullScreenMode = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get FullScreenMode Property
    VARIANT_BOOL bVal;
    bVal = pWin->get_FullScreenMode(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pWin->put_FullScreenMode (bVal) ;
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## KeepFontAspectRatio

Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a value that specifies whether HostExplorer keeps all fonts within a normal aspect ratio. By default, this property is set to VARIANT\_TRUE.

### Basic Syntax

```
Boolean = HEProfileSessionWindow.KeepFontAspectRatio  
HEProfileSessionWindow.KeepFontAspectRatio = Boolean
```

### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_KeepFontAspectRatio([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEProfileSessionWindow::put_KeepFontAspectRatio([in]  
VARIANT_BOOL newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer keeps all fonts within a normal aspect ratio. If *pVal* equals VARIANT\_FALSE, HostExplorer can create extra wide or extra tall fonts.

*newVal*—The set value. Set *newVal* to VARIANT\_TRUE to ensure that HostExplorer keeps all fonts within a normal aspect ratio; HostExplorer can then better match the fonts with the current window size. Set *newVal* to VARIANT\_FALSE to let HostExplorer create extra wide or extra tall fonts.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
' Get value  
bVal = SessWin.KeepFontAspectRatio  
If (bVal = False) Then  
    ' Set value  
    SessWin.KeepFontAspectRatio = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileSessionWindow Interface
IHEProfileSessionWindow* pSessionWindow;
pIProfile->get_SessionWindow ( & pSessionWindow );
//Get KeepFontAspectRatio Property
VARIANT_BOOL bVal;
bVal = pWin->get_KeepFontAspectRatio(&bVal);
if (bVal == VARIANT_FALSE)
{
    bVal = VARIANT_TRUE;
    pWin->put_KeepFontAspectRatio(bVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## LongName

### Property, IHEProfileSessionWindow 3270 5250 VT

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

#### Basic Syntax

```
String = HEProfileSessionWindow.LongName
```

```
HEProfileSessionWindow.LongName = String
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_LongName([out, retval] BSTR *pVal);
```

```
HRESULT IHEProfileSessionWindow::put_LongName([in] BSTR newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim Profile As HEProfile
Dim SessionWindow As HEProfileSessionWindow
Dim strVal As String
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessionWindow = Profile.SessionWindow
' Get value
strVal = SessWin.LongName
If (Len(strVal) = 0) Then
    ' Set value
    SessWin.LongName = "Sess-1"
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileSessionWindow Interface
IHEProfileSessionWindow* pSessionWindow;
pIProfile->get_SessionWindow ( & pSessionWindow );
//Get LongName Property
BSTR bstr ;
pWin->get_LongName(&bstr);
if (strlen(OLE2A(bstr)) == 0)
{
    if (bstr!=NULL)
        SysFreeString(bstr);
    bstr = SysAllocString(OLESTR("Sess-1"));
    pWin->put_LongName(bstr);
    SysFreeString(bstr);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## PromptOnClose

### Property, IHEProfileSessionWindow 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEProfileSessionWindow.PromptOnClose
```

```
HEProfileSessionWindow.PromptOnClose = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_PromptOnClose([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_PromptOnClose([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer prompts you before closing a window session. A returned value of VARIANT\_FALSE indicates that HostExplorer does not prompt you before closing a window session.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer prompts you before closing a window session. A value of VARIANT\_FALSE indicates that HostExplorer does not prompt you before closing a window session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
' Get value  
bVal = SessWin.PromptOnClose  
If (bVal = False) Then  
    ' Set value  
    SessWin.PromptOnClose = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get PromptOnClose Property
    VARIANT_BOOL bVal;
    bVal = pWin->get_PromptOnClose(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pWin->put_PromptOnClose(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ResizeBehavior

### Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a value that specifies how HostExplorer displays information in the session window when you resize the window. By default, this property is set to HOSTEX\_RESIZE\_BEHAVIOR\_CHANGE\_FONT.

#### Basic Syntax

```
HOSTEX_RESIZE_BEHAVIOR = HEProfileSessionWindow.ResizeBehavior  
HEProfileSessionWindow.ResizeBehavior = HOSTEX_RESIZE_BEHAVIOR
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_ResizeBehavior([out, retval]  
HOSTEX_RESIZE_BEHAVIOR *pVal);  
  
HRESULT IHEProfileSessionWindow::put_ResizeBehavior([in]  
HOSTEX_RESIZE_BEHAVIOR newVal);
```

#### Parameters

*pVal*—The returned value, specifying the current resize behavior.

*newVal*—The set value, specifying the resize behavior you want to apply to the session window.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim behavior As HOSTEX_RESIZE_BEHAVIOR  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
' Get value  
behavior = SessWin.ResizeBehavior  
If (behavior = HOSTEX_RESIZE_BEHAVIOR_DO_NOTHING) Then  
    ' Set value  
    SessWin.ResizeBehavior = HOSTEX_RESIZE_BEHAVIOR_NEGOTIATE_WIN_SIZE  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get ResizeBehavior Property
    HOSTEX_RESIZE_BEHAVIOR behavior;
    bVal = pWin->get_ResizeBehavior(&behavior);
    if (behavior == HOSTEX_RESIZE_BEHAVIOR_DO_NOTHING)
    {
        behavior = HOSTEX_RESIZE_BEHAVIOR_NEGOTIATE_WIN_SIZE;
        pWin->put_ResizeBehavior(behavior);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SaveProfOnClose

### Property, IHEProfileSessionWindow 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEProfileSessionWindow.SaveProfOnClose
```

```
HEProfileSessionWindow.SaveProfOnClose = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_SaveProfOnClose([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_SaveProfOnClose([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically saves the setting changes for the current profile when you close the session. A returned value of VARIANT\_FALSE indicates that HostExplorer does not save the setting changes for the current profile when you close the session.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically saves the setting changes for the current profile when you close the session. A value of VARIANT\_FALSE indicates that HostExplorer does not save the setting changes for the current profile when you close the session.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
' Get value  
bVal = SessWin.SaveProfOnClose  
If (bVal = False) Then  
    ' Set value  
    SessWin.SaveProfOnClose = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get SaveProfOnClose Property
    VARIANT_BOOL bVal;
    bVal = pWin->get_SaveProfOnClose(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pWin->put_SaveProfOnClose(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SaveFontOnExit

### Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically saves any changes you make to either the font or the window size to the current session profile. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HEProfileSessionWindow.SaveFontOnExit
```

```
HEProfileSessionWindow.SaveFontOnExit = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_SaveFontOnExit([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_SaveFontOnExit([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically saves any changes you make to either the font or the window size. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically save any changes you make to either the font or the window size.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically saves any changes you make to either the font or the window size. A value of VARIANT\_FALSE indicates that HostExplorer does not automatically save any changes you make to either the font or the window size.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
bVal = SessWin.SaveFontOnExit  
If (bVal = False) Then  
    SessWin.SaveFontOnExit = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get SaveFontOnExit Property
    VARIANT_BOOL bVal;
    pWin->get_SaveFontOnExit(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pWin->put_SaveFontOnExit(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SnapFrameBack

### Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays borders on the terminal screen.

#### Basic Syntax

```
Boolean = HEProfileSessionWindow.SnapFrameBack
```

```
HEProfileSessionWindow.SnapFrameBack = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_SnapFrameBack([out, retval]  
VARIANT_BOOL *pVal);  
  
HRESULT IHEProfileSessionWindow::put_SnapFrameBack([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer displays borders on the terminal screen. A returned value of VARIANT\_FALSE indicates that HostExplorer does not display borders on the terminal screen.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer displays borders on the terminal screen. A value of VARIANT\_FALSE indicates that HostExplorer does not display borders on the terminal screen.

#### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
bVal = SessWin.SnapFrameBack  
If (bVal = False) Then  
    SessWin.SnapFrameBack = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get SnapFrameBack Property
    VARIANT_BOOL bVal;
    pWin->get_SnapFrameBack(&bVal);
    if (bVal==VARIANT_FALSE)
    {
        bVal=VARIANT_TRUE;
        pWin->put_SnapFrameBack(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## SwitchScreenType

Property, IHEProfileSessionWindow **3270 5250 VT**

This property returns or sets a value specifying the type of information that HostExplorer retains when you resize the screen.

### Basic Syntax

```
HOSTEX_SWITCHSCREENTYPE = HEProfileSessionWindow.SwitchScreenType  
HEProfileSessionWindow.SwitchScreenType = HOSTEX_SWITCHSCREENTYPE
```

### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_SwitchScreenType([out, retval]  
HOSTEX_SWITCHSCREENTYPE *pVal);  
HRESULT IHEProfileSessionWindow::put_SwitchScreenType([in]  
HOSTEX_SWITCHSCREENTYPE newVal);
```

### Parameters

*pVal*—The returned value, specifying the type of information that HostExplorer retains when you resize the screen.

*newVal*—The set value, specifying the type of information that HostExplorer retains when you resize the screen.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim ScreenType As HOSTEX_SWITCHSCREENTYPE  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
ScreenType = SessWin.SwitchScreenType  
If (ScreenType = HOSTEX_SWITCHSCREENTYPE_KEEPFONT) then  
    SessWin.SwitchScreenType = HOSTEX_SWITCHSCREENTYPE_KEEPOLDINFO  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get SwitchScreenType Property
    HOSTEX_SWITCHSCREENTYPE ScreenType;
    pWin->get_SwitchScreenType(&ScreenType);
    if (ScreenType == HOSTEX_SWITCHSCREENTYPE_KEEPFONT)
    {
        ScreenType = HOSTEX_SWITCHSCREENTYPE_KEEPOLDINFO;
        pWin->put_SwitchScreenType(ScreenType);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WindowState

Property, IHEProfileSessionWindow **3270 5250 VT**

This property returns or sets a value that specifies the type of resizing for the session window. The resize type can be one of the following values:

- 0—Minimize
- 1—Restore original size
- 2—Maximize

### Basic Syntax

```
Integer = HEProfileSessionWindow.WindowState
```

```
HEProfileSessionWindow.WindowState = Integer
```

### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_WindowState([out, retval] short *pVal);
```

```
HRESULT IHEProfileSessionWindow::put_WindowState([in] short newVal);
```

### Parameters

*pVal*—The returned value, specifying the resize type for the session window.

*newVal*—The set value, specifying the type of resizing that you want to apply to the session window.

### Basic Example

```
Dim Profile As HEProfile
Dim SessionWindow As HEProfileSessionWindow
Dim ResizeType As Integer
'Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set SessionWindow = Profile.SessionWindow
ResizeType = SessWin.WindowState
If (ResizeType <> 0) Then
    SessWin.WindowState = 0
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get WindowState Property
    short ResizeType;
    pWin->get_WindowState(&ResizeType);
    if (ResizeType != 0)
    {
        ResizeType = 0;
        pWin->put_WindowState(ResizeType);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WindowTitle

### Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a string indicating the name and/or description displayed in the top right-hand corner of the session window.

#### Basic Syntax

```
String = HEProfileSessionWindow.WindowTitle  
HEProfileSessionWindow.WindowTitle = String
```

#### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_WindowTitle([out, retval] BSTR  
*pVal);  
HRESULT IHEProfileSessionWindow::put_WindowTitle([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, which indicates the name and/or description.

*newVal*—The set string, which indicates the name and/or description.

#### Basic Example

```
Dim Profile As HEPProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim strVal As String  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
' Get string  
strVal = SessWin.WindowTitle  
If (Len(strVal) = 0) Then  
    ' Set string  
    SessWin.WindowTitle = "HostExplorer"  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get WindowTitle Property
    BSTR bstr ;
    pWin->get_WindowTitle(&bstr);
    if (strlen(OLE2A(bstr)) == 0)
    {
        if (bstr!=NULL)
            SysFreeString(bstr);
        bstr = SysAllocString(OLESTR("Host Explorer"));
        pWin->put_WindowTitle(bstr);
        SysFreeString(bstr);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WorkSpaceBackgroundBitmap

Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a string specifying the bitmap file for the background of the terminal screen.

### Basic Syntax

```
String = HEProfileSessionWindow.WorkSpaceBackgroundBitmap  
HEProfileSessionWindow.WorkSpaceBackgroundBitmap = String
```

### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_WorkSpaceBackgroundBitmap([out,  
retval] BSTR *pVal);  
HRESULT IHEProfileSessionWindow::put_WorkSpaceBackgroundBitmap([in] BSTR  
newVal);
```

### Parameters

*pVal*—The returned value, specifying the bitmap file for the background of the terminal screen.

*newVal*—The set value, specifying the bitmap file for the background of the terminal screen.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
bVal = SessWin.WorkSpaceBackgroundBitmap  
If (bVal = False) Then  
    SessWin.WorkSpaceBackgroundBitmap = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEProfileSessionWindow Interface
    IHEProfileSessionWindow* pSessionWindow;
    pIProfile->get_SessionWindow ( & pSessionWindow );
    //Get WorkSpaceBackgroundBitmap Property
    VARIANT_BOOL bVal;
    pWin->get_WorkSpaceBackgroundBitmap(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pWin->put_WorkSpaceBackgroundBitmap(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WorkspaceBackgroundColor

Property, IHEProfileSessionWindow 3270 5250 VT

This property returns or sets a value specifying the background color of the terminal screen in OLE\_COLOR format.

### Basic Syntax

```
OLE_COLOR = HEProfileSessionWindow.WorkspaceBackgroundColor  
HEProfileSessionWindow.WorkspaceBackgroundColor = OLE_COLOR
```

### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_WorkspaceBackgroundColor([out,  
retval] OLE_COLOR *pVal);  
HRESULT IHEProfileSessionWindow::put_WorkspaceBackgroundColor([in]  
OLE_COLOR newVal);
```

### Parameters

*pVal*—The returned value, specifying the background color of the terminal screen.  
*newVal*—The set value, specifying the background color of the terminal screen.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim color As OLE_COLOR  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
color = SessWin.WorkspaceBackgroundColor  
' White is hex FFFFFF which is 16777215  
' when converted to Long  
If (color = 16777215) Then  
    ' Change from white to Green, which  
    ' is hex FF00, which converts to  
    ' Long 65280  
    SessWin.WorkspaceBackgroundColor = 65280  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileSessionWindow Interface
IHEProfileSessionWindow* pSessionWindow;
pIProfile->get_SessionWindow ( & pSessionWindow );
//Get WorkspaceBackgroundColor Property
long lVal;
// Get value
pWin->get_WorkspaceBackgroundColor(&lVal);
// White is hex FFFFFF which is 16777215
// when converted to Long
if (lVal == 16777215)
{
    // Change from white to Green, which
    // is hex FF00,which converts to
    // Long 65280
    lVal = 65280;
    pWin->put_WorkspaceBackgroundColor(lVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## WorkspaceForegroundColor

Property, IHEProfileSessionWindow **3270 5250 VT**

This property returns or sets a value specifying the foreground color of the terminal screen in OLE\_COLOR format.

### Basic Syntax

```
OLE_COLOR = HEProfileSessionWindow.WorkspaceForegroundColor  
HEProfileSessionWindow.WorkspaceForegroundColor = OLE_COLOR
```

### C++ Syntax

```
HRESULT IHEProfileSessionWindow::get_WorkspaceForegroundColor([out,  
retval] OLE_COLOR *pVal);  
HRESULT IHEProfileSessionWindow::put_WorkspaceForegroundColor([in]  
OLE_COLOR newVal);
```

### Parameters

*pVal*—The returned value, specifying the foreground color of the terminal screen.  
*newVal*—The returned value, specifying the foreground color of the terminal screen.

### Basic Example

```
Dim Profile As HEProfile  
Dim SessionWindow As HEProfileSessionWindow  
Dim color As OLE_COLOR  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set SessionWindow = Profile.SessionWindow  
color = SessWin.WorkspaceForegroundColor  
' White is hex FFFFFF which is 16777215  
' when converted to Long  
If (color = 16777215) Then  
    ' Change from white to Green, which  
    ' is hex FF00, which converts to  
    ' Long 65280  
    SessWin.WorkspaceForegroundColor = 65280  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
...
// Get HEProfile Interface
IHEProfile *pIProfile = NULL;
pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
// Get HEProfileSessionWindow Interface
IHEProfileSessionWindow* pSessionWindow;
pIProfile->get_SessionWindow ( & pSessionWindow );
//Get WorkspaceForegroundColor Property
long lVal;
// Get value
pWin->get_WorkspaceForegroundColor(&lVal);
// White is hex FFFFFF which is 16777215
// when converted to Long
if (lVal == 16777215)
{
    // Change from white to Green, which
    // is hex FF00,which converts to
    // Long 65280
    lVal = 65280;
    pWin->put_WorkspaceForegroundColor(lVal);
}
...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## ProfileSound Interface

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

### Properties

The ProfileSound interface consists of the following properties:

- Notify
- Sound

#### Notify

##### Property, IHEProfileSound 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEProfileSound.Notify  
HEProfileSound.Notify = Boolean
```

#### C++ Syntax

```
HRESULT IHEProfileSound::get_Notify([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEProfileSound::put_Notify([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Notify option is on. A returned value of VARIANT\_FALSE indicates that the Notify option is off.

*newVal*—A value of VARIANT\_TRUE indicates that the Notify option is on. A value of VARIANT\_FALSE indicates that the Notify option is off.

#### Basic Example

```
Dim Profile As HEProfile  
Dim Sound As HEProfileSound  
Dim bVal As Boolean  
'Set the appropriate type  
Terminal.TerminalType = HOSTEX_TERMINAL_3270  
Set Profile = Terminal.Session  
Set Sound = Profile.Sound  
' Get value  
bVal = ProgSnds.Notify  
If (bVal = False) Then  
    ' Set value  
    ProgSnds.Notify = True  
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileSound Interface
    IHEProfileSound* pSound;
    pIProfile->get_Sound ( &pSound );
    //Get Notify Property
    VARIANT_BOOL bVal;
    bVal = pSound->get_Notify(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSound->put_Notify(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## Sound

### Property, IHEProfileSound 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEProfileSound.Sound
```

```
HEProfileSound.Sound = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEProfileSound::put_Sound([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Sound option for program sounds is on. A returned value of VARIANT\_FALSE indicates that the Sound option for program sounds is off.

*newVal*—A value of VARIANT\_TRUE indicates that the Sound option for program sounds is on. A value of VARIANT\_FALSE indicates that the Sound option for program sounds is off.

#### Basic Example

```
Dim Profile As HEProfile
Dim Sound As HEProfileSound
Dim bVal As Boolean
' Set the appropriate type
Terminal.TerminalType = HOSTEX_TERMINAL_3270
Set Profile = Terminal.Session
Set Sound = Profile.Sound
' Get value
bVal = ProgSnds.Sound
If (bVal = False) Then
    ' Set value
    ProgSnds.Sound = True
End If
```

**C++ Example**

```
...
// Need to create the Com Interface IHETerminal
IHETerminal* pITerminal = NULL;
HRESULT hr = CoCreateInstance( CLSID_HETerminal, NULL,
    CLSCTX_INPROC_SERVER, IID_IHETerminal, (LPVOID*)&pITerminal );
...
if (!hr)
{
    // get session interface
    IDispatch *pIDispatch = NULL;
    pITerminal->get_Session( &pIDispatch );
    // By default the terminal type is HOSTEX_TERMINAL_3270
    // If you want to set the terminal type you have three choices
    // HOSTEX_TERMINAL_3270, HOSTEX_TERMINAL_VT or HOSTEX_TERMINAL_5250
    pITerminal->put_TerminalType(HOSTEX_TERMINAL_3270);
    ...
    // Get HEPProfile Interface
    IHEProfile *pIProfile = NULL;
    pIDispatch->QueryInterface( IID_IHEProfile, (void**) &pIProfile );
    // Get HEPProfileSound Interface
    IHEProfileSound* pSound;
    pIProfile->get_Sound ( & pSound );
    //Get Sound Property
    VARIANT_BOOL bVal;
    bVal = pSounds->get_Sound(&bVal);
    if (bVal == VARIANT_FALSE)
    {
        bVal = VARIANT_TRUE;
        pSounds->put_Sound(bVal);
    }
    ...
}
// Release pITerminal
if (pITerminal )
{
    pITerminal->Release();
}
```

## 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 HEPAR3270 and HEPAR5250 objects
- both the HEPAR3270 and HEPARVT objects
- only the HEPARVT object

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

## ***Properties and Data Types of the HEPAR3270 Object***

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

### **Properties**

- APILInputMode
- GraphicsModel
- PrinterDeInitString
- ProgramSymbols
- TransferMode
- EnableAPL
- NumericCharacters
- PrinterInitString
- TransferErrorCode
- ValidateNumericFieldData

### **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
- OnCopyReplaceFieldAttribute With
- PasteMode
- CutMode
- OnPasteFieldModeTabCharacter

## 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
- HostWritableString
- NRCID
- PrintDisableTranslation
- PrintLFtoCRLF
- SaveFileName
- TerminalID
- AutoWrap
- CaptureMode
- HistoryLines
- MoveCursorOnMouseClicked
- 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

- HEPARSER\_FEATURE Data Type
- HOSTEX\_CONNECT\_BY Data Type
- HOSTEX\_SELECTION\_MODE Data Type
- HOSTEX\_TERM\_MODEL Data Type

## Methods of the Parser Objects

The following are methods of the Parser objects:

- AsciiToHost
- ClearSel
- FindString
- GetCursorPosition
- GetFeature
- GetFieldAttribute
- GetFieldCount
- GetFieldExtAttribute
- GetFieldIndex
- GetFieldLength
- GetFieldPos
- GetFieldText
- GetScreenText
- GetSel
- GetSelectionArea
- 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

## AsciiToHost

### Method, IHEParser 3270 5250 VT

This method converts an ASCII character into the equivalent EBCDIC host character.

#### Basic Syntax

```
HEParser.AsciiToHost(inchar As BYTE) As BYTE
```

#### C++ Syntax

```
HRESULT IHEParser::AsciiToHost(  
    [in] BYTE inChar,  
    [out, retval] BYTE *outChar);
```

#### Parameters

*inChar*—The input character in ASCII format that you want to convert.

*outChar*—The return character that has been converted to EBCDIC format.

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
  
Dim asciiChar As Byte  
Dim HostChar As Byte  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
asciiChar = 65  
HostChar = Parser.AsciiToHost(asciiChar)  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BYTE AsciiChar = 65;
            BYTE HostChar;
            pParser->AsciiToHost( AsciiChar, &HostChar);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ClearSel

Method, IHEParser **3270 5250 VT**

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

**Basic Syntax**      `HEParser.ClearSel`

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

**Parameters**      This method has no parameters.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
Dim Parser As HEParser5250
```

```
strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect
    Set Parser = OSession.Parser
    Parser.ClearSel
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->ClearSel();
        }
        pSession->Disconnect();

    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

**FindString**

**Method, IHEParser 3270 5250 VT**

This method returns the row and column and buffer position where a specified string appears in the screen buffer.

**Basic Syntax**

```
HEParser.FindString( bstrString As String, bCaseSensitive As Boolean, vRow
As Variant, vCol As Variant) As Integer
```

**C++ Syntax**

```
HRESULT IHEParser::FindString(
    [in] BSTR bstrString,
    [in] VARIANT_BOOL bCaseSensitive,
    [in, out] LPVARIANT vRow,
    [in, out] LPVARIANT vCol,
    [out, retval] short *pVal);
```

**Parameters**

*bstrString*—The string that you want to locate in the screen buffer.

*bCaseSensitive*—The value that specifies whether you want the search to be case sensitive. If *bCaseSensitive* equals VARIANT\_TRUE, the buffer string must match the search string exactly in content and case. If *bCaseSensitive* equals VARIANT\_FALSE, the search is not case sensitive.

*vRow*—The value that specifies the row at which you want the search to begin; upon return, *vRow* specifies the actual row where the matching string begins in the buffer.

*vCol*—The value that specifies the column at which you want the search to begin; upon return, *vCol* specifies the actual column where the matching string begins in the buffer.

*pVal*—The return value that specifies the position of the matching string in the screen buffer.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

Dim Row As Integer
Dim Col As Integer
Dim iVal as Integer

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
Row = 3
Col = 5
iVal = Parser.FindString("Hello world", True, Row, Col)

OSession.Disconnect
```

**C++ Example**

```
IHEParse5250r* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            WORD shortVal;
            VARIANT vRow;
            VARIANT vCol;
            VariantInit(&vRow);
            VariantInit(&vCol);
            vRow.vt = VT_I2;
            vRow.iVal = 2;
            vCol.vt = VT_I2;
            vCol.iVal = 4;

            BSTR bstrText = SysAllocString( OLESTR("Hello world") );
            pParser->FindString( bstrText, TRUE, &vRow, &vCol, &shortVal);

            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetCursorPosition

Method, IHEParser 3270 5250 VT

This method gets the cursor position as a row and column value.

### Basic Syntax

```
HEParser.GetCursorPosition Row As Integer, Column As Integer
```

### C++ Syntax

```
HRESULT IHEParser::GetCursorPosition(  
    [out] short *Row,  
    [out] short *Column);
```

### Parameters

*Row*—The row position of the cursor.

*Column*—The column position of the cursor.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Parser.GetCursorPosition Row, Col  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Row = 0;
            short Col = 0;
            pParser->GetCursorPosition(&Row, &Col);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFeature

### Method, IHEParser 3270 5250 VT

This method lets you determine if a particular Parser feature has been enabled.

#### Basic Syntax

```
IHEParser.GetFeature(lType As HEPARSER_FEATURE) As Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::GetFeature(  
    [in] HEPARSER_FEATURE lType,  
    [out, retval] VARIANT_BOOL *pVal);
```

#### Parameters

*lType*—The Parser feature that you want to check.

*pVal*—The return value that indicates the status of the feature. If *pVal* equals VARIANT\_TRUE, the specified Parser feature is enabled; if *pVal* equals VARIANT\_FALSE, the Parser feature is disabled.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
  
Dim bVal As Boolean  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
        Set OManager = CreateObject("HEOhio.OhioManager")  
        Set OSession = OManager.OpenSession(strProfile, strSessionName)  
        OSession.Connect  
        Set Parser = OSession.Parser  
  
        bVal = Parser.GetFeature(HOSTEX_TYPE_AHEAD)  
        OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* Session = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->GetFeature(HOSTEX_TYPE_AHEAD, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldAttribute

### Method, IHEParser 3270 5250

This method returns the field attribute for a specified field. For the HEPAR3270 object, this method returns the 3270 attribute of the specified field. You can use the following literals to test the bit options:

- ATTR\_MODIFIED
- ATTR\_BOLD
- ATTR\_NUMERIC
- ATTR\_PROTECTED
- ATTR\_NONDISPLAY

For the HEPAR5250 object, this method returns the Field Format Word of the specified field. You can use the following literals to test the bit options:

- |                     |                            |
|---------------------|----------------------------|
| • FFW_BYPASS        | • FFW_SIGNEDNUMERIC        |
| • FFW_ALLOWDUP      | • FFW_AUTOENTERONEXIT      |
| • FFW_MDT           | • FFW_FIELDEXITREQUIRED    |
| • FFW_SHIFTEDITMASK | • FFW_MONOCASE             |
| • FFW_ALPHASHIFT    | • FFW_MANDATORYENTRY       |
| • FFW_ALPHAONLY     | • FFW_RIGHTFILLMASK        |
| • FFW_NUMERICSHIFT  | • FFW_NOADJUST             |
| • FFW_NUMERICONLY   | • FFW_RIGHTADJUSTZEROFILL  |
| • FFW_KATASHIFT     | • FFW_RIGHTADJUSTBLANKFILL |
| • FFW_DIGITSONLY    | • FFW_MANDATORYFILL        |
| • FFW_IO            |                            |

#### Basic Syntax

```
IHEParser.GetFieldAttribute(iFieldID As Integer) As Integer
```

#### C++ Syntax

```
HRESULT IHEParser::GetFieldAttribute(  
    [in] short iFieldID,  
    [out, retval] short *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value that specifies the field attribute of the specified field.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

Dim Attribute As Integer

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser

Attribute = Parser.GetFieldAttribute(2)
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short attr = 0;
            pParser->GetFieldAttribute(2, &attr);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldCount

Method, **IHEParser** [3270](#) [5250](#)

This method returns the number of fields on the current screen.

**Basic Syntax**

```
HEParser.GetFieldCount() As Integer
```

**C++ Syntax**

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

**Parameters**

*pVal*—The return value that specifies the number of fields on the current screen.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

Dim Count As Integer

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
Count = Parser.GetFieldCount()
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short count = 0;
            pParser->GetFieldCount( &count );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldExtAttribute

Method, IHEParser **3270 5250**

This method returns the extended attribute of the specified field. The extended attribute defines the color and highlighting for the field. For the HEPAR3270 object, this method returns the extended 3270 attribute of the specified field. You can use the following literals to test the bit options:

- ATTR\_REVERSE
- ATTR\_BLINK
- ATTR\_UNDERLINE

For the HEPAR5250 object, this method returns the Field Control Word of the specified field. You can use the following literals to test the bit options:

- FCW\_LIGHTPEN
- FCW\_STRIPEANDLIGHTPEN

### Basic Syntax

```
HEParser.GetFieldExtAttribute(iFieldID As Integer) As Integer
```

### C++ Syntax

```
HRESULT IHEParser::GetFieldExtAttribute(  
    [in] short iFieldID,  
    [out, retval] short *pVal);
```

### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value that specifies the extended attribute of the specified field.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250
Dim Attribute As Integer

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Attribute = Parser.GetFieldExtAttribute(3)

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();
        pSession->get_Parser(IDispatch **&pParser);
        if(pParser)
        {
            short attr = 0;
            pParser->GetFieldExtAttribute(0, &attr);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldIndex

### Method, IHParser 3270 5250

This method returns the field ID for a specified field position. IDs start at zero (0).

#### Basic Syntax

```
HEParser.GetFieldIndex(nPos As Integer) As Integer
```

#### C++ Syntax

```
HRESULT IHParser::GetFieldIndex(  
    [in] short nPos,  
    [out, retval] short *pVal);
```

#### Parameters

*nPos*—The position of the field that you want to check. Specify the position in terms of the screen buffer. For example, if the screen is 24x80, there are 1920 possible positions. The first position on the screen is 1.

*pVal*—The return value that specifies the field ID for the specified position.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
Dim Index As Integer  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Index = Parser.GetFieldIndex(3)  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short shortVal;
            pParser->GetFieldIndex(14, &shortVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldLength

Method, IHEParser 3270 5250 VT

This method returns the length of the specified field.

### Basic Syntax

```
HEParser.GetFieldLength(iFieldID As Integer) As Integer
```

### C++ Syntax

```
HRESULT IHEParser::GetFieldLength(  
    [in] short iFieldID,  
    [out, retval] short *pVal);
```

### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value that specifies the length of the specified field.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
Dim Length As Integer  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Length = Parser.GetFieldLength(2)  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Length;
            pParser->GetFieldLength(2, &Length);
        }
        pSession->Disconnect();
    }
    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldPos

### Method, IHParser 3270 5250

This method returns the position of the specified field. The returned position is specified in terms of the screen buffer. For example, if the screen is 24x80, there are 1920 possible positions. The first position on the screen is 1.

#### Basic Syntax

```
HEParser.GetFieldPos(iFieldID As Integer) As Integer
```

#### C++ Syntax

```
HRESULT IHParser::GetFieldPos(  
    [in] short iFieldID,  
    [out, retval] short *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value that specifies the position of the specified field in the screen buffer.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
Dim Position As Integer  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Position = Parser.GetFieldPos(3)  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Position = 0;
            pParser->GetFieldPos(2, &Position);

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetFieldText

### Method, IHParser 3270 5250

This method returns the text in the specified field as a string. The system converts all Nulls to spaces and removes all trailing spaces.

#### Basic Syntax

```
HEParser.GetFieldText(iFieldID As Integer) As String
```

#### C++ Syntax

```
HRESULT IHParser::GetFieldText(  
    [in] short iFieldID,  
    [out, retval] BSTR *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return string containing the text of the specified field.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSesession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
Dim str As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
OSesession.Connect  
  
Set Parser = OSesession.Parser  
str = Parser.GetFieldText(3)  
OSesession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrVal = NULL;
            pParser->GetFieldText(2, &bstrVal);

            SysFreeString(bstrVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetScreenText

### Method, IHEParser 3270 5250

This method returns a string containing the portion of screen text that begins at a specified row and column and spans a specified length. The specified row and column must be valid. The first position on the screen is Row 1 Column 1. The length value can be one of the following:

- |    |                                 |
|----|---------------------------------|
| 0  | Copy to End of Field            |
| -1 | Copy to End of Line             |
| -2 | Copy to End of Word             |
| -3 | Copy to End of Screen           |
| >0 | Copy the exact length specified |

#### Basic Syntax

```
HEParser.GetScreenText(iRow As Integer, iColumn As Integer, iLength  
As Integer) As String
```

#### C++ Syntax

```
HRESULT IHEParser::GetScreenText(  
    [in] short iRow,  
    [in] short iColumn,  
    [in] short iLength,  
    [out, retval] BSTR *pVal);
```

#### Parameters

*iRow*—The specified screen row number.

*iColumn*—The specified screen column number.

*iLength*—The specified length of text you want to return.

*pVal*—The return string containing the specified portion of screen text.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

Dim Str As String
Dim Row As Integer
Dim Column As Integer
Dim Length As Integer

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
Row = 2
Column = 23
Length = 8
Str = Parser.GetScreenText(Row, Column, Length)
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Row = 2;
            short Column = 23;
            short Length = 8;

            BSTR bstrVal = NULL;
            pParser->GetScreenText(Row, Column, Length, &bstrVal);
            SysFreeString(bstrVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetSel

Method, IHEParser 3270 5250 VT

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

### Basic Syntax

```
IHEParser.GetSel() As String
```

### C++ Syntax

```
HRESULT IHEParser::GetSel( [out, retval] BSTR *pData);
```

### Parameters

*pData*—The returned string.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim strText As String

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
StrText = Parser.GetSel

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstr = NULL;
            pParser->GetSel( &bstr );
            SysFreeString(bstr);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetSelectionArea

**Method, IHEParser 3270 5250 VT**

This method returns the coordinates (top, left, bottom, right) of the current screen selection.

### Basic Syntax

```
HEParser.GetSelectionArea pTop As Long, pLeft As Long, pBottom As Long,  
pRight As Long
```

### C++ Syntax

```
HRESULT IHEParser::GetSelectionArea(  
    [out] long *pTop,  
    [out] long *pLeft,  
    [out] long *pBottom,  
    [out] long *pRight);
```

### Parameters

*pTop*—The returned top coordinate of the selection.

*pLeft*—The returned left coordinate of the selection.

*pBottom*—The returned bottom coordinate of the selection.

*pRight*—The returned right coordinate of the selection.

### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
  
    Dim pTop As Long  
    Dim pLeft As Long  
    Dim pBottom As Long  
    Dim pRight As Long  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Parser.GetSelectionArea pTop, pLeft, pBottom, pRight  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            long pTop, pLeft, pBottom, pRight;
            pParser->GetSelectionArea( &pTop, &pLeft, &pBottom, &pRight);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetValue

### Method, IHEParser 3270 5250 VT

This method returns the current value of the specified Parser property.

#### Basic Syntax

```
HEParser.GetValue(lType As HEPARSER_VALUE) As Long
```

#### C++ Syntax

```
HRESULT IHEParser::GetValue(  
    [in] HEPARSER_VALUE lType,  
    [out, retval] LPARAM *pVal);
```

#### Parameters

*lType*—The Parser property you want to check.

*pVal*—The returned value of the specified property.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
Dim iVal as HEPARSER_VALUE  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
  
iVal = Parser.GetValue(HOSTEX_BELL_MARGIN)  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID id = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            LPARAM pProp;
            pParser->GetValue(HOSTEX_DISPLAY_IN_OIA, &pProp);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HostToAscii

### Method, IHEParser 3270 5250 VT

This method converts an EBCDIC host character into the equivalent ASCII character.

#### Basic Syntax

```
HEParser.HostToAscii(inchar As BYTE) As BYTE
```

#### C++ Syntax

```
HRESULT IHEParser::HostToAscii(  
    [in] BYTE inChar,  
    [out, retval] BYTE *outChar);
```

#### Parameters

*inChar*—The input character in EBCDIC format that you want to convert.

*OutChar*—The return character that has been converted to ASCII format.

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
  
Dim HostChar As Byte  
Dim AsciiChar As Byte  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
  
HostChar = 78  
AsciiChar = Parser.HostToAscii(HostChar)  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BYTE HostChar= 65;
            BYTE AsciiChar;
            pParser->HostToAscii(HostChar, &AsciiChar);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsFieldBold

### Method, IHEParser 3270 5250

This method indicates whether the specified field displays in bold.

#### Basic Syntax

```
HEParser.IsFieldBold(iFieldID As Integer) As Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::IsFieldBold(  
    [in] short iFieldID,  
    [out, retval] VARIANT_BOOL *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the specified field is bold. If *pVal* is VARIANT\_FALSE, the field is not bold.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim bVal As Boolean  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    bVal = Parser.IsFieldBold(3)  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->IsFieldBold(3, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsFieldHidden

### Method, IHParser 3270 5250

This method indicates whether the specified field is a non-display field.

#### Basic Syntax

```
HEParser.IsFieldHidden(iFieldID As Integer) As Boolean
```

#### C++ Syntax

```
HRESULT IHParser::IsFieldHidden([in] short iFieldID, [out, retval]  
VARIANT_BOOL *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the specified field is a non-display field. If *pVal* is VARIANT\_FALSE, the field is a display field.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim bVal As Boolean  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = IsFieldHidden (3)  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->IsFieldHidden(3, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsFieldModified

### Method, IHParser 3270 5250

This method indicates whether the specified field has been modified.

#### Basic Syntax

```
HEParser.IsFieldModified(iFieldID As Integer) As Boolean
```

#### C++ Syntax

```
HRESULT IHParser::IsFieldModified([in] short iFieldID, [out, retval]  
VARIANT_BOOL *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the specified field has been modified. If *pVal* is VARIANT\_FALSE, the field has not been modified.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim bVal As Boolean  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = Parser.IsFieldModified(3)  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->IsFieldModified(3, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsFieldNumeric

Method, IHEParser 3270 5250

This method indicates whether the specified field is numeric only.

### Basic Syntax

```
HEParser.IsFieldNumeric(iFieldID As Integer) As Boolean
```

### C++ Syntax

```
HRESULT IHEParser::IsFieldNumeric([in] short iFieldID, [out, retval]  
VARIANT_BOOL *pVal);
```

### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the specified field is numeric only. If *pVal* is VARIANT\_FALSE, the field contains non-numeric data.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim bVal As Boolean  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = Parser.IsFieldNumeric(3)  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->IsFieldNumeric(3, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsFieldPenSelectable

### Method, IHEParser 3270 5250

This method indicates whether the specified field is pen-selectable.

#### Basic Syntax

```
HEParser.IsFieldPenSelectable(iFieldID As Integer) As Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::IsFieldPenSelectable([in] short iFieldID, [out, retval]  
VARIANT_BOOL *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the specified field is pen-selectable. If *pVal* is VARIANT\_FALSE, the field is not pen-selectable.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim bVal As Boolean  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = Parser. IsFieldPenSelectable(3)  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->IsFieldPenSelectable(3, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsFieldProtected

### Method, IHParser 3270 5250

This method indicates whether the specified field is a protected field.

#### Basic Syntax

```
HEParser.IsFieldProtected(iFieldID As Integer) As Boolean
```

#### C++ Syntax

```
HRESULT IHParser::IsFieldProtected([in] short iFieldID, [out, retval]  
VARIANT_BOOL *pVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to check. IDs start at zero (0).

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the specified field is protected. If *pVal* is VARIANT\_FALSE, the field is not protected.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim bVal As Boolean  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = Parser. IsFieldProtected (3)  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->IsFieldProtected( 3, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## MoveCursorRelative

### Method, IHEParser 3270 5250 VT

This method moves the cursor a specified number of rows and columns from the current position.

#### Basic Syntax

```
IHEParser.MoveCursorRelative iRows As Integer, iCols As Integer
```

#### C++ Syntax

```
HRESULT IHEParser::MoveCursorRelative([in] short iRows, [in] short iCols);
```

#### Parameters

*iRows*—The number of rows by which you want to move the cursor from the current position.

*iCols*— The number of columns by which you want to move the cursor from the current position.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.MoveCursorRelative -1, 3
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->MoveCursorRelative( -1, 3 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PasteDataToScreen

### Method, IHEParser 3270 5250

This method copies data from the Clipboard to the current cursor position.

#### Basic Syntax

```
HEParser.PasteDataToScreen()
```

#### C++ Syntax

```
HRESULT IHEParser::PasteDataToScreen();
```

#### Parameters

This method has no parameters.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
Parser.PasteDataToScreen
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );
CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->PasteDataToScreen();
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PutString

Method, [IHEParser](#) [3270](#) [5250](#)

This method copies the specified string to the current cursor position.

### Basic Syntax

```
IHEParser.PutString(newVal As String)
```

### C++ Syntax

```
HRESULT IHEParser::PutString([in] BSTR newVal);
```

### Parameters

*newVal*—The string you want to copy to the current cursor position.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.PutString "Here is a string"

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrVal = SysAllocString( OLESTR("This is a string") );
            pParser->PutString(bstrVal);
            SysFreeString(bstrVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PutText

### Method, IHEParser 3270 5250 VT

This method lets you write text to a specified unprotected location on the screen. The first location on the screen is Row 1 Column 1.

#### Basic Syntax

```
IHEParser.PutText bstrText As String, vRow As Variant, vCol As Variant
```

#### C++ Syntax

```
HRESULT IHEParser::PutText([in] BSTR bstrText, [in] VARIANT vRow, [in] VARIANT vCol);
```

#### Parameters

*bstrText*—The text you want to write to the specified row and column.

*vRow*—The row in which you want to write the text.

*vCol*—The column in which you want to write the text.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSesession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim Row as Integer
Dim Col as Integer

strProfile = "c:\\\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSesession = OManager.OpenSession(strProfile, strSessionName)
OSesession.Connect

Set Parser = OSesession.Parser
Row = 3
Col = 5
Parser.PutText "Hello world", Row, Col
OSesession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = SysAllocString( OLESTR("Hello world") );
            VARIANT vRow;
            VARIANT vCol;
            VariantInit(&vRow);
            VariantInit(&vCol);

            vRow.vt = VT_I2;
            vRow.iVal = 3;
            vCol.vt = VT_I2;
            vCol.iVal = 5;

            pParser->PutText( bstrText, vRow, vCol);

            VariantClear(&vRow);
            VariantClear(&vCol);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }
    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ReceiveFile

### Method, IHEParser 3270 VT

This method lets you transfer (download) a file from the host system to your machine.

#### Basic Syntax

```
IHEParser.ReceiveFile bstrPCfileName As String, vHostName As Variant,  
vOptions As Variant
```

#### C++ Syntax

```
HRESULT IHEParser::ReceiveFile([in] BSTR bstrPCFileName, [in] VARIANT  
vHostName, [in] VARIANT vOptions);
```

#### Parameters

*bstrPCfileName*—The name of the file on your machine that receives the file on the host system.

*vHostName*—The name of the file on the host system that you want to download.

*vOptions*—The file transfer options you want to use for the download operation. If you are using CMS, you must precede the options list with an open parenthesis. For more information on the available options, see the following topics:

- General Options
- CMS-Specific Options on Upload
- TSO-Specific Options on Upload
- MUSIC-Specific Options on Upload

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser3270  
  
    strProfile = "c:\\Display3270.HEP"  
    strSessionName = "3270"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Parser.ReceiveFile "c:\\test.txt", "TEST TXT", "( ASCII CRLF"  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID id = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrFile = SysAllocString( OLESTR("c:\\test.txt") );

            VARIANT vHostFile;
            VariantInit(&vHostFile);
            vHostFile.vt = VT_BSTR;
            vHostFile.bstrVal = SysAllocString( OLESTR("CONFIG SYS A1") );
            VARIANT vOptions;
            VariantInit(&vOptions);
            vOptions.vt = VT_BSTR;
            vOptions.bstrVal = SysAllocString( OLESTR("( ASCII CRLF") );

            pParser->ReceiveFile( bstrFile, vHostFile, vOptions );

            VariantClear(&vHostFile);
            VariantClear(&vOptions);

            SysFreeString(bstrFile);
        }
        pSession->Disconnect();
    }
    pApplication->Release();
}
SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ReplaceSel

Method, IHEParser **3270 5250 VT**

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

### Basic Syntax

```
HEParser.ReplaceSel newText As String
```

### C++ Syntax

```
HRESULT IHEParser::ReplaceSel(BSTR newText);
```

### Parameters

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

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.ReplaceSel "this is a string"

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = SysAllocString( OLESTR("this is a string") );
            pParser->ReplaceSel(bstrText);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SendAid

### Method, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
HEParser.SendAid nAidKey As Long
```

#### C++ Syntax

```
HRESULT IHEParser::SendAid([in] long nAidKey);
```

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.SendAid VK3_PRINTSCREEN
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->SendAid( VK3_PRINTSCREEN);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## **TN3270 Keyboard Mapping**

The following list outlines the keyboard mapping for TN3270 terminals:

- ALA-Alternate-Input VK3\_ALA\_INPUT
- Change-Graphics-Cursor VK3\_TOGGLEGRAPHICSCCURSOR
- Check-Hotspot VK3\_CHECKHOTSPOT
- Close-Window VK3\_CLOSEWINDOW
- Cursor-Select VK3\_CURSORSELECT
- Dead-Key VK3\_DEAD
- Dlg-API-Settings IDM\_O\_EHLLAPI
- Dlg-Close-Session IDM\_F\_CLOSESESSION
- Dlg-Download IDM\_T\_RECEIVEDOWNLOAD
- Dlg-Edit-Session-Profile IDM\_O\_PREFERENCES
- Dlg-Exit IDM\_F\_EXIT
- Dlg-Font-Select IDM\_N\_SELECTFONT
- Dlg-Global IDM\_O\_GLOBAL
- Dlg-Hotspots IDM\_O\_HOTSPOTS
- Dlg-Keyboard-Mapper IDM\_O\_KEYMAPR
- Dlg-Open-Session IDM\_F\_OPENSESSION
- Dlg-Poppad-Configure IDM\_O\_POPPAD
- Dlg-Poppad-Custom IDM\_V\_CUSTOM\_POPPAD
- Dlg-Poppad-Default IDM\_V\_DEFAULT\_POPPAD
- Dlg-Print-Screen IDM\_F\_PRINTSCREEN
- Dlg-Prompt-Demo-File VK3\_ASKFORDEMOFILE
- Dlg-Prompt-Password VK3\_PROMPTFORPASSWORD
- Dlg-Quick-Key-Editor IDM\_O\_MACRO
- Dlg-Run-Macro IDM\_M\_MACRORUN

• Dlg-Run-Program	VK3_RUNPGMDLG
• Dlg-Save-Demo-File	VK3_SAVEDEMOFILE
• Dlg-Save-Profile	IDM_F_SAVESESSION
• Dlg-Save-Screen	IDM_F_SAVESCREEN
• Dlg-Translate-Tables	IDM_O_XLATEDLG
• Dlg-Upload	IDM_T_SENDUPLOAD
• End-Recording	IDM_M_MACROENDRECORDING
• Font-Larger	IDM_N_NEXTLARGERFONT
• Font-Smaller	IDM_N_NEXTSMALLERFONT
• Help-Index	IDM_INDEXHELP
• Help-Keys	IDM_KEYHELP
• Jump-To-Session	VK3_JUMPSESSION
• Insert-Field-Attribute	VK3_INSERTFIELDATTRIBUTE
• Ipause	VK3_IPAUSE
• Maximize-Font	IDM_N_MAXIMIZEFONT
• Mouse-To-Cursor	VK3_MOUSETOCURSOR
• Move-Cursor-Cursor-Select	VK3_MOVECURSORANDCURSORSELECT
• Move-Cursor-Enter	VK3_MOVECURSORANDENTER
• Next-Session	VK3_NEXT_SESSION
• Password	VK3_PASSWORD
• Pause	VK3_PAUSE
• Prev-Session	VK3_PREV_SESSION
• Print-Raw	VK3_PRINTRAW
• Print-Raw-LPT1	VK3_PRINTRAWLPT1
• Print-Raw-LPT2	VK3_PRINTRAWLPT2
• Print-Raw-LPT3	VK3_PRINTRAWLPT3
• Print-Screen	VK3_PRINTSCREEN

• Record-Macro	IDM_M_MACRORECORD
• Reset-Type-Ahead	VK3_RESET_TYPEAHEAD
• Restore-Cursor-Position	VK3_RESTORECURSORPOSITION
• Run	VK3_RUNPGM
• Run-Macro	VK3_RUNSCRIPT
• Save-Cursor-Position	VK3_SAVECURSORPOSITION
• Save-Screen	VK3_SAVESCREEN
• Show-Demo-File	VK3_SHOWDEMOFILE
• Show-Track-Menu	VK3_SHOWTRACKMENU
• Start-Session	VK3_STARTSESSION
• Toggle-APL-Keyboard	VK3_APLTOGGLE
• Toggle-Attribute	VK3_FLIP_ATTR
• Toggle-Capture	IDM_F_CAPTURE
• Toggle-Connection	VK3_TOGGLE_CONNECTION
• Toggle-CrossHair-Cursor	VK3_TOGGLECROSSHAIR
• Toggle-Cursor	VK3_TOGGLE_CURSOR
• Toggle-Entry-Assist	VK3_ENTRY_ASSIST
• Toggle-Full-Screen	VK3_FULLSCREEN
• Toggle-Recording-Pause	IDM_M_MACROPAUSERECORDING
• Toggle-Toolbar	IDM_V_TOOLBAR
• Toggle-Tracing	VK3_TOGGLETRACE
• Toggle-Word-Wrap	VK3_WORD_WRAP
• Whats-This?	VK3_WHATSTHIS

## **TN5250 Keyboard Mapping**

The following keystrokes are available for TN5250 terminals:

• Check-Hotspot	VK3_CHECKHOTSPOT
• Close-Window	VK3_CLOSEWINDOW
• Cursor-Select	VK3_CURSORSELECT
• Dead-Key	VK3_DEAD
• Dlg-API-Settings	IDM_O_EHLLAPI
• Dlg-Close-Session	IDM_F_CLOSESESSION
• Dlg-Download	IDM_T_RECEIVEDOWNLOAD
• Dlg-Edit-Session-Profile	IDM_O_PREFERENCES
• Dlg-Exit	IDM_F_EXIT
• Dlg-Font-Select	IDM_N_SELECTFONT
• Dlg-Global	IDM_O_GLOBAL
• Dlg-Hotspots	IDM_O_HOTSPOTS
• Dlg-Keyboard-Mapper	IDM_O_KEYMAPR
• Dlg-Open-Session	IDM_F_OPENSESSION
• Dlg-Poppad-Configure	IDM_O_POPPAD
• Dlg-Poppad-Custom	IDM_V_CUSTOM_POPPAD
• Dlg-Poppad-Default	IDM_V_DEFAULT_POPPAD
• Dlg-Print-Screen	IDM_F_PRINTSCREEN
• Dlg-Prompt-Demo-File	VK3_ASKFORDEMOFILE
• Dlg-Prompt-Password	VK3_PROMPTFORPASSWORD
• Dlg-Quick-Key-Editor	IDM_O_MACRO
• Dlg-Run-Macro	IDM_M_MACRORUN
• Dlg-Run-Program	VK3_RUNPGMDLG
• Dlg-Save-Demo-File	VK3_SAVEDEMOFILE

• Dlg-Save-Profile	IDM_F_SAVESESSION
• Dlg-Save-Screen	IDM_F_SAVESCREEN
• Dlg-Translate-Tables	IDM_O_XLATEDLG
• Dlg-Upload	IDM_T_SENDUPLOAD
• End-Recording	IDM_M_MACROENDRECORDING
• Font-Larger	IDM_N_NEXTLARGERFONT
• Font-Smaller	IDM_N_NEXTSMALLERFONT
• Help-Index	IDM_INDEXHELP
• Help-Keys	IDM_KEYHELP
• Insert-Field-Attribute	VK3_INSERTFIELDATTRIBUTE
• IPause	VK3_IPAUSE
• Jump-To-Session	VK3_JUMPSESSION
• Maximize-Font	IDM_N_MAXIMIZEFONT
• Mouse-To-Cursor	VK3_MOUSETOCURSOR
• Move-Cursor-Cursor-Select	VK3_MOVECURSORANDCURSORSELECT
• Move-Cursor-Enter	VK3_MOVECURSORANDENTER
• Next-Session	VK3_NEXT_SESSION
• Password	VK3_PASSWORD
• Pause	VK3_PAUSE
• Prev-Session	VK3_PREV_SESSION
• Print-Raw	VK3_PRINTRAW
• Print-Raw-LPT1	VK3_PRINTRAWLPT1
• Print-Raw-LPT2	VK3_PRINTRAWLPT2
• Print-Raw-LPT3	VK3_PRINTRAWLPT3
• Print-Screen	VK3_PRINTSCREEN
• Record-Macro	IDM_M_MACRORECORD
• Reset-Type-Ahead	VK3_RESET_TYPEAHEAD

• Restore-Cursor-Position	VK3_RESTORECURSORPOSITION
• Run	VK3_RUNPGM
• Run-Macro	VK3_RUNSCRIPT
• Save-Cursor-Position	VK3_SAVECURSORPOSITION
• Save-Screen	VK3_SAVESCREEN
• Show-Demo-File	VK3_SHOWDEMOFILE
• Show-Track-Menu	VK3_SHOWTRACKMENU
• Start-Session	VK3_STARTSESSION
• Toggle-Attribute	VK3_FLIP_ATTR
• Toggle-Capture	IDM_F_CAPTURE
• Toggle-Connection	VK3_TOGGLE_CONNECTION
• Toggle-CrossHair-Cursor	VK3_TOGGLECROSSHAIR
• Toggle-Cursor	VK3_TOGGLE_CURSOR
• Toggle-Entry-Assist	VK3_ENTRY_ASSIST
• Toggle-Full-Screen	VK3_FULLSCREEN
• Toggle-Recording-Pause	IDM_M_MACROPAUSERECORDING
• Toggle-Toolbar	IDM_V_TOOLBAR
• Toggle-Tracing	VK3_TOGGLETRACE
• Toggle-Word-Wrap	VK3_WORD_WRAP
• Whats-This?	VK3_WHATSTHIS

## SendFile

### Method, IHEParser 3270 VT

This method lets you transfer (upload) a specified file from your machine to the host system.

#### Basic Syntax

```
IHEParser.SendFile bstrPCFileName As String, vHostName As Variant,  
vOptions As Variant
```

#### C++ Syntax

```
HRESULT IHEParser::SendFile([in] BSTR bstrPCFileName, [in] VARIANT  
vHostName, [in] VARIANT vOptions);
```

#### Parameters

*bstrPCFileName*—The name of the file on your machine that you want to transfer to the host system.

*vHostName*—The name of the file on the host system that receives the uploaded file.

*vOptions*—The file transfer options you want to use for the upload operation. If you are using CMS, you must precede the options list with an open parenthesis. For more information on the available options, see the following topics:

- General Options
- CMS-Specific Options on Upload
- TSO-Specific Options on Upload
- MUSIC-Specific Options on Upload

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser3270  
  
    strProfile = "c:\\Display3270.HEP"  
    strSessionName = "3270"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Parser.SendFile "c:\\test.txt", "TEST TXT", "( ASCII CRLF"  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();
        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrFile = SysAllocString( OLESTR("c:\\test.txt") );
            VARIANT vHostFile;
            VariantInit(&vHostFile);

            vHostFile.vt = VT_BSTR;
            vHostFile.bstrVal = SysAllocString( OLESTR("CONFIG SYS A1") );

            VARIANT vOptions;
            VariantInit(&vOptions);

            vOptions.vt = VT_BSTR;
            vOptions.bstrVal = SysAllocString( OLESTR("( ASCII CRLF") );

            pParser->SendFile( bstrFile, vHostFile, vOptions );

            VariantClear(&vHostFile);
            VariantClear(&vOptions);
            SysFreeString(bstrFile);
        }
        pSession->Disconnect();
    }
    pApplication->Release();
}
SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SendKeys

Method, IHEParser 3270 5250 VT

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

### Basic Syntax

```
HEParser.SendKeys pBuffer As String
```

### C++ Syntax

```
HRESULT IHEParser::SendKeys([in] BSTR pBuffer);
```

### Parameters

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

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

        Set Parser = OSession.Parser
        Parser.SendKeys "[enter]"

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = SysAllocString( OLESTR("[enter]") );
            pParser->SendKeys( bstrText );
            SysFreeString( bstrText );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetCursorPosition

### Method, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
HEParser.SetCursorPosition Row As Integer, Column As Integer
```

#### C++ Syntax

```
HRESULT IHEParser::SetCursorPosition([in] short Row, [in] short Column);
```

#### Parameters

*Row*—The row position of the cursor.

*Column*—The column position of the cursor.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim Row as Integer  
Dim Col as Integer
```

```
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
    Set Parser = OSession.Parser  
    Row = 3  
    Col = 5  
    Parser.SetCursorPosition Row, Col
```

```
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
ID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Row = 3;
            short Col = 5 ;
            pParser->SetCursorPosition( Row, Col );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetFeature

Method, IHEParser 3270 5250 VT

This method lets you enable or disable a particular Parser feature.

### Basic Syntax

```
HEParser.SetFeature lType As HEPARSER_FEATURE, newVal As Boolean
```

### C++ Syntax

```
HRESULT IHEParser::SetFeature([in] HEPARSER_FEATURE lType, [in] VARIANT_BOOL newVal);
```

### Parameters

*lType*—The Parser feature that you want to enable or disable.

*newVal*—The new status for the specified feature. To enable the feature, set *newVal* to VARIANT\_TRUE. To disable the feature, set *newVal* to VARIANT\_FALSE.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser. SetFeature HOSTEX_TYPE_AHEAD, True

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->SetFeature( HOSTEX_TYPE_AHEAD, TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetFieldText

### Method, IHEParser 3270 5250

This method lets you specify the text for a particular field.

#### Basic Syntax

```
HEParser.SetFieldText iFieldID As Integer, newVal As String
```

#### C++ Syntax

```
HRESULT IHEParser::SetFieldText([in] short iFieldID, [in] BSTR newVal);
```

#### Parameters

*iFieldID*—The ID of the field you want to modify. IDs start at zero (0).

*newVal*—The text you want to place in the specified field.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser. SetFieldText 3, "this is a new string"

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = SysAllocString( OLESTR("Hello world") );
            pParser->SetFieldText(2, bstrText);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetSel

### Method, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
IHEParser.SetSel StartRow As Integer, StartCol As Integer, EndRow As Integer, EndCol As Integer
```

#### C++ Syntax

```
HRESULT IHEParser::SetSel([in] short StartRow, [in] short StartCol, [in] short EndRow, [in] short EndCol);
```

#### Parameters

*StartRow*—The first row of the selection.

*StartCol*—The first column of the selection.

*EndRow*—The last row of the selection

*EndCol*—The last column of the selection.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.SetSel 6, 17, 8, 33

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->SetSel(6, 17, 8, 33);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetValue

### Method, IHEParser 3270 5250 VT

This method lets you set the value of a particular Parser property.

#### Basic Syntax

```
HEParser.SetValue lType As HEPARSER_VALUE, newVal As Long
```

#### C++ Syntax

```
HRESULT IHEParser::SetValue([in] HEPARSER_VALUE lType, [in] LPARAM  
newVal);
```

#### Parameters

*lType*—The Parser property you want to set.

*newVal*—The new value for the specified property.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.SetValue HOSTEX_BELL_MARGIN, 1
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            LPARAM pProp = 1;
            pParser->SetValue( HOSTEX_DISPLAY_IN_OIA, pProp );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitConnected

### Method, IHEParser 3270 5250 VT

This method lets you specify a maximum wait time for establishing a connection to the host system.

#### Basic Syntax

```
HEParser.WaitConnected lIdleTime As Long
```

#### C++ Syntax

```
HRESULT IHEParser::WaitConnected([in] long lIdleTime);
```

#### Parameters

*lIdleTime*—The wait time in seconds.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.WaitConnected 4
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->WaitConnected( 4 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitForCursor

### Method, IHEParser 3270 5250 VT

This method lets you specify a wait time for the cursor to move into a particular row (and, optionally, a particular column).

#### Basic Syntax

```
IHEParser.WaitForCursor(iRow As Integer, vCol As Variant, lTimeout As Long)  
As Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::WaitForCursor([in] short iRow, [in] VARIANT vCol, [in]  
long lTimeOut, [out, retval] VARIANT_BOOL *pVal);
```

#### Parameters

*iRow*—The row you want to check.

*vCol*—The optional column you want to check.

*lTimeOut*—The wait time in seconds.

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the cursor moved into the specified row (and column) within the specified time. If *pVal* is VARIANT\_FALSE, the cursor did not move into the row (and column) within the time period.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSesession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
    Dim Row as Integer  
    Dim Col As Integer  
    Dim Timeout As Long  
    Dim bVal As Boolean  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
    OSesession.Connect  
  
    Set Parser = OSesession.Parser  
  
    Row = 5  
    Col = 31  
    Timeout = 2000  
    bVal = Parser.WaitForCursor( Row, Col, Timeout)  
  
    OSesession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Row = 10;
            long Timeout = 5000;

            VARIANT_BOOL vbVar;

            VARIANT vCol;
            VariantInit( &vCol );

            vCol.vt = VT_I2;
            vCol.iVal = 5;
            pParser->WaitForCursor(Row, vCol, Timeout, &vbVar);

            VariantClear(&vCol);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitForCursorMove

### Method, IHEParser 3270 5250 VT

This method lets you specify a wait time for the cursor to move through a particular number of rows (and, optionally, a particular number of columns).

#### Basic Syntax

```
HEParser.WaitForCursorMove(iRows As Integer, iCols As Variant, lTimeout As Long) As Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::WaitForCursorMove([in] short iRows, [in] VARIANT iCols, [in] long lTimeOut, [out, retval] VARIANT_BOOL *pVal);
```

#### Parameters

*iRows*—The number of rows through which the cursor must move.

*iCols*—The optional number of columns through which the cursor must move.

*lTimeOut*—The wait time in seconds.

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the cursor moved through the specified number of rows (and columns) within the specified time. If *pVal* is VARIANT\_FALSE, the cursor did not move the required amount within the time period.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSesession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    Dim Row as Integer
    Dim Col As Integer
    Dim Timeout As Long
    Dim bVal As Boolean

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSesession = OManager.OpenSession(strProfile, strSessionName)
    OSesession.Connect

    Set Parser = OSesession.Parser
    Row = 6
    Col = 30
    Timeout = 2000
    bVal = Parser.WaitForCursorMove(Row, Col, Timeout)

    OSesession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Row = 10;
            long Timeout = 5000;

            VARIANT_BOOL vbVar;
            VARIANT vCol;
            VariantInit( &vCol );

            vCol.vt = VT_I2;
            vCol.iVal = 5;

            pParser->WaitForCursorMove(Row, vCol, Timeout, &vbVar);
            VariantClear(&vCol);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitForIO

### Method, IHParser 3270 5250

This method lets you specify a wait time for the user to press an action key such as Enter or PFx.

#### Basic Syntax

```
HEParser.WaitForIO lTimeout As Long
```

#### C++ Syntax

```
HRESULT IHParser::WaitForIO([in] long lTimeout);
```

#### Parameters

*lTimeOut*—The wait time in seconds.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.WaitForIO 4

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->WaitForIO( 4 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitForString

Method, IHEParser **3270 5250 VT**

This method lets you specify a wait time for a particular string to appear in a particular row and column. You can limit the search for the string to a certain row or specify a certain row as the upper limit for the search. You can also force case-sensitive searches.

### Basic Syntax

```
IHEParser.WaitForString(szString As String, iRow As Integer, iCol As Integer, iFlag As Integer, lTimeOut As Long, bCaseSensitive As Boolean) As Integer
```

### C++ Syntax

```
HRESULT IHEParser::WaitForString([in] BSTR szString, [in] short iRow, [in] short iCol, [in] short iFlag, [in] long lTimeOut, [in] VARIANT_BOOL bCaseSensitive, [out, retval] short *pVal);
```

### Parameters

*szString*—The string for which you want to wait.

*iRow*—The row in which the string must appear (or, if *iFlag* = 2, the uppermost row in which the string can appear). *iRow* must be  $\geq 1$ .

*iCol*—The column in which the string must appear. *iCol* must be  $\geq 1$ .

*iFlag*—The row options flag. If *iFlag* = 1, the specified string must appear in *iRow*. If *iFlag* = 2, the string must appear in any row greater than or equal to *iRow*.

*lTimeOut*—The wait time in seconds.

*bCaseSensitive*—The case-sensitive option. If *bCaseSensitive* is VARIANT\_TRUE, the on-screen string must match *szString* in case and content. If *bCaseSensitive* is VARIANT\_FALSE, the on-screen string does not need to have the same lettercase as *szString*.

*pVal*—The search result. If *pVal* = 0, the specified string did not appear within the required time. If *pVal* > 0, *pVal* records the row in which the string did appear.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    Dim Row As Integer
    Dim Col As Integer
    Dim Flag As Integer
    Dim iVal As Integer
    Dim Timeout As Integer
    Dim bCase As Boolean

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Row = 1
    Col = 20
    Flag = 1
    Timeout = 4000
    bCase = True

    iVal = Parser.WaitForString("Sign On", Row, Col, Flag, Timeout, bCase)
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BOOL bCaseSensitive = TRUE;
            short iFlag = 1;
            short Row = 1;
            short Col = 3;
            short shortVal;
            long Timeout = 5000;

            BSTR bstrVal = SysAllocString( OLESTR("Sign On"));
            pParser-
>WaitForString(bstrVal, Row, Col, iFlag, Timeout, bCaseSensitive, &shortV
al);

            SysFreeString(bstrVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitHostQuiet

Method, IHEParser **3270 5250 VT**

This method lets you specify a wait time for the host to be idle after you have connected to the host. You also specify the length of idle time for which you want to wait.

### Basic Syntax

```
HEParser.WaitHostQuiet(vTime As Variant, lTimeout As Long) As Boolean
```

### C++ Syntax

```
HRESULT IHEParser::WaitHostQuiet([in] VARIANT vTime, [in] long lTimeOut,  
[out, retval] VARIANT_BOOL *pVal);
```

### Parameters

*vTime*—The length of idle time (in seconds) on the host for which you want to wait.

*lTimeOut*—The wait time in seconds.

*pVal*—The return value. If *pVal* is VARIANT\_TRUE, the required period of idle time occurred on the host during the specified time. If *pVal* is VARIANT\_FALSE, the required period of idle time did not occur on the host within the specified time.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
    Dim iVal As Integer  
    Dim iVal2 As Integer  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    iVal2 = 1000  
    iVal = Parser.WaitHostQuiet(iVal2, 5000)  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            long lTimeOut = 5000;
            VARIANT_BOOL vbVal;

            VARIANT vTime;
            VariantInit(&vTime);

            vTime.vt = VT_I2;
            vTime.iVal = 5000;

            pParser->WaitHostQuiet(vTime, lTimeOut, &vbVal);
            VariantClear(&vTime);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitIdle

### Method, IHParser 3270 5250 VT

This method lets you specify a period of idle time for the screen. An idle screen is one that does not change during the specified period.

#### Basic Syntax

```
HEParser.WaitIdle lIdleTime As Long
```

#### C++ Syntax

```
HRESULT IHParser::WaitIdle([in] long lIdleTime);
```

#### Parameters

*lIdleTime* —The length of idle time (in milliseconds) on the screen for which you want to wait.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.WaitIdle 3000

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->WaitIdle(3000);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitPSUpdated

### Method, IHEParser 3270 5250 VT

This method lets you specify a wait time for the screen to be updated.

#### Basic Syntax

```
HEParser.WaitPSUpdated(lIdleTime As Long, vWaitKeyb As Variant) As Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::WaitPSUpdated([in] long lIdleTime, [in] VARIANT vWaitKeyb, [out, retval] short *pVal);
```

#### Parameters

*lIdleTime*—The length of time (in seconds) for which you want to wait for the screen to update.

*vWaitKeyb*—A value that specifies whether HostExplorer waits until the keyboard unlocks.

*pVal*—The return value. If *pVal* = 0, the screen updated within the specified time. If *pVal* = 1, the screen did not update within the specified time.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

Dim bVal As Boolean

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    bVal = Parser.WaitPSUpdated(1000, True)

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short iRet = 0;
            long lTimeOut = 5000;

            VARIANT vWaitkey;
            VariantInit(&vWaitkey);

            vWaitkey.vt = VT_I2;
            vWaitkey.iVal = 1;

            pParser->WaitPSUpdated(lTimeOut, vWaitkey, &iRet);
            VariantClear(&vWaitkey);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WaitXfer

**Method, IHEParser 3270 5250 VT**

This method suspends program operation until the preceding file transfer is complete. You can use this method to synchronize your program with the host system.

**Basic Syntax**      `IHEParser.WaitXfer`

**C++ Syntax**      `HRESULT IHEParser::WaitXfer();`

**Parameters**      This method has no parameters.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser5250

    strProfile = "c:\\\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.WaitXfer
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->WaitXfer();
        }
        pSession->Disconnect();

    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## WriteProtectedText

### Method, IHEParser 3270 5250

This method writes specified text at a specified position (row and column) in the screen buffer.

#### Basic Syntax

```
IHEParser.WriteProtectedText iRow As Integer, iCol As Integer, bstrText as String
```

#### C++ Syntax

```
HRESULT IHEParser::WriteProtectedText([in] short iRow, [in] short iCol, [in] BSTR bstrText);
```

#### Parameters

*iRow*—The row in the screen buffer where you want to write the text.

*iCol*—The column in the screen buffer where you want to write the text.

*bstrText*—The string containing the text you want to write to the screen buffer.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim Row As Integer
Dim Col As Integer

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
Row = 3
Col = 5
Parser.WriteProtectedText Row, Col, "Hello"

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Row = 0;
            short Col = 0;

            BSTR bstrText = SysAllocString( OLESTR("Hello world") );
            pParser->WriteProtectedText(Row, Col, bstrText);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Properties of the Parser Objects

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

- Answerback
- APILInputMode
- AutoWrap
- BufRows
- CanChangeScreen
- CaptureMode
- CellCopyMode
- ConnectBy
- ConnectErrorStatus
- ConnectRC
- ConvertNulls
- CutMode
- EnableAPL
- EnablePrinterTimeout
- GraphicsModel
- HistoryLines
- HLLAPIName
- HostResponseTime
- HostWritableString
- KeyboardLocked
- ModelColumns
- ModelRows
- MoveCursorOnMouseClicked
- NRCID
- NumericCharacters
- NVTMode
- OIAString
- OIAStringW
- OnCopyReplaceFieldAttributeWith
- OnPasteFieldModeTabCharacter
- PasteMode
- PrintByPassWindows
- PrinterDeInitString
- PrintDisableTranslation
- PrinterInitString
- PrinterTimeout
- PrinterTimeoutValue
- PrintLFtoCRLF
- ProgramSymbols
- SaveAppend
- SaveFileName
- ScreenChanged
- ScreenCol
- ScreenRow
- SelectionMode
- SessionName
- SoftCharacterSetID
- StatusLineMode
- TerminalID
- TerminalModel
- Text
- Transport
- TypeAheadTimeout
- UPSS
- ValidateNumericFieldData
- TransferErrorCode
- TransferMode

## Answerback

### Property, IHEParser VT

This property sets the Answer Back message. The message includes control codes for the VT host.

#### Basic Syntax

```
HEParser.Answerback = String
```

#### C++ Syntax

```
HRESULT IHEParser::put_Answerback([in] BSTR newVal);
```

#### Parameters

*newVal*—The Answer Back message.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.Answerback = "use a special string with control codes"

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = SysAllocString( OLESTR("Special string with contr
ol codes") );
            pParser->Answerback( bstrText );
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## APLInputMode

### Property, IHEParser 3270

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

#### Basic Syntax

```
Boolean = HEParse.APLInputMode
```

#### C++ Syntax

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

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the session is in APL input mode. A returned value of VARIANT\_FALSE indicates that the session is not in APL input mode.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse3270
Dim bVal As Boolean

strProfile = "c:\\Display3270.HEP"
strSessionName = "3270"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
bVal = Parser.APLInputMode

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->get_APILInputMode( &vbVal);

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## BufRows

### Property, IHEParser VT

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

#### Basic Syntax

```
Integer = HEParse.BufRows
```

```
HEParse.BufRows = Integer
```

#### C++ Syntax

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

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

#### Parameters

*pVal*—The returned maximum number of rows.

*newVal*—The maximum number of rows that you set.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Parser As HEParseVT
```

```
Dim iVal As Integer
```

```
strProfile = "c:\\DisplayVT.HEP"
```

```
strSessionName = "VT"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Parser = OSession.Parser
```

```
iVal = Parser.BufRows
```

```
Parser.BufRows = 100
```

```
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Rows = 0;
            pParser->get_BufRows(&Rows);
            // pParser->put_BufRows( 10 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## CanChangeScreen

**Property, IHEParser 3270 5250 VT**

This property returns or sets a value that specifies whether the screen can change.

### Basic Syntax

```
Boolean = HEParse.CanChangeScreen
```

```
HEParse.CanChangeScreen = Boolean
```

### C++ Syntax

```
HRESULT IHEParser::get_CanChangeScreen([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEParser::put_CanChangeScreen([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, you can change the screen contents. If *pVal* equals VARIANT\_FALSE, you cannot change the screen contents.

*newVal*—The new value (VARIANT\_TRUE or VARIANT\_FALSE) you set to indicate if the screen contents can be changed.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim bVal As Boolean  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    bVal = Parser.CanChangeScreen  
    Parser.CanChangeScreen = False  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbChanged;
            pParser->get_CanChangeScreen(&vbChanged);
            pParser->put_CanChangeScreen(VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## CaptureMode

### Property, IHEParser VT

This property specifies how to capture selected text.

#### Basic Syntax

```
HOSTEX_CAPTURE_MODE = HEParse.CaptureMode  
HEParse.CaptureMode = HOSTEX_CAPTURE_MODE
```

#### C++ Syntax

```
HRESULT IHEParser::get_CaptureMode([out, retval] HOSTEX_CAPTURE_MODE  
*pVal);  
HRESULT IHEParser::put_CaptureMode([in] HOSTEX_CAPTURE_MODE newVal);
```

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim Capture As HOSTEX_CAPTURE_MODE  
  
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Capture = Parser. CaptureMode  
Parser. CaptureMode = HOSTEX_CAPTURE_MODE_TEXT  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_CAPTURE_MODE Val;
            pParser->get_CaptureMode(&Val);

            pParser->put_CaptureMode( HOSTEX_CAPTURE_MODE_TEXT );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## CellCopyMode

Property, IHEParser 3270 5250 VT

This property sets or returns a value that specifies whether HostExplorer parses data by words or by field attributes when copying data to the Clipboard in cell-delimited format. Parsing by field attributes lets you copy data to the Clipboard in CSV (Comma-Separated Value) and BIF (Binary Interchange Format) formats. CSV and BIF are common formats used by spreadsheet applications.

### Basic Syntax

```
HOSTEX_CELL_DELIMITED = HEParse.CellCopyMode  
HEParse.CellCopyMode = HOSTEX_CELL_DELIMITED
```

### C++ Syntax

```
HRESULT IHEParser::get_CellCopyMode([out, retval] HOSTEX_CELL_DELIMITED  
*pVal);  
HRESULT IHEParser::put_CellCopyMode([in] HOSTEX_CELL_DELIMITED newVal);
```

### Parameters

*pVal*—The returned value which indicates the cell copy mode currently in effect.  
*newVal*—The set value which specifies the new cell copy mode.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim CopyCell As HOSTEX_CELL_DELIMITED  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
CopyCell = Parser.CellCopyMode  
Parser.CellCopyMode = HOSTEX_CELL_DELIMITED_WORD  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_CAPTURE_MODE Val;
            pParser->get_CaptureMode(&Val);

            pParser->put_CaptureMode( HOSTEX_CAPTURE_MODE_TEXT );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ConnectBy Property, IHEParser 3270 5250 VT

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

**Basic Syntax**

```
HOSTEX_CONNECT_BY = HEParse.ConnectBy  
HEParse.ConnectBy = HOSTEX_CONNECT_BY
```

**C++ Syntax**

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

**Parameters**

*pVal*—The returned value, indicating the connection.

*newVal*—The value that you set, indicating the connection.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim ConnectBy As HOSTEX_CONNECT_BY  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
ConnectBy = Parser.ConnectBy  
Parser.ConnectBy = HOSTEX_CONNECT_BY_TELNET  
OSession.Disconnect
```

**C++ Example**

```
IHEParser*      pParser      = NULL;
IOhioManager*   pApplication = NULL;
IOhioSession*   pSession     = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid,  NULL,  CLSCTX_SERVER,  iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_CONNECT_BY ConnectBy;
            pParser->get_ConnectBy( &ConnectBy);

            // pParser->put_ConnectBy(HOSTEX_CONNECT_BY_TELNET);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ConnectErrorStatus

### Property, IHEParser 3270 5250 VT

This property returns the connection error status for TAPI (Telephony Application Programming Interface). TAPI is used for connections over a modem.

#### Basic Syntax

```
Integer = HEParse.ConnectErrorStatus
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value that indicates the connection error status.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim iVal As Integer

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    iVal = Parser.ConnectErrorStatus
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );
CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Val;
            pParser->get_ConnectErrorStatus(&Val);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ConnectRC

**Property, IHEParser 3270 5250 VT**

This property returns a code that indicates whether the terminal is connecting, already connected, or neither.

### Basic Syntax

```
Integer = HEParse.ConnectRC
```

### C++ Syntax

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

### Parameters

*pVal*—The returned value that indicates the connection status.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim iVal As Integer

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    iVal = Parser.ConnectRC
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Val;
            pParser->get_ConnectRC(&Val);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ConvertNulls

### Property, IHEParser 3270 5250

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

#### Basic Syntax

```
Boolean = HEParser.ConvertNulls
```

```
HEParser.ConvertNulls = Boolean
```

#### C++ Syntax

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

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

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the system converts nulls to blank characters in input fields. A returned value of VARIANT\_FALSE indicates that the system does not convert nulls to blank characters in input fields.

*newVal*—A value of VARIANT\_TRUE indicates that the system converts nulls to blank characters in input fields. A value of VARIANT\_FALSE indicates that the system does not convert nulls to blank characters in input fields.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSesession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250
```

```
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
OSesession.Connect
```

```
Set Parser = OSesession.Parser  
Parser.ConvertNulls = True  
OSesession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID id = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->get_ConvertNulls(&vbVal);

            pParser->put_ConvertNulls(VARIANT_FALSE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## CutMode

### Property, IHEParser 3270 5250

This property removes selected text from unprotected areas of the screen. By default, this property is set to HOSTEX\_CUT\_MODE\_DELETE\_TEXT.

#### Basic Syntax

```
HOSTEX_CUT_MODE = HEParse.CutMode  
HEParse.CutMode = HOSTEX_CUT_MODE
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim CutMode As HOSTEX_CUT_MODE  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    CutMode = Parser.CutMode  
    Parser.CutMode = HOSTEX_CUT_MODE_REPLACE_WITH_SPACES  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_CUT_MODE CutMode;
            pParser->get_CutMode( &CutMode );

            pParser->put_CutMode( HOSTEX_CUT_MODE_REPLACE_WITH_SPACES );

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## EnableAPL

### Property, IHEParser 3270

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

#### Basic Syntax

```
Boolean = HEParse.EnableAPL  
HEParse.EnableAPL = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::get_EnableAPL([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEParser::put_EnableAPL([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the system enables APL mode. A returned value of VARIANT\_FALSE indicates that the system does not enable APL mode.

*newVal*—A set value of VARIANT\_TRUE indicates that the system enables APL mode. A set value of VARIANT\_FALSE indicates that the system does not enable APL mode.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSesession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse3270  
Dim bAPL As Boolean  
  
strProfile = "c:\\Display3270.HEP"  
strSessionName = "3270"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
OSesession.Connect  
  
Set Parser = OSesession.Parser  
bAPL = Parser.EnableAPL  
Parser.EnableAPL = True  
OSesession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->get_EnableAPL(&vbVal);

            pParser->put_EnableAPL(VARIANT_FALSE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## EnablePrinterTimeout

### Property, IHEParser VT

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

#### Basic Syntax

```
Boolean = HEParse.EnablePrinterTimeout  
HEParse.EnablePrinterTimeout = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::get_EnablePrinterTimeout ([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHEParser::put_EnablePrinterTimeout ([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer enables the delay of printer output. A returned value of VARIANT\_FALSE indicates that HostExplorer disables the delay of printer output.

*newVal*—A set value of VARIANT\_TRUE indicates that HostExplorer enables the delay of printer output. A set value of VARIANT\_FALSE indicates that HostExplorer disables the delay of printer output.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim bVal As Boolean  
  
    strProfile = "c:\\DisplayVT.HEP"  
    strSessionName = "VT"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    bVal = Parser. EnablePrinterTimeout  
    Parser. EnablePrinterTimeout = True  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->get_EnablePrinterTimeout (&vbVal);
            pParser->put_EnablePrinterTimeout (VARIANT_FALSE);
        }

        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GraphicsModel

### Property, IHEParser 3270

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

#### Basic Syntax

```
HOSTEX_GRAPHICS_MODEL = HEParse.GraphicsModel  
HEParse.GraphicsModel = HOSTEX_GRAPHICS_MODEL
```

#### C++ Syntax

```
HRESULT IHEParser::get_GraphicsModel([out, retval] HOSTEX_GRAPHICS_MODEL  
*pVal);  
HRESULT IHEParser::put_GraphicsModel([in] HOSTEX_GRAPHICS_MODEL newVal);
```

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse3270  
Dim Model As HOSTEX_GRAPHICS_MODEL  
  
strProfile = "c:\\Display3270.HEP"  
strSessionName = "3270"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Model = Parser.GraphicsModel  
' Parser.GraphicsModel = HOSTEX_GRAPHICS_MODEL_3179G  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_GRAPHICS_MODEL Val;
            pParser->get_GraphicsModel(&Val);

            // pParser->put_GraphicsModel(HOSTEX_GRAPHICS_MODEL_3179G );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HistoryLines

### Property, IHEParser VT

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

#### Basic Syntax

```
Integer = HEParse.HistoryLines  
HEParse.HistoryLines = Integer
```

#### C++ Syntax

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

#### Parameters

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

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim Lines As Integer  
  
    strProfile = "c:\\DisplayVT.HEP"  
    strSessionName = "VT"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Lines = Parser.HistoryLines  
    ' Parser.HistoryLines = 2000  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short HistoryLines;
            pParser->get_HistoryLines( &HistoryLines );

            // pParser->put_HistoryLines( 2000 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HLLAPIName

Property, IHEParser 3270 5250 VT

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

### Basic Syntax

```
String = HEParser.HLLAPIName  
HEParser.HLLAPIName = String
```

### C++ Syntax

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

### Parameters

*pVal*—The returned HLLAPI name.

*newVal*—The HLLAPI name that you set.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim str As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
str = Parser.HLLAPIName  
Parser.HLLAPIName = "A"  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrName = NULL;
            pParser->get_HLLAPIName(&bstrName);
            SysFreeString(bstrName);

            BSTR bstrText = SysAllocString( OLESTR("A") );
            pParser->put_HLLAPIName(bstrText);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HostResponseTime

### Property, IHEParser 3270 5250

This property returns the response time of the host.

#### Basic Syntax

```
Long = HEParse.HostResponseTime
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value that records the response time of the host.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim LVal As Long

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
LVal = Parser.HostResponseTime

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
STR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid,  NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            long HostResponseTime;
            pParser->get_HostResponseTime(&HostResponseTime);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HostWritableString

### Property, IHEParser VT

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

#### Basic Syntax

```
String = HEParse.HostWritableString
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParseVT
Dim strHost As String

strProfile = "c:\\DisplayVT.HEP"
strSessionName = "VT"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
strHost = Parser.HostWritableString

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = NULL;
            pParser->get_HostWritableString( &bstrText);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## KeyboardLocked

**Property, IHEParser 3270 5250 VT**

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

**Basic Syntax**

```
Integer = HEParse.KeyboardLocked  
HEParse.KeyboardLocked = Integer
```

**C++ Syntax**

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

**Parameters**

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

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim Locked As Integer  
  
    strProfile = "c:\\\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Locked = Parser.KeyboardLocked  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Status;
            pParser->get_KeyboardLocked(&Status);

            // pParser->put_KeyboardLocked( Status );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ModelColumns

### Property, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
Integer = HEParse.ModelColumns  
HEParse.ModelColumns = Integer
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim Columns As Integer  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Columns = Parser.ModelColumns  
    ' Parser.ModelColumns = 66  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Columns;
            parser->get_ModelColumns (&Columns);

            // parser->put_ModelColumns (132);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ModelRows

### Property, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
Integer = HEParse.ModelRows  
HEParse.ModelRows = Integer
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim ModelRows As Integer  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    ModelRows = Parser.ModelRows  
    ' Parser.ModelRows = 20  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short Rows;
            pParser-> get_ModelRows(&Rows);
            // pParser-> put_ModelRows(20);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## NRCID

### Property, IHEParser VT

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

#### Basic Syntax

```
Integer = HEParser.NRCID  
HEParser.NRCID = Integer
```

#### C++ Syntax

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

#### Parameters

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

*newVal*—The set string for the NRC set.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim iNRCID As Integer  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
iNRCID = Parser.NRCID  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short NRCID;
            pParser->get_NRCID(&NRCID);

            pParser->put_NRCID(NRCID);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## NumericCharacters

### Property, IHEParser 3270

This property lets you specify which characters are valid numeric characters.

#### Basic Syntax

```
HEParser.NumericCharacters = String
```

#### C++ Syntax

```
HRESULT IHEParser::put_NumericCharacters([in] BSTR newVal);
```

#### Parameters

*newVal*—The string containing the set of valid numeric characters.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParser3270

    strProfile = "c:\\Display3270.HEP"
    strSessionName = "3270"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.NumericCharacters = "1234567890"
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrText = SysAllocString( OLESTR("1234567890") );
            pParser->put_NumericCharacters(bstrText);
            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## NVTMode

### Property, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
Boolean = HEParse.NVTMode
```

#### C++ Syntax

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

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the system is in NVT mode. A returned value of VARIANT\_FALSE indicates that the system is not in NVT mode.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim bNVT As Boolean

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    bNVT = Parser.NVTMode

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->get_NVTMode( &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## OIAString

### Property, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
String = HEParse.OIAString
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim sOIA As String

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
sOIA = Parser.OIAString
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrOIA;
            pParser->get_OIAString( &bstrOIA);
            SysFreeString(bstrOIA);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## OIAStringW

Property, IHEParser 3270 5250 VT

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

### Basic Syntax

```
String = HEParse.OIAStringW
```

### C++ Syntax

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

### Parameters

*pVal*—The returned wide character string of the host status line in the OIA.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim strW As String

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
strW = Parser.OIAStringW

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrOIA;
            pParser->get_OIAStringW( &bstrOIA);

            SysFreeString(bstrOIA);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PasteMode

### Property, IHEParser 3270 5250

This property lets you determine how HostExplorer pastes the contents of the clipboard to the current cursor location. By default, this property is set to HOSTEX\_PASTE\_MODE\_PASTE\_BLOCK.

#### Basic Syntax

```
HOSTEX_PASTE_MODE = HEParse.PasteMode
```

```
HEParse.PasteMode = HOSTEX_PASTE_MODE
```

#### C++ Syntax

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

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

#### Parameters

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

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim iMode As HOSTEX_PASTE_MODE

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
iMode = Parser.PasteMode
Parser.PasteMode = HOSTEX_PASTE_MODE_PASTE_BLOCK
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_PASTE_MODE PasteMode;
            pParser->get_PasteMode(&PasteMode);
            // pParser->put_PasteMode ( HOSTEX_PASTE_MODE_PASTE_BLOCK );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrintByPassWindows

### Property, IHEParser VT

This property lets you specify whether HostExplorer bypasses the Windows print driver.

#### Basic Syntax

```
IHEParser.PrintByPassWindows = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::put_PrintByPassWindows([in] VARIANT_BOOL newVal);
```

#### Parameters

*newVal*—The set value. If *newVal* equals VARIANT\_TRUE, HostExplorer bypasses the Windows print driver. If *newVal* equals VARIANT\_FALSE, HostExplorer does not bypass the Windows print driver.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParserVT

    strProfile = "c:\\DisplayVT.HEP"
    strSessionName = "VT"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.PrintByPassWindows = True
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->put_PrintByPassWindows(VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrinterDeInitString

### Property, IHEParser 3270

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

#### Basic Syntax

```
String = HEParse.PrinterDeInitString
```

```
HEParse.PrinterDeInitString = String
```

#### C++ Syntax

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

```
HRESULT IHEParser::put_PrinterDeInitString([in] BSTR newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Parser As HEParse3270
```

```
strProfile = "c:\\Display3270.HEP"
```

```
strSessionName = "3270"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
DeInit = "String formatted with escape sequence for printer"
```

```
Set Parser = OSession.Parser
```

```
Parser.PrinterDeInitString = "^[E"
```

```
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstr = NULL;
            pParser->get_PrinterDeInitString( &bstr );
            SysFreeString( bstr );

            BSTR bstrDeInit = SysAllocString( OLESTR( "[E]" ) );
            pParser->put_PrinterDeInitString( bstrDeInit );
            SysFreeString( bstrDeInit );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrintDisableTranslation

### Property, IHEParser VT

This property lets you specify whether HostExplorer disables Host-to-PC translation.

#### Basic Syntax

```
IHEParser.PrintDisableTranslation = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::put_PrintDisableTranslation([in] VARIANT_BOOL newVal);
```

#### Parameters

*newVal*—The set value. If *newVal* equals VARIANT\_TRUE, HostExplorer disables Host-to-PC translation. If *newVal* equals VARIANT\_FALSE, HostExplorer enables Host-to-PC translation.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParseVT

    strProfile = "c:\\DisplayVT.HEP"
    strSessionName = "VT"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser

    Parser.PrintDisableTranslation = True
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->put_PrintDisableTranslation( VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrinterInitString

### Property, IHEParser 3270

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

#### Basic Syntax

```
String = HEParse.PrinterInitString
```

```
HEParse.PrinterInitString = String
```

#### C++ Syntax

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

```
HRESULT IHEParser::put_PrinterInitString([in] BSTR newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Parser As HEParse3270
```

```
Dim InitStr As String
```

```
strProfile = "c:\\Display3270.HEP"
```

```
strSessionName = "3270"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Parser = OSession.Parser
```

```
Set Parser = OSession.Parser
```

```
InitStr = Parser.PrinterInitString
```

```
Parser.PrinterInitString = "^[&L0H" ' To eject page
```

```
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrInitString = NULL;
            pParser->get_PrinterInitString( &bstrInitString );
            SysFreeString(bstrInitString);

            // BSTR bstrInit = SysAllocString( OLESTR( "^[%lOH" ) );
            // pParser->put_PrinterInitString(bstrInit);
            // SysFreeString(bstrInit);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrinterTimeout

### Property, IHEParser VT

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

#### Basic Syntax

```
Long = HEParse.PrinterTimeout  
HEParse.PrinterTimeout = Long
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim Timeout As Long  
  
    strProfile = "c:\\DisplayVT.HEP"  
    strSessionName = "VT"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Parser = OSession.Parser  
    Timeout = Parser.PrinterTimeOut  
    'Parser.PrinterTimeOut = 4000  
  
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            long Timeout = 0;
            pParser ->get_PrinterTimeout( &Timeout );

            // pParser ->put_PrinterTimeout( 1000 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrinterTimeoutValue

### Property, IHEParser VT

This property lets you specify the timeout value for the DO\_PRNCTRL command.

#### Basic Syntax

```
HEParser.PrinterTimeoutValue = Integer
```

#### C++ Syntax

```
HRESULT IHEParser::put_PrinterTimeoutValue([in] short newVal);
```

#### Parameters

*NewVal*—The timeout value for the DO\_PRNCTRL command.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParserVT

    strProfile = "c:\\DisplayVT.HEP"
    strSessionName = "VT"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.PrinterTimeoutValue = 3999

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->put_PrinterTimeoutValue( 3999 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PrintLFtoCRLF

### Property, IHEParser VT

This property lets you specify whether HostExplorer expands linefeed (LF) characters to carriage return + linefeed (CR + LF) characters when printing.

#### Basic Syntax

```
IHEParser.PrintLFtoCRLF = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::put_PrintLFtoCRLF([in] VARIANT_BOOL newVal);
```

#### Parameters

*newVal*—The set value. If *newVal* equals VARIANT\_TRUE, HostExplorer prints each linefeed character as a carriage return followed by a linefeed character. If *newVal* equals VARIANT\_FALSE, HostExplorer does not expand linefeed characters.

#### Basic Example

```
Dim OManager As HEOHIOlib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParserVT

    strProfile = "c:\\DisplayVT.HEP"
    strSessionName = "VT"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Parser = OSession.Parser
    Parser.PrintLFtoCRLF = True

    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->put_PrintLFtoCRLF( VARIANT_TRUE );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ProgramSymbols

### Property, IHEParser 3270

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

#### Basic Syntax

```
Boolean = HEParse.ProgramSymbols
```

```
HEParser.ProgramSymbols = Boolean
```

#### C++ Syntax

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

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

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer supports Program symbols. A returned value of VARIANT\_FALSE indicates that HostExplorer does not support Program symbols.

*newVal*—A set value of VARIANT\_TRUE indicates that HostExplorer supports Program symbols. A set value of VARIANT\_FALSE indicates that HostExplorer does not support Program symbols.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Parser As HEParse3270
```

```
Dim bProgram As Boolean
```

```
    strProfile = "c:\\Display3270.HEP"
```

```
    strSessionName = "3270"
```

```
    Set OManager = CreateObject("HEOhio.OhioManager")
```

```
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
    OSession.Connect
```

```
        Set Parser = OSession.Parser
```

```
        bProgram = Parser.ProgramSymbols
```

```
        Parser.ProgramSymbols = False
```

```
    OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbProgram;
            pParser-> get_ProgramSymbols (vbProgram);
            //  pParser-> put_ProgramSymbols (VARIANT_FALSE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## OnCopyReplaceFieldAttributeWith

### Property, IHEParser 3270 5250

This property replaces field attributes with a tab, comma, or paragraph mark. By default, this property is set to  
HOSTEX\_FIELD\_ATTR\_REPLACEMENT\_COMM.

<b>Basic Syntax</b>	HOSTEX_FIELD_ATTR_REPLACEMENT = HEParse.OnCopyReplaceFieldAttributeWith HEParse.OnCopyReplaceFieldAttributeWith = HOSTEX_FIELD_ATTR_REPLACEMENT
<b>C++ Syntax</b>	HRESULT IHEParser::get_OnCopyReplaceFieldAttributeWith([out, retval] HOSTEX_FIELD_ATTR_REPLACEMENT *pVal);  HRESULT IHEParser::put_OnCopyReplaceFieldAttributeWith([in] HOSTEX_FIELD_ATTR_REPLACEMENT newVal);
<b>Parameters</b>	<i>pVal</i> —The returned value, indicating how the field attribute is replaced. <i>newVal</i> —The set value, indicating how the field attribute is replaced.
<b>Basic Example</b>	Dim OManager As HEOHIOLib.OhioManager Dim OSession As OhioSession  Dim strProfile, strSessionName As String Dim Parser As HEParse5250 Dim FieldAttr As HOSTEX_FIELD_ATTR_REPLACEMENT  strProfile = "c:\\Display5250.HEP" strSessionName = "as400"  Set OManager = CreateObject("HEOhio.OhioManager") Set OSession = OManager.OpenSession(strProfile, strSessionName) OSession.Connect  Set Parser = OSession.Parser FieldAttr = Parser.OnCopyReplaceFieldAttributeWith Parser.OnCopyReplaceFieldAttributeWith = HOSTEX_FIELD_ATTR_REPLACEMENT_COMM  OSession.Disconnect

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get(Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_FIELD_ATTR_REPLACEMENT Val;
            pParser->getOnCopyReplaceFieldAttributeWith( &Val);

            pParser-
>putOnCopyReplaceFieldAttributeWith( HOSTEX_FIELD_ATTR_REPLACEMENT_COMMA
);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## OnPasteFieldModeTabCharacter

### Property, IHEParser 3270 5250

This property lets you set and retrieve the character that HostExplorer uses as a tab character when you paste data from the Clipboard. This property applies only if the Paste Mode is `HOSTEX_PASTE_MODE_PASTE_FIELD`.

#### Basic Syntax

```
HOSTEX_NEXT_FIELD_KEY = HEParse.OnPasteFieldModeTabCharacter  
HEParse.OnPasteFieldModeTabCharacter = HOSTEX_NEXT_FIELD_KEY
```

#### C++ Syntax

```
HRESULT IHEParser::get_OnPasteFieldModeTabCharacter ([out, retval]  
HOSTEX_NEXT_FIELD_KEY *pVal);  
HRESULT IHEParser::put_OnPasteFieldModeTabCharacter ([in]  
HOSTEX_NEXT_FIELD_KEY newVal);
```

#### Parameters

*pVal*—The returned value indicating the character HostExplorer uses as a tab character between pasted fields.

*newVal*—The character you want HostExplorer to use as a tab between pasted fields.

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim lPasteFiled As HOSTEX_NEXT_FIELD_KEY  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
  
IPasteField = Parser.OnPasteFieldModeTabCharacter  
Parser.OnPasteFieldModeTabCharacter = HOSTEX_NEXT_FIELD_KEY_COMMA  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_NEXT_FIELD_KEY Val;
            pParser->get_OnPasteFieldModeTabCharacter ( &Val);

            // pParser-
>put_OnPasteFieldModeTabCharacter (HOSTEX_NEXT_FIELD_KEY_COMMA);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SaveAppend

### Property, IHEParser VT

This property lets you specify whether HostExplorer appends the current screen capture to previously saved screen captures.

#### Basic Syntax

```
IHEParser.SaveAppend = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::put_SaveAppend([in] VARIANT_BOOL newVal);
```

#### Parameters

*newVal*—The value you set. If *newVal* equals VARIANT\_TRUE, HostExplorer appends the current screen capture to previous ones. If *newVal* equals VARIANT\_FALSE, HostExplorer overwrites the previous screen captures with the current one.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParserVT

strProfile = "c:\\DisplayVT.HEP"
strSessionName = "VT"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser

Parser.SaveAppend = False
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            pParser->put_SaveAppend(VARIANT_FALSE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SaveFileName

### Property, IHEParser VT

This property sets the name of the file that HostExplorer uses to store screen captures.

#### Basic Syntax

```
HEParser.SaveFileName = String
```

#### C++ Syntax

```
HRESULT IHEParser::put_SaveFileName([in] BSTR newVal);
```

#### Parameters

*newVal*—The name of the file.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParserVT

strProfile = "c:\\DisplayVT.HEP"
strSessionName = "VT"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
Parser.SaveFileName = "c:\\filename.txt"

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrFile = SysAllocString( OLESTR("c:\\\\Filename.txt") );
            pParser->put_SaveFileName(bstrFile);
            SysFreeString(bstrFile);

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ScreenChanged

### Property, IHEParser 3270 5250 VT

This property returns or sets a value that specifies whether the contents of the screen have changed.

#### Basic Syntax

```
Boolean = HEParse.ScreenChanged
```

```
HEParser.ScreenChanged = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEParser::put_ScreenChanged([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, the screen contents have changed. If *pVal* equals VARIANT\_FALSE, the screen contents have not changed.

*newVal*—The new value (VARIANT\_TRUE or VARIANT\_FALSE) you set to indicate if the screen contents have changed.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250
Dim bChanged As Boolean

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser

bChanged = Parser.ScreenChanged
Parser.ScreenChanged = False

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbVal;
            pParser->get_ScreenChanged( &vbVal);

            pParser->put_ScreenChanged( VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ScreenCol

### Property, IHParser

This property returns or sets the column position of the cursor.

#### Basic Syntax

```
Integer = HEParser.ScreenCol  
HEParser.ScreenCol = Integer
```

#### C++ Syntax

```
HRESULT IHParser::get_ScreenCol([out, retval] short *pVal);  
HRESULT IHParser::put_ScreenCol([in] short newVal);
```

#### Parameters

*pVal*—The returned value indicating the current column position of the cursor.  
*newVal*—The new column position you want to set for the cursor. Column positions begin at 1.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
  
Dim iCol As Integer  
  
StrProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
iCol = Parser.ScreenCol  
' Parser.ScreenCol = 22  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short iCol;
            pParser->get_ScreenCol( &iCol );

            // pParser->put_ScreenCol( 22 );

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ScreenRow

### Property, IHEParser 3270 5250 VT

This property returns or sets the row position of the cursor.

#### Basic Syntax

```
Integer = HEParse.ScreenRow  
HEParse.ScreenRow = Integer
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned value indicating the current row position of the cursor.

*newVal*—The new row position you want to set for the cursor.

Row positions begin at 1.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim Row As Integer  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Row = Parser.ScreenRow  
Parser.ScreenRow = 22  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();
        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short iRow;
            pParser->get_ScreenRow( &iRow );

            // pParser->put_ScreenRow( 22 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SelectionMode

### Property, IHEParser 3270 5250 VT

This property returns or sets the type of selection (that is, stream or rectangular). By default, this property is set to HOSTEX\_SELECTION\_MODE\_BLOCK.

#### Basic Syntax

```
HOSTEX_SELECTION_MODE = HEParse.SelectionMode  
HEParse.SelectionMode = HOSTEX_SELECTION_MODE
```

#### C++ Syntax

```
HRESULT IHEParser::get_SelectionMode([out, retval] HOSTEX_SELECTION_MODE  
*pVal);  
HRESULT IHEParser::put_SelectionMode([in] HOSTEX_SELECTION_MODE newVal);
```

#### Parameters

*pVal*—The returned value, indicating the selection type.

*newVal*—The set value, indicating the selection type.

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim Selection As HOSTEX_SELECTION_MODE  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Selection = Parser.SelectionMode  
Parser.SelectionMode = HOSTEX_SELECTION_MODE_STREAM  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_SELECTION_MODE Mode;
            pParser->get_SelectionMode (&Mode) ;

            pParser->put_SelectionMode (HOSTEX_SELECTION_MODE_STREAM) ;
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SessionName

### Property, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
String = HEParse.SessionName  
HEParse.SessionName = String
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HECHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim Name As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Name = Parser.SessionName  
Parser.SessionName = strSessionName  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrName = NULL;
            pParser->get_SessionName ( &bstrName );
            SysFreeString( bstrName );

            pParser->put_SessionName (bstrSessionName);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SoftCharacterSetID

### Property, IHEParser VT

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

#### Basic Syntax

```
Integer = HEParse.SoftCharacterSetID
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParseVT
Dim iSoft As Integer

strProfile = "c:\\DisplayVT.HEP"
strSessionName = "VT"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
iSoft = Parser.SoftCharacterSetID

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser= NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short CharacterSet ;
            pParser->get_SoftCharacterSetID( &CharacterSet);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## StatusLineMode

### Property, IHEParser 3270 5250 VT

This property returns or sets the type of status linemode (that is, terminal or window). By default, this property is set to HOSTEX\_STATUS\_LINE\_MODE\_WINDOWSTATUSBAR.

#### Basic Syntax

```
HOSTEX_STATUS_LINE_MODE = HEParse.StatusLineMode  
HEParse.StatusLineMode = HOSTEX_STATUS_LINE_MODE
```

#### C++ Syntax

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

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
  
Dim Linemode As HOSTEX_STATUS_LINE_MODE  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Linemode = Parser.StatusLineMode  
Parser.StatusLineMode = HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_STATUS_LINE_MODE LineMode;
            pParser->get_StatusLineMode( &LineMode);

            pParser-
>put_StatusLineMode (HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE)
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TerminalID

### Property, IHEParser VT

This property returns or sets the terminal ID response that HostExplorer sends to the host. By default, this property is set to HOSTEX\_TERMINAL\_ID\_VT220.

#### Basic Syntax

```
HOSTEX_TERMINAL_ID = HEParse.TerminalID
```

```
HEParse.TerminalID = HOSTEX_TERMINAL_ID
```

#### C++ Syntax

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

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

#### Parameters

*pVal*—The returned terminal ID response.

*newVal*—The set terminal ID response.

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT
```

```
Dim Terminal As HOSTEX_TERMINAL_ID
```

```
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
Set Parser = OSession.Parser  
Terminal = Parser.TerminalID  
' Parser.TerminalID = HOSTEX_TERMINAL_ID_VT102
```

```
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_TERMINAL_ID Terminal;
            pParser->get_TerminalID( &Terminal);

            // pParser->put_TerminalID( HOSTEX_TERMINAL_ID_VT102);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TerminalModel

### Property, IHEParser 3270 5250 VT

This property returns or sets the terminal model supported by HostExplorer. For TN3270 and TN5250 terminals, the default for this property is set to HOSTEX\_TERM\_MODEL\_2. For TNVT terminals, the default is set to HOSTEX\_TERM\_MODEL\_VT220.

#### Basic Syntax

```
HOSTEX_TERM_MODEL = HEParse.TerminalModel
```

```
HEParse.TerminalModel = HOSTEX_TERM_MODEL
```

#### C++ Syntax

```
HRESULT IHEParser::get_TerminalModel([out, retval] HOSTEX_TERM_MODEL  
*pVal);
```

```
HRESULT IHEParser::put_TerminalModel([in] HOSTEX_TERM_MODEL newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse5250

Dim TermModel As HOSTEX_TERM_MODEL

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
TermModel = HEParse.TerminalModel
' Parser.TerminalModel = HOSTEX_TERM_MODEL_5

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_TERM_MODEL Model;
            pParser->get_TerminalModel( &Model);

            // pParser->get_TerminalModel( HOSTEX_TERM_MODEL_5);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Text

### Property, IHEParser 3270 5250 VT

This property sets or returns the entire presentation space as a string value.

#### Basic Syntax

```
String = HEParse.Text  
HEParse.Text = String
```

#### C++ Syntax

```
HRESULT IHEParser::get_Text([out, retval] BSTR *pVal);  
HRESULT IHEParser::put_Text([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value containing the presentation space as a string.  
*newVal*—The string that you want to set as the presentation space.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim strText As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
strText = Parser.Text  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            BSTR bstrFull = NULL;
            pParser->get_Text( &bstrFull );
            SysFreeString( bstrFull );

            // BSTR bstrNewPS = SysAllocString( OLESTR("New Text to write on screen") );
            // pParser->put_Text( bstrNewPS1 );
            // SysFreeString( bstrNewPS1 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Transport

### Property, IHEParser 3270 5250 VT

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

#### Basic Syntax

```
Object = HEParse.Transport  
HEParser.Transport = Object
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned callback pointer to the Transport interface.

*newVal*—The set callback pointer to the Transport interface.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse5250  
Dim Transport As HETransportTN  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Set Transport = Parser.Transport  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            IHETransport* pTransport2 = NULL;
            pParser->get_Transport( (IUnknown**) &pTransport2);

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TypeAheadTimeout

Property, IHEParser 3270 5250 VT

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

### Basic Syntax

```
Long = HEParser.TypeAheadTimeout
```

```
HEParser.TypeAheadTimeout = Long
```

### C++ Syntax

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

### Parameters

*pVal*—The returned value, indicating the time delay.

*newVal*—The set value, indicating the time delay.

### Basic Example

```
Dim OManager As HEOHOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser5250  
Dim TypeTimout As Long  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
TypeTimeout = Parser.TypeAheadTimeout  
Parser.TypeAheadTimeout = 3000  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            long    TypeTimeout;
            pParser->get_TypeAheadTimeout( &TypeTimeout);

            pParser->put_TypeAheadTimeout(2000);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ValidateNumericFieldData

### Property, IHEParser 3270

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

#### Basic Syntax

```
Boolean = HEParse.ValidateNumericFieldData  
HEParser.ValidateNumericFieldData = Boolean
```

#### C++ Syntax

```
HRESULT IHEParser::get_ValidateNumericFieldData([out, retval]  
VARIANT_BOOL *pVal);  
HRESULT IHEParser::put_ValidateNumericFieldData([in] VARIANT_BOOL  
newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that only numeric characters are accepted in the input field.

A returned value of VARIANT\_FALSE indicates that any character can be accepted in the input field.

*newVal*—A set value of VARIANT\_TRUE indicates that only numeric characters are accepted in the input field.

A set value of VARIANT\_FALSE indicates that any character can be accepted in the input field.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParse3270  
Dim bVal As Boolean  
  
strProfile = "c:\\Display3270.HEP"  
strSessionName = "3270"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = Parser.ValidateNumericFieldData  
Parser.ValidateNumericFieldData = True  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbValidate;
            pParser->get_ValidateNumericFieldData( &vbValidate);

            pParser->put_ValidateNumericFieldData( VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## MoveCursorOnMouseClicked

### Property, IHEParser VT

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

#### Basic Syntax

```
Boolean = HEParse.MoveCursorOnMouseClicked
```

```
HEParser.MoveCursorOnMouseClicked = Boolean
```

#### C++ Syntax

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

```
HRESULT IHEParser::put_MoveCursorOnMouseClicked([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically moves the cursor on a mouse-click. A returned value of VARIANT\_FALSE indicates that HostExplorer does not automatically move the cursor on a mouse-click.

*newVal*—A set value of VARIANT\_TRUE indicates that HostExplorer automatically moves the cursor on a mouse-click.

A set value of VARIANT\_FALSE indicates that HostExplorer does not automatically move the cursor on a mouse-click.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim bVal As Boolean  
  
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bVal = Parser.MoveCursorOnMouseClicked  
Parser.MoveCursorOnMouseClicked = True  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbMove;
            pParser->get_MoveCursorOnMouseClicked( &vbMove);

            pParser->put_MoveCursorOnMouseClicked( VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## UPSS

### Property, IHEParser VT

This property returns or sets the User Preferred Supplemental Character Set (UPSS). By default, this property is set to VTCS\_ISO\_8859\_1.

#### Basic Syntax

```
Integer = HEParse.**UPSS**
```

```
HEParse.**UPSS** = Integer
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned UPSS index.

*newVal*—The set UPSS index.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim UPSS as Integer  
  
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
UPSS = Parser.UPSS  
Parser.UPSS = 123  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            short UPSSValue;
            pParser->get_UPSS( &UPSSValue );

            ' pParser->put_UPSS( 123 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## AutoWrap

### Property, IHEParse VT

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

#### Basic Syntax

```
Boolean = HEParse.AutoWrap  
HEParse.AutoWrap = Boolean
```

#### C++ Syntax

```
HRESULT IHEParse::get_AutoWrap([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHEParse::put_AutoWrap([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that HostExplorer automatically wraps lines that extend past the last column on the screen. A returned value of VARIANT\_FALSE indicates that HostExplorer discards data that extends past the end of the screen.

*newVal*—A value of VARIANT\_TRUE indicates that HostExplorer automatically wraps lines that extend past the last column on the screen. A value of VARIANT\_FALSE indicates that HostExplorer discards data that extends past the end of the screen.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParseVT  
Dim bAutoWrap As Boolean  
  
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
bAutoWrap = Parser.AutoWrap  
Parser.AutoWrap = True  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            VARIANT_BOOL vbAutoWrap;
            pParser->get_AutoWrap( &vbAutoWrap );

            pParser->put_AutoWrap( VARIANT_TRUE );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TransferErrorCode

### Property, IHEParser 3270

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

#### Basic Syntax

```
Long = HEParse.TransferErrorCode
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned error code.

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Parser As HEParse3270
Dim ErrorCode As Long

strProfile = "c:\\Display3270.HEP"
strSessionName = "3270"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Parser = OSession.Parser
ErrorCode = Parser.TransferErrorCode

OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            // do some file transfer.
            long Error;
            pParser->get_TransferErrorCode( &Error);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TransferMode

### Property, IHEParser 3270

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

#### Basic Syntax

```
HOSTEX_TRANSFER = HEParser.TransferMode  
HEParser.TransferMode = HOSTEX_TRANSFER
```

#### C++ Syntax

```
HRESULT IHEParser::get_TransferMode([out, retval] HOSTEX_TRANSFER *pVal);  
HRESULT IHEParser::put_TransferMode([in] HOSTEX_TRANSFER newVal);
```

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Parser As HEParser3270  
Dim Transfer As HOSTEX_TRANSFER  
  
strProfile = "c:\\Display3270.HEP"  
strSessionName = "3270"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Parser = OSession.Parser  
Transfer = Parser.TransferMode  
Parser.TransferMode = HOSTEX_TRANSFER_DOWNLOAD  
  
OSession.Disconnect
```

**C++ Example**

```
IHEParser* pParser = NULL;
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Parser((IDispatch **)&pParser);
        if(pParser)
        {
            HOSTEX_TRANSFER TransferMode;
            pParser->get_TransferMode( &TransferMode );

            pParser->put_TransferMode( HOSTEX_TRANSFER_DOWNLOAD );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Data Types of the Parser Objects

The Parser objects contain the following data types:

- HEPARSER\_FEATURE Data Type
- 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 HETP5250 object
- both the HETP3270 and HETP5250 objects
- only the HETPVT object
- both the HETP3270 and HETPVT objects

There are also methods and properties common to the HETP3270, HETP250, and HETPVT objects.

## ***Properties of the HETP3270 Object***

The following properties are specific to the HETP3270 object:

- EnableEMode
- TNESession

## ***Properties of the HETP5250 Object***

The following properties are specific to the HETP5250 object:

- |                       |                    |
|-----------------------|--------------------|
| • LUNNameRequested    | • Keyboard         |
| • MessageQueueLibrary | • MessageQueueName |
| • Password            | • Username         |

## ***Properties of the HETP3270/5250 Objects***

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

- DeviceType
- LUNNameRequested

## ***Properties and Data Types of the HETPVT Object***

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

### **Properties**

- LineMode
- TerminalOnline

### **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 HETPVVT 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 HETPVVT 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

## Methods of the Transport Objects

The following properties are methods of the Transport objects:

- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>• GetFeature</li><li>• NegotiateNAWS</li><li>• SendFunctionKey</li><li>• SetFeature</li><li>• SetKerberosInfo</li></ul> | <ul style="list-style-type: none"><li>• GetStatusString</li><li>• SendData</li><li>• SendKeepAlive</li><li>• SetHostPrintTransformInfo</li><li>• ToggleBlockReceive</li></ul> |
|---|---|

### **GetFeature**

**Method, IHETransport 3270 5250 VT**

This method returns the current value of a specified Transport feature.

#### **Basic Syntax**

```
HETransport.GetFeature(lType As HETRANSPORT_FEATURE) As Boolean
```

#### **C++ Syntax**

```
HRESULT IHETransport::GetFeature(  
    [in] HETRANSPORT_FEATURE lType,  
    [out, retval] VARIANT_BOOL *pVal);
```

#### **Parameters**

*lType*—The name of the feature that you want to check.

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, the feature is enabled. If *pVal* equals VARIANT\_FALSE, the feature is disabled.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

Dim bVal As Boolean

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport
bVal = Transport.GetFeature(HOSTEX_EAB)

OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbVal;
            pTransport->GetFeature(HOSTEX_EAB, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## GetStatusString

Method, IHETransport 3270 5250 VT

This method specifies the status of the connection to the host.

### Basic Syntax

```
HETransport.GetStatusString(1param1 As Long, 1param2 As Long) As String
```

### C++ Syntax

```
HRESULT IHETransport::GetStatusString(  
    [in] long 1param1,  
    [in] long 1Param2,  
    [out, retval] BSTR *pStatus);
```

### Parameters

*1param1*—The user must set this parameter to 0.

*1param2*—The user must set this parameter to 0.

*pStatus*—The returned string, indicating the current connection status, or the last error that occurred.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim StrStatus As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
Set Transport = OSession.Transport  
StrStatus = Transport.GetStatusString(0, 0)  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrText = NULL;
            pTransport->GetStatusString(0, 0, &bstrText);

            SysFreeString(bstrText);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## NegotiateNAWS

### Method, IHETransport VT

This method renegotiates the window size using specified row and column values (if they are different from the current values).

#### Basic Syntax

```
HETransport.NegotiateNAWS(iRows As Integer, iCols As Integer)
```

#### C++ Syntax

```
HRESULT IHETransport::NegotiateNAWS(  
[in] int iRows,  
[in] int iCols);
```

#### Parameters

*iRows*—The number of rows you want in the window.

*iCols*—The number of columns you want in the window.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
  
Transport. NegotiateNAWS 20, 30  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            pTransport->NegotiateNAWS( 20, 30 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SendData

### Method, IHETransport 3270 5250 VT

This method sends the formatted data to the host.

#### Basic Syntax

```
HETransport.SendData (pBuffer As String)
```

#### C++ Syntax

```
HRESULT IHETransport::SendData( [in] BSTR pBuffer);
```

#### Parameters

*pBuffer*—The modified fields being sent.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport
Dim StrData As String

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport
StrData = "Data formatted to be sent to host"
Transport.SendData StrData
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrData = SysAllocString( OLESTR("Data formatted to be sent to host") );
            pTransport->SendData(bstrData);

            SysFreeString (bstrData);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString (bstrProfile);
SysFreeString (bstrSessionName );
```

## SendFunctionKey

### Method, IHETransport 3270

This method sends a particular function to the host.

#### Basic Syntax

```
HETransport.SendFunctionKey(IKey As HOSTEX_FUNCTION_KEY)
```

#### C++ Syntax

```
HRESULT IHETransport::SendFunctionKey([in] HOSTEX_FUNCTION_KEY IKey);
```

#### Parameters

*IKey*—The function key that is sent to the host.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

strProfile = "c:\\Display3270.HEP"
strSessionName = "3270"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport
Transport.SendFunctionKey HOSTEX_FUNCTION_KEY_SEND_ATTENTION

OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            pTransport-
>SendFunctionKey( HOSTEX_FUNCTION_KEY_SEND_ATTENTION);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName);
```

## SendKeepAlive

### Method, IHETransport 3270 5250 VT

A repeated message that is sent to the host, indicating that you are still connected.

**Basic Syntax**      `HETransport.SendKeepAlive`

**C++ Syntax**      `HRESULT IHETransport::SendKeepAlive();`

**Parameters**      This method has no parameters.

**Basic Example**      `Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession`

```
Dim strProfile, strSessionName As String  
Dim Transport As HETransport
```

```
strProfile = "c:\\\\Display5250.HEP"  
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
Set Transport = OSession.Transport  
Transport.SendKeepAlive  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            pTransport->SendKeepAlive();
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetFeature

Method, IHETransport 3270 5250 VT

This method lets you enable or disable various features of the Transport objects.

### Basic Syntax

```
HETransport.SetFeature lType As HETRANSPORT_FEATURE, newVal As Boolean
```

### C++ Syntax

```
HRESULT IHETransport::SetFeature(  
    [in] HETRANSPORT_FEATURE lType,  
    [in] VARIANT_BOOL newVal);
```

### Parameters

*lType*—The name of the feature that you want to enable or disable.

*newVal*—The value you want to apply to the feature. Set *newVal* to VARIANT\_TRUE to enable the feature. Set *newVal* to VARIANT\_FALSE to disable the feature.

### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
Transport.SetFeature HOSTEX_KEEP_ALIVE, True  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            pTransport->SetFeature( HOSTEX_KEEP_ALIVE, TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SetHostPrintTransformInfo

### Method, IHETransport 5250

This method lets you specify the information required to negotiate the Host Print transform with an AS/400 Telnet server.

#### Basic Syntax

```
HETransport.SetHostPrintTransformInfo (bHostPrintTransform As Boolean,  
PrinterModel As String, Drawer1 As Byte, Drawer2 As Byte,  
EnvelopeHopper As Byte, bASCII899 As Boolean, CustomizingObject As  
String, CustomizingLibrary As String)
```

#### C++ Syntax

```
HRESULT IHETransport::SetHostPrintTransformInfo (  
    [in] VARIANT_BOOL bHostPrintTransform,  
    [in] BSTR PrinterModel,  
    [in] unsigned char Drawer1,  
    [in] unsigned char Drawer2,  
    [in] unsigned char EnvelopeHopper,  
    [in] VARIANT_BOOL bASCII899,  
    [in] BSTR CustomizingObject,  
    [in] BSTR CustomizingLibrary);
```

#### Parameters

*bHostPrintTransform*—The value that specifies whether the AS/400 Telnet server will format the data in ASCII for the specified printer. If *bHostPrintTransform* equals VARIANT\_TRUE, the AS/400 Telnet server will format the data in ASCII for the specified printer. If *bHostPrintTransform* equals VARIANT\_FALSE, the AS/400 Telnet server will not format the data in ASCII for the specified printer.

*PrinterModel*—The printer string supported by the AS/400 server (as listed in printer session properties).

*Drawer1*—Paper Type Codes

*Drawer2*—Paper Type Codes

*EnvelopeHopper*—Envelope Type Codes

*bASCII899*—The value that specifies whether the server supports the ASCII 899 code page to format the print job. Set it to VARIANT\_TRUE if you want the server to use the ASCII 899 code page to format the print job. Set it to VARIANT\_FALSE if you do not want the server to use the ASCII 899 code page to format the print job.

*CustomizingObject*—The string that specifies the name of the AS/400 customizing object.

*CustomizingLibrary*—The string that specifies the name of the AS/400 system library that contains the customizing object.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

    strProfile = "c:\\\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Transport = OSession.Transport
    Transport.SetHostPrintTransformInfo True, "IBM2380",
    1, 255, 255, True, "", ""

    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;
BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );
CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
    (void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);
    if (pSession)
    {
        pSession->Connect();
        pSession->get_Transport((IDispatch **)&pTransport);
        if (pTransport)
        {
            VARIANT_BOOL vbHostPrintTransform = VARIANT_TRUE;
            BSTR bstrPrinterModel = SysAllocString( OLESTR("*IBM2380") );
            BYTE Drawer1 = 1;
            BYTE Drawer2 = 255;
            BYTE EnvelopeHopper = 255;
            VARIANT_BOOL vbASCII899 = VARIANT_FALSE;
            BSTR bstrCustomizingObject = NULL;
            BSTR bstrCustomizingLibrary = NULL;

            pTransport->SetHostPrintTransformInfo( vbHostPrintTransform,
                bstrPrinterModel,
                Drawer1,
                Drawer2,
                EnvelopeHopper,
                vbASCII899,
                bstrCustomizingObject,
                bstrCustomizingLibrary);

            SysFreeString(bstrPrinterModel);
            SysFreeString(bstrCustomizingObject);
            SysFreeString(bstrCustomizingLibrary);
        }
        pSession->Disconnect();
    }
    pApplication->Release();
}
SysFreeString(bstrProfile);
SysFreeString(bstrSessionName);
```

## **Paper Type Codes**

The following table lists the hexadecimal values (and their decimal equivalents) that you use to specify paper types in the SetHostPrintTransformInfo method.

<b>Paper Type</b>	<b>Hexadecimal Value</b>	<b>Decimal Value</b>
None	FF	255
MFRTYPMDL	00	0
Letter	01	1
Legal	02	2
Executive	03	3
A4	04	4
A5	05	5
B5	06	6
CONT80	07	7
CONT132	08	8
A3	0E	14
B4	0F	15
Ledger	10	16

## Envelope Type Codes

The following table lists the hexadecimal values (and their decimal equivalents) that you use to specify envelope types in the SetHostPrintTransformInfo method.

Envelope Type	Hexadecimal Value	Decimal Value
None	FF	255
MFRTYPMDL	00	0
B5	06	6
Monarch	09	9
Number 9	0A	10
Number 10	0B	11
C5	0C	12
DL	0D	13

## SetKerberosInfo

### Method, IHETransport 3270 VT

This method lets you specify your Kerberos information, including the version of Kerberos you want to use and your authentication name(s).

**Basic Syntax**      `HETransport.SetKerberosInfo Version As Integer, AltUserName As String, UserName As String`

**C++ Syntax**      `HRESULT IHETransport::SetKerberosInfo(  
 [in] short Version,  
 [in] BSTR AltUserName,  
 [in] BSTR UserName);`

**Parameters**      *Version*—The Kerberos version to use for the current session. By default, this property is set to 4.

*AltUserName*—The string that specifies your alternate user name for Kerberos authentication.

*UserName*—The string that specifies your user name for Kerberos authentication.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Transport = OSession.Transport
    Transport.SetKerberosInfo 4, "Ella", "Helen"

    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL,
    CLSCTX_SERVER, iid, (void**)&pApplication);

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstr1 = SysAllocString( OLESTR("Ella") );
            BSTR bstr2 = SysAllocString( OLESTR("Helen") );

            pTransport->SetKerberosInfo( 4, bstr1, bstr2 );
            SysFreeString(bstr1);
            SysFreeString(bstr2);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

**ToggleBlockReceive**  
**Method, IHETransport 3270 5250 VT**

This method toggles the blocking of the receipt of data from the Transport objects.

**Basic Syntax**      `HETransport.ToggleBlockReceive(ToggleMode As HOSTEX_TOGGLE_RECEIVE) As Boolean`

**C++ Syntax**      `HRESULT IHETransport::ToggleBlockReceive(  
[in] HOSTEX_TOGGLE_RECEIVE ToggleMode,  
[out, retval] VARIANT_BOOL *pVal);`

**Parameters**      *ToggleMode*—The toggle mode.

*pVal*—The returned value of the state after executing the method.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Transport As HETransport
```

```
Dim bVal As Boolean
```

```
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
Set Transport = OSession.Transport  
bVal = Transport.ToggleBlockReceive (HOSTEX_TOGGLE_RECEIVE_OFF)
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbVal;
            pTransport-
>ToggleBlockReceive (HOSTEX_TOGGLE_RECEIVE_OFF, &vbVal);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## 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
- LUNNameRequested
- DeviceType
- EnableEMode
- EnableSSH
- EnableTracing
- HostAddress
- HostName
- IsEncrypted
- IsReceiveBlocked
- Keyboard
- LineMode
- MaxBlockSize
- MessageQueueLibrary
- MessageQueueName
- ModelColumns
- ModelRows
- NumberOfRetries
- Password
- PerformingTransfer
- Port
- PortList
- RetryDelayTimeBetweenHosts
- SecurityOption
- SessionKeepAlive
- TelnetEcho
- TelnetIsLineMode
- TelnetIsLocalEcho
- TelnetName
- TerminalModel
- TerminalOnline
- TerminalType
- TNESession
- TraceFilename
- Username

## AttentionFormat

### Property, IHETransport 3270 5250 VT

This property returns or sets a value that enables compatibility with other servers or gateways. By default, this property is set to HOSTEX\_ATN\_FORMAT\_ATTACHMATE.

#### Basic Syntax

```
HOSTEX_ATN_FORMAT = HETransport.AttentionFormat  
HETransport.AttentionFormat = HOSTEX_ATN_FORMAT
```

#### C++ Syntax

```
HRESULT IHETransport::get_AttentionFormat([out, retval] HOSTEX_ATN_FORMAT  
*pVal);  
HRESULT IHETransport::put_AttentionFormat([in] HOSTEX_ATN_FORMAT newVal);
```

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim Format As HOSTEX_ATN_FORMAT  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
Format = Transport.AttentionFormat  
' Transport.AttentionFormat = HOSTEX_ATN_FORMAT_IBM  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_ATN_FORMAT Format;
            pTransport->get_AttentionFormat(&Format);

            //pTransport->put_AttentionFormat(HOSTEX_ATN_FORMAT_IBM);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## CharSet

### Property, IHETransport 5250

This property returns or sets character-set information to the host.

#### Basic Syntax

```
String = HETransport.CharSet
```

```
HETransport.CharSet = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_CharSet([out, retval] BSTR *pVal);
```

```
HRESULT IHETransport::put_CharSet([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, indicating the character-set information.

*newVal*—The set string, indicating the character-set information.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim Charset As String
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
Charset = Transport.Charset
```

```
' Transport.Charset = "00697"
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrCharset = NULL;
            pTransport->get_CharSet(&bstrCharset);
            SysFreeString(bstrCharset);

            // BSTR bstr = SysAllocString( OLESTR ("00697") ) ;
            // pTransport->put_CharSet(bstr);
            // SysFreeString(bstr);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## CodePage

### Property, IHETransport 5250

This property returns or sets a string that specifies what code page to use.

#### Basic Syntax

```
String = HETransport.CodePage  
HETransport.CodePage = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_CodePage([out, retval] BSTR *pVal);  
HRESULT IHETransport::put_CodePage([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the code page.

*newVal*—The set string, specifying the code page.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim CodePage As String  
  
    strProfile = "c:\\\\Display5250.HEP"  
strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
    Set Transport = OSession.Transport  
CodePage = Transport.CodePage  
` Transport.CodePage = "00037"  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrCodePage = NULL;
            pTransport->get_CodePage(&bstrCodePage);
            SysFreeString(bstrCodePage);

            // BSTR bstr = SysAllocString( OLESTR ( "00037" ) ) ;
            // pTransport->put_CodePage( bstr );
            // SysFreeString(bstr);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Connected

### Property, IHETransport 3270 5250 VT

This property returns or sets a value that indicates whether you are connected to the host.

#### Basic Syntax

```
Boolean = HETransport.Connected
```

```
HETransport.Connected = Boolean
```

#### C++ Syntax

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

```
HRESULT IHETransport::put_Connected([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that you want to connect to the host. A returned value of VARIANT\_FALSE indicates that you want to disconnect from the host.

*newVal*—A value of VARIANT\_TRUE indicates that you want to connect to the host. A value of VARIANT\_FALSE indicates that you want to disconnect from the host.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim bConnected As Boolean
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
bConnected = Transport.Connected
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbConnected;
            pTransport->get_Connected (&vbConnected);
            //      pTransport->put_Connected (VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ConnectionStatus

### Property, IHETransport 3270 5250 VT

This property returns a value that indicates the status of the connection to the host.

#### Basic Syntax

```
HOSTEX_CON_STATUS = HETransport.ConnectionStatus
```

#### C++ Syntax

```
HRESULT IHETransport::get_ConnectionStatus([out, retval]  
HOSTEX_CON_STATUS *pVal);
```

#### Parameters

*pVal*—The returned value, indicating the connection status to the host.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim ConnectionStatus As HOSTEX_CON_STATUS  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
  
ConnectionStatus = Transport.ConnectionStatus  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_CON_STATUS iStatus;
            pTransport->get_ConnectionStatus(&iStatus);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## LUNameRequested

### Property, IHETransport 3270 5250

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

#### Basic Syntax

```
String = HETransport.LUNameRequested  
HETransport.LUNameRequested = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_LUNameRequested([out, retval] BSTR *pVal);  
HRESULT IHETransport::put_LUNameRequested([in] BSTR newVal);
```

#### Parameters

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

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim StrLuname As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
StrLuname = Transport.LUNameRequested  
' Transport.LUNameRequested = "ValidName"  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrLUName = NULL;
            pTransport->get_LUNameRequested(&bstrLUName);

            // pTransport->put_LUNameRequested(bstrLUName);
            SysFreeString(bstrLUName);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## DeviceType

### Property, IHETransport 3270 5250

This property returns or sets a value that specifies whether the device type to be used with the Transport objects is a printer or display device.

#### Basic Syntax

```
HOSTEX_DEVICE_TYPE = HETransport.DeviceType
```

```
HETransport.DeviceType = HOSTEX_DEVICE_TYPE
```

#### C++ Syntax

```
HRESULT IHETransport::get_DeviceType([out, retval] HOSTEX_DEVICE_TYPE  
*pVal);
```

```
HRESULT IHETransport::put_DeviceType([in] HOSTEX_DEVICE_TYPE newVal);
```

#### Parameters

*pVal*—The returned value, indicating the device type to be used.

*newVal*—The set value, indicating the device type to be used.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim DeviceType As HOSTEX_DEVICE_TYPE
```

```
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
Set Transport = OSession.Transport  
DeviceType = Transport.DeviceType
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_DEVICE_TYPE Type;
            pTransport->get_DeviceType( &Type);

            // pTransport->put_DeviceType( Type);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## EnableEMode

### Property, IHETransport 3270

This property returns or sets a value that enables the features for the TN3270E (Enhanced) terminal. By default, this property is set to VARIANT\_TRUE.

#### Basic Syntax

```
Boolean = HETransport.EnableEMode  
HETransport.EnableEMode = Boolean
```

#### C++ Syntax

```
HRESULT IHETransport::get_EnableEMode ([out, retval] VARIANT_BOOL *pVal);  
HRESULT IHETransport::put_EnableEMode ([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value, which enables the features for the TN3270E terminal.

*newVal*—The set value, which enables the features for the TN3270E terminal.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSesession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim Emode As Boolean  
  
strProfile = "c:\\Display3270.HEP"  
strSessionName = "3270"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
OSesession.Connect  
  
Set Transport = OSesession.Transport  
Emode = Transport.EnableEMode  
' Transport.EnableEMode = True  
  
OSesession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication);

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbemode;
            pTransport->get_EnableEMode( &vbemode );

            // pTransport->put_EnableEMode( vbemode );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## EnableSSH

### Property, IHETransport 3270 VT

This property returns or sets a value that indicates whether HostExplorer uses SSH (Secure Shell) encryption between the server and client.

#### Basic Syntax

```
Boolean = HETransport.EnableSSH
```

```
HETransport.EnableSSH = Boolean
```

#### C++ Syntax

```
HRESULT IHETransport::get_EnableSSH([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT IHETransport::put_EnableSSH([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, HostExplorer uses SSH encryption. If *pVal* equals VARIANT\_FALSE, HostExplorer does not use SSH.

*newVal*—The new value that enables or disables SSH encryption. Set *newVal* to VARIANT\_TRUE to enable SSH. Set *newVal* to VARIANT\_FALSE to disable SSH.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim SSH as Boolean
```

```
strProfile = "c:\\Display3270.HEP"
```

```
strSessionName = "3270"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
SSH = Transport.EnableSSH
```

```
' Transport.EnableSSH = False
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbSSH;
            pTransport->get_EnableSSH(&vbSSH);
            // pTransport->put_EnableSSH(VARIANT_FALSE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## EnableTracing

Property, IHETransport **3270 5250 VT**

This property returns or sets a value that specifies whether you want to enable tracing. Tracing is the process of writing the information that is sent to and received from the host into a file. By default, this property is set to VARIANT\_FALSE.

### Basic Syntax

```
Boolean = HETransport.EnableTracing
```

```
HETransport.EnableTracing = Boolean
```

### C++ Syntax

```
HRESULT IHETransport::get_EnableTracing([out, retval] VARIANT_BOOL  
*pVal);
```

```
HRESULT IHETransport::put_EnableTracing([in] VARIANT_BOOL newVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE enables tracing. A returned value of VARIANT\_FALSE disables tracing.

*newVal*—A value of VARIANT\_TRUE enables tracing. A value of VARIANT\_FALSE disables tracing.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim EnableTrace As Boolean  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
  
EnableTrace = Transport.EnableTracing  
Transport.EnableTracing = False  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbEnable;
            pTransport->get_EnableTracing (&vbEnable);

            pTransport->put_EnableTracing (VARIANT_TRUE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HostAddress

**Property, IHETransport 3270 5250 VT**

This property returns a string that specifies the address of the host to which you are currently connected.

### Basic Syntax

```
String = HETransport.HostAddress
```

### C++ Syntax

```
HRESULT IHETransport::get_HostAddress([out, retval] BSTR *pVal);
```

### Parameters

*pVal*—The returned string specifying the address of the host to which you are connected.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport
Dim Address As String

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport
Address = Transport.HostAddress

OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrAddress = NULL;
            pTransport->get_HostAddress(&bstrAddress);
            SysFreeString(bstrAddress);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## HostName

### Property, IHETransport 3270 5250 VT

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

#### Basic Syntax

```
String = HETransport.HostName
```

```
HETransport.HostName = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_HostName([out, retval] BSTR *pVal);
```

```
HRESULT IHETransport::put_HostName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string indicating the host name.

*newVal*—The set string indicating the host name.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim Hostname As String
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
Hostname = Transport.HostName
```

```
' Transport.HostName = "YourHostName"
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrHostname = NULL;
            pTransport->get_HostName(&bstrHostname);
            // pTransport->put_HostName( bstrHostname );

            SysFreeString (bstrHostname);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString (bstrProfile);
SysFreeString (bstrSessionName );
```

## IsEncrypted

### Property, IHETransport 3270 VT

This property returns the level of encryption of the message that is sent from the terminal to the host.

#### Basic Syntax

```
HOSTEX_ENCRYPTED = HETransport.IsEncrypted
```

#### C++ Syntax

```
HRESULT IHETransport::get_IsEncrypted([out, retval] HOSTEX_ENCRYPTED  
*pVal);
```

#### Parameters

*pVal*—The returned value, which indicates the level of encryption.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim Encrypted As HOSTEX_ENCRYPTED  
  
strProfile = "c:\\Display3270.HEP"  
strSessionName = "3270"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
Encrypted = HETransport.IsEncrypted  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_ENCRYPTED Status;
            pTransport->get_IsEncrypted(&Status);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## IsReceiveBlocked

Property, IHETransport **3270 5250 VT**

This property returns the state of whether the receive is blocked.

### Basic Syntax

```
Boolean = HETransport.IsReceiveBlocked
```

### C++ Syntax

```
HRESULT IHETransport::get_IsReceiveBlocked([out, retval] VARIANT_BOOL  
*pVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the receive is blocked. A returned value of VARIANT\_FALSE indicates that the receive is not blocked.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim bVal As Boolean  
  
    strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
    Set Transport = OSession.Transport  
bVal = Transport.IsReceiveBlocked  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbReceived;
            pTransport->get_IsReceiveBlocked(&vbReceived);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Keyboard

### Property, IHETransport 5250

This property returns or sets a value that specifies the type of keyboard being used.

#### Basic Syntax

```
String = HETransport.Keyboard  
HETransport.Keyboard = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_Keyboard([out, retval] BSTR *pVal);  
HRESULT IHETransport::put_Keyboard([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, specifying the keyboard type.  
*newVal*—The set value, specifying the keyboard type.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim Keyboard As String  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Transport = OSession.Transport  
    Keyboard = Transport.Keyboard  
    ' Transport.Keyboard = "USB"  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrKeyboard = NULL;
            pTransport->get_Keyboard(&bstrKeyboard );
            SysFreeString(bstrKeyboard);

            // BSTR bstr = SysAllocString( OLESTR("USB") );
            // pTransport->put_Keyboard( bstr );
            // SysFreeString(bstr);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## MaxBlockSize

Property, IHETransport **3270 5250 VT**

This property returns or sets the maximum Winsock receive block size.

### Basic Syntax

```
Long = HETransport.MaxBlockSize
```

```
HETransport.MaxBlockSize = Long
```

### C++ Syntax

```
HRESULT IHETransport::get_MaxBlockSize([out, retval] long *pVal);
```

```
HRESULT IHETransport::put_MaxBlockSize([in] long newVal);
```

### Parameters

*pVal*—The returned value, indicating the maximum Winsock receive block size.

*newVal*—The set value, indicating the maximum Winsock receive block size.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSesession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim Size As Long
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSesession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSesession.Connect
```

```
Set Transport = OSesession.Transport
```

```
Size = Transport.MaxBlockSize
```

```
' Transport.MaxBlockSize = 16000
```

```
OSesession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            long Size;
            pTransport->get_MaxBlockSize(&Size);

            // pTransport->put_MaxBlockSize( 16000 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## MessageQueueLibrary

### Property, IHETransport 5250

This property returns or sets a value that specifies the library containing the queue object that directs the print output.

#### Basic Syntax

```
String = HETransport.MessageQueueLibrary  
HETransport.MessageQueueLibrary = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_MessageQueueLibrary([out, retval] BSTR *pVal);  
HRESULT IHETransport::put_MessageQueueLibrary([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, indicating the library that contains the queue object.  
*newVal*—The set value, indicating the library that contains the queue object.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim Message As String  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Transport = OSession.Transport  
    Message = Transport.MessageQueueLibrary  
    ' Transport.MessageQueueLibrary = "Your message Queue Library"  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrMessageQueueLibrary = NULL;
            pTransport->get_MessageQueueLibrary(&bstrMessageQueueLibrary );

            //  pTransport-
            >put_MessageQueueLibrary( bstrMessageQueueLibrary );

            SysFreeString(bstrMessageQueueLibrary );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## MessageQueueName

### Property, IHETransport 5250

This property returns or sets a value that specifies the queue object that directs the print output.

#### Basic Syntax

```
String = HETransport.MessageQueueName  
HETransport.MessageQueueName = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_MessageQueueName ([out, retval] BSTR *pVal);  
HRESULT IHETransport::put_MessageQueueName ([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned value, indicating the queue object.  
*newVal*—The set value, indicating the queue object.

#### Basic Example

```
Dim OManager As HECHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim QueueName As String  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Transport = OSession.Transport  
    QueueName = Transport.MessageQueueName  
    ' Transport.MessageQueueName = "Your message queue name"  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrMessageQueueName = NULL;
            pTransport->get_MessageQueueName(&bstrMessageQueueName );
            //    pTransport->put_MessageQueueName(bstrMessageQueueName );

            SysFreeString( bstrMessageQueueName );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

**ModelColumns****Property, IHETransport 3270 5250 VT**

This property returns or sets the number of columns required for the terminal. By default, this property is set to 80 columns.

**Basic Syntax**

```
Integer = HETransport.ModelColumns  
HETransport.ModelColumns = Integer
```

**C++ Syntax**

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

**Parameters**

*pVal*—The returned number of columns required for the terminal.  
*newVal*—The set number of columns required for the terminal.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim Columns As Integer  
  
    strProfile = "c:\\\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Transport = OSession.Transport  
    Columns = Transport.ModelColumns  
    ' Transport.ModelColumns = 132  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, id,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            short Col;
            pTransport->get_ModelColumns(&Col );

            // pTransport->put_ModelColumns(132 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## ModelRows

### Property, IHETransport 3270 5250 VT

This property returns or sets the number of rows required for the terminal. By default, this property is set to 24 rows.

#### Basic Syntax

```
Integer = HETransport.ModelRows  
HETransport.ModelRows = Integer
```

#### C++ Syntax

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

#### Parameters

pVal—The returned number of rows required for the terminal.  
newVal—The set number of rows required for the terminal.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim Rows As Integer  
  
    strProfile = "c:\\Display5250.HEP"  
    strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
    OSession.Connect  
  
    Set Transport = OSession.Transport  
    Rows = Transport.ModelRows  
    'Transport.ModelRows = 28  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            short Rows;
            pTransport->get_ModelRows(&Rows);

            // pTransport->put_ModelRows(28);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## NumberOfRetries

Property, IHETransport **3270 5250 VT**

This property returns or sets the number of additional times that the session tries to connect to the host (if the first attempt fails).

### Basic Syntax

```
Integer = HETransport.NumberOfRetries
```

```
HETransport.NumberOfRetries = Integer
```

### C++ Syntax

```
HRESULT IHETransport::get_NumberOfRetries([out, retval] short *pVal);
```

```
HRESULT IHETransport::put_NumberOfRetries([in] short newVal);
```

### Parameters

*pVal*—The returned number of retries. If *pVal* equals -1, the number of retries is infinite.

*newVal*—The number of retries you want to set. Set *newVal* to -1 to specify infinite retries.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim Retries As Integer
```

```
    strProfile = "c:\\Display5250.HEP"
```

```
    strSessionName = "as400"
```

```
    Set OManager = CreateObject("HEOhio.OhioManager")
```

```
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
    OSession.Connect
```

```
        Set Transport = OSession.Transport
```

```
        Retries = Transport.NumberOfRetries
```

```
        ' Transport.NumberOfRetries = 2
```

```
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            short Retries;
            pTransport->get_NumberOfRetries(&Retries);

            //pTransport->put_NumberOfRetries(2);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Password

### Property, IHETransport 5250

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

#### Basic Syntax

```
String = HETransport.Password  
HETransport.Password = String
```

#### C++ Syntax

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

#### Parameters

*pVal*—The returned string that specifies the password.

*newVal*—The set string that specifies the password.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim Password As String  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
Password = Transport.Password  
' Transport.Password = "New Password"  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrPassword = NULL;
            pTransport->get_Password(&bstrPassword);
            //pTransport->put_Password(bstrPassword);
            SysFreeString(bstrPassword);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PerformingTransfer

### Property, IHETransport VT

This property returns or sets a value that indicates whether the Transport object is in file transfer mode.

#### Basic Syntax

```
Boolean = HETransport.PerformingTransfer  
HETransport.PerformingTransfer = Boolean
```

#### C++ Syntax

```
HRESULT IHETransport::get_PerformingTransfer([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHETransport::put_PerformingTransfer([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the Transport object is in file transfer mode. A returned value of VARIANT\_FALSE indicates that the Transport object is not in file transfer mode.

*newVal*—A value of VARIANT\_TRUE indicates that the Transport object is in file transfer mode. A value of VARIANT\_FALSE indicates that the Transport object is not in file transfer mode.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim bXfer As Boolean  
  
    strProfile = "c:\\DisplayVT.HEP"  
    strSessionName = "VT"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
    Set OSession = OManager.OpenSession(strProfile, strSessionName)  
  
    OSession.Connect  
    Set Transport = OSession.Transport  
    bXfer = Transport.PerformingTransfer  
    'Transport.PerformingTransfer = False  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbXfer;
            pTransport->get_PerformingTransfer(&bXfer);
            //pTransport->put_PerformingTransfer(VARIANT_FALSE);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Port

### Property, IHETransport 3270 5250 VT

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

#### Basic Syntax

```
Long = HETransport.Port
```

```
HETransport.Port = Long
```

#### C++ Syntax

```
HRESULT IHETransport::get_Port([out, retval] long *pVal);
```

```
HRESULT IHETransport::put_Port([in] long newVal);
```

#### Parameters

*pVal*—The returned value, specifying the Internet port.

*newVal*—The set value, specifying the Internet port.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim Port As Long
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
Port = Transport.Port
```

```
'Transport.Port = 23
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID     iid     = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            long Port;
            pTransport->get_Port(&Port);
            // pTransport->put_Port( 23 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## PortList

### Property, IHETransport 3270 5250 VT

This property lets you specify a string that lists the port numbers you want to use. Use a semicolon (;) to separate successive port numbers in the string.

#### Basic Syntax

```
HETransport.PortList = String
```

#### C++ Syntax

```
HRESULT IHETransport::put_PortList([in] BSTR newVal);
```

#### Parameters

*newVal*—The list of ports.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

    strProfile = "c:\\Display5250.HEP"
    strSessionName = "as400"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Transport = OSession.Transport
    Transport.PortList = "1;23;80;8080"

    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrPorts = SysAllocString( OLESTR("1;23;80;8080") );
            pTransport->put_PortList(bstrPorts);
            SysFreeString(bstrPorts);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

**RetryDelayTimeBetweenHosts**

**Property, IHETransport 3270 5250 VT**

This property returns or sets the number of seconds that the session waits before connecting to the next host.

**Basic Syntax**

```
Integer = HETransport.RetryDelayTimeBetweenHosts  
HETransport.RetryDelayTimeBetweenHosts = Integer
```

**C++ Syntax**

```
HRESULT IHETransport::get_RetryDelayTimeBetweenHosts([out, retval] short  
*pVal);  
HRESULT IHETransport::put_RetryDelayTimeBetweenHosts([in] short newVal);
```

**Parameters**

*pVal*—The returned value, specifying the number of seconds that the session waits.  
*newVal*—The number of seconds that you want the session to wait before connecting to the next host.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim RetryDelay As Integer  
  
    strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
    Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
    Set Transport = OSession.Transport  
RetryDelay = Transport.RetryDelayTimeBetweenHosts  
' Transport.RetryDelayTimeBetweenHosts = 200  
  
    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            short      RetryDelay;
            pTransport->get_RetryDelayTimeBetweenHosts( &RetryDelay );
            // pTransport->put_RetryDelayTimeBetweenHosts( RetryDelay );

        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SecurityOption

Property, IHETransport 3270 5250 VT

This property returns or sets a value that specifies the traffic for securing the channel between the server and the client. By default, this option is set to HOSTEX\_SECURITY\_NO\_SECURITY.

### Basic Syntax

```
HOSTEX_SECURITY_OPTIONS = HETransport.SecurityOption  
HETransport.SecurityOption = HOSTEX_SECURITY_OPTIONS
```

### C++ Syntax

```
HRESULT IHETransport::get_SecurityOption([out, retval]  
HOSTEX_SECURITY_OPTIONS *pVal);  
HRESULT IHETransport::put_SecurityOption([in] HOSTEX_SECURITY_OPTIONS  
newVal);
```

### Parameters

*pVal*—The returned value, specifying the security option.

*newVal*—The set value, specifying the security method.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSesession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim Options As HOSTEX_SECURITY_OPTIONS  
  
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
OSesession.Connect  
  
Set Transport = OSesession.Transport  
Options = Transport.SecurityOption  
' Transport.SecurityOption = HOSTEX_SECURITY_SSL_TLS  
  
OSesession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_SECURITY_OPTIONS Options;
            pTransport->get_SecurityOption( &Options);

            // pTransport->put_SecurityOption(HOSTEX_SECURITY_SSL_TLS);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## SessionKeepAlive

Property, IHETransport **3270 5250 VT**

This property returns or sets a value that indicates the time interval (in seconds) before the Transport object sends Keep Alive messages to the host. By default, this property is set to 30 seconds.

### Basic Syntax

```
Long = HETransport.SessionKeepAlive
```

```
HETransport.SessionKeepAlive = Long
```

### C++ Syntax

```
HRESULT IHETransport::get_SessionKeepAlive([out, retval] long *pVal);
```

```
HRESULT IHETransport::put_SessionKeepAlive([in] long newVal);
```

### Parameters

*pVal*—The returned value, indicating the time interval before the Transport object sends Keep Alive messages to the host.

*newVal*—The set value, indicating the time interval before the Transport object sends Keep Alive messages to the host.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim KeepAlive As Long
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
KeepAlive = Transport.SessionKeepAlive
```

```
' Transport.SessionKeepAlive = 32
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            long KeepAlive;
            pTransport->get_SessionKeepAlive(&KeepAlive);

            //pTransport->put_SessionKeepAlive(35);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TelnetEcho

### Property, IHETransport 3270 5250 VT

This property returns or sets a value that indicates how HostExplorer responds to remote echo negotiation with a Telnet host. By default, this property is set to HOSTEX\_TELNETECHO\_AUTOMATIC.

#### Basic Syntax

```
HOSTEX_TELNETECHO = HETransport.TelnetEcho
```

```
HETransport.TelnetEcho = HOSTEX_TELNETECHO
```

#### C++ Syntax

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

```
HRESULT IHETransport::put_TelnetEcho([in] HOSTEX_TELNETECHO newVal);
```

#### Parameters

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

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

#### Basic Example

```
Dim OManager As HEHIOLib.OhioManager  
Dim OSesession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
Dim TelnetEcho As HOSTEX_TELNETECHO
```

```
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSesession = OManager.OpenSession(strProfile, strSessionName)  
OSesession.Connect
```

```
Set Transport = OSesession.Transport  
TelnetEcho = Transport.TelnetEcho  
' Transport.TelnetEcho = HOSTEX_TELNETECHO_NO  
OSesession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_TELNETECHO TelnetEcho
            pTransport->get_TelnetEcho(&TelnetEcho);
            // pTransport->put_TelnetEcho( HOSTEX_TELNETECHO_NO );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TelnetIsLineMode

Property, IHETransport **3270 5250 VT**

This property returns a value that indicates whether HostExplorer stores characters in a buffer until you send a carriage return to the host. When enabled, Linemode forces HostExplorer to send characters one line at a time rather than as individual characters.

**Basic Syntax**

```
Boolean = HETransport.TelnetIsLineMode
```

**C++ Syntax**

```
HRESULT IHETransport::get_TelnetIsLineMode([out, retval] VARIANT_BOOL *pVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that you are in Linemode. A returned value of VARIANT\_FALSE indicates that you are not in Linemode.

**Basic Example**

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

Dim LineMode As Boolean

    strProfile = "c:\\DisplayVT.HEP"
    strSessionName = "VT"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Transport = OSession.Transport
    LineMode = Transport.TelnetIsLineMode

    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR( "VT" ) );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbLineMode;
            pTransport->get_TelnetIsLineMode( &vbLineMode );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TelnetIsLocalEcho

Property, IHETransport **3270 5250 VT**

This property returns a value that indicates whether you are currently in local echo mode.

### Basic Syntax

```
Boolean = HETransport.TelnetIsLocalEcho
```

### C++ Syntax

```
HRESULT IHETransport::get_TelnetIsLocalEcho([out, retval] VARIANT_BOOL *pVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that you are in local echo mode.  
A returned value of VARIANT\_FALSE indicates that you are not in local echo mode.

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

Dim LocalEcho As Boolean

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport
LocalEcho = Transport.TelnetIsLocalEcho

OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbLocalEcho;
            pTransport->get_TelnetIsLocalEcho( &vbLocalEcho );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TelnetName

### Property, IHETransport 3270 5250 VT

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

#### Basic Syntax

```
String = HETransport.TelnetName
```

```
HETransport.TelnetName = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_TelnetName([out, retval] BSTR *pVal);
```

```
HRESULT IHETransport::put_TelnetName([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying a Telnet name.

*newVal*—The set string, specifying a Telnet name.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Transport As HETransport
```

```
Dim TelnetName As String
```

```
strProfile = "c:\\Display5250.HEP"  
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
Set Transport = OSession.Transport  
TelnetName = Transport.TelnetName  
'Transport.TelnetName = "Some Telnet Name"
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrTelnetName = NULL;
            pTransport->get_TelnetName( &bstrTelnetName);

            // pTransport->put_TelnetName( bstrTelnetName);
            SysFreeString(bstrTelnetName) ;
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TerminalModel

### Property, IHETransport 3270 5250 VT

This property returns or sets a value that indicates the terminal model that you are using to connect to the host. For TN3270 and TN5250 terminals, the default for this property is set to HOSTEX\_TERM\_MODEL\_2. For TNVT terminals, the default is set to HOSTEX\_TERM\_MODEL\_VT220.

#### Basic Syntax

```
HOSTEX_TERM_MODEL = HETransport.TerminalModel
```

```
HETransport.TerminalModel = HOSTEX_TERM_MODEL
```

#### C++ Syntax

```
HRESULT IHETransport::get_TerminalModel([out, retval] HOSTEX_TERM_MODEL  
*pVal);
```

```
HRESULT IHETransport::put_TerminalModel([in] HOSTEX_TERM_MODEL newVal);
```

#### Parameters

*pVal*—The returned value, indicating the terminal model.

*newVal*—The set value, indicating the terminal model.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager  
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String  
Dim Transport As HETransport
```

```
Dim Terminal As HOSTEX_TERM_MODEL
```

```
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect
```

```
Set Transport = OSession.Transport  
Terminal = Transport.TerminalModel  
' Transport.TerminalModel = HOSTEX_TERM_MODEL_VT102
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_TERM_MODEL Terminal;
            pTransport->get_TerminalModel( &Terminal );
            // pTransport->put_TerminalModel( HOSTEX_TERM_MODEL_VT102 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TerminalOnline

### Property, IHETransport VT

This property returns or sets a value that specifies whether the VT terminal is online.

#### Basic Syntax

```
Boolean = HETransport.TerminalOnline  
HETransport.TerminalOnline = Boolean
```

#### C++ Syntax

```
HRESULT IHETransport::get_TerminalOnline([out, retval] VARIANT_BOOL  
*pVal);  
HRESULT IHETransport::put_TerminalOnline([in] VARIANT_BOOL newVal);
```

#### Parameters

*pVal*—The returned value. If *pVal* equals VARIANT\_TRUE, the terminal is online. If *pVal* equals VARIANT\_FALSE, the terminal is not online.  
*newVal*—The new online status (either VARIANT\_TRUE or VARIANT\_FALSE) for the terminal.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager  
Dim OSession As OhioSession  
  
Dim strProfile, strSessionName As String  
Dim Transport As HETransport  
  
Dim Online As Boolean  
  
strProfile = "c:\\DisplayVT.HEP"  
strSessionName = "VT"  
  
Set OManager = CreateObject("HEOhio.OhioManager")  
Set OSession = OManager.OpenSession(strProfile, strSessionName)  
OSession.Connect  
  
Set Transport = OSession.Transport  
  
Online = Transport.TerminalOnline  
Transport.TerminalOnline = False  
  
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbOnline;
            pTransport->get_TerminalOnline( &vbOnline);
            // pTransport->put_TerminalOnline( VARIANT_FALSE);
        }      pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TerminalType

Property, IHETransport **3270 5250 VT**

This property lets you specify the terminal type you want to use (one of Display 3270, Display 5250, Display VT, Printer 5250, or Printer 3270).

### Basic Syntax

```
HETransport.TerminalType = Integer
```

### C++ Syntax

```
HRESULT IHETransport::put_TerminalType([in] short newVal);
```

### Parameters

*newVal*—The terminal type. Use the following values to specify the terminal type:

- 1—Terminal 3270
- 2—Terminal VT
- 4—Terminal 5250
- 8—Printer 3270
- 16—Printer 5250

### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

strProfile = "c:\\Display5250.HEP"
strSessionName = "as400"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport
Transport.TerminalType = 4

OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            pTransport->put_TerminalType( 4 );
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TNESession

### Property, IHETransport 3270

This property returns or sets a value that indicates whether the terminal is in TN3270E mode.

#### Basic Syntax

```
Boolean = HETransport.TNESession
```

#### C++ Syntax

```
HRESULT IHETransport::get_TNESession([out, retval] VARIANT_BOOL *pVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the terminal is in TN3270E mode. A returned value of VARIANT\_FALSE indicates that the terminal is not in TN3270E mode.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

Dim TNSession As Boolean

strProfile = "c:\\Display3270.HEP"
strSessionName = "3270"

Set OManager = CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession(strProfile, strSessionName)
OSession.Connect

Set Transport = OSession.Transport

TNSession = Transport.TNESession

OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display3270.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("3270") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**) &pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            VARIANT_BOOL vbTNSession;
            pTransport->get_TNESession(&vbTNSession);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## TraceFilename

### Property, IHETransport 3270 5250 VT

This property returns or sets a string that specifies the name of the file that contains the information that is sent to and received from the host. By default, this property is set to C:\HETRACE.TRC.

#### Basic Syntax

```
String = HETransport.TraceFilename
```

```
HETransport.TraceFilename = String
```

#### C++ Syntax

```
HRESULT IHETransport::get_TraceFilename([out, retval] BSTR *pVal);
```

```
HRESULT IHETransport::put_TraceFilename([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying the file name.

*newVal*—The set string, specifying the file name.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
Dim TraceFile As String
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
TraceFile = Transport.TraceFilename
```

```
Transport.TraceFilename = "C:\\File.txt"
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrTraceFile = NULL;
            pTransport->get_TraceFilename(&bstrTraceFile);

            // pTransport.put_TraceFilename (bstrTraceFile);
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Username

### Property, IHETransport 5250

This property sets a string that specifies the user name that you use to connect to the host.

#### Basic Syntax

```
HETransport.Username = String
```

#### C++ Syntax

```
HRESULT IHETransport::put_Username([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned string, specifying your user name.

*newVal*—The set string, specifying your user name.

#### Basic Example

```
Dim OManager As HEOHIOLib.OhioManager
```

```
Dim OSession As OhioSession
```

```
Dim strProfile, strSessionName As String
```

```
Dim Transport As HETransport
```

```
strProfile = "c:\\Display5250.HEP"
```

```
strSessionName = "as400"
```

```
Set OManager = CreateObject("HEOhio.OhioManager")
```

```
Set OSession = OManager.OpenSession(strProfile, strSessionName)
```

```
OSession.Connect
```

```
Set Transport = OSession.Transport
```

```
Transport.Username = "Joe"
```

```
OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\Display5250.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("as400") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            BSTR bstrUsername = SysAllocString( OLESTR("Joe") );
            pTransport->put_Username( bstrUsername ) ;

            SysFreeString( bstrUsername ) ;
        }
        pSession->Disconnect();
    }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## LineMode

### Property, IHETransport VT

This property returns or sets a value that indicates whether or not you are currently in Linemode. By default, this property is set to HOSTEX\_LINEMODE\_DONTDOLINEMODE.

#### Basic Syntax

```
HOSTEX_LINEMODE = HETransport.LineMode
```

```
HETransport.LineMode = HOSTEX_LINEMODE
```

#### C++ Syntax

```
HRESULT IHETransport::get_LineMode([out, retval] HOSTEX_LINEMODE *pVal);
```

```
HRESULT IHETransport::put_LineMode([in] HOSTEX_LINEMODE newVal);
```

#### Parameters

*pVal*—A returned value of TRUE indicates that you are currently in Linemode. A returned value of FALSE indicates you are not currently in Linemode.

*newVal*—A value of TRUE indicates that you are currently in Linemode. A value of FALSE indicates you are not currently in Linemode.

#### Basic Example

```
Dim OManager As HEOhioLib.OhioManager
Dim OSession As OhioSession

Dim strProfile, strSessionName As String
Dim Transport As HETransport

Dim LineMode As HOSTEX_LINEMODE

    strProfile = "c:\\\\DisplayVT.HEP"
    strSessionName = "VT"

    Set OManager = CreateObject("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession(strProfile, strSessionName)
    OSession.Connect

    Set Transport = OSession.Transport

        LineMode = Transport.LineMode
        ' Transport.LineMode = HOSTEX_LINEMODE_ALWAYS

    OSession.Disconnect
```

**C++ Example**

```
IOhioManager* pApplication = NULL;
IOhioSession* pSession = NULL;
IHETransport* pTransport = NULL;

BSTR bstrProfile = SysAllocString( OLESTR("c:\\DisplayVT.HEP") );
BSTR bstrSessionName = SysAllocString( OLESTR("VT") );

CLSID clsid = CLSID_OhioManager;
IID iid = IID_IOhioManager;
HRESULT hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, iid,
(void**)&pApplication );

If ( SUCCEEDED(hr) )
{
    pApplication->OpenSession(bstrProfile, bstrSessionName, &pSession);

    if(pSession)
    {
        pSession->Connect();

        pSession->get_Transport((IDispatch **)&pTransport);
        if(pTransport)
        {
            HOSTEX_LINEMODE LineMode;
            pTransport->get_LineMode( &LineMode);

            // pTransport->put_LineMode( HOSTEX_LINEMODE_ALWAYS);
        }
        pSession->Disconnect(); }

    pApplication->Release();
}

SysFreeString(bstrProfile);
SysFreeString(bstrSessionName );
```

## Data Types of the Transport Objects

The Transport objects contain the following data types:

- HETRANSPORT\_FEATURE Data Type
- 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
- HOSTEX\_TELNETECHO Data Type
- HOSTEX\_TERM\_MODEL Data Type
- HOSTEX\_TOGGLE\_RECEIVE Data Type

## About OHIO

OHIO (Open Host Interface Objects) is a standardized programming interface to the host data. OHIO provides a common access method to the data when it arrives at the client and divides the data into logical objects. Using OHIO, you can write a script that can run in any type of emulator that supports OHIO.

The HostExplorer object Ohio is a base interface for eight Ohio interfaces, which use a specific inheritance hierarchy.

Ohio is used to create a screen that lets you communicate with, and connect to or disconnect from, the host. Ohio functions are used to call Transport and Parser functions.

Sample files of Ohio are available in the `HostExplorer\SDK\Samples\OHIO` directory where the program files are stored on your machine.

## OhioManager Interface

The OhioManager interface is a central repository that provides a list of all available OhioSession objects. It allows you to open and close sessions.

### Methods

The OhioManager interface consists of the following methods:

- CloseSession
- OpenSession

### Properties

The OhioManager interface consists of the following properties:

- OhioVersion
- Sessions
- VendorName
- VendorObject
- VendorProductVersion

## CloseSession

### Method, IOhioManager

This method closes an OhioSession object. The OhioSession is considered invalid and is removed from the list of OhioSession objects.

**Basic Syntax**      `OhioManager.CloseSession(inSession As Variant)`

**C++ Syntax**      `HRESULT IOhioManager::CloseSession([in] VARIANT inSession);`

**Parameters**      *inSession*—The IOhioSession that you want to close.

**Basic Example**

```
Dim OManager As OhioManager
Dim OSession As OhioSession
.
.
OManager.CloseSession (OSession)
'close by session name
OManager.CloseSession ("aix")
```

**C++ Example**

```
IOhio
Session* pOhioSession;
IOhioManager* pOhioManager;
VARIANT vr;
.
.
vr.vt = VT_DISPATCH;
pOhioSession->QueryInterface
    (IID_IDispatch, (void **)
     &vr.pdispVal);
vr.pdispVal->Release();
pOhioManager->CloseSession(vr);
```

## OpenSession

### Method, IOhioManager

This method creates an OhioSession object.

**Basic Syntax**      `OhioManager.OpenSession(inConfigResource As String, inSessionName As String) As OhioSession`

**C++ Syntax**      `HRESULT IOhioManager::OpenSession([in] BSTR inConfigResource, [in] BSTR inSessionName, [out, retval] IOhioSession * * outSession);`

**Parameters**

*inConfigResource*—The Hummingbird profile name.  
*inSessionName*—The name of the IOhioSession object to be created.  
*outSession*—The returned address of the IOhioSession object that is created.

**Basic Example**

```
Dim OManager As Object
Dim OSession As Object
Dim strProfile, strSessionName As String
.
.
.
strProfile = "C:\\\\Aix.hep"
strSessionName = "aix"
Set OManager =
    CreateObject("HEOhio.OhioManager")
Set OSession = OManager.OpenSession
    (strProfile,strSessionName)
```

**C++ Example**

```
IOhioManager* pOhioManager = NULL;
.
.
.
HRESULT hr = CoCreateInstance
    (CLSID_OhioManager, NULL,
     CLSCTX_INPROC_SERVER, IID_IOhioManager,
     reinterpret_cast<void**>
    (&pOhioManager));

if (FAILED(hr)){...} //error

BSTR l_bstrHEProfile, l_bstrSessionName;
l_bstrSessionName = SysAllocString
    (OLESTR("aix"));
l_bstrHEProfile = SysAllocString
    (OLESTR("C:\\\\Aix.hep"));
hr = pOhioManager->OpenSession
    (l_bstrHEProfile, l_bstrSessionName,
     &pOhioSession);
if (FAILED(hr)) {...} //error

SysFreeString(l_bstrHEProfile);
SysFreeString(l_bstrSessionName);
```

## OhioVersion

### Property, IOhioManager

This property returns the Ohio version level of the implementation.

<b>Basic Syntax</b>	String = OhioManager. <b>OhioVersion</b>
<b>C++ Syntax</b>	HRESULT IOhioManager:: <b>get_OhioVersion</b> ([out, retval] BSTR * <i>pVal</i> );
<b>Parameters</b>	<i>pVal</i> —The returned Ohio version level.
<b>Basic Example</b>	Dim OManager As OhioManager Dim strVersion As String . . . strVersion = OManager.OhioVersion
<b>C++ Example</b>	IOhioManager* pOhioManager; . . . BSTR bstrVersion; HRESULT hr = pIOhioManager-> <b>get_OhioVersion</b> (&bstrVersion); if (FAILED (hr)) {...} //error

## Sessions

### Property, IOhioManager

This property returns an OhioSessions object containing the OhioSession objects available on the system. Once the OhioSessions object is created, the list of objects is static. Use the OhioSessions. Refresh method to update the list.

<b>Basic Syntax</b>	OhioSessions = OhioManager. <b>Sessions</b>
<b>C++ Syntax</b>	HRESULT IOhioManager:: <b>get_Sessions</b> ([out, retval] IOhioSessions ** <i>pVal</i> );
<b>Parameters</b>	<i>pVal</i> —The returned address of the IOhioSessions object.

**Basic Example**

```
Dim OManager As OhioManager
Dim OSessions As OhioSessions
Dim OSess1, OSess2 As OhioSession
Set OSessions = OManager.Sessions
'retrieve OhioSession object by session
'name
Set OSess = OManager.Sessions("aix")
'retrieve OhioSession object by index
Set OSess2 = OManager.Sessions(2)
```

**C++ Example**

```
IOhioManager* pOhioManager;
IOhioSessions* pIOhioSessions = NULL;
. . .
HRESULT = pIOhioManager->
    get_Sessions(&pIOhioSessions);
if (FAILED(hr)) {...} //error
long iCount;
pIOhioSessions->get_Count(&iCount);
```

**VendorName****Property, IOhioManager**

This property returns the name of the vendor that is providing the OHIO implementation.

**Basic Syntax**

```
String = OManager.VendorName
```

**C++ Syntax**

```
HRESULT IOhioManager::get_VendorName([out, retval] BSTR * pVal);
```

**Parameters**

*pVal*—The returned name of the vendor.

**Basic Example**

```
Dim OManager As OhioManager
Dim strVendorName As String
. . .
strVendorName = OManager.VendorName
```

**C++ Example**

```
IOhioManager* pOhioManager;
BSTR bstrVendorName;
. . .
HRESULT hr = pOhioManager->
    get_VendorName(&bstrVendorName);
if (FAILED(hr)) {...} //error
printf("vendor name is %s\n", OLE2A(bstrVendorName));
SysFreeString(bstrVendorName);
```

## VendorObject

### Property, IOhioManager

This property returns the address of the vendor object that provides non-standard, vendor-specific extensions to the Ohio object.

#### Basic Syntax

```
Dispatch = OhioManager.VendorObject
```

#### C++ Syntax

```
HRESULT IOhioManager::get_VendorObject([out, retval] IDispatch ** pVal);
```

#### Parameters

*pVal*—There are currently no extensions implemented in the property VendorObject; therefore, this property returns a null pointer.

## VendorProductVersion

### Property, IOhioManager

This property returns the vendor product version that is providing the OHIO implementation.

#### Basic Syntax

```
String = OhioManager.VendorProductVersion
```

#### C++ Syntax

```
HRESULT OhioManager.get_VendorProductVersion([out, retval] BSTR * pVal);
```

#### Parameters

*pVal*—The returned vendor product version.

#### Basic Example

```
Dim OManager As OhioManager
Dim strVendorProdVer As String
.
.
strVendorProdVer =
    OManager.VendorProductVersion
```

#### C++ Example

```
BSTR bstrVendorProdVer;
IOhioManager* pOhioManager;
.
.
pOhioManager->get_VendorProductVersion
    (&bstrVendorProdVer);
printf("Product version is %d\n", bstrVendorProdVer);
SysFreeString(bstrVendorProdVer);
```

## OhioSessions Interface

The OhioSessions interface contains a collection of all the available sessions. You can use this interface to request a specific session or determine the number of sessions available. Because the list of sessions is static, you can use a refresh function to get an updated list.

### Methods

The OhioSessions interface consists of the following method:

Refresh

### Properties

The OhioSessions interface consists of the following properties:

- Count
- Item

## Refresh

### Method, IOhioSessions

This method updates the list of sessions with all the available OhioSession objects. Updating does not preserve the indexing of OhioSession objects.

#### Basic Syntax

OhioSessions.**Refresh**

#### C++ Syntax

HRESULT IOhioSessions::**Refresh()**;

#### Parameters

This method has no parameters.

**Basic Example**

```
'Here we refresh the OhioSessions object
'before retrieving its third OhioSession
;element, if any
Dim OSessions As OhioSessions
Dim OSess As OhioSession

. . .

OSessions.Refresh
'get the OhioSession object at index 3
Set OSess = OSessions(3)
If (OSess Is Nothing) Then
    'no session at index
Else
    'session at index
```

**C++ Example**

```
//CloseSession decrements the count. If
//we do not call refresh, the previous
//count will be retrieved.
IOhioManager* pIOhioManager;
IOhioSessions* pIOhioSessions
long lCount;
pIOhioManager->CloseSession(vr); pIOhioSessions->Refresh();
HRESULT hr = pIOhioSessions->get_Count
(&lCount);
if (FAILED(hr)) {...} //error
```

**Count****Property, IOhioSessions**

This property returns the number of OhioSession objects contained in the collection.

**Basic Syntax**

```
Long = OhioSessions.Count
```

**C++ Syntax**

```
HRESULT IOhioSessions::get_Count([out, retval] long * pVal);
```

**Parameters**

*pVal*—The returned value of the IOhioSession objects contained in the collection.

**Basic Example**

```
Dim OManager As OhioManager
Dim OSessions As OhioSessions
Dim num As Long
.
.
.
'access the count from OhioSessions object
num = OSessions.Count
'access count directly from OhioManager
num = OManager.OSessions.Count
```

**C++ Example**

```
long lCount;
IOhioSessions* pIOhioSessions;
.
.
.
pIOhioSessions->Refresh();
HRESULT hr = pIOhioSessions->get_Count
(&lCount);
if (FAILED (hr)) {...} //error
```

**Item****Property, IOhioSessions**

This property returns the OhioSession object at the specified index (one-based indexing) or name of the session.

**Basic Syntax**

```
OhioSession = OhioSessions.Item
```

```
OhioSessions.Item = VARIANT
```

**C++ Syntax**

```
HRESULT IOhioSessions::get_Item([in] VARIANT inIndexOrKey, [out, retval]
IOhioSession ** pVal);
```

**Parameters**

*inIndexOrKey*—The index of the session.

*IOhioSession*—The name of the session.

*pVal*—The returned address of the IOhioSession object.

**Basic Example**

```
Dim OSessions As OhioSessions
Dim OManager As OhioManager
Dim OSess As OhioSession
.
.
.
OSessions.Refresh
'access OhioSession object by numeric
'index
Set OSess = OSessions.Item("1")
'access OhioSession object by session name
Set OSess = OSessions.Item("aix")
'access OhioSession object directly from
'OhioManager
Set Osess = OManager.Sessions.Item("1");
```

**C++ Example**

```
IOhioSessions* pIOhioSessions;
IOhioSession* p0Sess;
.
.
.
BSTR bstrSessName;
VARIANT vrSessions;
vrSessions.vt=VT_I4;
vrSessions.lVal=1;
pIOhioSessions->get_Item
(vrSessions,&p0Sess);
p0Sess->get_SessionName(&bstrSessName);
USES_CONVERSION;
char* szSessName = OLE2A(bstrSessName);
printf("Session Name is %s\n",szSessName);
SysFreeString(bstrSessName)
```

## OhioSession Interface

The OhioSession interface allows you to connect to or disconnect from the host and access the host through the screen. It can provide you with the following data:

- session type (for example, 3270, 5250, VT)
- session name (for example, Session 1, Session 2)
- session status (for example, connected or disconnected)

### Methods

The OhioSession interface consists of the following methods:

- Connect
- Disconnect
- isConnected
- OnSessionChanged

### Properties

- CanChangeScreen
- ConfigurationResource
- Screen
- SessionName
- SessionType

## Connect

### Method, IOhioSession

This method starts the communications link to the host.

**Basic Syntax**      `OhioSession.Connect`

**C++ Syntax**      `HRESULT IOhioSession::Connect();`

**Parameters**      This method has no parameters.

**Basic Example**

```
Dim OManager As Object
Dim OSession As Object
Dim strProfile, strSessionName As String
.
.
.
strProfile = "C:\\Aix.hep"
strSessionName = "aix"
Set OManager = CreateObject
    ("HEOhio.OhioManager")
Set OSession = OManager.OpenSession
    (strProfile, strSessionName)
OSession.Connect
```

**C++ Example**

```
IOhioManager* pIOhioManager;
IOhioSession* pIOhioSession;

. . .

HRESULT hr = CoCreateInstance
    (CLSID_OhioManager, NULL,
    CLSCTX_INPROC_SERVER, IID_IOhioManager,
    reinterpret_cast<void**>
    (&pIOhioManager));

if (FAILED(hr)) {...} //error

BSTR bstrHEProfile, bstrSessionName;
bstrSessionName = SysAllocString
    (OLESTR("aix"));
bstrHEProfile = SysAllocString
    (OLESTR("C:\\\\Aix.hep"));

pIOhioManager->OpenSession
    (l_bstrHEProfile, bstrSessionName,
    &pIOhioSession);

SysFreeString(bstrHEProfile);
SysFreeString(bstrSessionName);

pIOhioSession->Connect();
pIOhioSession->Disconnect();
```

## Disconnect Method, IOhioSession

This method stops the communications link to the host.

**Basic Syntax**

OhioSession.**Disconnect**

**C++ Syntax**

HRESULT IOhioSession::**Disconnect()**;

**Parameters**

This method has no parameters.

**Basic Example**

```
Dim OManager As Object
Dim OSession As Object
Dim strProfile, strSessionName As String
.
.
.
strProfile = "C:\\Aix.hep"
strSessionName = "aix"
Set OManager = CreateObject
    ("HBOhio.OhioManager")
Set OSession = OManager.OpenSession
    (strProfile, strSessionName)
OSession.Connect
OSession.Disconnect
```

**C++ Example**

```
IOhioSession* pIOhioSession;
IOhioManager* pIOhioManager;
.
.
.
HRESULT hr = CoCreateInstance
    (CLSID_OhioManager, NULL,
    CLSCTX_INPROC_SERVER, IID_IOhioManager,
    reinterpret_cast<void**>
    (&pIOhioManager));

if (FAILED(hr)) {...} //error

BSTR bstrHEProfile, bstrSessionName;
bstrSessionName = SysAllocString
    (OLESTR("aix"));
bstrHEProfile = SysAllocString
    (OLESTR("C:\\Aix.hep"));

pIOhioManager->OpenSession
    (l_bstrHEProfile, bstrSessionName,
    &pIOhioSession);

SysFreeString(bstrHEProfile);
SysFreeString(bstrSessionName);

pIOhioSession->Connect();
pIOhioSession->Disconnect();
```

## isConnected

### Method, IOhioSession

This method indicates whether the OhioSession object is connected to a host.

#### Basic Syntax

```
Boolean = OhioSession.isConnected
```

#### C++ Syntax

```
HRESULT IOhioSession::isConnected([out, retval] VARIANT_BOOL * outValue);
```

#### Parameters

*outValue*—A value of VARIANT\_TRUE indicates that the object is connected to a host. A value of VARIANT\_FALSE indicates that the object is not connected to a host.

#### Basic Example

```
Dim OSession As OhioSession
Dim bIsConnected As Boolean
.
.
.
bIsConnected = OSession.isConnected
If (bIsConnected = True) Then
    'connected
Else
    'not connected
End If
```

#### C++ Example

```
IOhioSession* pIOhioSession;
VARIANT_BOOL bConnected;
.
.
.
pIOhioSession->isConnected(&bConnected);
if (bConnected == VARIANT_FALSE)
{
    printf("Session disconnected.\n");
}
```

## OnSessionChanged

### Method, IOhioSession

This method is called when the state of the session changes.

#### Basic Syntax

```
OhioSession.OnSessionChanged(inState As OHIO_STATE)
```

#### C++ Syntax

```
HRESULT IOhioSession::OnSessionChanged([in] OHIO_STATE inState);
```

#### Parameters

*inState*—The changed state of the session.

**Basic Example**

```
Dim WithEvents oIOhioScr As
    HEOHILib.OhioScreen
Dim WithEvents oIOhioSess As
    HEOHILib.OhioSession
Dim OManager As OhioManager
.....
Private Sub Form_Load()
    strProfile = "C:\\\\Aix.hep"
    strSessionName = "aix"
    Set OManager = CreateObject
        ("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession
        (strProfile, strSessionName)
    Set oIOhioSess = Osession
End Sub
.....
Private Sub oIOhioSess_OnSessionChanged
    (ByVal inState As HEOHILib.OHIO_STATE)
    Debug.Print "OnSessionChanged called."
    If (inState = OHIO_STATE_CONNECTED) Then
        Debug.Print "Connected"
    Else
        Debug.Print "Disconnected"
    End If
End Sub
```

## CanChangeScreen Property, IOhioSession

This property indicates whether you or the host can change the contents of the screen.

**Basic Syntax**

```
Boolean = OhioSession.CanChangeScreen
OhioSession.CanChangeScreen = Boolean
```

**C++ Syntax**

```
HRESULT IOhioSession::get_CanChangeScreen ([out, retval] VARIANT_BOOL *  
pVal);
HRESULT IOhioSession::put_CanChangeScreen ([in] VARIANT_BOOL newVal);
```

**Parameters**

*pVal*—The system returns a value of VARIANT\_TRUE if you or the host can change the screen contents. It returns VARIANT\_FALSE if you or the host cannot change the screen contents.

*newVal*—A value of VARIANT\_TRUE indicates that you or the host can change the screen contents. A value of VARIANT\_FALSE indicates that you or the host cannot change the screen contents.

**Basic Example**

```
Dim bOSession As OhioSession
Dim bCanChangeScreen As Boolean
.
.
.
bCanChangeScreen=OSession.CanChangeScreen
If (bCanChangeScreen = True) Then
.
.
.
Else
.
.
.
End If
'Set the value of OSession.CanChangeScreen
'to False as default
OSession.CanChangeScreen = False
```

**C++ Example**

```
IOhioSession* pIOhiosession;
VARIANT_BOOL bCanChangeSceen;
.
.
.
HRESULT hr = pIOhioSession->
    get_CanChangeScreen(&bCanChangeSceen);
if (FAILED(hr)) {...} //error
if (bCanChangeSceen)
    printf("Can change screen.\n");
else
    printf("Cannot change screen.\n");
//set to false
bCanChangeSceen = FALSE;
pIOhioSession->put_CanChangeScreen
    (bCanChangeScreen);
```

## ConfigurationResource

### Property, IOhioSession

This property returns a vendor-specific string (Hummingbird profile) used to indicate configuration information.

#### Basic Syntax

```
String = OhioSession.ConfigurationResource
```

#### C++ Syntax

```
HRESULT IOhioSession::get_ConfigurationResource([out, retval] BSTR *  
pVal);
```

#### Parameters

*pVal*—The returned vendor-specific string.

#### Basic Example

```
Dim OSesession As OhioSession  
Dim strConfigRes As String  
strConfigRes=OSession.ConfigurationResource()
```

#### C++ Example

```
IOhioSession* pIOhioSession;  
. . .  
BSTR bstrConfigurationResource;  
HRESULT hr = pIOhioSession->  
    get_ConfigurationResource  
    (&bstrConfigurationResource);  
if (FAILED(hr)) {...} //error  
SysFreeString(bstrSessName);
```

## Screen

### Property, IOhioSession

This property returns the OhioScreen object for the session.

#### Basic Syntax

```
OhioScreen = OhioSession.Screen
```

#### C++ Syntax

```
HRESULT IOhioSession::get_Screen([out, retval] IOhioScreen ** pVal);
```

#### Parameters

*pVal*—The returned address of the IOhioScreen object.

#### Basic Example

```
Dim OSesession As OhioSession  
Dim OScreen As OhioScreen  
. . .  
Set OScreen = OSesession.Screen
```

**C++ Example**

```
IOhioScreen* pIOhioScreen=NULL;
IOhioPosition* pIOhioPosition=NULL;
. . .
BSTR bstrWaitFor;
bstrWaitFor = SysAllocString
    (OLESTR("login"));
HRESULT hr = pIOhioSession->get_Screen
    (&pIOhioScreen);
if (SUCCEEDED(hr))
{
    hr = pIOhioScreen (pIOhioScreen->
        WaitForString(bstrWaitFor,24,1,2,20,
        TRUE, &pIOhioPosition);
    if (FAILED(hr)) {...} //error
}
SysFreeString(bstrWaitFor);
```

## SessionName Property, IOhioSession

This property returns a unique name associated with an OhioSession object.

**Basic Syntax**

```
String = OhioSession.SessionName
```

**C++ Syntax**

```
HRESULT IOhioSession::get_SessionName([out, retval] BSTR * pVal);
```

**Parameters**

*pVal*—The returned name of the IOhioSession object.

**Basic Example**

```
Dim OSess As OhioSession
Dim strSessName As String
. . .
strSessName = OSess.SessionName
```

**C++ Example**

```
IOhioSession* pIOhioSession;
. . .
BSTR bstrSessName;
HRESULT hr = pIOhioSession->
    get_SessionName(&bstrSessName);
    if (FAILED(hr)) {...} //error
USES_CONVERSION;
char* szSessName = OLE2A(bstrSessName);
printf("Session Name is %s\n",szSessName);
SysFreeString(bstrWaitFor);
```

## SessionType

### Property, IOhioSession

This property returns the session type for the OhioSession object.

<b>Basic Syntax</b>	<code>OHIO_TYPE = OhioSession.SessionType</code>
<b>C++ Syntax</b>	<code>HRESULT IOhioSession::get_SessionType([out, retval] OHIO_TYPE * pVal);</code>
<b>Parameters</b>	<i>pVal</i> —The returned session type for the IOhioSession object.
<b>Basic Example</b>	<pre>Dim OSession As OhioSession Dim OTy whole="pe As OHIO_TYPE . . OType = OSession.SessionType If (OType = OHIO_TYPE_VT) Then . . Else . . End If</pre>
<b>C++ Example</b>	<pre>IOhioSession* pIOhioSession; OHIO_TYPE pSessionType; . . HRESULT hr = pIOhioSession-&gt;     get_SessionType(&amp;pSessionType);     if (FAILED(hr)) {...} //error if (pSessionType==OHIO_TYPE_VT) . . else . .</pre>

## OhioScreen Interface

The OhioScreen interface is the host's virtual screen. It contains all the characters and attributes that would be seen on a traditional emulator screen. This interface is the primary object for text-based interactions with the host.

The interface provides methods such as manipulating text, searching the screen, sending keystrokes to the host, and handling the cursor. It lets you request an object that contains a collection of fields. Specifically, it can return the OIA (Operator Information Area) object.

**Note:** You can obtain an OhioScreen object from the Screen property of an instance of OhioSession.

The data on the screen is maintained in a series of planes, which can be accessed by various methods within the OhioScreen interface. Most of the methods in this interface work with the text plane, which contains the actual characters in the presentation place. The remaining planes (color, field, and extended) contain the corresponding attributes for each character in the text plane.

## Methods

The OhioScreen interface consists of the following methods:

- FindString
- OnCursorMoved
- OnSizeChanged
- SendKeys
- WaitForString
- GetData
- OnScreenChanged
- PutString
- WaitForInput
- WaitIdle

## Properties

The OhioScreen interface consists of the following properties:

- Columns
- Fields
- Rows
- Cursor
- OIA
- String

## FindString

### Method, IOhioScreen

This method searches the text plane for the target string. If it finds the string, the method returns an OhioPosition object containing the target location. If it does not find the string, the method returns a null. The target string must be completely contained in the target area for the search to be successful.

#### Basic Syntax

```
OhioScreen.FindString(inText As String, inStart As IOhioPosition, inLen As Long, inDir As OHIO_DIRECTION, inIgnoreCase As Boolean) As OhioPosition
```

#### C++ Syntax

```
HRESULT IOhioScreen::FindString([in] BSTR inText, [in] IOhioPosition * inStart, [in] long inLen, [in] OHIO_DIRECTION inDir, [in] VARIANT_BOOL inIgnoreCase, [out, retval] IOhioPosition ** outPos);
```

#### Parameters

*inText*—The target string.

*inStart*—The row and column where the search is to start. The position is inclusive.

*inLen*—The length from the starting position to include in the search.

*inDir*—An OHIO\_DIRECTION data type value.

*inIgnoreCase*—A value of VARIANT\_TRUE indicates that the search ignores case sensitivity. A value of VARIANT\_FALSE indicates that the search is case-sensitive.

*outPos*—The returned address of the IOhioPosition object.

#### Basic Example

```
Dim OSession As OhioSession
Dim OScreen As OhioScreen
Dim ODir As OHIO_DIRECTION
Dim OPos, OPosStart As OhioPosition
Dim iLen As Long
...
ODir = OHIO_DIRECTION_FORWARD
ILen = 100

Set OPosStart = OSession.CreateOhioPosition(0, 0)
Set OPos = OScreen.FindString("login", OPosStart, 100, ODir, True)

If (OPos Is Nothing) Then
    'String not found
Else
    'Found String
End If
```

**C++ Example**

```
IOhioSession* pIOhioSession;
IOhioScreen* pIOhioScreen=NULL;
IOhioPosition* pIOhioPosition=NULL;
...
BSTR bstrWaitFor;
bstrWaitFor = SysAllocString(OLESTR("login"));
pIOhioSession->get_Screen(&pIOhioScreen);
HRESULT hr = pIOhioSession->
    CreateOhioPosition
    (20,1,&pIOhioPosition);
if (FAILED(hr)) {...} //error
pIOhioScreen->FindString(bstrWaitFor
    ,pIOhioPosition, 1000,
    OHIO_DIRECTION_FORWARD, TRUE,
    &pIOhioPosition);

if (pIOhioPosition==NULL)
    //String not found
else
    ...
SysFreeString(bstrWaitFor);
```

**GetData****Method, IOhioScreen**

This method returns a byte array from the text, color, field, or extended plane of the virtual screen by specifying the starting and ending position.

**Basic Syntax**

```
OhioScreen.GetData(inStart As IOhioPosition, inEnd As IOhioPosition,
inPlane As OHIO_PLANE) As VARIANT
```

**C++ Syntax**

```
HRESULT IOhioScreen::GetData([in] IOhioPosition * inStart, [in]
IOhioPosition * inEnd, [in] OHIO_PLANE inPlane, [out, retval] VARIANT *  
outData);
```

**Parameters**

*inStart*—The row and column where the search is to begin. The position is inclusive.  
*inEnd*—The row and column where the search is to end. The position is inclusive.  
*inPlane*—The type of plane of the virtual screen.  
*outData*—The data contained in the plane.

**Basic Example**

```
Dim OSession As OhioSession
Dim OScreen As OhioScreen
Dim OPosEnd As OhioPosition
Dim strPlane As String
Dim ByteArray As Variant
.
.
.
Set OPosStart = OSession.CreateOhioPosition(0, 0)
Set OPosEnd = OSession.CreateOhioPosition(24, 1)
ByteArray = OScreen.GetData(OPosStart, OPosEnd, OHIO_PLANE_TEXT)
```

**C++ Example**

```
IOhioSession* pIOhioSession;
IOhioScreen* pIOhioscreen;
IOhioPosition* pOhioPosStart, pOhioPosEnd;
VARIANT ByteArray;
.
.
.
pIOhioSession->CreateOhioPosition(0,0,&pOhioPosStart);
pIOhioSession->CreateOhioPosition(24,1,
    &pOhioPosEnd);
pIOhioScreen->GetData(pOhioPosStart,
    pOhioPosEnd, OHIO_PLANE_TEXT,
    &ByteArray);
```

## OnCursorMoved

### Method, IOhioScreen

This method is called when the cursor moves.

**Basic Syntax**

```
OhioScreen.OnCursorMoved(inNewPosition As IOhioPosition)
```

**C++ Syntax**

```
HRESULT IOhioScreen::OnCursorMoved([in] IOhioPosition ** inNewPosition);
```

**Parameters**

*inNewPosition*—The address of the new cursor position.

**Basic Example**

```
Dim WithEvents oIOhioScr As
    HEOHIOLib.OhioScreen
Dim OManager As OhioManager
Dim OSession As OhioSession
.....
Private Sub Form_Load()
    strProfile = "C:\\\\Aix.hep"
    strSessionName = "Aix"
    Set OManager = CreateObject
        ("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession
        (strProfile, strSessionName)
    Set oIOhioScr = OSession.Screen
End Sub
.....
Private Sub oIOhioScr_OnCursorMoved
    (ByVal inNewPosition As
        HEOHIOLib.IOhioPosition)
    Debug.Print(vbCrLf & "OnCursorMoved
    called")
    Debug.Print(vbCrLf & "inUpdate: " &
    inUpdate)
    Debug.Print(vbCrLf & "newPos (rc): " &
    inNewPosition.Row & ", " &
    inNewPosition.Column)
End Sub
```

## OnScreenChanged

### Method, IOhioScreen

This method is called when the screen changes.

**Basic Syntax**

```
OhioScreen.OnScreenChanged(inUpdate As OHIO_UPDATE, inStart As
IOhioPosition, inEnd As IOhioPosition)
```

**C++ Syntax**

```
HRESULT IOhioScreen::OnScreenChanged([in] OHIO_UPDATE inUpdate,
[in] IOhioPosition * inStart, [in] IOhioPosition * inEnd);
```

**Parameters**

*inUpdate*—Whether you or the host changed the screen.

*inStart*—The starting position of the location where the screen changed.

*inEnd*—The ending position of the location where the screen changed.

**Basic Example**

```
Dim WithEvents oIOhioScr As
    HEOHIOLib.OhioScreen
Dim OManager As OhioManager
Dim OSession As OhioSession
.
.
.
Private Sub Form_Load()
    strProfile = "C:\\\\Aix.hep"
    strSessionName = "aix"
    Set OManager = CreateObject
        ("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession
        (strProfile, strSessionName)
    Set oIOhioScr = OSession.Screen
End Sub
.
.
.
Private Sub
    oIOhioScr_OnScreenChanged(ByVal
        inUpdate As HEOHIOLib.OHIO_UPDATE,
        ByVal inStart As
            HEOHIOLib.IOhioPosition, ByVal
            inEnd As HEOHIOLib.IOhioPosition)
    Debug.Pring (vbCrLf &
        "OnScreenChanged called")
    Debug.Pring (vbCrLf & vbCrLf & "
        inUpdate: " & inUpdate)
    Debug.Pring (vbCrLf & vbCrLf & "
        instart (rc): " & inStart.Row & ", " &
        inStart.Column)
    Debug.Pring (vbCrLf & vbCrLf & "
        inEnd (rc): " & inEnd.Row & ", " &
        inEnd.Column)
End Sub
```

## OnSizeChanged

### Method, IOhioScreen

This method is called when the size of the screen changes.

**Basic Syntax**      `OhioScreen.OnSizeChanged(inNewRows As Long, inNewColumns As Long)`

**C++ Syntax**      `HRESULT IOhioScreen::OnSizeChanged([in] long inNewRows, [in] long inNewColumns);`

**Parameters**      *inNewRows*—The number of rows in the changed screen.

*inNewColumns*—The number of columns in the changed screen.

**Basic Example**

```
Dim WithEvents oIOhioScr As
    HEOHIOLib.OhioScreen
Dim OManager As OhioManager
Dim OSession As OhioSession
.....
Private Sub Form_Load()
    strProfile = "C:\\\\Aix.hep"
    strSessionName = "aix"
    Set OManager = CreateObject
        ("HEOhio.OhioManager")
    Set OSession = OManager.OpenSession
        (strProfile, strSessionName)
    Set oIOhioScr = OSession.Screen
End Sub
.....
Private Sub oIOhioScr_OnSizeChanged(ByVal
    inNewRows As Long, ByVal inNewColumns As
    Long)
    Debug.Print (vbCrLf & " OnSizeChanged
    called")
    Debug.Print (vbCrLf & vbCrLf &
        inNewRows: " & inUpdate)
    Debug.Print (vbCrLf & vbCrLf &
        inNewColumns: " & inUpdate)
End Sub
```

## PutString

### Method, IOhioScreen

This method sends a string to the virtual screen at the specified location. The string displays only in unprotected fields; any parts of the string in protected fields are discarded.

<b>Basic Syntax</b>	OhioScreen.PutString( <i>inText</i> As String, <i>inStart</i> As IOhioPosition)
<b>C++ Syntax</b>	HRESULT IOhioScreen::PutString([in] BSTR <i>inText</i> , [in] IOhioPosition * <i>inStart</i> );
<b>Parameters</b>	<i>inText</i> —The string to place in the virtual screen. <i>inStart</i> —The starting position of the string.
<b>Basic Example</b>	Dim OScreen As OhioScreen Dim OPos As OhioPosition . . . 'get current cursor location Set OPos = OScreen.Cursor 'send to current cursor location OScreen.PutString "jack123", OPos 'OR 'Easier way to send to current cursor 'location OScreen.PutString "jack123", Nothing
<b>C++ Example</b>	//Send string to specific cursor location IOhioSession* pIOhioSession; IOhioScreen* pIOhioScreen; IOhioPosition* pIOhioPos; . . . HRESULT hr = pIOhioSession-> CreateOhioPosition(0, 0, &pIOhioPos); if (FAILED(hr)) {...} //error  BSTR bstrText; bstrText=SysAllocString(OLESTR("jack123")); pIOhioScreen->PutString(bstrText, pIOhioPos);  //Send string to current cursor location pIOhioScreen->PutString(bstrText,NULL);  SysFreeString(bstrText);

## SendKeys

### Method, IOhioScreen

This method sends a string of keys to the virtual screen as if keystrokes were being typed from the keyboard. The keystrokes are sent to the location you specify. If you do not provide a location, the keystrokes are sent to the current cursor location.

<b>Basic Syntax</b>	OhioScreen. <b>SendKeys</b> ( <i>inText</i> As String, <i>inPos</i> As IOhioPosition)
<b>C++ Syntax</b>	HRESULT IOhioScreen:: <b>SendKeys</b> ([in] BSTR <i>inText</i> , [in] IOhioPosition * <i>inPos</i> );
<b>Parameters</b>	<i>inText</i> —The text that is typed from the keyboard. <i>inPos</i> —The position where the keystrokes will appear.
<b>Basic Example</b>	Dim OScreen As OhioScreen Dim OPos As OhioPosition . . . 'get current cursor location Set OPos = OScreen.Cursor OScreen.SendKeys "ls -lrt[enter]", OPos if (FAILED(hr)) {...} //error 'Tip -When Nothing is the second 'parameter, by default the keys are sent 'to the current cursor location OScreen.SendKeys "ls -lrt[enter]", Nothing 'send the [pf1] key Oscreen.SendKeys [pf1], Nothing

**C++ Example**

```
IOhioScreen* pIOhioScreen
IOhioSession* pIOhioSession
IOhioPosition* pOhioPos;
. .
//if the second parameter is NULL, the
//keys will be sent to the current cursor
//location
BSTR bstrText, bstrEnter;
bstrText=SysAllocString(OLESTR("tester"));
bstrEnter=SysAllocString(OLESTR("[enter]"));
HRESULT hr = pIOhioScreen->SendKeys
    (bstrText,NULL);
if (FAILED(hr)) {...} //error
hr = pIOhioScreen->SendKeys
    (bstrEnter,NULL);
if (FAILED(hr)) {...} //error
//send both strings together
bstrText= SysAllocString(OLESTR("tester[enter]"));
pIOhioScreen->SendKeys(bstrText,NULL);

//send to a cursor location different from
//the current location
pIOhioSession->CreateOhioPosition(0, 0, &pOhioPos);
hr = pIOhioScreen->SendKeys
    (bstrText,pOhioPos);
if (FAILED(hr)) {...} //error

SysFreeString(bstrText);
SysFreeString(bstrEnter);
```

## ***Mnemonic Keywords***

In the method OhioScreen.SendKeys, HostExplorer sends a string of keys so that it appears on the virtual screen. You can use mnemonic keywords to send special functions or AID keys.

The following rules apply to mnemonic keywords:

- Enclose keywords in square brackets ([ ]).
- Keywords are not case-sensitive. For example, [Attn] is identical to [attn].
- Square brackets that should appear as literal values need to be escaped by doubling them (for example, ]). The first bracket escapes the second.

Examples of strings for the SendKeys method are as follows:

String	Description
abc [pf1]	Sends the characters a-b-c.
[backspace] [[x]]	Sends the 3270 backspace key followed by three "[x]" characters.
xyz[[[CLEAR]]	Sends the characters x-y-z- [ followed by the 3270 Clear AID key.

The following list contains mnemonic keywords that are valid for the method OhioScreen.SendKeys:

Mnemonic	3270 Function Description
[attn]	Attention AID key
[clear]	Clear AID key
[cursorselect]	Cursor Select AID key
[enter]	Enter AID key
[sysreq]	System Request
[pa1]	Program Attention 1 AID key
[pa2]	Program Attention 2 AID key
[pa3]	Program Attention 3 AID key
[pf1]	Program Function 1 AID key
[pf2]	Program Function 2 AID key
[pf3]	Program Function 3 AID key
[pf4]	Program Function 4 AID key
[pf5]	Program Function 5 AID key
[pf6]	Program Function 6 AID key
[pf7]	Program Function 7 AID key
[pf8]	Program Function 8 AID key

Mnemonic	3270 Function Description
[pf9]	Program Function 9 AID key
[pf10]	Program Function 10 AID key
[pf11]	Program Function 11 AID key
[pf12]	Program Function 12 AID key
[pf13]	Program Function 13 AID key
[pf14]	Program Function 14 AID key
[pf15]	Program Function 15 AID key
[pf16]	Program Function 16 AID key
[pf17]	Program Function 17 AID key
[pf18]	Program Function 18 AID key
[pf19]	Program Function 19 AID key
[pf20]	Program Function 20 AID key
[pf21]	Program Function 21 AID key
[pf22]	Program Function 22 AID key
[pf23]	Program Function 23 AID key
[pf24]	Program Function 24 AID key
[tab]	Tab forward to next unprotected field
[backtab]	Tab backward to previous unprotected field
[up]	Cursor up
[down]	Cursor down
[right]	Cursor right
[left]	Cursor left
[fastup]	Cursor up two rows
[fastdown]	Cursor down two rows
[fastright]	Cursor right two positions
[fastleft]	Cursor left two positions

Mnemonic	3270 Function Description
[home]	Home
[newline]	New line—move to first unprotected field on next or subsequent line
[reset]	Reset—clear keyboard lock and clear insert mode
[insert]	Insert mode—turns on insert mode for all subsequent keystrokes until [reset]
[backspace]	Destructive Backspace—move cursor one position left, delete character, shift remainder of field one position left
[delete]	Delete at cursor, shift remainder of field one position left
[eraseinput]	Erase input—clear all unprotected fields
[eraseof]	Erase from cursor to end of field, inclusive
[dup]	Duplicate
[fieldmark]	Field Mark

## **WaitForInput** **Method, IOhioScreen**

This method allows HostExplorer to wait for the keyboard to be unlocked.

### **Basic Syntax**

OhioScreen.**WaitForInput**(*inTimeOut* As Long)

### **C++ Syntax**

```
HRESULT IOhioScreen::WaitForInput(long inTimeOut, IOhioPosition **  
outPos);
```

### **Parameters**

*inTimeOut*—The specified expiry time (in seconds).

*outPos*—The returned value, indicating the current cursor position of the session. If the time expires, the value returned is null.

**Basic Example**

```
Dim OScreen As OhioScreen
Dim OPos As OhioPosition
Dim iTimeOut As Long
.
.
iTimeOut = 10
Set OPos = OScreen.WaitForInput(iTimeOut)

If (OPos Is Nothing) Then
    Debug.Print "null value returned"
End If
```

## **WaitForString** **Method, IOhioScreen**

This method allows HostExplorer to wait for the specified string to appear on the screen.

**Basic Syntax**

```
OhioScreen.WaitForString(inString As String, inRow As Integer, inCol As
Integer, inFlag As Long, inTimeOut As Long, inIgnoreCase As Boolean) As
OhioPosition
```

**C++ Syntax**

```
HRESULT IOhioScreen::WaitForString(BSTR inString, short inRow, short
inCol, long inFlag, long inTimeOut, VARIANT_BOOL inIgnoreCase,
IOhioPosition * * outPos);
```

**Parameters**

*inString*—The specified string.

*inRow*—The row that contains the specified string. The *inRow* parameter can have the following values:

- >0—Search the row with the parameter value.
- 0—Search the entire screen.
- -1—Search the row containing the cursor.

*inCol*—The column that contains the specified string.

*inFlag*—The *inFlag* parameter can have the following values:

- 1—Search using the *inRow* parameter only.
- 2—Search starting from the *inRow* parameter value.

*inTimeOut*—The specified expiry time (in seconds).

*inIgnoreCase*—A value of VARIANT\_TRUE indicates that the search is not case-sensitive. A value of VARIANT\_FALSE indicates that the search is case-sensitive.

*outPos*—The returned value, indicating the position of the string. If the time has expired, or if no string is found, the value returned is null.

**Basic Example**

```
Dim OPos As OhioPosition
Dim iRow, iCol, iFlag As Integer
Dim iTimeOut As Long
Dim bIgnoreCase As Boolean
Dim searchStr As Str
.
.
.
iRow = 24
iCol = 1
iFlag = 2
iTimeOut = 100
bIgnoreCase = True
searchStr = "Password"
Set OPos = OScreen.WaitForString
    (searchStr, iRow, iCol, iFlag, iTimeOut,
    bIgnoreCase)

If (OPos Is Nothing) Then
    'String not found
Else
    'String found
End If
```

**C++ Example**

```
IOhioSession* pIOhioSession;
IOhioScreen* pIOhioScreen;
IOhioPosition* pIOhioPosition=NULL;
. . .
BSTR bstrWaitFor;
bstrWaitFor = SysAllocString
    (OLESTR("login"));
HRESULT hr = pIOhioSession->get_Screen
    (&pIOhioScreen);
if (FAILED(hr)) {...} //error
pIOhioScreen->WaitForString(bstrWaitFor,24,
    1,2,20,TRUE,&pIOhioPosition);

if (pIOhioPosition==NULL)
    //String not found
else
    //String found
SysFreeString(bstrWaitFor);
```

**WaitIdle****Method, IOhioScreen**

This method allows the session to idle for a specified number of seconds.

**Basic Syntax**

```
OhioScreen.WaitIdle(inIdleTime As Long)
```

**C++ Syntax**

```
HRESULT IOhioScreen::WaitIdle(long inIdleTime);
```

**Parameters**

*inIdleTime*—The number of milliseconds that the session is idle.

**Basic Example**

```
Dim iTimeOut As Long
Dim OScreen As OhioScreen
. . .
iTimeOut = 10
OScreen.WaitIdle(iTimeOut)
```

## Columns

### Property, IOhioScreen

This property returns the number of columns in the virtual screen.

#### Basic Syntax

```
Long = OhioScreen.Columns
```

#### C++ Syntax

```
HRESULT IOhioScreen::get_Columns([out, retval] long * pVal);
```

#### Parameters

*pVal*—The returned number of columns in the screen.

#### Basic Example

```
Dim OScreen As OhioScreen
Dim nCol As Long
.
.
nCol = OScreen.Columns
strText ="number of columns is " & nCol
```

#### C++ Example

```
IOhioScreen* pIOhioScreen
.
.
long numCols;
HRESULT hr = pIOhioScreen->get_Columns
    (&numCols);
if SUCCEEDED(hr)
    printf("number of cols is %d",numCols);
```

## Cursor

### Property, IOhioScreen

This property returns or sets the location of the cursor in the virtual screen. The OhioPosition object contains the row and column of the cursor.

#### Basic Syntax

```
OhioPosition = OhioScreen.Cursor
```

```
OhioScreen.Cursor = IOhioPosition
```

#### C++ Syntax

```
HRESULT IOhioScreen::get_Cursor([out, retval] IOhioPosition ** pVal);
```

```
HRESULT IOhioScreen::put_Cursor([in] IOhioPosition * newVal);
```

#### Parameters

*pVal*—The returned address of the cursor position.

*newVal*—The address of the cursor position that you set.

**Basic Example**

```
'Get the current cursor position
Dim OScreen As OhioScreen
Dim OSession As OhioSession
Dim OPos As OhioPosition
.
.
.
Set OPos = OScreen.Cursor
'Set the current cursor position
Set OPos =
    OSession.CreateOhioPosition(0, 0)
OScreen.Cursor = OPos
```

**C++ Example**

```
//get current cursor position
IOhioSession* pIOhioSession;
IOhioScreen* pIOhioScreen;
IOhioPosition* pOhioPos;
.
.
.
pIOhioScreen->get_Cursor(&pOhioPos);
//set current cursor position
pIOhioSession->CreateOhioPosition(0, 0,
    &pOhioPos);
pIOhioScreen->put_Cursor(pOhioPos);
```

## Fields

### Property, **IOhioScreen**

This property returns the **OhioFields** object associated with the virtual screen, providing another way to access the data on the screen. The **OhioFields** object contains a static view of all the fields in the current virtual screen. Fields provide methods for interpreting the data in the non-text planes.

**Note:** Zero-length fields (caused by adjacent field attributes) are not returned in the **OhioFields** collection. For unformatted screens, the returned collection contains only one **OhioField** object, which contains the whole virtual screen.

**Basic Syntax**

```
OhioFields = OhioScreen.Fields
```

**C++ Syntax**

```
HRESULT IOhioScreen::get_Fields([out, retval] IOhioFields ** pVal);
```

**Parameters**

*pVal*—The returned address of the **IOhioFields** object.

**Basic Example**

```
Dim OScreen As OhioScreen
Dim OFields As OhioFields
Dim OField As OhioField
Dim iFieldCount As Integer
.
.
Set OFields = OScreen.Fields
iFCOUNT = 0
For Each OField In OFields
    iFieldCount = iFieldCount + 1
Next
strText = strText & "Number of Fields is " & str(iFieldCount)
```

**C++ Example**

```
long lIndex = 1;
long lLen;
IOhioScreen* pIOhioScreen;
IOhioFields* pIOhioFields;
IOhioField* pIOhioField;
.
.
HRESULT hr = pIOhioScreen->get_Fields
    (&pIOhioFields);
if (FAILED(hr)) {...} //error
pIOhioFields->get_Item
    (lIndex,&pIOhioField);
if (pIOhioField != NULL)
{
    pIOhioField->get_Length(&lLen);
    printf("length of field is %d\n",lLen);
}
else
    //error
```

## OIA

### Property, IOhioScreen

This property returns the OhioOIA object associated with the virtual screen. You can use this object to query the status of the Operator Information Area (OIA).

**Basic Syntax**

```
OhioOIA = OhioScreen.OIA
```

**C++ Syntax**

```
HRESULT IOhioScreen::get_OIA([out, retval] IOhioOIA ** pVal);
```

**Parameters**

*pVal*—The returned address of the IOhioOIA object.

**Basic Example**

```
Dim OIA As OhioOIA
Dim bAlphaNumeric As Boolean
.
.
Set OIA = OScreen.OIA
bAlphaNumeric = OIA.AlphaNumeric
If (bAlphaNumeric = True) Then
    'cursor is in an alphanumeric field
Else
    'cursor is in a non-alphanumeric field
End If
```

**C++ Example**

```
IOhioOIA* pIOhioOIA;
VARIANT_BOOL bAlphanumeric;
.
.
HRESULT hr = pIOhioScreen->get_OIA
    (&pIOhioOIA);
if (FAILED(hr)) {...} //error
pIOhioOIA->
    get_Alphanumeric(&bAlphanumeric);
if (bAlphanumeric)
    //cursor is in an alphanumeric field
else
    //cursor is in a non-alphanumeric field
```

## Rows

### Property, IOhioScreen

This property returns the number of rows in the virtual screen.

**Basic Syntax**

```
Long = OhioScreen.Rows
```

**C++ Syntax**

```
HRESULT IOhioScreen::get_Rows([out, retval] long * pVal);
```

**Parameters**

*pVal*—The returned number of rows in the screen.

**Basic Example**

```
Dim OScreen As OhioScreen
Dim nRows As Long
.
.
nRows = OScreen.Rows
strText = strText & "Rows =" & nRows
```

**C++ Example**

```
long lRows;
IOhioScreen* pIOhioScreen;
. . .
HRESULT hr = pIOhioScreen->get_Rows
(&lRows);
if (FAILED(hr)) {...} //error
printf("number of rows: %d\n",lRows);
```

**String****Property, IOhioScreen**

This property returns the entire text plane of the virtual screen as a string. This property returns all null characters and field attribute characters as blank space characters.

**Basic Syntax**

```
String = OhioScreen.String
```

**C++ Syntax**

```
HRESULT IOhioScreen::get_String([out, retval] BSTR * pVal);
```

**Parameters**

*pVal*—The returned text plane.

**Basic Example**

```
Dim OScreen As OhioScreen
Dim strDisplayScreen As String
. . .
StrDisplayScreen = OScreen.String
```

**C++ Example**

```
IOhioScreen* pIOhioScreen;
BSTR bstrString;
. . .
HRESULT hr = pIOhioScreen->get_String
(&bstrString);
if (FAILED(hr)) {...} //error
USES_CONVERSION
printf("text plane: %s\n",OLE2A
(bstrString));
SysFreeString(bstrString);
```

## OhioOIA Interface

The OhioOIA interface returns the Operator Information Area (OIA) object of a host session. The OhioOIA object contains the information displayed at the bottom of the screen, which provides the user with the session name, IP address, column and row numbers, and cursor position.

### Properties

The OhioOIA interface consists of the following properties:

- Alphanumeric
- CommCheckCode
- InsertMode
- Numeric
- ProgCheckCode
- APL
- InputInhibited
- MachineCheckCode
- Owner

### Alphanumeric Property, IOhioOIA

This property indicates whether the cursor is in an alphanumeric field.

#### Basic Syntax

```
Boolean = OhioOIA.AlphaNumeric
```

#### C++ Syntax

```
HRESULT IOhioOIA::get_Alphanumeric([out, retval] VARIANT_BOOL * pVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field that contains the cursor is an alphanumeric field. A returned value of VARIANT\_FALSE indicates that the field that contains the cursor is not an alphanumeric field.

#### Basic Example

```
Dim OScreen As OhioScreen
Dim OIA As OhioOIA
Dim bAlphaNumeric As Boolean
.
.
Set OIA = OScreen.OIA
bAlphaNumeric = OIA.AlphaNumeric
If (bAlphaNumeric = True) Then
    'cursor is in an alphanumeric field
Else
    'cursor is in a non-alphanumeric field
End If
```

## API

### Property, IOhioOIA

This property indicates whether the session is in A Program Language (APL) input mode.

#### Basic Syntax

```
Boolean = OhioOIA.APL
```

#### C++ Syntax

```
HRESULT IOhioOIA::get_APL([out, retval] VARIANT_BOOL * pVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the session is in APL input mode. A returned value of VARIANT\_FALSE indicates that the session is not in APL input mode.

#### Basic Example

```
Dim OScreen As OhioScreen
Dim OIA As OhioOIA
.
.
Set OIA = OScreen.OIA
Dim bAPL As Boolean
bAPL = OIA.APL
If (bAPL = True) Then
    'in APL input mode
Else
    'not in APL input mode
End If
```

#### C++ Example

```
IOhioScreen* pIOhioScreen;
IOhioOIA* pIOhioOIA;
VARIANT_BOOL bAPL;
.
.
pIOhioScreen->get_OIA(&pIOhioOIA);
pIOhioOIA->get_APL(&bAPL);
if (bAPL)
    //in APL input mode
else
    //not in APL input mode
```

## CommCheckCode

### Property, IOhioOIA

This property returns the communication check code if the InputInhibited property returns OHIO\_INPUTINHIBITED\_COMMCHECK.

#### Basic Syntax

```
Long = OhioOIA.CommCheckCode
```

#### C++ Syntax

```
HRESULT IOhioOIA::get_CommCheckCode([out, retval] long * pVal);
```

#### Parameters

*pVal*—The returned communication check code.

#### Basic Example

```
Dim OIA As OhioOIA
Dim InputInhibited As OHIO_INPUTINHIBITED
Dim ChkCode As Long
.
.
.
InputInhibited = OIA.InputInhibited
If (InputInhibited =
    OHIO_INPUTINHIBITED_COMMCHECK) Then
    ChkCode = OIA.CommCheckCode
.
.
.
End If
```

#### C++ Example

```
IOhioOIA* pIOhioOIA;
long lCommCheckCode;
OHIO_INPUTINHIBITED InputInhibited;
.
.
.
pIOhioOIA->get_InputInhibited
    (&InputInhibited);
if(InputInhibited == OHIO_INPUTINHIBITED_COMMCHECK)
{
    pIOhioOIA->get_CommCheckCode
    (&lCommCheckCode);
.
.
.
}
```

## InputInhibited Property, IOhioOIA

This property indicates whether input is inhibited. If input is inhibited, the system prohibits SendKeys or SendAid methods to the OhioScreen interface. The returned value can indicate the reason that input is inhibited. If input is inhibited for more than one reason, the highest value is returned.

### Basic Syntax

```
OHIO_INPUTINHIBITED = OhioOIA.InputInhibited
```

### C++ Syntax

```
HRESULT IOhioOIA::get_InputInhibited([out, retval] OHIO_INPUTINHIBITED *  
pVal);
```

### Parameters

*pVal*—The returned value, indicating whether input is inhibited.

### Basic Example

```
Dim OIA As OhioOIA  
Dim ChkCode As Long  
Dim InputInhibited As OHIO_INPUTINHIBITED  
.  
.  
.  
InputInhibited = OIA.InputInhibited  
If (InputInhibited = OHIO_INPUTINHIBITED_COMMCHECK) Then  
    'input inhibited  
    ChkCode = OIA.CommCheckCode  
    ...  
End If
```

### C++ Example

```
IOhioOIA* pIOhioOIA;  
long lChkCode;  
OHIO_INPUTINHIBITED InputInhib;  
.  
. .  
pIOhioOIA->get_InputInhibited  
    (&InputInhib);  
if (InputInhib ==  
    OHIO_INPUTINHIBITED_COMMCHECK)  
{  
    pIOhioOIA->get_CommCheckCode  
    (&lChkCode);  
    . . .  
}
```

## InsertMode

### Property, IOhioOIA

This property indicates if the session is in Insert mode.

#### Basic Syntax

```
Boolean = OhioOIA.InsertMode
```

#### C++ Syntax

```
HRESULT IOhioOIA::get_InsertMode([out, retval] VARIANT_BOOL * pVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the session is in Insert mode. A returned value of VARIANT\_FALSE indicates that the session is not in Insert mode.

#### Basic Example

```
Dim OIA As OhioOIA
Dim bInsertMode As Boolean
.
.
.
bInsertMode = OIA.InsertMode
If (bInsertMode = True) Then
    'session in insert mode
Else
    'session not in insert mode
End If
```

#### C++ Example

```
IOhioScreen* pIOhioScreen;
IOhioOIA* pIOhioOIA;
VARIANT_BOOL bInsert;
.
.
.
pIOhioScreen->get_OIA(&pIOhioOIA);
pIOhioOIA->get_InsertMode(&bInsert);
if (bInsert)
    //session in insert mode
else
    //session not in insert mode
```

## MachineCheckCode

### Property, IOhioOIA

This property returns the machine check code if the InputInhibited property returns OHIO\_INPUTINHIBITED\_MACHINECHECK.

#### Basic Syntax

```
Long = OhioOIA.MachineCheckCode
```

#### C++ Syntax

```
HRESULT IOhioOIA::get_MachineCheckCode([out, retval] long * pVal);
```

**Parameters**

*pVal*—The returned machine check code.

**Basic Example**

```
Dim OIA As OhioOIA
Dim InputInhibited As OHIO_INPUTINHIBITED
Dim ChkCode As Long
.
.
.
InputInhibited = OIA.InputInhibited
If (InputInhibited = OHIO_INPUTINHIBITED_MACHINECHECK) Then
    ChkCode = OIA.MachineCheckCode
.
.
.
End If
```

**C++ Example**

```
IOhioOIA* pIOhioOIA;
long lChkCode;
OHIO_INPUTINHIBITED Inhib;
.
.
.
pIOhioOIA->get_InputInhibited(&Inhib);
if
    (Inhib==OHIO_INPUTINHIBITED_MACHINECHECK)
{
    pIOhioOIA->get_MachineCheckCode
    (&lChkCode);
.
.
.
}
```

## Numeric

### Property, IOhioOIA

This property indicates whether the field that contains the cursor is a numeric-only field.

**Basic Syntax**

```
Boolean = OhioOIA.Numeric
```

**C++ Syntax**

```
HRESULT IOhioOIA::get_Numeric([out, retval] VARIANT_BOOL * pVal);
```

**Parameters**

*pVal*—A returned value of VARIANT\_TRUE indicates that the field that contains the cursor is a numeric-only field. A returned value of VARIANT\_FALSE indicates that the field is not a numeric-only field.

**Basic Example**

```
Dim OIA As OhioOIA
Dim bNumeric As Boolean
.
.
.
bNumeric = OIA.Numeric
If (bNumeric = True) Then
    'cursor is in a numeric-only field
End If
```

**C++ Example**

```
IOhioScreen* pIOhioScreen;
IOhioOIA* pIOhioOIA;
VARIANT_BOOL bNumeric;
.
.
.
HRESULT hr = pIOhioScreen->get_OIA
    (&pIOhioOIA);
if (FAILED(hr)) {...} //error
pIOhioOIA->get_Numeric(&bNumeric);
if (bNumeric)
    //cursor is in a numeric-only field
else
    //cursor is not in a numeric-only field
```

## Owner Property, IOhioOIA

This property specifies the owner of the host connection.

**Basic Syntax**

```
OHIO_OWNER = OhioOIA.Owner
```

**C++ Syntax**

```
HRESULT IOhioOIA::get_Owner([out, retval] OHIO_OWNER * pVal);
```

**Parameters**

*pVal*—The returned owner of the host connection.

**Basic Example**

```
Dim OIA As OhioOIA
Dim Owner As OHIO_OWNER
.
.
.
Owner = OIA.Owner
If (Owner = OHIO_OWNER_UNKNOWN) Then
    'Add code
Else
    'Add code
End If
```

**C++ Example**

```
IOhioOIA* pIOhioOIA;
OHIO_OWNER OwnerOIA;
...
HRESULT hr = pIOhioOIA->get_Owner
    (&OwnerOIA);
if (FAILED(hr)) {...} //error
if (OwnerOIA==OHIO_OWNER_UNKNOWN)
    //Add code
```

## ProgCheckCode

### Property, IOhioOIA

This property returns the program check code if the InputInhibited property returns OHIO\_INPUTINHIBITED\_PROGCHECK.

**Basic Syntax**

```
Long = OhioOIA.ProgCheckCode
```

**C++ Syntax**

```
HRESULT IOhioOIA::get_ProgCheckCode([out, retval] long * pVal);
```

**Parameters**

*pVal*—The returned program check code.

**Basic Example**

```
Dim OIA As OhioOIA
Dim InputInhibited As OHIO_INPUTINHIBITED
Dim ChkCode As Long
...
InputInhibited = OIA.InputInhibited
If (InputInhibited = OHIO_INPUTINHIBITED_PROGCHECK) Then
    ChkCode = OIA.ProgCheckCode
Else
    'handle other codes
End If
```

**C++ Example**

```
IOhioOIA* pIOhioOIA;
long lChkCode;
OHIO_INPUTINHIBITED InputInhibited;
...
HRESULT hr = pIOhioOIA->get_InputInhibited
    (&InputInhibited);
if (FAILED(hr)) {...} //error
if (InputInhibited ==
    OHIO_INPUTINHIBITED_PROGCHECK)
    pIOhioOIA->get_ProgCheckCode
        (&lChkCode);
```

## OhioFields Interface

The OhioFields interface contains a collection of the fields in the virtual screen. Each element of the collection is an instance of OhioField. Through this interface, you can iterate through the fields and find fields based on location and string.

You can access the OhioFields interface only through the OhioScreen interface using the Fields property.

**Note:** OhioFields is a static view of the virtual screen. It does not reflect your changes until you update the field list with a new view of the virtual screen using the Refresh method.

The OhioFields interface returns the number of fields in the screen.

### Methods

The OhioFields interface consists of the following methods:

- FindByPosition
- FindByString
- Refresh

### Properties

The OhioFields interface consists of the following properties:

- Count
- Item

## FindByPosition

### Method, IOhioFields

This method searches the collection for the requested position and returns the OhioField object containing that position. If the position is not found, the method returns a null.

#### Basic Syntax

```
OhioFields.FindByPosition(inPosition As IOhioPosition) As OhioField
```

#### C++ Syntax

```
HRESULT IOhioFields::FindByPosition([in] IOhioPosition * inPosition, [out, retval] IOhioField ** outField);
```

#### Parameters

*inPosition*—The target row and column.

*outField*—The returned address of the IOhioField object.

#### Basic Example

```
Dim OSession As OhioSession
Dim OFields As OhioFields
Dim OField As OhioField
.
.
Set OFields = OScreen.Fields
Set OPos = OSession.CreateOhioPosition(15, 15)
Set OField = OFields.FindByPosition(OPos)
```

#### C++ Example

```
long lLen;
IOhioSession* pIOhioSession;
IOhioPosition* inPos;
.
.
pIOhioSession->CreateOhioPosition
    (24,1,&inPos);
HRESULT hr = pIOhioFields->FindByPosition
    (inPos, &pIOhioField);
if (FAILED(hr)) {...} //error
pIOhioField->get_Length(&lLen);
```

## FindByString

### Method, IOhioFields

This method searches the collection for the requested string and returns the OhioField object containing that string. To be considered a match, the string must be contained completely within the field. If the string is not found, the method returns a null.

You can also specify case-sensitivity and whether the search is performed backward or forward.

#### Basic Syntax

```
OhioFields.FindByString(inString As String, inStart As IOhioPosition,  
inLen As Long, inIgnoreCase As Boolean) As OhioField
```

#### C++ Syntax

```
HRESULT IOhioFields::FindByString([in] BSTR inString, [in] IOhioPosition *  
inStart, [in] long inLen, [in] OHIO_DIRECTION inDir, [in] VARIANT_BOOL  
inIgnoreCase, [out, retval] IOhioField ** outField
```

#### Parameters

*inString*—The target string.

*inStart*—The row and column where the search is to start.

*inLen*—The length, from the starting position, to include in the search.

*nDir*—The OHIO\_DIRECTION data type value.

*inIgnoreCase*—Whether the search is case-sensitive. VARIANT\_TRUE indicates that case will be ignored. VARIANT\_FALSE indicates that the search will be case-sensitive.

*outField*—The returned address of the OhioField object.

#### Basic Example

```
Dim OSession As OhioSession  
Dim OFields As OhioFields  
Dim OField As OhioField  
.  
Set OFields = OScreen.Fields  
Set OPos = OSession.CreateOhioPosition(1, 1)  
Set OField = OFields.FindByString("Next", OPos, 2000, OHIO_DIRECTION_FORWARD, True)  
If (OField Is Nothing) Then  
    'the string is not found  
Else  
    'string found  
End If
```

**C++ Example**

```
IOhioSession* pIOhioSession;
IOhioFields* pIOhioFields;
long lLen;
BSTR bstrSearch;
IOhioPosition* inPos;
. .
bstrSearch=SysAllocString(OLESTR("test"));
pIOhioSession->CreateOhioPosition
(24,1,&inPos);
HRESULT hr = pIOhioFields->FindByString
(bstrSearch,inPos,400,
OHIO_DIRECTION_FORWARD,TRUE,&pIOhioField);
if (FAILED(hr)) {...} //error
pIOhioField->get_Length(&lLen);
SysFreeString(bstrSearch);
```

## Refresh

### Method, IOhioFields

This method updates the collection of OhioField objects, adding all OhioField objects in the current virtual screen to the collection. This method does not preserve indexing of OhioField objects.

**Basic Syntax**

```
OhioFields.Refresh
```

**C++ Syntax**

```
HRESULT IOhioFields::Refresh();
```

**Parameters**

This method has no parameters.

**Basic Example**

```
Dim OSession As OhioSession
Dim OFields As OhioFields
Dim OField As OhioField
. .
Set OPos = OSession.CreateOhioPosition(1, 1)
'update OFields collection before
'accessing it
OFIELDS.Refresh
Set OField = OFIELDS.FindByPosition(OPos)
```

**C++ Example**

```
IOhioSession* pIOhioSession;
IOhioFields* pIOhioFields;
IOhioField* pIOhioField;
long lLen;
BSTR bstrSearch;
IOhioPosition* inPos;
. . .
bstrSearch=SysAllocString(OLESTR("test"));
pIOhioSession->CreateOhioPosition(24, 1,
    &inPos);
pIOhioFields->Refresh();
HRESULT hr = pIOhioFields->FindByString
    (bstrSearch,inPos,400,
    OHIO_DIRECTION_FORWARD, TRUE,
    &pIOhioField);
if (FAILED(hr)) {...} //error
pIOhioField->get_Length(&lLen);
SysFreeString(bstrSearch);
```

## Count

### Property, IOhioFields

This property returns the number of OhioField objects contained in the OhioFields collection.

**Basic Syntax**

```
Long = OhioFields.Count
```

**C++ Syntax**

```
HRESULT IOhioFields::get_Count([out, retval] long * pVal);
```

**Parameters**

*pVal*—The returned number of IOhioField objects.

**Basic Example**

```
Dim OFields As OhioFields
Dim nCount As Integer
. . .
nCount = OFields.Count
strText = "Number of fields is " & nCount
```

**C++ Example**

```
IOhioScreen* pIOhioScreen;
long lcount;
IOhioFields* pIOhioFields;
. . .
HRESULT hr = pIOhioScreen->get_Fields
    (&pIOhioFields);
if (FAILED(hr)) {...} //error
pIOhioFields->get_Count(&lcount);
```

**Item****Property, IOhioFields**

This property returns the OhioField object at the specified index (one-based indexing).

**Basic Syntax**

```
Long = OhioFields.Item
OhioFields.Item = Long
```

**C++ Syntax**

```
HRESULT IOhioFields::get_Item([in] long inIndexOrKey, [out, retval]
IOhioField ** pVal);
```

**Parameters**

*inIndexOrKey*—The index of the returned IOhioField object.  
*pVal*—The returned address of the IOhioField object.

**Basic Example**

```
Dim OField As OhioField
Dim strFieldText As String
. . .
Set OField = OFields.Item(4)
'retrieve text from field number 4
strFieldText = OField.String
```

**C++ Example**

```
long lIndex = 1;
long lLen=0;
IOhioFields* pIOhioFields;
IOhioField* pIOhioField;
. . .
HRESULT hr = pIOhioScreen->get_Fields
    (&pIOhioFields);
if (FAILED(hr)) {...} //error
pIOhioFields->get_Item
    (lIndex,&pIOhioField);
pIOhioField->get_Length(&lLen);
```

## OhioField Interface

The OhioField interface is a virtual-screen field that includes both data and attributes describing the field. The interface provides methods for accessing and manipulating field attributes and data.

**Note:** For VT terminals, the interface returns the entire screen because the VT terminal does not have fields.

You can access OhioField methods and properties only through the OhioFields interface.

### Methods

The OhioField interface consists of the following method:

GetData

### Properties

The OhioField interface consists of the following properties:

- Attribute
- HighIntensity
- Modified
- Numeric
- Protected
- String
- End
- Length
- Normal
- PenSelectable
- Start

## GetData

### Method, IOhioField

This method lets you request different types of data, such as text, color, and field.

#### Basic Syntax

```
OhioField.GetData(inPlane As OHIO_PLANE) As Variant
```

#### C++ Syntax

```
HRESULT IOhioField::GetData([in] OHIO_PLANE inPlane, [out, retval] VARIANT* outData);
```

#### Parameters

*inPlane*—The type of plane.

*outData*—The returned data contained in the plane.

#### Basic Example

```
Dim OField As OhioField  
Dim aByteArray As Variant  
. . .  
ByteArray = OField.GetData  
(OHIO_PLANE_TEXT)
```

#### C++ Example

```
IOhioField* pIOhioField;  
VARIANT ByteArray;  
OHIO_PLANE OhioPlane;  
OhioPlane = OHIO_PLANE_TEXT;  
pIOhioField->GetData(OhioPlane,  
&ByteArray);
```

## Attribute

### Property, IOhioField

This property returns the attribute byte for the field.

#### Basic Syntax

```
OHIO_FIELD = OhioField.Attribute
```

#### C++ Syntax

```
HRESULT IOhioField::get_Attribute([out, retval] OHIO_FIELD * pVal);
```

#### Parameters

*pVal*—The returned attribute byte for the field.

#### Basic Example

```
Dim OField As OhioField
Dim OAtt As OHIO_FIELD
OAtt = OField.Attribute
Dim lResult As Long

'check if an attribute is enabled with a
'bitwise And
lResult = OAtt And
    OHIO_FIELD_PEN_SELECTABLE
If (lResult = 0) Then
    'Attribute not enabled
Else
    'Attribute enabled
End If

'Another way of doing the same thing
lResult = OAtt And
    OHIO_FIELD_PEN_SELECTABLE
If (lResult = OHIO_FIELD_PEN_SELECTABLE) Then
    'Attribute enabled
Else
    'Attribute not enabled
End If
```

**C++ Example**

```
IOhioField* pIOhioField;
long lResult;
OHIO_FIELD OAtt;
. . .
HRESULT hr = pIOhioField->
    get_Attribute(&OAtt);
if (SUCCEEDED(hr))
{
    lResult = OAtt &&
        OHIO_FIELD_PEN_SELECTABLE;

    if (lResult ==
        OHIO_FIELD_PEN_SELECTABLE)
        //Attribute enabled
    else
        //Attribute not enabled

    //Another way of doing the same thing
    lResult = OAtt &&
        OHIO_FIELD_PEN_SELECTABLE;
    if (lResult == 0)
        //Attribute not enabled
    else
        //Attribute enabled
}
```

**End****Property, IOhioField**

This property returns the ending position of the field, which is the last character in the field. The position can range from 1 to the size of the virtual screen.

**Basic Syntax**

```
OhioPosition = OhioField.End
```

**C++ Syntax**

```
HRESULT IOhioField::get_End([out, retval] IOhioPosition ** pVal);
```

**Parameters**

*pVal*—The returned value of the ending position of the field.

**Basic Example**

```
Dim OField As OhioField
Dim OPos As OhioPosition
Dim row, col As Long
.
.
Set OPos = OField.End
row = OPos.row
col = OPos.Column
```

**C++ Example**

```
IOhioField* pIOhioField;
IOhioPosition* PosEnd;
long row,col;
.
.
HRESULT hr = pIOhioField->get_End
    (&PosEnd);
if (FAILED(hr)) {...} //error
PosEnd->get_Row(&row);
PosEnd->get_Column(&col);
```

## HighIntensity Property, IOhioField

This property indicates whether the field is high-intensity.

### Basic Syntax

```
Boolean = OhioField.HighIntensity
```

### C++ Syntax

```
HRESULT IOhioField::get_HighIntensity([out, retval] VARIANT_BOOL * pVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field is high-intensity. A returned value of VARIANT\_FALSE indicates that the field is not high-intensity.

### Basic Example

```
Dim OField As OhioField
Dim bHighIntensity As Boolean
.
.
.
bHighIntensity = OField.HighIntensity
If (bHighIntensity = True) Then
    'field is high-intensity
Else
    'field is not high-intensity
End If
```

### C++ Example

```
IOhioField* pIOhioField;
VARIANT_BOOL bHighIntensity;
.
.
.
HRESULT hr=pIOhioField->get_HighIntensity
    (&bHighIntensity);
if (FAILED(hr)) {...} //error
if (bHighIntensity)
    //field is high-intensity
else
    //field is not high-intensity
```

## Length

### Property, IOhioField

This property returns the length of the field. The length of a field can range from 1 to the size of the virtual screen.

#### Basic Syntax

```
Long = OhioField.Length
```

#### C++ Syntax

```
HRESULT IOhioField::get_Length([out, retval] long * pVal);
```

#### Parameters

*pVal*—The returned value of the length of the field.

#### Basic Example

```
Dim OField As OhioField  
Dim Length As Long  
. . .  
Length = OField.Length
```

#### C++ Example

```
IOhioField* pIOhioField;  
Long lLen;  
. . .  
HRESULT hr = pIOhioField->get_Length  
(&lLen);  
if (FAILED(hr)) {...} //error
```

## Modified Property, IOhioField

This property indicates whether the field has been modified.

### Basic Syntax

```
Boolean = OhioField.Modified
```

### C++ Syntax

```
HRESULT IOhioField::get_Modified([out, retval] VARIANT_BOOL * pVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field has been modified.  
A returned value of VARIANT\_FALSE indicates that the field has not been modified.

### Basic Example

```
Dim OField As OhioField
Dim bModified As Boolean
.
.
.
bModified = OField.Modified
If (bModified = True) Then
    'field has been modified
Else
    'field not modified
End If
```

### C++ Example

```
IOhioField* pIOhioField;
VARIANT_BOOL bModified;
.
.
.
HRESULT hr=pIOhioField->get_HighIntensity
(&bModified);
if (FAILED(hr)) {...} //error
if (bModified)
    //field has been modified
else
    //field not modified
```

## Normal

### Property, IOhioField

This property indicates whether the field is normal (that is, not protected and not high-intensity).

#### Basic Syntax

```
Boolean = OhioField.Normal
```

#### C++ Syntax

```
HRESULT IOhioField::get_Normal([out, retval] VARIANT_BOOL * pVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field is normal. A returned value of VARIANT\_FALSE indicates that the field is not normal.

#### Basic Example

```
Dim OField As OhioField
Dim bNormal As Boolean
.
.
.
bNormal = OField.Normal
If (bNormal = True) Then
    'field is normal
Else
    'field is not normal
End If
```

#### C++ Example

```
IOhioField* pIOhioField;
VARIANT_BOOL bNormal;
.
.
.
HRESULT hr = pIOhioField->get_Normal
    (&bNormal);
if (FAILED(hr)) {...} //error
if (bNormal)
    //field is normal
else
    //field is not normal
```

## Numeric

### Property, IOhioField

This property indicates whether the field is numeric-only.

#### Basic Syntax

```
Boolean = OhioField.Numeric
```

#### C++ Syntax

```
HRESULT IOhioField::get_Numeric([out, retval] VARIANT_BOOL * pVal);
```

#### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field is numeric-only. A returned value of VARIANT\_FALSE indicates that the field is not numeric-only.

#### Basic Example

```
Dim OField As OhioField
Dim bNumeric As Boolean
.
.
.
bNumeric = OField.Numeric
If (bNumeric = True) Then
    'field is numeric-only
Else
    'field is not numeric-only
End If
```

#### C++ Example

```
IOhioField* pIOhioField;
VARIANT_BOOL bNumeric;
.
.
.
HRESULT hr = pIOhioField->get_Numeric
    (&bNumeric);
if (FAILED(hr)) {...} //error
if (bNumeric)
    //field is numeric-only
else
    //field is not numeric-onlly
```

## PenSelectable Property, IOhioField

This property indicates whether the field is pen-selectable.

### Basic Syntax

```
Boolean = OhioField.PenSelectable
```

### C++ Syntax

```
HRESULT IOhioField::get_PenSelectable([out, retval] VARIANT_BOOL * pVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field is pen-selectable. A returned value of VARIANT\_FALSE indicates that the field is not pen-selectable.

### Basic Example

```
Dim OField As OhioField
Dim bPenSelectable As Boolean
.
.
.
bPenSelectable = OField.PenSelectable
If (bPenSelectable = True) Then
    'field is pen-selectable
Else
    'field is not pen-selectable
End If
```

### C++ Example

```
IOhioField* pIOhioField;
VARIANT_BOOL bPenSelectable;
.
.
.
HRESULT hr = pIOhioField->get_Numeric
    (&bPenSelectable);
if (FAILED(hr)) {...} //error
if (bPenSelectable)
    //field is pen-selectable
else
    //field is not pen-selectable
```

## Protected Property, IOhioField

This property indicates whether the field is protected.

### Basic Syntax

```
Boolean = OhioField.Protected
```

### C++ Syntax

```
HRESULT IOhioField::get_Protected([out, retval] VARIANT_BOOL * pVal);
```

### Parameters

*pVal*—A returned value of VARIANT\_TRUE indicates that the field is protected. A returned value of VARIANT\_FALSE indicates that the field is not protected.

### Basic Example

```
Dim OField As OhioField
Dim bProtected As Boolean
.
.
.
bProtected = OField.Protected
If (bProtected = True) Then
    'Field is protected
Else
    'Field is unprotected
End If
```

### C++ Example

```
IOhioField* pIOhioField;
VARIANT_BOOL bProtected;
.
.
.
HRESULT hr = pIOhioField->get_Numeric
    (&bProtected);
if (FAILED(hr)) {...} //error
if (bProtected)
    //Field is protected
else
    //Field is unprotected
```

## Start

### Property, IOhioField

This property returns the starting position of the field, which is the first character of the field. The position can range from 1 to the size of the virtual screen.

#### Basic Syntax

```
OhioPosition = OhioField.Start
```

#### C++ Syntax

```
HRESULT IOhioField::get_Start([out, retval] IOhioPosition ** pVal);
```

#### Parameters

*pVal*—The returned starting position of the field.

#### Basic Example

```
Dim OField As OhioField
Dim OPos As OhioPosition
Dim row, col As Long
.
.
Set OPos = OField.Start
row = OPos.row
col = OPos.Column
```

#### C++ Example

```
IOhioField* pIOhioField;
IOhioPosition* PosStart;
long row,col;
.
.
HRESULT hr = pIOhioField->get_Start
(&PosStart);
if (FAILED(hr)) {...} //error
PosStart->get_Row(&row);
PosStart->get_Column(&col);
```

## String

### Property, IOhioField

This property returns or sets the text-plane data for the field as a string. If you set the property with a value shorter than the field, the system clears the rest of the field. If you set the property with a value longer than the field, the text is shortened. To view the changed text, you need to refresh the OhioFields collection.

#### Basic Syntax

```
String = OhioField.String
```

```
OhioField.String = String
```

#### C++ Syntax

```
HRESULT IOhioField::get_String([out, retval] BSTR * pVal);
```

```
HRESULT OhioField.put_String([in] BSTR newVal);
```

#### Parameters

*pVal*—The returned text-plane data for the field.

*newVal*—The data to appear in the text plane.

#### Basic Example

```
Dim OField As OhioField
Dim OFields As OhioFields
Dim strFieldText As String
.
.
Set OField = OFields(4)
strFieldText = OField.String
```

**C++ Example**

```
IOhioField* pIOhioField;
//get string
BSTR bstrString;
HRESULT hr = pIOhioField->get_String
    (&bstrString);
if (FAILED(hr)) {...} //error
printf("field-get_string is: %s\n",OLE2A(bstrString));
SysFreeString(bstrString);

//put string
VARIANT_BOOL bProtected;
hr = pIOhioField->get_Protected
    (&bProtected);
if (FAILED(hr)) {...} //error
if (bProtected)
    //Add code
else
{
    BSTR inBstr;
    inBstr=SysAllocString(OLESTR
        ("sample text"));
    hr = pIOhioField->put_String(inBstr);
    if (FAILED(hr)) {...} //error
    SysFreeString(inBstr);
}
```

## OhioPosition Interface

The OhioPosition interface provides the row and column coordinates of the cursor position. You can create an OhioPosition by using the CreateOhioPosition method. For more information, see “CreateOhioPosition” on page 1354.

The interface is used by the following sub-interfaces:

- OhioScreen Interface
- OhioFields Interface
- OhioField Interface

### Methods

The OhioPosition interface consists of the following method:

CreateOhioPosition

### Properties

The OhioPosition interface consists of the following properties:

- Column
- Row

## CreateOhioPosition

### Method, IOhioPosition

This method allows you to create an OhioPosition object that can be passed to all interfaces. The object consists of a row and column coordinate.

#### Basic Syntax

```
OhioPosition.CreateOhioPosition(inRow As Long, inCol As Long) As  
OhioPosition
```

#### C++ Syntax

```
HRESULT IOhioPosition::CreateOhioPosition ([in] long inRow, [in] long  
inCol, [out, retval] IOhioPosition ** outPosition);
```

#### Parameters

*inRow*—The row coordinate.

*inCol*—The column coordinate.

*outPosition*—The address of the new OhioPosition object.

#### Basic Example

```
Dim OSession As OhioSession  
Dim OFields As OhioFields  
Dim OField As OhioField  
. . .  
Set OFields = OScreen.Fields  
Set OPos = OSession.CreateOhioPosition(1, 1)  
Set OField = OFields.FindByPosition(OPos)
```

#### C++ Example

```
IOhioSessions* pIOhioSessions;  
IOhioFields* pIOhioFields;  
IOhioField* pIOhioField;  
IOhioPosition* inPos;  
. . .  
pIOhioSession->CreateOhioPosition  
    (24,1,&inPos);  
HRESULT hr = pIOhioFields->FindByPosition  
    (inPos, &pIOhioField);  
if (FAILED(hr)) {...} //error
```

## Column

### Property, IOhioPosition

This property returns or sets the column coordinate of the OhioPosition object.

#### Basic Syntax

```
Long = OhioPosition.Column
```

```
OhioPosition.Column = Long
```

#### C++ Syntax

```
HRESULT IOhioPosition::get_Column([out, retval] long * pVal);  
HRESULT IOhioPosition::put_Column([in] long newVal);
```

#### Parameters

*pVal*—The returned column coordinate.

*newVal*—The value of the column coordinate.

#### Basic Example

```
Dim OSession As OhioSession  
Dim OPos As OhioPosition  
Dim iCol As Long  
. . .  
Set OPos = OSession.CreateOhioPosition(1, 1)  
'set column  
OPos.Column = 5  
'get column  
iCol = OPos.Column
```

#### C++ Example

```
long outCol, inCol;  
IOhioSession* pIOhioSession;  
IOhioPosition* OPos;  
. . .  
pIOhioSession->CreateOhioPosition  
    (5,8,&OPos);  
//get the current Column  
OPos->get_Column(&outCol);  
//set the Column to another value  
inCol = 14;  
OPos->put_Column(inCol);
```

## Row

### Property, IOhioPosition

This property returns or sets the row coordinate of the OhioPosition object.

#### Basic Syntax

```
Long = OhioPosition.Row  
OhioPosition.Row = Long
```

#### C++ Syntax

```
HRESULT IOhioPosition::get_Row([out, retval] long * pVal);  
HRESULT IOhioPosition::put_Row([in] long newVal);
```

#### Parameters

*pVal*—The returned row coordinate.

*newVal*—The value of the row coordinate.

#### Basic Example

```
Dim OSession As OhioSession  
Dim OPos As OhioPosition  
Dim iRow As Long  
. . .  
Set OPos =  
    OSession.CreateOhioPosition(1, 1)  
'set row  
OPos.row = 5  
'get row  
iRow = OPos.Row
```

#### C++ Example

```
IOhioSession* pIOhioSession;  
long outRow, inRow;  
IOhioPosition* OPos;  
. . .  
pIOhioSession->CreateOhioPosition  
    (5,8,&OPos);  
//get the current row value  
OPos->get_Row(&outRow);  
//set the row value  
inRow = 14;  
OPos->put_Row(inRow);
```

## Data Types of OHIO

Ohio contains the following read-only data types:

### **Ohio interface**

OHIO\_DIRECTION Data Type

### **OhioSession interface**

- OHIO\_STATE Data Type
- OHIO\_TYPE Data Type

### **OhioScreen interface**

- OHIO\_COLOR Data Type
- OHIO\_EXTENDED Data Type
- OHIO\_FIELD Data Type
- OHIO\_PLANE Data Type
- OHIO\_UPDATE Data Type

### **OhioOIA interface**

- OHIO\_INPUTINHIBITED Data Type
- OHIO\_OWNER Data Type

## ***OHIO\_COLOR Data Type***

The OHIO\_COLOR data type specifies the color of the text in the entire field.

It has the following values:

<b>Value</b>	<b>Description</b>
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.

## ***OHIO\_DIRECTION Data Type***

The OHIO\_DIRECTION data type specifies the direction of the search.

It has the following values:

<b>Value</b>	<b>Description</b>
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).

## ***OHIO\_EXTENDED Data Type***

The OHIO\_EXTENDED data type specifies the extended attribute of the field.

It has the following values:

<b>Value</b>	<b>Description</b>
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.

## ***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:

<b>Value</b>	<b>Description</b>
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.

## ***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.

## ***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.

## ***OHIO\_INPUTINHIBITED Data Type***

The OHIO\_INPUTINHIBITED data type specifies what is inhibiting the input.

It has the following values:

<b>Value</b>	<b>Description</b>
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.

## ***OHIO\_OWNER Data Type***

The OHIO\_OWNER data type specifies the owner of the Ohio session.

It has the following values:

<b>Value</b>	<b>Description</b>
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).

## ***OHIO\_PLANE Data Type***

The OHIO\_PLANE data type specifies the plane from which to retrieve the data.

It has the following values:

<b>Value</b>	<b>Description</b>
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.

## ***OHIO\_STATE Data Type***

The OHIO\_STATE data type specifies the status of the communication link to the host.

It has the following values:

<b>Value</b>	<b>Description</b>
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.

## ***OHIO\_TYPE Data Type***

The OHIO\_TYPE data type specifies the type of host.

It has the following values:

<b>Value</b>	<b>Description</b>
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.

## ***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:

<b>Value</b>	<b>Description</b>
OHIO_UPDATE_HOST	Indicates that the host initiated the update.
OHIO_UPDATE_CLIENT	Indicates that the client initiated the update.

## About Legacy APIs

HostExplorer provides the following existing or “legacy” APIs:

- EHLLAPI (Extended High Level Language Application Programming Interface) and WinHLLAPI (Windows HLLAPI)—Allow other Windows programs (for example, Attachmate® Extra! for Windows) to communicate and control HostExplorer terminal emulators.
- DDE (Dynamic Data Exchange)—A tool that allows programs (for example, Microsoft Excel, Word, and Visual Basic) to communicate with the HostExplorer 3270 emulator.

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

### **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. For more information, see “Window Close (201)” on page 1366.
- an extended ConnectPS (1) function. For more information, see “ConnectPS (1)” on page 1365.

## ConnectPS (1)

HostExplorer supports a modified version of the ConnectPS function, which lets you dynamically create a new session to any host from EHLLAPI/WinHLLAPI. The calling sequence is:

**Function Number**

1

**Data String**

Pointer to the following structure:

```
typedef struct _AUTOSTART_
{
    charcPSID;// Sessions ID = '*' (Asterisk)
    charcModelType;// Emulation Type = '3' for 3270, '5' for 5250, or 'V'
                    // for VT
    charcReserved1;
    charcReserved2;
    charcReserved3;
    characHostName[128];// Name/IP Address of host
} AUTOSTART, FAR *LPAUTOSTART;
```

**Length**

Not applicable.

**PS Position**

Not applicable.

**Return**

0—ConnectPS successful.  
1—Invalid PS short name specified  
9—No sessions active. Emulator not loaded.

**Example**

```
HLLFunc = 1;
HLLDataString[0] = '*';
HLLDataString[1] = '3';
strcpy( &HLLDataString[5], "1.2.3.4" );
HLLAPI(&HLLFunc, HLLDataString, &HLLDataLength, &PsPos );
```

## Window Close (201)

HostExplorer includes a special EHLLAPI/WinHLLAPI function to close a given window, regardless of any of the system flags. You can use Function 201 to close a specific short name window. The calling sequence is:

<b>Function Number</b>	201
<b>Data String</b>	A one-character presentation space name. Short name must be a letter of the alphabet (A--Z) or a number (1 to 5) for dynamic sessions.
<b>Length</b>	Not applicable.
<b>PS Position</b>	Not applicable.
<b>Return</b>	0—Window closed successfully. 1—Invalid PS short name specified. 9—No sessions active. Emulator not loaded.
<b>Example</b>	<pre>HLLFunc = 201; HLLDataString[0] = 'A'; HLLAPI (&amp;HLLFunc, HLLDataString, &amp;HLLDataLength, &amp;PsPos );</pre>

## 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
- Auto Unload
- Return Extra Session Info
- Allow Connect Physical
- Update Screen After Copy
- Start Minimized
- Yield Wait
- Auto Sync
- Convert Nulls

## EHLLAPI Support

### 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.

### 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.Settings` 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 `EHLLAPISStartup` before calling any EHLLAPI function and call `EHLLAPICleanup` as the last call in your program.

The following EHLLAPI calls are available with HostExplorer:

- EHLLAPIConnect
- EHLLAPICreatePosToRowCol
- EHLLAPICreateRowColToPosition
- EHLLAPICopyFieldToString
- EHLLAPICopyOIA
- EHLLAPICopyPS
- EHLLAPICopyPSToString
- EHLLAPICopyStringToField
- EHLLAPICopyStringToPS
- EHLLAPIDisconnect
- EHLLAPIFindFieldPosition
- EHLLAPIGetRowString
- EHLLAPIGetVersion
- EHLLAPIPause
- EHLLAPIQueryCursorLocation
- EHLLAPIQueryFieldAttribute
- EHLLAPIQuerySessions
- EHLLAPIQuerySessionStatus
- EHLLAPIReceiveFile
- EHLLAPIRelease
- EHLLAPIReserve
- EHLLAPIReset
- EHLLAPISearchField
- EHLLAPISearchPS
- EHLLAPISendFile
- EHLLAPISendKey
- EHLLAPISetCursorPosition
- EHLLAPISetSessionParameters
- EHLLAPIWait

## **EHLLAPIConnect**

**3270 5250 VT**

This function is used to download a file from the host system. The valid options are ASCII, CRLF, and APPEND. Separate all options with spaces, not commas.

### **Function**

**EHLLAPIReceiveFile** (*strPCFileName* As String, *idSession* As String, *StrHostFileName* As String, *strOptions* As String) As Integer

### **Input**

*StrPCFileName*—contains the name of the computer file to receive the data.

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.

*StrHostFileName*—contains the name of the HOST file to be downloaded.

*strOptions*—contains the file-transfer options, separated by spaces.

**Example**

The following example transfers the host file PROFILE EXEC to the computer file C:\PROF.AUT.

```
IRc% = EHLLAPIReceiveFile( "C:\PROF.AUT", "A", "PROFILE EXEC",
"ASCII CRLF" )
```

**EHLLAPIConvertPosToRowCol****3270 5250 VT**

This function is used to convert a presentation-space (PS) value to a row-and-column value.

**Function**

**EHLLAPIConvertPositionToRowCol** (*idSession* As String, *iPos* As Integer, *iRow* As Integer, *iColumn* As Integer) As Integer

**Input**

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.  
*iPos*—the PS (1 to screen size) to be converted.

**Output**

*iRow*—the row position.  
*iColumn*—the column position.

**Example**

The following example converts the PS position 1761 on session "A" to a row-and-column value:

```
Rc% = EHLLAPIConvertPositionToRowCol( "A", 1761, iRow%, iCol% )
```

**EHLLAPIConvertRowColToPosition****3270 5250 VT**

This function is used to convert a row-and-column value to a presentation-space (PS) value. Unlike other functions, the return code of this function is the new PS position.

**Function**

**EHLLAPIConvertRowColToPosition** (*idSession* As String, *iRow* As Integer, *iColumn* As Integer) As Integer

**Input**

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.  
*iRow*—the row value between 1 and the maximum number of rows (typically 24).  
*iColumn*—the column value between 1 and the maximum number of columns (typically 80).

**Output**

*iPos*—the PS (1 to screen size) to be converted.

**Example**

The following example converts the row-and-column on session "A" value to a PS position:

```
IPos% = EHLLAPICConvertRowColToPosition( "A", 24, 1 )
```

## **EHLLAPICopyFieldToString**

**3270 5250 VT**

This function copies a field (or portion of a field) to a Visual Basic (VB) string. Attributes are translated to blanks, and no extended attributes are returned.

**Function**

```
EHLLAPICopyFieldToString (strString As String, iMaxLen As Integer, iPos As Integer) As Integer
```

**Input**

*iPos*—the presentation-space (PS) position (1 to the maximum screen size) of the field to copy.

*iMaxLen*—the length of data to copy from *iPos*.

**Output**

*StrString*—the VB string to receive the field content. If the screen is unformatted, the result of this may be up to 3564 bytes.

**Example**

This example copies the contents of the field at position 81 for a maximum length of 80 bytes:

```
iRc% = EHLLAPICopyFieldToString( strDest$, 80, 81 )
```

## **EHLLAPICopyOIA**

**3270 5250 VT**

This function copies the Operator Information Area (OIA) to a string. The contents of this string are in the special EBCDIC format, as described in the *EHLLAPI Programming Guide*.

**Function**

```
EHLLAPICopyOIA (strOIA As String) As Integer
```

**Output**

*StrOIA*—the VB string to receive the OIA string. Length is always 103 bytes.

**Example**

This example copies the current OIA into a string:

```
iRc% = EHLLAPICopyOIA( strOIA$ )
```

**EHLLAPICopyPS****3270 5250 VT**

This function copies the entire presentation space (PS) to a Visual Basic (VB) string. Attributes are translated to blanks, and no extended attributes are returned.

**Function****EHLLAPICopyPS** (*strScreen* As String) As Integer**Output**

*StrScreen*—the VB string to receive the screen image. The screen sizes are Model 2 (24x80) 1920, Model 3 (32x80) 2560, Model 4 (43x80) 3440, and Model 5 (27x132) 3564.

**Example**

This example copies the entire PS to a VB string:

```
iRc% = EHLLAPICopyPS( strPS$ )
```

**EHLLAPICopyPSToString****3270 5250 VT**

This function copies a portion of the presentation space (PS) to a Visual Basic (VB) string. Attributes are translated to blanks, and no extended attributes are returned.

**Function****EHLLAPICopyPSToString** (*strString* As String, *iMaxLen* As Integer, *iPos* As Integer) As Integer**Input**

*iPos*—the PS position (1 to the maximum screen size) to start copying.

*iMaxLen*—the length of data to copy from *iPos*.

**Output**

*strString*—the VB string to receive the partial screen image.

**Example**

This example copies the PS space at starting at position 1700 for 220 bytes:

```
iRc% = EHLLAPICopyPS( strPartial$, 220, 1700 )
```

**EHLLAPICopyStringToField****3270 5250 VT**

This function copies a Visual Basic (VB) string to a 3270 field. Attributes are translated to blanks, and no extended attributes are returned.

**Function****EHLLAPICopyStringToField** (*strString* As String, *iPos* As Integer) As Integer**Input***StrString*—the VB string to copy to the 3270 field.*iPos*—the PS position (1 to the maximum screen size) to which the information is copied.**Example**

This example copies a string to position 1761:

```
iRc% = EHLLAPICopyStringToField( "Hello World", 1761 )
```

**EHLLAPICopyStringToPS****3270 5250 VT**

This function copies a string to the presentation space (PS) at the location specified. Attributes are translated to blanks, and no extended attributes are returned.

**Function****EHLLAPICopyStringToPS** (*strString* As String, *iPos* As Integer) As Integer**Input***StrString*—the Visual Basic (VB) string to copy to the PS.*iPos*—the PS position (1 to the maximum screen size) to copy the info to.**Example**

This example copies a string to position 1761:

```
iRc% = EHLLAPICopyStringToPS( "Hello World", 1761 )
```

**EHLLAPIDisconnect****3270 5250 VT**

This function disconnects the Visual Basic (VB) interface from a session.

**Function****EHLLAPIDisconnect** (*idSession* As String) As Integer**Input***IdSession*—the session short-name identifier. It must be a single, uppercase letter.

**Example**

The following example disconnects from session "A":

```
iRc% = EHLLAPIDisconnect( "A" )
```

## **EHLLAPIFindFieldPosition**

**3270 5250 VT**

This function is used to get the presentation-space (PS) position of the previous or next protected or unprotected field. The search types are defined in the Declarations section of the HLLCALLS.BAS module.

**Function**

**EHLLAPIFindFieldPosition** (*strSearchType* As String, *iPos* As Integer) As Integer

**Input**

*StrSearchType*—the string-search type, which corresponds to the direction and field type.

*iPos*—the PS position (1 to the maximum screen size) to begin the search.

**Output**

*iPos*—the PS position of the field.

**Example**

The following example searches for the next unprotected field starting at position 81:

```
iPos% = 81  
iRc% = EHLLAPIFindFieldPosition( EHLLAPI_NEXTUNPROT, iPos)
```

## **EHLLAPIGetString**

**3270 5250 VT**

This function (which is not a true EHLLAPI call) is used to retrieve the contents of a specific row in the PS. Attributes are translated to blanks and no extended attributes are returned.

**Function**

**EHLLAPIGetString** (*idSession* As String, *strString* As String, *iRow* As Integer) As Integer

**Input**

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.

*iRow*—the row value between 1 and the maximum number of rows.

**Output**

*strString*—the VB string to receive the contents of the row.

**Example**

The following example retrieves the contents of row 23 into a VB string:

```
iRc% = EHLLAPIGetString( "A", strRow$, 23)
```

**EHLLAPIGetVersion****3270 5250 VT**

This function is used to retrieve the version of the EHLLAPI interface.

**Function****EHLLAPIGetVersion (hiVer As Integer, loVer As Integer) As Integer****Output**

*HiVer*—the high-order digit of the version.

*LoVer*—the low-order digit of the version.

**Example**

The following example retrieves the current version of the EHLLAPI interface:

```
iRc% = EHLLAPIGetVersion( hiVer%, loVer% )
```

**EHLLAPIPause****3270 5250 VT**

This function is used to pause for a specific number of half-second increments.

**Function****EHLLAPIPause (iHalfSeconds As Integer) As Integer****Input**

*iHalfSeconds*—the number of half-seconds to wait.

**Example**

The following example waits for 1.5 seconds:

```
iRc% = EHLLAPIPause( 3 )
```

**EHLLAPIQueryCursorPosition****3270 5250 VT**

This function returns the current cursor location.

**Function****EHLLAPIQueryCursorPosition (iPos As Integer) As Integer****Output**

*iPos*—the presentation-space (PS) position of the cursor.

**Example**

The following example retrieves the cursor position:

```
iRc% = EHLLAPIQueryCursorPosition( iPos% )
```

## **EHLLAPIQueryFieldAttribute**

**3270 5250 VT**

This function is used to get the attribute of the field at the location specified.

**Function**

**EHLLAPIQueryFieldAttribute** (*iPos* As Integer, *iAttr* As Integer) As Integer

**Input**

*iPos*—the presentation-space (PS) position to search for the field attribute.

**Output**

*iAttr*—the 3270 attribute returned.

**Example**

The following example retrieves the attribute at position 1920:

```
iRc% = EHLLAPIQueryFieldAttribute( 1920, iAttr% )
```

## **EHLLAPIQuerySessions**

**3270 5250 VT**

This function returns a string containing the short names of all available sessions. For example, if the system returns an "AC" string, sessions "A" and "C" are available.

**Function**

**EHLLAPIQuerySessions** (*strAllSessions* As String) As Integer

**Output**

*StrAllSessions*—the VB string to receive the session short names.

**Example**

The following example retrieves the list of sessions.

```
iRc% = EHLLAPIQuerySessions( strAll$ )
```

## **EHLLAPIQuerySessionStatus**

**3270 5250 VT**

This function returns the session long name and the session screen size.

**Function**

**EHLLAPIQuerySessionStatus** (*idSession* As String, *strLongName* As String, *iRows* As Integer, *iColumns* As Integer) As Integer

**Input**

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.

**Output**

*StrLongName*—contains the session long name (up to 8 characters).

*iRows*—contains the number of rows in the presentation-space (PS).

*iColumns*—contains the number of columns in the PS.

**Example**

The following example retrieves the info for session "A".

```
iRc% = EHLLAPIQuerySessionStatus( "A", strLongName$, iRows%, iCols%)
```

## **EHLLAPIReceiveFile**

**3270 5250 VT**

This function is used to download a file from the host. Valid options are ASCII, CRLF, and APPEND. Separate all options with spaces, not commas.

**Function**

**EHLLAPIReceiveFile** (*strPCFileName* As String, *idSession* As String, *StrHostFileName* As String, *strOptions* As String) As Integer

**Input**

*StrPCFileName*—contains the name of the computer file to receive the data.

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.

*StrHostFileName*—contains the name of the HOST file to be downloaded.

*StrOptions*—contains the file-transfer options, separated by spaces.

**Example**

The following example transfers the host file PROFILE EXEC to the computer file C:\PROF.AUT:

```
iRc% = EHLLAPIReceiveFile( "C:\PROF.AUT", "A", "PROFILE EXEC", "ASCII  
CRLF" )
```

**EHLLAPIRelease****3270 5250 VT**

This function is used to release the 3270 keyboard. Make this call only after a call to EHLLAPISetup.

**Function****EHLLAPIRelease () As Integer****Example**

The following example releases the keyboard:

```
iRc% = EHLLAPIRelease()
```

**EHLLAPISetup****3270 5250 VT**

This function is used to reserve the 3270 keyboard. This prevents the user from accessing the 3270 session from the keyboard.

**Function****EHLLAPISetup () As Integer****Example**

The following example reserves the keyboard:

```
iRc% = EHLLAPISetup()
```

**EHLLAPISet****3270 5250 VT**

This function is used to reset the EHLLAPI interface back to the default values.

**Function****EHLLAPISet () As Integer****Example**

The following example resets the interface:

```
iRc% = EHLLAPISet()
```

**EHLLAPISearchField****3270 5250 VT**

This function is used to search a field for a given string.

**Function****EHLLAPISearchField** (*strString* As String, *iPos* As Integer) As Integer**Input***StrString*—the Visual Basic (VB) string to search.*iPos*—the presentation-space (PS) position to begin the search.**Output***iPos*—the PS position of the text, if found (if iRc = 0).**Example**

The following example searches for the string "More..." in the field, starting at position 1841:

```
iPos% = 1841  
iRc% = EHLLAPISearchField( "More...", iPos% )
```

**EHLLAPISearchPS****3270 5250 VT**

This function is used to search the entire presentation space (PS) for a given string.

**Function****EHLLAPISearchPS** (*strString* As String, *iPos* As Integer) As Integer**Input***StrString*—the Visual Basic (VB) string to search.**Output***iPos*—the PS position of the text, if found (if iRc = 0).**Example**

The following example searches for the string "CP READ":

```
iRc% = EHLLAPISearchField( "More...", iPos% )
```

## **EHLLAPISendFile**

**3270 5250 VT**

This function is used to upload a file to the host. Common valid options are ASCII, CRLF, and APPEND. Other options are specific to the operating system. Separate all options with spaces, not commas.

### **Function**

**EHLLAPISendFile** (*strPCFileName* As String, *strSessionID* As String,  
*StrHostFileName* As String, *strOptions* As String) As Integer

### **Input**

*StrPCFileName*—contains the name of the computer file to upload.

*IdSession*—the session short-name identifier. It must be a single, uppercase letter.

*StrHostFileName*—contains the name of the HOST file to receive the data.

*StrOptions*—contains the file-transfer options, separated by spaces.

### **Example**

The following example transfers the computer file C:\AUTOEXEC.BAT to the host file PROFILE EXEC:

```
iRc% = EHLLAPIReceiveFile( "C:\AUTOEXEC.BAT", "A", "PROFILE EXEC", "ASCII  
CRLF" )
```

## **EHLLAPISendKey**

**3270 5250 VT**

This function is used to send a sequence of keystrokes to the session. Although you can transfer data into the session using the CopyString functions, only this function allows you to press action keys such as Enter and PFxx.

### **Function**

**EHLLAPISendKey** (*strKeyString* As String) As Integer

### **Input**

*StrKeyString*—the VB string that contains the list of keys to press.

### **Example**

The following example types the string "LOGIN PIERRE" and then presses the Enter key:

```
iRc% = EHLLAPISendKey( "LOGIN PIERRE@E" )
```

## ***Creating Special Key Strings***

The following entries assume that the ESC (escape) key is mapped to the '@' symbol.

To use a special key string, insert the @ symbol twice in the string by pressing the ESC key twice.

For example, type:

"PIERRE@@CC0" as "PIERRE@CC0", not "PIERRE", followed by the Clear key.

### **EHLLAPISetCursorPosition**

**3270 5250 VT**

This function is used to set the cursor position.

**Function**

**EHLLAPISetCursorPosition** (*iNewPos* As Integer) As Integer

**Input**

*iPos*—the presentation-space (PS) position of the cursor.

**Example**

The following example sets the cursor position:

```
iRc% = EHLLAPISetCursorPosition( 1761 )
```

### **EHLLAPISetSessionParameters**

**3270 5250 VT**

This function is used to set the current session parameters.

**Function**

**EHLLAPISetSessionParameters** (*strParameters* As String) As Integer

**Input**

*StrParameters*—the Visual Basic (VB) string containing the comma-separated options.

**Example**

The following example translates Extended Attributes (EABs) into CGA colors:

```
iRc% = EHLLAPISetSessionParameters( "XLATE,NOEAB" )
```

## *Valid Options*

### **3270 5250 VT**

You can use the following options when setting current session parameters:

**ATTRB**—Return attributes in Copy functions.

**NOATTRB**—Convert attributes to blanks.

**EAB**—Return extended attributes in functions. Although supported by EHLLAPI, the current VB functions currently disable this option.

**NOEAB**—Do not return extended attributes.

**XLATE**—Translate all unknown EBCDIC data to blanks.

**NOXLATE**—Do not translate any PS data.

**AUTORESET**—Unlock the keyboard before executing any EHLLAPISendKey function.

**NORESET**—Do not unlock the keyboard before executing EHLLAPISendKey.

**SRCHALL**—Search the entire PS space for strings.

**SRCHFROM**—Search from the specified position for strings.

**SRCHFRWD**—Search forward for a string.

**SRCHBKWD**—Search backward for a string.

**FPAUSE**—Insert a full pause for EHLLAPIPause.

**IPAUSE**—Insert an interruptible pause (if the PS is updated) for EHLLAPIPause.

**QUIET**—Perform Quiet mode file transfers.

**NOQUIET**—Perform a normal file transfer (a dialog box appears).

**ESC=x**—Set the EHLLAPISendKey escape character. The default value is @.

**TIMEOUT=x**—Specify the timeout for a file transfer, for example:

- 0—None
- 1—30 seconds
- 2—1 minute
- 3—1.5 minutes
- 4—2 minutes
- 5—2.5 minutes
- 6—3 minutes
- 7—3.5 minutes
- 8—4 minutes
- 9—4.5 minutes
- J—5 minutes
- K—5.5 minutes
- L—6 minutes
- M—6.5 minutes
- N—7 minutes

## **EHLLAPIWait**

### **3270 5250 VT**

This function is used to wait for an event to occur on the host session. The function can check the status, wait for a certain amount of time, or wait indefinitely for the host PS to change. The wait types are defined in the Declarations section of the HLLCALLS.BAS module. Use the EHLLAPISetSessionParameters call to set the timeout used by the EHLLAPI\_TWAIT option.

**Function**

**EHLLAPIWait** (*iWaitType* As Integer) As Integer

**Input**

*iWaitType*—the wait type as defined in the Declarations section of HLLCALLS.BAS.

**Example**

The following example checks if the PS updates:

```
iRc% = EHLLAPIWait( EHLLAPI_NWAIT )
```

**Valid Options**

- EHLLAPI\_NWAIT*—no wait. Check module status and return immediately.
- EHLLAPI\_TWAIT*—timed wait. Wait until the host updates or wait up to 60 seconds, whichever comes first.
- EHLLAPI\_LWAIT*—wait until the host system updates (forever).

## 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.

### *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.

<b>Return Code</b>	<b>Description</b>
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.

## ***Auto Start Delay***

### **To add an additional wait command:**

- 1 Launch HostExplorer, then establish a remote host connection.
- 2 Add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:
  - [EHLLAPI.Settings]
  - Auto Start Delay = x
  - This delays the ConnectPS return after the connection is complete.
- 3 Replace x with the number of seconds.

**Note:** x is the number of seconds you want the system to wait. By default, this value is 1.

## ***Start Minimized***

When HLLAPI spawns a new session automatically by starting a profile, that window is kept hidden by the emulator because it is under HLLAPI control.

To force newly spawned sessions in minimized mode (iconized and visible), add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Start Minimized = On
```

## ***Auto Unload***

When you issue a DisconnectPS, HLLAPI terminates that terminal session automatically if the session was spawned by HLLAPI.

To prevent HLLAPI from terminating the session, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Auto Unload = Off
```

## ***Yield Wait***

By default, the functions that require the DLL to wait for some event to complete (such as Wait, Pause, Send File, and Receive File) use a PeekMessage loop in order to let all applications process messages. However, you can set a loop call to yield the wait.

To set a loop call, set the following line in the EHLLAPI.Settings section of the HOSTEX.INI file:

```
[EHLLAPI.Settings]  
Yield Wait = On
```

## ***Return Extra Session Info***

**To set the last byte of the 18-byte structure:**

1 Add one of the following lines to the EHLLAPI.Settings section in the HOSTEX.INI file:

- 'T'—Idle - Configured but not loaded and not connected.
- 'R'—Ready - Session connected to host but not connected to HLLAPI.
- 'C'—Connect - Session connected to host and connected to HLLAPI (Connect PS).

This sets the last byte of the 18-byte structure, normally reserved to a flag providing this information.

2 Add the following lines to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]  
Return Extra Session Info = On  
This enables this extra flag byte.
```

**Note:** By default, the HLLAPI standard interface does not provide any mechanism to know whether a session is: connected (with HLLAPI), loaded and connected to a host but not to HLLAPI, or simply configured and not connected at all.

## ***Auto Sync***

The SendKey function (in its current design) does not allow for automatic pacing when you press AID generating keys. Therefore, if you want to send two sets of strings in a row, such as *XYZ@E*, you must place a WAIT(TWAIT) command between them. You can instruct HLLAPI to wait until the keyboard unlocks before returning from the SendKey function when you press an AID key. This extension provides an automatic synchronization with the host and simplifies your HLLAPI application.

### **To enable Auto Sync:**

- 1 Enable the Type Ahead feature in the profile assigned to the HLLAPI short name. You can set the Type Ahead feature by double-clicking the Session folder and clicking General in the Session Profile dialog box.
- 2 Add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Auto Sync = On
```

## ***Allow Connect Physical***

According to common specifications, the CONPHYS flag that is used in DOS to perform a physical connect (bring window to front) is not supported in EHLLAPI (ACS3EHAP.DLL).

To bring the window to the front, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Allow Connect Physical= On
```

## ***Convert Nulls***

The CopyPS and CopyPSToString functions normally convert 3270/5250 Nulls to ASCII blanks when you copy text.

To prevent HLLAPI from converting nulls, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]  
Convert Nulls = Off
```

**Note:** If you disable the Convert Nulls HLLAPI feature, you must use the STRLEN option to use explicit string lengths or change the EOT character from the default value of Null.

## ***Update Screen After Copy***

This option forces the emulator to repaint the screen when an HLLAPI application copies data to the screen buffer using the CopyString to PS and CopyStringToField functions.

To force the emulator to repaint the screen, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]  
Update Screen After Copy = On
```

**Note:** This function dramatically reduces performance. Enable this option for debugging purposes only.

## **EHLLAPI Support in VT and NVT Modes**

HostExplorer supports the EHLLAPI interface while in VT and NVT (Network Virtual Terminal), or ANSI terminal modes. This allows your application to interact with a front-end system without requiring a new interface to work with the ANSI-terminal portion of the emulator.

While in NVT mode, only a subset of the EHLLAPI functions 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
```

## 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)

## 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)

## 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.

## 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 . For more information, see “Default Locations for User Files” on page 1407.

```
[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)

# Dynamic Data Exchange (DDE)

## 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.

## 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

### ***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}
```

### ***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)]" )
```

## **Poke Message**

You can use a Poke message to send information to the DDE server HOSTEX. The Poke message lets you send information to identify a new cursor position, press keys, and set the search string. Poke does not return any information.

### **Examples**

The following function sets the cursor position to position 1761. The variable ChanNum represents the DDE conversation ID.

#### **Word for Windows**

```
DDEPoke ChanNum, "Cursor", "1761"
```

#### **Microsoft Excel 4.0**

```
=POKE( A1, "Cursor", B1 )
```

#### **Microsoft Excel 5.0 and**

**7.0**

You can poke only data that exists in a cell.  
DDEPoke ChanNum, " Cursor ", Range("a1")

If you want to Poke data in a text string, you must use the v4.0 macros.

## **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" )
```

## ***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.

## ***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.

## ***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. For more information, see “Receive File Command Syntax” on page 1398.

**Send File**—Uploads the specified file to the host from your computer. This requires command syntax. For more information, see “Send File Command Syntax” on page 1398.

**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)]

## *Receive File Command Syntax*

### **Command Syntax**

The command syntax for TSO and MUSIC is:

```
[receive file(PC_filename host_filename transfer_options)]
```

The command syntax for CMS and CICS is:

```
[receive file(PC_filename host_filename ( transfer_options ))]
```

### **Examples**

TSO or MUSIC:

```
[receive file(c:\martin host.martin ASCII CRLF)]
```

CMS:

```
[receive file(c:\vincent.txt vincent text ( ASCII CRLF))]
```

CICS:

```
[receive file(c:\test.txt test ( ASCII CRLF))]
```

## *Send File Command Syntax*

### **Command Syntax**

The command syntax for TSO and MUSIC is:

```
[send file(PC_filename host_filename transfer_options)]
```

The command syntax for CMS and CICS is:

```
[send file(PC_filename host_filename ( transfer_options ) )]
```

### **Examples**

TSO or MUSIC:

```
[send file(c:\martin host.martin ASCII CRLF)]
```

CMS:

```
[send file(c:\vincent.txt vincent text ( ASCII CRLF))]
```

CICS:

```
[send file(c:\test.txt test ( ASCII CRLF))]
```

## Poke Commands

You can send the following items to a Session topic:

**Cursor**—Sets the cursor position to the new value in the presentation space. The syntax for the value field is either nnn or Fnn[U/P]. nnn sets the position to the numeric value specified whereas Fnn[U/P] sets the position to the first position of the specified field. For additional information, see the Cursor example.

**EscChar**—Sets the escape character used for sending keys. The default escape character is '@'.

**Keystroke**—Presses a collection of keystrokes. The string can contain up to 255 characters. The string format for keys is in HLLAPI mnemonic format. This allows you to enter normal text and press 3270 action keys such as Home, Pfx, and Clear. For additional information, see the Keystroke example.

**PS**—Inserts the string into the entire presentation space. In this mode, data over protected fields is ignored. Therefore, you can retrieve the entire PS, update certain portions, 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.

### *Cursor Example*

The following is a Cursor example:

- "1"—sets the cursor to position 1.
- "F2"—sets the cursor to the first position of the second field in the presentation space.
- "F2U"—sets the cursor to the first position of the second unprotected field in the presentation space.
- "F2P"—sets the cursor to the first position of the second protected field in the presentation space.

### *Keystroke Example*

The following is a Keystroke example:

"LOGIN PIERRE@E"

## ***Request Commands***

You can request the following items from a Session topic:

**Columns**—Returns the number of columns in the current presentation space.

**Cursor**—Returns the current cursor location in the presentation space. 1 is the first position.

**Emulator**—Returns the window handle of the window displaying the presentation space.

**File Transfer**— Returns the short name of the presentation space and either 0 if no transfer is occurring or 1 if a transfer is occurring. For additional information, see the File Transfer example.

**Keyboard**—Returns the status of the 3270 keyboard. Valid return values are Clear and Locked.

**Model**—Returns the 3270 model for the presentation space. Valid values are 2, 3, 4, and 5.

**OIA**—Returns the operator information area (OIA) in ASCII format.

**Power**—Always returns “On”.

**Profile Name**—Returns the name of the profile used for the presentation space; for example, DEFAULT.

**Rows**—Returns the number of rows in the current presentation space.

**Search**—Returns the position of the search string in the presentation space (PS). The search string is defined using the POKE command.

**PS**—Returns the entire contents of the PS. The entire space is returned as one string. Nulls are converted to blanks. Therefore, if you are using a Model 2 terminal (24x80), a string of 1920 bytes is returned.

**Pnnnn[F/Lmmm]**—Returns a portion of the PS. Nulls are converted to blanks. For additional information, refer to the Pnnnn[F/Lmmm] example.

**Fnn[U/P]**—Returns the contents of the field specified. Nulls are converted to blanks. For additional information, refer to the Fnn example.

**Rnn**—Returns the contents of the specified row. Nulls are converted to blanks.

**R2**—Returns the contents of the second row in the presentation space. The length is dependent on which 3270 model you are using. Models 2, 3, and 4 return 80 characters, whereas a model 5 returns 132 characters.

### *File Transfer Example*

The following is a File Transfer example:

A 1—Transfer occurring on session A.

### *Pnnnn[F/Lmmm] Examples*

The following list consists of Pnnnn[F/Lmmm] examples:

- P100—Returns the presentation space from position 100 to the end of the field containing position 100.
- P100F—Returns the entire field that contains position 100.
- P100L20—Returns data from position 100 for a length of 20 characters regardless of field positions.

### *Fnn Examples*

The following list consists of Fnn examples:

- F2— Returns the contents of the second field (unprotected or protected) in the presentation space.
- F2U—Returns the contents of the second unprotected field in the presentation space.
- F2P—Returns the contents of the second protected field in the presentation space.

## DDE Terminology

In DDE, the screen is called the presentation space (PS). When copying information to or from the PS, the indices used always begin at 1 and end at the last value of the PS. For example, a 24x80 screen has 1920 addressable positions, which range from 1 to 1920.

Some of the requests return multiple items back to you in a single string. Each item is delimited using the carriage return byte (“\r” in standard C syntax), value 0x0D, or 13 decimal.

DDE lets you start a conversation with:

- HostExplorer
- Word for Windows
- Microsoft Excel

### *Starting a Conversation with HostExplorer*

To start a conversation with HostExplorer, send an INITIATE message with HOSTEX as the application name and a topic that can be one of the session short names or System.

### *Starting a Conversation with Word for Windows*

To start a conversation with Word for Windows, use the following syntax in a macro:

```
iChanNum = DDEInitiate( "HOSTEX", "A" )
```

You can replace the topic "A" with any existing session short name or "System" to access the system topic.

### *Starting a Conversation with Microsoft Excel*

To start a conversation with Microsoft Excel, use the following syntax in a cell:

```
=INITIATE( "HOSTEX", "A" )
```

### ***Creating DDE Keystroke Mnemonics***

The following assumes that the ESC (escape) key is mapped to the @ symbol.

To create a DDE keystroke mnemonic, use the @ symbol. For example:

- To send an Enter command, send the string @E.
- To send a normal @ sign, send two @ symbols—@@.

## 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
    DDETerninate ChanNum
    MessageBox "Logged into CMS successful", "Information"
End Sub
Sub Main
    On Error Goto ErrorHandler
    crlf$ = Chr$(13) + Chr$(10)
    iChanNum = DDEInitiate("HOSTEX", "A")
    iNumRows = Val(DDERequest$(iChanNum, "Rows"))
    iNumCols = Val(DDERequest$(iChanNum, "Columns"))

    If iChanNum Then
        For row = 1 To iNumRows
            request$ = "P" + Mid$(Str$(1 + ((row - 1) * iNumCols)), 2) +
            "L" + LTrim$(Str$(iNumCols))
            Data$ = Data$ + DDERequest$(iChanNum, request$) + crlf$
        Next row
        Insert Data$
    Else 'could not open session
        MsgBox "Could not open DDE Session with program."
    End If
ErrorHandler:
    DDETerninate iChanNum
End Sub
```

## 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).

### ***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".

### ***Start Session Example***

The following is a Start Session example:

```
[Start Session(Profile_Name.Profile_Folder [,IP Host/Gateway  
Name [,IP Port] ])]
```

## *Default Locations for User Files*

Per-user files are all application or service files that, when changed, affect only the user who is making the change (that is, the currently logged in user). An example of a user-specific file is exceed.xcfg. If you configure exceed.xcfg with Xconfig to use a certain display, then other users of the machine are not affected.

**Note:** Each user of the product on the machine receives a personal user directory.

The following are the default locations for user files:

<b>Operating System</b>	<b>Per-User Files—Default Location (Current User)</b>
Windows 98/Me	C:\Windows\Application Data\Hummingbird\Connectivity\version\
Windows 98/Me (user profiles enabled)	C:\Windows\Profiles\user\Application Data\Hummingbird\ Connectivity\version\
Windows NT 4.0	C:\Winnt\Profiles\user\Application Data\Hummingbird\ Connectivity\version\
Windows 2000/XP Windows Server 2003	C:\Documents and Settings\user\Application Data\Hummingbird\ Connectivity\version\

**Note:** This location is usually hidden (by default).

## **Current User versus “All Users”**

Starting with version 8, Hummingbird Connectivity products use individual or personal profiles even when a product is installed for all users of the machine.

For Windows NT/2000/XP/Server 2003 platforms, Hummingbird Setup Wizard prompts you to choose between installing the product on the computer for the currently logged in user, or for all users. For the current user, shortcuts are created in the appropriate user profile folder, along with copies of all other user files. For all users (anyone who uses the computer), shortcuts are created in the “All Users” profile folder, and user files are created for each user of the machine (when they first use the product).

# **Chapter 3**

---

## **Customizing FTP**

## Introducing FTP API

The FTP API is a non-OLE interface that lets you build Hummingbird Basic scripts to perform local and remote disk and directory operations. The FTP API functions correspond to functionality in the 5.3 and earlier versions of FTP. For information on the 6.0 version of FTP API, see Introducing FTP OLE API. If you require more information on the function of a specific FTP API scripting command, refer to the FTP online Help.

**Note:** You should also familiarize yourself with the following reference manuals for HLLAPI from IBM:

- IBM 3270 Personal Computer High Level Language Application Programming Interface
- AIX Version 3.2 High Level Application Programming Interface (HLLAPI) Programming

### FTP script requirements

When creating an FTP script, you must fulfill certain requirements:

- All of the FTP API functions are prototyped in the `ftp rtns.ebh` header file. In order to access the FTP API methods in Hummingbird DLL's, you must include the `ftp rtns.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.

## Customizing FTP

### Initialization and Termination Functions

#### InitFTP Function

This function initializes the environment for the FTP scripting functions. This function must be called before you make any FTP scripting request. Otherwise, the scripting subroutines behave abnormally. A nonzero integer return value specifies a successful initialization, otherwise a zero value is returned.

##### **BASIC Syntax**

```
RetVal = InitFTP()
```

##### **Parameters**

This function has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## DeinitFTP Function

This function de-initializes the environment of the FTP-specific environment variables, and must be the last method executed in the Basic main method.

**BASIC Syntax**

```
call DeinitFTP()
```

**Parameters**

This function has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

# Directory Functions

## MakeHostDir Function

This function makes a new directory on the host. The argument must not be empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call MakeHostDir(BYVAL dirName$)
```

**Parameters**

*dirName*—specifies the new directory name or full path.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## ChangeHostDir Function

This function changes the current directory to the specified directory. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call ChangeHostDir (BYVAL dirName$)
```

### Parameters

*dirName*—specifies the directory name or full path to which to change the Host Directory.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## PrintWorkingDir Function

This function acquires the full path of the current directory on the host. The returned string may include FTP command codes, and can be processed further, if required.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
Text$ = PrintWorkingHostDir
```

### Parameters

This function has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## RemoveHostDir Function

This function removes the specified directory from the host. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call RemoveHostDir(BYVAL dirName$)
```

**Parameters**

*dirName*—specifies the directory name or full path to be removed.

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Connection and Access Functions

### ConnectToHost Function

This function connects to a remote host. You must supply the server port in order for this command to make the connection. The host name can either be an IP address or a server name specified in the host table.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call ConnectToHost(BYVAL host$, BYVAL serverPort as portint)
```

**Parameters**

*host*—specifies the remote host string name or IP address to which you want to connect.

*serverPort*—specifies the port you want to use to make the connection.

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## UserLogin Function

This function logs you on to the remote server. If your server requires a password or account, you should type them as specified below; otherwise, leave these strings empty. This function generates an error if there are no established connections.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call UserLogin(BYVAL userName$, BYVAL password$,  
BYVAL account$)
```

**Parameters**

*userName*—specifies the username that should be used to log on to the remote host.  
*password*—specifies the password to be used for login.  
*account*—specifies the account to be used for login.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## DisconnectFromHost Function

This function disconnects the current connection. It generates an error if there are no established connections. You can retrieve the text associated with the function using the error-handling routine.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call DisconnectFromHost()
```

**Parameters**

This function has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## File Transfer Functions

### GetFile Function

This function gets a single file from the host to the PC. To specify the destination as the current directory on the PC, pass an empty string as the destination path. If the current host directory does not include the source file, you should specify the full path. `GetFile` does not accept wildcard specifications.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call GetFile(BYVAL sourcePath$, BYVAL destPath$)
```

**Parameters**

*sourcePath*—specifies the full path to the file on the host.

*destPath*—specifies the full path to where you want to put the file on the PC.

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

### MGet Function

This function gets multiple files from the host to the current PC directory.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call Mget(BYVAL pattern$)
```

**Parameters**

*pattern*—specifies the pattern used by the host to get all matching file. The pattern may include wildcards native to the host.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## PutFile Function

This function moves a single file from the PC to the host. To specify the destination as the current directory on the host, pass an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call PutFile (BYVAL sourcePath$, BYVAL destPath$)
```

### Parameters

*sourcePath*—specifies the full source path of the PC file name to be sent to the remote host.

*destPath*—specifies the full destination path of the host file name.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## MPut Function

This function puts multiple files to the current directory on the host from the PC.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call Mput (BYVAL pattern$)
```

### Parameters

*pattern*—specifies the pattern used by the PC to get all matching file. The pattern may include wildcards native to the local operating system.

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## AppendFile Function

This function appends a single file from your PC to the host. To specify the destination as the current directory on the host, type an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call AppendFile( BYVAL sourcePath$, BYVAL  
destPath$ )
```

### Parameters

*sourcePath*—specifies the full source path of the PC file name you want to send to the remote host.

*destPath*—specifies the full destination path of the host file name to which you want to append the file.

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Deleting and Renaming Functions

### DeleteHostFile Function

This function deletes the specified file from the host. The argument may include any wildcard characters supported by the host.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call DeleteHostFile(BYVAL sourcePath$)
```

### Parameters

*sourcePath*—specifies the path of the file to be deleted from the host.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## RenameHostFile Function

This function renames the specified file to a new name. The subroutine does not attempt to rename files unless you specify both arguments.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call RenameHostFile(BYVAL sourcePath$, BYVAL  
destPath$)
```

### Parameters

*sourcePath*—specifies the path of the file to be renamed.  
*destPath*—specifies the path of the file with the new name.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Tree Functions

### GetTree Function

This function gets an entire tree from the host to the PC. To specify the current directory as the destination on the PC, pass an empty string as the destination path. If the current host directory does not include the source directory, you should specify the full path. `GetTree` does not accept wildcard specifications.

### BASIC Syntax

```
call GetTree(BYVAL sourcePath$, BYVAL destPath$)
```

### Parameters

*sourcePath*—specifies the full source path to be received from the host.  
*destPath*—specifies the full destination path on the PC.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## PutTree Function

This function puts an entire tree from the PC to the host. To specify the destination as the current directory on the host, pass an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call PutTree(BYVAL sourcePath$, BYVAL destPath$)
```

### Parameters

*sourcePath*—specifies the full source path of the PC path to be sent to the remote host.

*destPath*—specifies the full destination path on the host.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## DelTree Function

This function deletes an entire tree from the host. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call DelTree(BYVAL dirName$)
```

### Parameters

*dirName*—specifies the directory name and its subdirectories that you will delete

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## LocalDelTree Function

This function deletes an entire tree from the local PC. The argument must not be empty. `LocalDelTree` is a supplement to the current Hummingbird Scripting Language API. No connection to any host is required, however, you must use the `FTPIInit` and `Deinit` API.

Errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call LocalDelTree(BYVAL dirName$)
```

### Parameters

*dirName*—specifies the path of the tree to be deleted from the PC.

**Note:** This function is prototyped in the `ftpRtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Transfer Type Settings

### ASCIIType Setting

This setting changes the current file transfer type to ASCII format. The default transfer type for FTP scripts is AUTO.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call ASCIIType()
```

### Parameters

This method has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## BinaryType Setting

This setting changes the current file transfer type to binary format. The default transfer type for FTP scripts is AUTO.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

### BASIC Syntax

```
call BinaryType()
```

### Parameters

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## SetTransferType Setting

This setting sets the current transfer type to ASCII, BINARY or AUTO DETECT type. If the function is successful, the previous transfer type is returned; otherwise, a negative value returns. In AUTO DETECT, transfer type reverts to ASCII when transferring text files, and to BINARY when transferring binary files (e.g. executables). The default transfer type for FTP scripts is AUTO.

### BASIC Syntax

```
ReturnVal = SetTransferType (BYVAL transferType as  
portint)
```

### Parameters

*transferType*—specifies a valid integer value to be set as transfer type.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script.

## GetTransferType Setting

This function returns the current transfer type. One of the following values will be returned: ASCII, BINARY or AUTO DETECT. In AUTO DETECT transfer type reverts to ASCII when transferring text files, and to BINARY when transferring binary files (e.g. executables).

The default transfer type for FTP scripts is AUTO.

### BASIC Syntax

```
RetVal = GetTransferType()
```

### Parameters

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Local File Creation Settings

### CreateValidFATFiles Setting

This function sets a flag which the FTP script uses during PC file creation to create valid FAT file names. If the parameter is nonzero the flag is set, otherwise it is reset. By default this flag is set and the FTP script creates valid FAT file names. The `CreateValidFATFiles` method returns a nonzero value upon a successful setting, or zero to indicate an error.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
returnVal=CreateVAlidFATFiles (BYVAL setVal as  
portint)
```

**Parameters**

*setVal*—specifies an integer that turns valid file creation on the PC on or off.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## ValidFATFileCreate Setting

This setting returns the current state of the flag that FTP scripting uses during PC file creation to create valid FAT file names. The function either returns a nonzero value indicating that the flag is set, or a zero value indicating that the flag is not set.

**BASIC Syntax**

```
ReturnVal = ValidFATFileCreate()
```

**Parameters**

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script.

## SetReplyTimeoutValue Setting

This setting sets how many seconds FTP scripting waits for a reply from the remote host after a command is sent. If the parameter is a positive, nonzero, long integer, the timeout value will be set. Otherwise, the function returns a zero, which indicates an error. By default the timeout value is set to 120 (secs).

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
ReturnVal = SetReplyTimeoutValue(BYVAL value as  
long)
```

**Parameters**

*value*—specifies a long value to be set as the timeout.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script.

## GetReplyTimeoutValue Setting

This setting returns the current timeout value used by the FTP scripting tool. The value returned represents how long the FTP scripting tool waits for a reply from the remote host after an FTP command is sent. The `GetReplyTimeoutValue` function returns a long value indicating the timeout.

**BASIC Syntax**

```
ReturnVal = GetReplyTimeoutValue()
```

**Parameters**

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Remote Command Functions

### LS Function

This function gets the directory listing of the specified directory, with or without options in the argument. If required, you can process the returned text further using the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
Text$ = LS(BYVAL options$)
```

**Parameters**

*options*—specifies the full path, or options that may be supplied to the host (this varies with each host).

**Note:** This function is prototyped in the `ftp rtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## FTPDIR Function

This function gets the directory listing of the specified directory with or without options in the argument. The returned text will include a full description of each file on the host. If required, you can further process the returned text using the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
Text$ = FTPDIR(BYVAL options$)
```

**Parameters**

*options*—specifies the full path, or options that may be supplied to the host (this varies with each host).

**Note:** This function is prototyped in the `ftp rtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## FTP Session Settings

### FTPACTIVE Setting

This setting forces the FTP client session to establish the data connection actively. This makes the server (host) passive. By default, FTP scripting sessions are active.

**BASIC Syntax**

```
call FTPACTIVE()
```

**Parameters**

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

### FTPPASSIVE Setting

This setting forces the FTP session to establish the data connection passively. This makes the server (host) active and the FTP client passive. By default, FTP scripting sessions are active.

**BASIC Syntax**

```
call FTPPASSIVE()
```

**Parameters**

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## SessionType Setting

This setting determines the active/passive status of the FTP client session. If the function returns a nonzero value, then the FTP client session is active and the remote host (server) is passive. Otherwise, a zero value indicates that the remote host (server) is active and the FTP client session is passive.

**BASIC Syntax**

```
ReturnVal=SessionType()
```

**Parameters**

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script.

## CTRLZStatus Setting

This function determines whether or not the CTRL-Z character is being inserted at the end of ASCII files, when they are transferred from the PC to the server. A nonzero value indicates that the FTP session will insert a CTRL-Z character at the end of transferred files. Otherwise, a returned zero indicates that CTRL-Z is not inserted at the end of transferred files.

**BASIC Syntax**

```
returnVal=CTRLZStatus()
```

**Parameters**

This setting has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## GetLocalFilenamesStatus Setting

This setting returns the current state of the flag which FTP scripting uses to transfer local files with lowercase file names during a PUT operation. The function will either return a nonzero value indicating that the flag is set, or a zero value to indicate that the flag is not set.

**BASIC Syntax**

```
returnVal = GetLocalFilenamesStatus()
```

**Parameters**

This setting has no parameters.

## SetLocalFilenamesLowercase Setting

This setting sets a flag that FTP scripting uses during a put operation in order to transfer local files to the host with lowercase file names. If the parameter is nonzero, the flag sets. Otherwise, it resets. By default this flag is set, and FTP scripting will transfer lowercase file names.

**BASIC Syntax**

```
ReturnVal = SetTransferType(BYVAL transferType as  
portint)
```

**Parameters**

*transferType*—specifies an integer that turns lowercase transfer on or off.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script.

## StripCtrlZ Setting

This setting removes the CTRL-Z if it is at the end of ASCII files being transferred from the PC to the server. This ensures that CTRL-Z does not exist in the server's file. When this flag is set, `hclftp.dll` ignores the state of the WriteCtrlZ flag. By default, CTRL-Z is inserted at the end of files.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

call StripCtrlZ(BYVAL *setVal* as portint)

**Parameters**

*setVal*—specifies the Boolean variable used to set or reset the option.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## WriteCtrlZ Setting

This setting controls whether or not a CTRL-Z end-of-file marker is inserted at the end of ASCII files being transferred from the PC to the server. By default, CTRL-Z is inserted at the end of files.

**BASIC Syntax**

call WriteCtrlZ(BYVAL *setVal* as portint)

**Parameters**

*setVal*—specifies the Boolean variable used to set or reset the option.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Miscellaneous FTP Command Functions and Settings

### Account Function

This function sends an FTP account command to the host. In order to transfer files, you must supply the account command to some hosts.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call Account (BYVAL account$)
```

**Parameters**

*account*—specifies the account string you want to send to the host.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## SystemType Function

This function requests the host to specify the underlying operating system of the host. The subroutine sends an FTP (SYST) command to the host. The server either returns an error specifying that the command has not been implemented, or returns the text that includes the server system type. This text can be retrieved with the `GetOutputText` function, and can be processed further by the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
call SystemType()
```

**Parameters**

This function has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## GetOutputText Function

This function retrieves the replies made by the server with respect to any other FTP API function.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
Text$ = GetOutputText
```

**Parameters**

This function has no parameters.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

The `GetOutputText` function is used to retrieve the replies made by the server to any used subroutines or functions. It returns the string echoed by the server when a command is sent, allowing you to see what the server has returned.

## QuoteCommand Function

This function sends arbitrary FTP commands to the host. The returned string may include FTP command codes, and, if required, can be processed further.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

**BASIC Syntax**

```
Text$ = QuoteCommand( BYVAL commandText$ )
```

**Parameters**

*commandText*—specifies the command text to be sent to the host

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Error Handling Function

### GetErrorText Function

This function retrieves the error text associated with the last FTP subroutine or function call. The FTP scripting engine generates the text.

#### BASIC Syntax

```
Text$ = GetErrorText (BYVAL errorCode as portint)
```

#### Parameters

*errorCode*—specifies the trapped error code generated as a result of the previous FTP functions or subroutines.

**Note:** This function is prototyped in the `ftprtns.ebh` header file. In order to access the prototype for this API command, you must include this header file in your script file.

## Customizing FTP OLE

### 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`.

## 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
```

## 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.

### **IHclFtpEngine Object Method**

The following is the Engine method:

Quit Method

## IHclFtpEngine Object Properties

The following are the Engine properties:

- Application Property
- FullName Property
- Name Property (IHclFtpEngine)
- Parent Property
- Sessions Property
- Visible Property

## 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.

## IHclFtpSession Object Methods

The following are the Session methods:

- AppendFile Method
- ChangeHostDir Method
- ConnectToHost Method
- DeleteHostFile Method
- DelTree Method
- DisconnectFromHost Method
- FTPDIR Method
- GetFile Method
- GetTree Method
- HostToHostFileAppend Method
- HostToHostFileTransfer Method
- HostToHostMultipleFileTransfer Method
- HostToHostTreeTransfer Method
- LS Method
- MakeHostDir Method
- MGet Method
- MPut Method
- NewUserLogin Method
- ParentDir Method
- PrintHostFile Method
- PutFile Method
- PutTree Method
- QuoteCommand Method
- RemoteHelp Method
- RemoveHostDir Method
- RenameHostFile Method
- ReplyText Method
- SendAccountCommand Method
- ServerBanner Method
- SystemType Method
- UserLogin Method
- WorkingHostDirectory Method

## IHclFtpSession Object Properties

The following are the Session properties:

- Account Property
- Connected Property
- DefaultSystemType Property
- DropExtension Property
- DropVersion Property
- FirewallPassword Property
- FirewallServerName Property
- FirewallServerPort Property
- FirewallType Property
- FirewallUserName Property
- ForceDefaultSystemType Property
- InitialHostDirectory Property
- Name Property (IHclFtpSession)
- Password Property
- PASVMode Property
- ReplyTimeout Property
- ServerName Property
- ServerPort Property
- SSHMode Property
- StripCtrlZ Property
- TraceFileName Property
- TraceMode Property
- TransferType Property
- UseFirewall Property
- UserName Property
- WriteCtrlZ Property

## 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.

## IHclFtpSessions Object Methods

The following are the Sessions methods:

- NewSession Method
- RemoveAllSessions Method
- RemoveSession Method
- RemoveSessionByName Method
- RemoveSessionByNumber Method

## IHclFtpSessions Object Properties

The following are the Sessions properties:

- Count Property
- EMailAddress Property
- LocalDefaultDirectory Property
- LocalFileNamesCase Property
- SessionByName Property
- SessionByNumber Property
- ValidLocalFileNames Property

## FullName Property

This property returns the full name of the application.

**BASIC Syntax**

```
variable = FullName
```

**Parameters**

This property has no parameters.

**Data Type**

String

**Property Type**

Read Only

## Name Property (IHclFtpEngine)

This property returns the name of the application.

**BASIC Syntax**

```
variable = Name
```

**Parameters**

This property has no parameters.

**Data Type**

String

**Property Type**

Read Only

## Parent Property

This property returns the application object of IHclFtpEngine type.

**BASIC Syntax**

```
variable = Parent
```

**Parameters**

This property has no parameters.

**Data Type**

IHclFtpEngine

**Property Type**

Read Only

## Visible Property

This property returns the state of the main window. If the window is visible, then the boolean is TRUE; if the window is not visible, then the boolean is FALSE.

**BASIC Syntax**      variable = Visible

**Parameters**      This property has no parameters.

**Data Type**      Boolean

**Property Type**      Read Only

## Sessions Property

This is the most useful of the IHclFtpEngine properties. Returns an object of IHclFtpSessions type. This is the collection of IHclFtpSession objects.

**BASIC Syntax**      Sessions = variable

variable = Sessions

**Parameters**      This property has no parameters.

**Data Type**      IHclFtpSessions

**Property Type**      Read Only

## Quit Method

This method exits the application.

**BASIC Syntax**      Quit

**Parameters**      This method has no parameters.

**Property Type**      None

## ServerName Property

This property returns the name of the FTP server to which the session is connected. The default value is an empty string.

**BASIC Syntax**      variable = ServerName  
                        ServerName = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## UserName Property

This property determines the login user name . The default value is an empty string.

**BASIC Syntax**      variable = UserName  
                        UserName = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## Password Property

This property determines the password for login authentication . The default value is an empty string.

**BASIC Syntax**      variable = Password  
                        Password = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## Account Property

This property sends an FTP account command to the host. In order to transfer files, you must supply the account command to some hosts. The default value is an empty string.

**BASIC Syntax**      variable = Account  
                        Account = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## InitialHostDirectory Property

This property identifies the directory to which the initial host directory is changed after the connection is made to the server at login. As soon as you log in, the host directory is changed automatically.

**BASIC Syntax**      variable = InitialHostDirectory  
                        InitialHostDirectory = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## ServerPort Property

This property identifies the port used for FTP connections. The default port is 21.

**BASIC Syntax**      variable = ServerPort

ServerPort = variable

**Parameters**      This property has no parameters.

**Data Type**      Long

**Property Type**      Read/Write

## ReplyTimeout Property

This property identifies the amount of time in seconds the FTP engine waits to receive a reply to the command sent to the FTP server before terminating. The default value is 120 seconds.

**BASIC Syntax**      variable = ReplyTimeout

ReplyTimeout = variable

**Parameters**      This property has no parameters.

**Data Type**      Long

**Property Type**      Read/Write

## TransferType Property

This property identifies the transfer type. The default is AUTO\_TRANSFER.

**BASIC Syntax**      variable = TransferType  
                          TransferType = variable

**Parameters**      This property has no parameters.

**Data Type**      TTransfer

**Property Type**      Read/Write

## PASVMode Property

This property helps transfer data through any possible firewalls. The default value is TRUE.

**BASIC Syntax**      variable = PASVMode  
                          PASVMode = variable

**Parameters**      This property has no parameters.

**Data Type**      Boolean

**Property Type**      Read/Write

## SSHMode Property

Determines if the FTP communications and file transfers are encrypted. The default value is FALSE.

<b>BASIC Syntax</b>	SSHMode=variable Variable=SSHMode
<b>Parameters</b>	TRUE—The FTP client encrypts the data connection. FALSE—The server does not encrypt the data connection.
<b>Data Type</b>	Boolean
<b>Property Type</b>	Read/Write

## Connected Property

This property determines if there is a connection. A TRUE value means there is a connection; a FALSE value means there is no connection. The default value is FALSE.

<b>BASIC Syntax</b>	variable = Connected
<b>Parameters</b>	This property has no parameters.
<b>Data Type</b>	Boolean
<b>Property Type</b>	Read Only

## WriteCtrlZ Property

This property controls whether or not a CTRL-Z end-of-file marker is inserted at the end of ASCII files being transferred from the PC to the server. A TRUE value indicates that the character is inserted. A FALSE value indicates the character is not inserted. The default value is FALSE.

**BASIC Syntax**

```
variable = WriteCtrlZ  
WriteCtrlZ = variable
```

**Parameters**

This property has no parameters.

**Data Type**

Boolean

**Property Type**

Read/Write

## StripCtrlZ Property

This property determines if CTRL-Z characters are stripped from the end of ASCII files transferred from the PC to the server. If the property is set to TRUE, FTP ignores the WriteCtrlZ value. The default value is FALSE.

**BASIC Syntax**

```
variable = StripCtrlZ  
StripCtrlZ = variable
```

**Parameters**

This property has no parameters.

**Data Type**

Boolean

**Property Type**

Read/Write

## DropExtension Property

This property determines if it is going to drop file extensions from file names transferred from the PC to the server. The default value is FALSE. Use this property only with mainframes such as IBM and TANDEM.

**BASIC Syntax**      variable = DropExtension  
                          DropExtension = variable

**Parameters**      This property has no parameters.

**Data Type**      Boolean

**Property Type**      Read/Write

## DropVersion Property

This property determines if FTPAPI is going to drop file version numbers from VMS file names transferred to a PC. The default is FALSE.

**BASIC Syntax**      Variable = DropVersion  
                          DropVersion = variable

**Parameters**      This property has no parameters.

**Data Type**      Boolean

**Property Type**      Read/Write

## TraceMode Property

This property determines if the script is going to perform tracing of FTP protocol commands. The default value is FALSE.

**BASIC Syntax**

```
Variable = TraceMode  
TraceMode = variable
```

**Parameters**

This property has no parameters.

**Data Type**

Boolean

**Property Type**

Read/Write

## TraceFileName Property

This property determines the trace file name used for tracing. The default value is an empty string.

**BASIC Syntax**

```
variable = TraceFileName  
TraceFileName = variable
```

**Parameters**

This property has no parameters.

**Data Type**

String

**Property Type**

Read/Write

## UseFirewall Property

This property determines if firewall support is needed, or if the script is going to use firewall support. The default value is FALSE.

**BASIC Syntax**

```
variable = UseFirewall  
UseFirewall = variable
```

**Parameters**

This property has no parameters.

**Data Type**

Boolean

**Property Type**

Read/Write

## FirewallServerPort Property

This property determines the server port used to connect to the firewall. The default port is 21.

**BASIC Syntax**      variable = FirewallServerPort  
                          FirewallServerPort = variable

**Parameters**      This property has no parameters.

**Data Type**      Long

**Property Type**      Read/Write

## FirewallServerName Property

This property determines the FTP firewall server to which the session is connected. The default value is an empty string.

**BASIC Syntax**      variable = FirewallServerName  
                          FirewallServerName = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## FirewallUserName Property

This property determines the user name used to authenticate the firewall login. The default value is an empty string.

**BASIC Syntax**      variable = FirewallUserName  
                          FirewallUserName = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## FirewallPassword Property

This property determines the string for firewall login authentication. The default value is an empty string.

**BASIC Syntax**      variable = FirewallPassword  
                          FirewallPassword = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## FirewallType Property

This property determines the firewall type. The default value is SITE\_SERVERNAME.

**BASIC Syntax**      variable = FirewallType  
                          FirewallType = variable

**Parameters**      This property has no parameters.

**Data Type**      TFirewall

**Property Type**      Read/Write

## Name Property (IHclFtpSession)

This property sets and returns the session name. The default value is Session [index in the collection].

**BASIC Syntax**      variable = Name  
                          Name = variable

**Parameters**      This property has no parameters.

**Data Type**      String

**Property Type**      Read/Write

## DefaultSystemType Property

This property determines the default system type. The default value is UNIX. If FTP cannot detect the system type, it will use the value you set as the system default.

**BASIC Syntax**      variable = DefaultSystemType  
                          DefaultSystemType = variable

**Parameters**      This property has no parameters.

**Data Type**      TDSystem

**Property Type**      Read/Write

## ForceDefaultSystemType Property

This property forces the client to accept the default system type you define. The default value is FALSE.

**BASIC Syntax**      Variable = ForceDefaultSystemType  
                          ForceDefaultSystemType = variable

**Parameters**      This property has no parameters.

**Data Type**      Boolean

**Property Type**      Read/Write

## DelTree Method

This method removes an entire local tree structure.

**BASIC Syntax**      DelTree [*directory path*]

**Parameters**      *directory path*—Specifies the path of the directory tree you want to delete.

**Return Value**      None

## DeleteHostFile Method

This method deletes a file on the host.

<b>BASIC Syntax</b>	DeletHostFile[ <i>file path</i> ]
<b>Parameters</b>	<i>file path</i> —Specifies the path of the file you want to delete.
<b>Return Value</b>	None

## ChangeHostDir Method

This method changes to another directory on the host.

<b>BASIC Syntax</b>	ChangeHostDir [ <i>directory path</i> ]
<b>Parameters</b>	<i>directory path</i> —Specifies the path of the directory to which you want to change.
<b>Return Value</b>	None

## ServerBanner Method

This method returns the server banner. The method returns an error when there is no connection.

<b>BASIC Syntax</b>	Variable = ServerBanner
<b>Parameters</b>	This method has no parameters.
<b>Return Value</b>	String

## WorkingHostDirectory Method

This method returns the path of the current working host directory.

**BASIC Syntax**      Variable = WorkingHostDirectory

**Parameters**      This method has no parameters.

**Return Value**      String

## ReplyText Method

This method returns the server reply text for the last FTP command.

**BASIC Syntax**      variable = ReplyText

**Parameters**      This method has no parameters.

**Return Value**      String

## SystemType Method

This method returns the name of the host operating system.

**BASIC Syntax**      variable = SystemType

**Parameters**      This method has no parameters.

**Return Value**      String

## RemoteHelp Method

This method returns help text from the server.

**BASIC Syntax**

`variable = RemoteHelp`

**Parameters**

*RemoteHelp*—The parameters and arguments to the help command which you want to execute on the remote system.

**Return Value**

String

## RemoveHostDir Method

This method removes a directory from the host.

**BASIC Syntax**

`RemoveHostDir [directory path]`

**Parameters**

*directory path*—Specifies the path of the directory you want to delete.

**Return Value**

None

## RenameHostFile Method

This method renames a file on the host.

**BASIC Syntax**

`RenameHostFile [source, destination path]`

**Parameters**

*source*—The path to the file that is to be renamed.

*destination path*—The path to where the renamed file will be stored.

**Return Value**

None

## ConnectToHost Method

This method connects to the host. It uses the values of the properties specified in the FTP Site Properties dialog box of Humminbird FTP for Windows Explorer. For more information about FTP for Windows Explorer, refer to the Hummingbird Neighborhood Help which is available when you install Hummingbird Neighborhood.

**BASIC Syntax** Not applicable

**Parameters** This method has no parameters.

**Return Value** None

## DisconnectFromHost Method

This method disconnects from the host.

**BASIC Syntax** Not applicable

**Parameters** This method has no parameters.

**Return Value** None

## UserLogin Method

This method is used to log in to the remote server. If the server requires a password or account, use the values for the UserName Property, Password Property, Account Property, and InitialHostDirectory Property to authenticate login.

**BASIC Syntax** Not applicable

**Parameters** This method has no parameters.

**Return Value** None

## NewUserLogin Method

This method uses the parameters to authenticate another login with the new user name. All required parameters must be included.

<b>BASIC Syntax</b>	<code>NewUserLogIn [username, password, account, initialhost directory]</code>
<b>Parameters</b>	<p><i>username</i>—Specifies the user name that should be used to log in to the remote host.</p> <p><i>password</i>—Specifies the password to be used for login.</p> <p><i>account</i>—Specifies the account to be used for login.</p> <p><i>initialhost directory</i>—Specifies the directory to which the initial host directory is changed after the connection is made to the server at login.</p>
<b>Return Value</b>	None

## SendAccountCommand Method

This method sends the account information to the server.

<b>BASIC Syntax</b>	<code>SendAccountCommand [account info]</code>
<b>Parameters</b>	<p><i>account info</i>—The account information you want to send to the host.</p>
<b>Return Value</b>	None

## LS Method

This method returns the host directory listing with the specified options. The script can process the returned text further, if required.

<b>BASIC Syntax</b>	<code>variable = LS [options]</code>
<b>Parameters</b>	<p><i>options</i>—Options are defined by the FTP server and vary depending on the server.</p>
<b>Return Value</b>	String

## FTPDIR Method

This method retrieves the directory listing of the specified directory, with or without options in the argument. The returned text includes a full description for each file on the host. The script can process the text further, if required.

**BASIC Syntax**

```
variable = FTPDIR [options]
```

**Parameters**

*options*—Options are defined by the FTP server and vary depending on the server.

**Return Value**

String

## GetFile Method

This method retrieves a file from the host to the PC. If the destination path is empty, the script sends the file to the current working directory.

**BASIC Syntax**

```
GetFile [source path, destination path]
```

**Parameters**

*source path*—Specifies the full path to the file on the host.  
*destination path*—Specifies the full path to where you want to put the file on the PC.

**Return Value**

None

## GetTree Method

This method retrieves an entire directory tree structure from the host to the PC. If the destination is empty, the script uses the local default directory.

**BASIC Syntax**

```
GetTree [source path, destination path]
```

**Parameters**

*source path*—Specifies the full path of the directory tree on the host.  
*destination path*—Specifies the full path to where you want to put the directory tree on the PC.

**Return Value**

None

## PutFile Method

This method sends a single file from the PC to the host. If the destination path is empty, the script sends the file to the current host working directory.

**BASIC Syntax**

`PutFile [source path, destination path]`

**Parameters**

*source path*—Specifies the full source path of the PC file name you want to send to the remote host.

*destination path*—Specifies the full destination path of the host file name.

**Return Value**

None

## PutTree Method

This method sends an entire directory tree from the PC to the host. If the destination path is empty, the script uses the local default directory.

**BASIC Syntax**

`PutTree [source, destination]`

**Parameters**

*source*—Specifies the full source path of the PC file name you want to send to the remote host.

*destination*—Specifies the full destination path of the host file name.

**Return Value**

None

## QuoteCommand Method

This method enables the user to send arbitrary protocol commands to the server.

**BASIC Syntax**

`variable = QuoteCommand [command text]`

**Parameters**

*command text*—The protocol commands you want to send to the server.

**Return Value**

String

## AppendFile Method

This method appends a PC file to a host file. If the destination path is empty, the script sends the file to the current host working directory.

**BASIC Syntax**

`AppendFile [source path, destination path]`

**Parameters**

*source path*—Specifies the full source path of the PC file that you want to append to the remote host file.

*destination path*—Specifies the full destination path of the host file to which you want to append the PC file.

**Return Value**

None

## MGet Method

This method retrieves multiple files from the host to the default local directory on the PC.

**BASIC Syntax**

`MGet [pattern]`

**Parameters**

*pattern*—Specifies the pattern used by the host to get all matching file(s). The pattern may include wildcards native to the host.

**Return Value**

None

## MPut Method

This method sends multiple files from the PC to the host working directory.

**BASIC Syntax**

`MPut [pattern]`

**Parameters**

*pattern*—Specifies the pattern used by the PC to get all matching files. The pattern may include wildcards native to the PC.

**Return Value**

None

## HostToHostFileTransfer Method

This method transfers a file from one session to another.

**BASIC Syntax**

`HostToHostFileTransfer [source path, destination path, destination session object]`

**Parameters**

*source path*—Specifies the full source path of the file name you want to transfer to another session.

*destination path*—Specifies the full destination path of the file name that you want to transfer.

*destination session object*—Specifies the HclFtpSession object for the host containing the destination.

**Return Value**

None

## HostToHostTreeTransfer Method

This method transfers an entire directory tree to another session.

**BASIC Syntax**

`HostToHostTreeTransfer [source path, destination path, destination session object]`

**Parameters**

*source path*—Specifies the full source path of the tree you want to transfer to another session.

*destination path*—Specifies the full destination path of the tree that you want to transfer.

*destination session object*—Specifies the HclFtpSession object for the host containing the destination.

**Return Value**

None

## HostToHostMultipleFileTransfer Method

This method transfers multiple files from one session to the other.

**BASIC Syntax**      HostToHostMultipleFileTransfer [*pattern, destination session object*]

**Parameters**      *pattern*—Specifies the pattern used by the current host to get all matching files. The pattern may be dependent on the server.

*destination session object*—Specifies the HciFtpSession object that represents the destination host.

**Return Value**      None

## HostToHostFileAppend Method

This method appends a session file to another session file.

**BASIC Syntax**      HostToHostFileAppend [*source, destination, path, destination session object*]

**Parameters**      *source*—Specifies the full source path of the files you want to append to another session file.

*destination*—Specifies the full destination path of the file that you want to append.

*destination session object*—Specifies the HciFtpSession object that represents the host where the destination file exists.

**Return Value**      None

## ParentDir Method

This method changes from the current directory to the nearest parent directory.

**BASIC Syntax**      ParentDir

**Parameters**      This method has no parameters.

**Return Value**      None

## MDelete Method

This method deletes multiple files on the host.

**BASIC Syntax**

MDelete [*Pattern*]

**Parameters**

*Pattern*—Specifies the pattern used by the host to get all matching file(s). The pattern can include wildcards native to the host.

**Return Value**

None

## PrintHostFile Method

This method prints a host file.

**BASIC Syntax**

PrintHostFile [*file path*]

**Parameters**

*file path*—Specifies the full path of the file you want to print.

**Return Value**

None

## MakeHostDir Method

This method creates a host directory.

**BASIC Syntax**

MakeHostDir [*directory path*]

**Parameters**

*directory path*—Specifies the full path of the directory you want to create.

**Return Value**

None

## TTransfer Data Type

<b>Data Type</b>	Integer
<b>Parameters</b>	This data type has no parameters.
<b>Possible Values</b>	BINARY_TRANSFER—Transfers a file as a binary image file. ASCII_TRANSFER—Transfers a file as ASCII (text). AUTO_TRANSFER—Automatically detects whether files are Binary or ASCII format.

## TDSystem Data Type

<b>Data Type</b>	Integer
<b>Parameters</b>	This data type has no parameters.
<b>Possible Values</b>	UNIX—UNIX System Type. VMS_VAX—VAX VMX system type. DOS—DOS system type. OS_2—OS/2 System Type. WIN_NT—Windows NT System Type. MVS_SYS—MVS system type. VM_CMS_SYS—VM/CMS system type. VM_VPS_SYS—VM/VPS SYS system type. TANDEM OS_9 OTHER

## TFirewall Data Type

<b>Data Type</b>	Integer
<b>Parameters</b>	This data type has no parameters.
<b>Possible Values</b>	SITE_SERVERNAME—Firewall Type SITE_SERVERNAME. USER_AFTERLOGIN—Firewall Type USER_AFTERLOGIN. USER_WITHOUTLOGIN—Firewall Type USER_WITHOUTLOGIN. PROXY_OPEN—Firewall Type PROXY_OPEN. PROXY_CONNECT—Firewall Type PROXY_CONNECT.

## TLocalCase Data Type

<b>Data Type</b>	Integer
<b>Parameters</b>	This data type has no parameters.
<b>Possible Values</b>	SAME—Keeps original case. LOWERCASE—Sets all characters in the file name to lowercase. UPPERCASE—Sets all characters in the file name to uppercase.

## EMailAddress Property

This property sets or returns the e-mail address. This address is used as the password for all sessions using "anonymous" as the user name. It is also used for anonymous sessions. The default value is an empty string.

**BASIC Syntax**

```
EmailAddress = variable  
variable = EmailAddress
```

**Parameters**

This property has no parameters.

**Data Type**

String

**Property Type**

Read/Write

## ValidLocalFileNames Property

This property creates valid local file names when files are transferred from the host to the PC. If the value is TRUE, the FTP engine creates the valid local file names. If the value is FALSE, the FTP engine does not check for valid names. By default, the property is set to TRUE.

**BASIC Syntax**

```
ValidLocalFileNames = variable  
variable = ValidLocalFileNames
```

**Parameters**

This property has no parameters.

**Data Type**

Boolean

**Property Type**

Read/Write

## LocalFileNamesCase Property

This property defines the case of the PC file names sent to the host. The default is lowercase.

**BASIC Syntax**

```
LocalFileNamesCase = variable  
variable = LocalFileNamesCase
```

**Parameters**

This property has no parameters.

**Data Type**

TLocalCase

**Property Type**

Read/Write

## LocalDefaultDirectory Property

This property identifies and sets the local default directory for all sessions. If the file name or directory name does not include the full path, the script automatically sends files to or retrieves files from the default directory.

**BASIC Syntax**

```
LocalDefaultDirectory = variable  
variable = LocalDefaultDirectory
```

**Parameters**

This property has no parameters.

**Data Type**

String

**Property Type**

Read/Write

## SessionByNumber Property

This property returns the session object identified by the index you provide.

**BASIC Syntax**

```
SessionByNumber (ByVal index as Long) = variable  
variable = SessionByNumber (ByVal index As Long)
```

**Parameters**

This property has no parameters.

**Data Type**

IHclFtpSession

**Property Type**

Read Only

## SessionByName Property

This property returns the session object identified by the name you provide.

**BASIC Syntax**

```
SessionByName (By Val Name as String) = variable  
variable = SessionByName (ByVal name as String)
```

**Parameters**

This property has no parameters.

**Data Type**

IHclFtpSession

**Property Type**

Read OnlyRead Only

## NewSession Method

This method creates a new FTP session with default properties.

**BASIC Syntax**

```
variable = NewSession
```

**Parameters**

This method has no parameters.

**Return Value**

IHclFtpSession object

## RemoveSession Method

This method disconnects and removes a session identified by the IHclFtpSession object.

**BASIC Syntax**

```
RemoveSession variable
```

**Parameters**

IHclFtpSession object

**Return Value**

None

## RemoveSessionByNumber Method

This method disconnects and removes a session identified by the index number.

**BASIC Syntax**      RemoveSessionByNumber index

**Parameters**      Long Integer

**Return Value**      None

## RemoveSessionByName Method

This method disconnects and removes a session identified by the name.

**BASIC Syntax**      RemoveSessionByName variable

**Parameters**      String

**Return Value**      None

## RemoveAllSessions Method

This method disconnects and removes all sessions.

**BASIC Syntax**      variable = RemoveAllSessions

**Parameters**      This method has no parameters.

**Return Value**      None

## LocalDelTree Method

This method deletes the entire local directory tree identified by the parameter.

**BASIC Syntax**      LocalDelTree variable

**Parameters**      String

**Return Value**      None

## Application Property

This property returns the application object of IHclFtpEngine type.

**BASIC Syntax**      variable = Application

**Parameters**      This property has no parameters.

**Data Type**      IHclFtpSession

**Property Type**      Read Only

## Count Property

This property returns the number of current sessions.

**BASIC Syntax**      Count = variable

variable = Count

**Parameters**      This property has no parameters.

**Data Type**      Long Integer

**Property Type**      Read Only

## User-Defined Types

The following are the user-defined types:

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

# Chapter 4

---

## Customizing WyseTerm

## 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.

# Configuring Telnet Using DDE

## 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

## 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
- SysItems Item
- Formats Item

### **Topics Item**

Lists the available topics as follows:

- System
- Terminal
- EmulationType
- Page

**Syntax**                      Topics

**Parameters**                 This method has no parameters.

## SysItems Item

Lists the available request items in the System Topic.

You can choose one of the following:

- Topics item
- SysItems item
- Formats item

**Syntax**                      SysItems

**Parameters**                 This method has no parameters.

## Formats Item

Lists the supported return formats. Telnet supports only text format.

**Syntax**                      Formats

**Parameters**                 This method has no parameters.

## Terminal Topic Items (HTelnet|Terminal)

The Terminal topic lets the client control the terminal emulation mode (ANSI-BBS, SCOANSI, VT220, VT320, WYSE50, WYSE60, or DG210) and other connection settings in the Telnet program. It also lets the client load previously saved settings files. The Terminal topic includes both request and execute items.

## Request Items

Terminal Topic has following request transaction items:

- ConnectionStatus Item
- EmulationType Item
- ExecReturn Item
- Formats Item
- SaveData Item
- TopicItemList Item

## Execute Items

To perform an execute transaction, you must fill the data with text corresponding to the command. To confirm that the command succeeded, you must perform a request transaction using the ExecReturn item (see the preceding list) to confirm that the execute transaction was successful.

The Exec command should consist only of the command name and any parameters. Do not include any spaces.

Terminal Topic has the following Execute transaction items:

- LoadSetupFile Item
- ConnectRlogin Item
- ConnectTelnet Item
- Disconnect Item
- SendString Item
- SendStringToTerminal Item
- LookForString Item
- StartDataSave Item
- StopDataSave Item

## TopicItemList Item

Lists the available request items in the Terminal Topic. You can choose one of the following:

- TopicItemList
- Formats
- EmulationType
- ConnectionStatus
- ExecReturn
- SaveData

**Syntax** TopicItemList

**Parameters** This method has no parameters.

## EmulationType Item

Returns the current emulation type. Telnet supports the following types:

- ANSI-BBS
- SCIANSI
- VT220
- VT320
- Wyse50
- Wyse60
- DG210

**Syntax** EmulationType

**Parameters** This method has no parameters.

## ConnectionStatus Item

Returns the current state of the connection to the Telnet application. Telnet returns one of the following three states:

- connected
- connecting
- disconnecting

**Syntax** ConnectionStatus

**Parameters** This method has no parameters.

## ExecReturn Item

Returns the result of the last called Execute transaction.

**Syntax** ExecReturn

**Parameters** This method has no parameters.

## SaveData Item

Returns text that was previously saved with the StartDataSave execute item.

**Syntax** SaveData

**Parameters** This method has no parameters.

## LoadSetupFile Item

Loads a Telnet settings file. If you do not specify the path, the script uses the USER directory. If the file loads, the item returns a TRUE value. If the file does not load, the script returns a FALSE value.

**Syntax**

LoadSetupFile (*Filename.pts*)

**Parameters**

*Filenname.pts*—is the name of the settings file you want to load.

## ConnectRlogin Item

Establishes a connection using the RLOGIN protocol. Use the ConnectStatus request item to determine if the connection is successful. The item returns a TRUE value if the connection is successful and a FALSE value if the connection fails.

**Syntax**

ConnectRlogin (*hostname.port*)

**Parameters**

*hostname*—is the name of the host to which you want to connect.

*port*—is the port you through which you want to make the connection.

## ConnectTelnet Item

Establishes a connection using the TELNET protocol. Use the ConnectStatus request item to determine if the connection is successful. The item returns a TRUE value if the connection is successful and a FALSE value if the connection fails.

**Syntax**

ConnectTelnet (*hostname.port*)

**Parameters**

*hostname*—is the name of the host to which you want to connect.

*port*—is the port you through which you want to make the connection.

## Disconnect Item

Disconnects you from your current host session. Use the ConnectStatus request item to determine if the disconnection is successful. The item returns a TRUE value if the disconnection was successful and a FALSE value if the disconnection fails.

**Syntax**                  `Disconnect ()`

## SendString Item

Sends the character string to the host as it appears in the `str` argument. This item returns a TRUE value if successful and a FALSE value if not successful.

**Syntax**                  `SendString (str)`

**Parameters**              `str`—is any character string consisting of letters, numbers, and symbols.

## SendStringToTerminal Item

Sends the character string to a terminal as if sent by the host. This item sends the string as it appears in the `str` argument. This item returns a TRUE value if successful and a FALSE value if not successful.

**Syntax**                  `SendStringToTerminal (str)`

**Parameters**              `str`—is any character string consisting of letters, numbers, and symbols.

## LookForString Item

Looks for a string (as it appears in the `str` argument) in the incoming data stream from the host. This item creates a new item with the same name as `str`. To delete the new item, set its value to an empty string. This item returns a TRUE value if successful and a FALSE value if not successful.

**Syntax**

`LookForString (str)`

**Parameters**

`str`—is any character string consisting of letters, numbers, and symbols.

## StartDataSave Item

Initializes the `SaveData` item (to blank) and starts saving the host data stream to it. This item returns a TRUE value if successful and a FALSE value if not successful.

**Syntax**

`StartDataSave`

**Parameters**

This method has no parameters.

## StopDataSave Item

Stops saving the data stream to the `SaveData` item. This item returns a TRUE value if successful and a FALSE value if not successful.

**Syntax**

`StopDataSave`

**Parameters**

This method has no parameters.

## 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:

- TopicItemList Item
- Formats Item
- CursorPosition Item
- ExecReturn Item

### Execute Items

To perform an execute transaction, you must fill the data with text corresponding to the command. To confirm that the command succeeded, you must perform a request transaction using the ExecReturn item (see the preceding list) to confirm that the execute transaction was successful.

The Exec command should consist only of the command name and any parameters. Do not include any spaces.

The Emulation Type topic has the following Execute transaction items:

- GetOption Item
- SetOption Item
- ReadKeyboardFile Item
- SetAttributeColor Item
- SetAttributeUsage Item

## CursorPosition Item

Returns the current cursor position as a row, column or string. You can change the cursor position with a poke transaction to this item name. Specify the row and column position as the poke data.

**Syntax** CursorPosition

**Parameters** This method has no parameters.

## GetOption Item

Gets the value of the option specified by the emulation option parameter. Generic options are available in all emulation modes. Each mode has specific options that are available only when you select that mode.

**Syntax** GetOption (*emulationoption*)

**Parameters** *emulationoption*—is any terminal emulation-specific setting.

## SetOption Item

Sets the value of the option specified by the emulation option parameter. Generic options are available in all emulation modes. Each mode has specific options that are available only when you select that mode.

**Syntax** SetOption (*emulationoption, value*)

**Parameters** *emulationoption*—is any terminal emulation-specific setting.  
*value*—is the value assigned to the option.

## ReadKeyboardFile Item

Reads a keyboard file in order to change the mapping of the PC key combination that is needed to forward host keys to the host.

**Note:** The HKeyMap program must create these keyboard files. This program can be started only in Telnet. The HKeyMap program is not DDE enabled. As a result, this item can read only existing keyboard files.

### Syntax

`ReadKeyboardFile(filename)`

*emulationoption*—is any terminal emulation-specific setting.

### Parameters

*filename*—is the name and path of the keyboard file you want to read.

## SetAttributeColor Item

Sets the color you want shown for the attribute you specify. The screen updates to reflect this change. The color consists of the foreground color and the background color. For example, `SetAttributeColor(bold,red,blue)` causes all bolded characters to display with red text and blue background. You can define the characteristics of this attribute using the `SetAttributeUsage` command.

### Syntax

`SetAttributeColor(attr, fgcolor, bgcolor)`

*attr*—is the name of the attribute.

### Parameters

*fgcolor*—is the foreground color.

*bgcolor*—is the background color.

## **SetAttributeUsage Item**

Sets the display of the attribute you specify. You can change the foreground color or the actual screen attribute characteristic itself. For example, you can change all bold characters to display underlined. The screen updates to reflect this change.

### **Syntax**

`SetAttributeUsage(attr, realAttrib, realcolor)`

*attr*—is the name of the attribute.

### **Parameters**

*realAttr*—is the attribute characteristic you want to assign.

*bgcolor*—is the foreground color of the attribute.

## **Emulation Options**

The `GetOption()` and `SetOption()` items require that you specify generic emulation options. Some of the options you can use with these items are listed below by category:

- Generic options—These options are available for all emulation modes.
- VT 220 and VT320 options
- Wyse50 and Wyse60 options

See the appropriate section for a complete listing of available options.

## Generic Options

Option Name	Return Values
Interpret	<b>TRUE</b> —control codes are interpreted and not displayed <b>FALSE</b> —control codes are displayed and not interpreted
AutoWrap	<b>TRUE</b> —lines are automatically wrapped <b>FALSE</b> —no autowrap is used
BackSpaceDelete	<b>TRUE</b> —backspace sends delete <b>FALSE</b> —backspace sends backspace
WarningBell	<b>TRUE</b> —rings bell <b>FALSE</b> —does not ring bell
NewLine	<b>TRUE</b> —CRLF (Carriage Return - Line Feed) <b>FALSE</b> —CR (Carriage Return only)
TermStringId	<b>The terminal string ID</b> —the host uses this string to identify what type of terminal emulation the client (Telnet) is using
LocalEcho	<b>TRUE</b> —characters typed are displayed locally at the client (Telnet) <b>FALSE</b> —characters typed are sent to the host and the host determines if the client displays the character locally
Online	<b>TRUE</b> —characters are sent to the host and the terminal receives these characters <b>FALSE</b> —characters are sent to the host only
DisplayLines	<b>1-100</b> —the number of lines that can be displayed at one time
DisplayWidth80	<b>TRUE</b> —80 columns are displayed <b>FALSE</b> —132 columns are displayed
SmoothScroll	<b>TRUE</b> —scrolls smoothly <b>FALSE</b> —scrolls jump

JumpScrollNumber	<b>2-10</b> —the number of lines jumpscroll uses. Only applicable when Smooth Scroll option is FALSE.
History	<b>TRUE</b> —saves the scrolled data in a scroll back buffer <b>FALSE</b> —does not save the scrolled data
HistSize	<b>1-1000</b> —the number of lines to keep in the history buffer. Only applicable when History option is TRUE.
CaptureMode	<b>OFF/ON/ONHOSTCONTROL</b> —determines if the incoming data is captured to a file
CaptureDestination	<b>FILE PATHNAME</b> —the destination of the capture output. Only applicable when CaptureMode is ON

## VT220 and VT320 Options

Option Name	Return Values
UfLock	<b>TRUE</b> —user features are locked <b>FALSE</b> —user features are not locked
UdkLock	<b>TRUE</b> —user-defined keys are locked <b>FALSE</b> —user-defined keys are not locked
EmulationMode	<b>VT52/VT100/VT2007bit/VT2008bit</b> —emulation mode is set to one of the corresponding settings
NationalSet	North American/UK/Dutch/Finnish/French/French Canadian/German/Italian/Norwegian or Danish/Portuguese/Spanish/Swedish/Swiss—National Character Set is set to one of the corresponding settings
Multinational	<b>TRUE</b> —DEC multinational character set is used <b>FALSE</b> —only National character set is used
*UserPrefCharSetISO	<b>TRUE</b> —ISO LATIN1 is used <b>FALSE</b> —DEC multinational is used
*StatusLineMode	<b>OFF</b> —terminal information is not designated to be displayed <b>ON</b> —terminal information is designated to be displayed <b>HOSTWRITEABLE</b> —the host has been designated to change what is displayed in the status line directly

\* VT320 options only.

## WYSE50 and WYSE60 Options

Option Name	Return Values
ApplicationMode	<b>TRUE</b> —keys are in application mode <b>FALSE</b> —keys are in normal mode
EditModeLocal	<b>TRUE</b> —the client (Telnet) is working in local edit mode (that is, they can make local edits to a remote file) <b>FALSE</b> —the client (Telnet) is not working in local mode
BlockEndUSCR	<b>TRUE</b> —the USCR is used as the block terminator <b>FALSE</b> —the USCR character is not used as the block terminator
AutoPageMode	<b>TRUE</b> —moves to the next page automatically <b>FALSE</b> —does not move to the next page automatically
AutoScroll	<b>TRUE</b> —when text reaches the bottom of the page, the page scrolls so that the new text appears on a new bottom line <b>FALSE</b> —when text reaches the bottom of the page, the new text wraps to the top of the page
CommMode	<b>FULL DUPLEX/HALF DUPLEX/BLOCK</b> —the current communication mode is set to one of the corresponding settings
ActivePage	<b>0-3</b> —the number of Wyse pages that are active. There are four total pages
ActivePageSize	<b>0-100</b> —the page size of the active page. Note that each page can have a different size

## Page Topic Items (HTelnet|Page)

The Page topic lets the client control the current displayable properties such as color and attribute settings. These can be changed manually in Telnet in the Modify Terminal Setup dialog box. It also lets the client retrieve blocks of characters from anywhere on the display screen.

The Page topic has the following request transaction items:

- TopicItemList Item
- Formats Item
- CurrentFGColor Item
- CurrentBGColor Item
- CurrentAttribute Item
- CharacterAt Item
- CharacterAttributeAt Item
- CharacterFGColorAt Item
- CharacterBGColorAt Item

### **CurrentFGColor Item**

Determines the current foreground color given to new characters. Nocolor is the default.

**Syntax**      CurrentFGColor

**Parameters**      This method has no parameters.

### **CurrentBGColor Item**

Determines the current background color given to new characters. Nocolor is the default.

**Syntax**      CurrentBGColor

**Parameters**      This method has no parameters.

### **CurrentAttribute Item**

Determines the current attribute applied to new characters.

**Syntax**      CurrentAttribute

**Parameters**      This method has no parameters.

## CharacterAt Item

Returns all the characters between the first and second R?C?. The top left corner of the screen has a value of r,c=1,1, and the bottom right is R,C=24,80, or R,C=24,132 depending upon the current column width. For Example, CharacterAtR1C1R24C80 returns all characters between the x, y coordinates of (1,1) and (24,80).

**Note:** You can access the history buffer by specifying a number between 0 and -HistorySize + 1.

**Syntax**`CharacterAtR?C?R?C?`**Parameters**

This method has no parameters.

## CharacterAttributeAt Item

Returns the attribute of all characters between the first and second R?C?. The top left corner of the screen has a value of R,C=1,1, and the bottom right is R,C=24,80, or R,C=24,132 depending upon the current column width. For example, CharacterAttributeAtR1C1R24C80 returns the attribute of all characters between the x, y coordinates of (1,1) and (24,80).

**Note:** You can access the history buffer by specifying a number between 0 and -HistorySize + 1.

**Syntax**`CharacterAttributeAtR?c?R?C?`**Parameters**

R?—is the number of the row.

C?—is the number of the column.

## CharacterFGColorAt Item

Returns all the foreground colors of the characters between the first and second R?C?. The top left corner of the screen has a value of R,C=1,1 and the bottom right is, c=24,80, or r,c=24,132 depending on the current column width. For example, CharacterFGColorAtR1C1R24C80 returns the foreground color of all characters between the x, y coordinates of (1,1) and (24,80).

**Note:** You can access the history buffer by specifying a number between 0 and -HistorySize+1.

**Syntax**`CharacterFGColorAtR?C?R?C?`**Parameters**

R?—is the number of the row.

C?—is the number of the column.

## CharacterBGColorAt Item

Returns all the background colors of the characters between the first and second R?C?. The top left corner of the screen has a values of r,c=1,1, and the bottom right is r,c,+24,80, or r,c=24,132 depending on the current column width.

**Syntax**`CharacterBGColorAtR?C?R?C?`**Parameters**

R?—is the number of the row.

C?—is the number of the column.

## Valid Colors and Attributes

Various items require that you specify either a valid color and/or attribute as parameter of its function. In other cases, a color or an attribute may be returned as part of a transaction. The supported colors and attributes, with their numeric values, are listed below.

**Note:** To create a multi-characteristic attribute in C/C++, attributes are bit ORed together for the final attribute. To create a multi-characteristic attribute in Basic, they are added together to generate a final attribute.

### Colors

Color	DDE String Values	OLE Decimal Value
black	Black	0
blue	Blue	1
green	Green	2
cyan	Cyan	3
red	Red	4
magenta	Magenta	5
brown	Brown	6
dark white	Dkwhite	7
grey	grey	8
light blue	light blue	9
light green	light green	10
light cyan	light cyan	11
light red	light red	12

<b>Color</b>	<b>DDE String Values</b>	<b>OLE Decimal Value</b>
light magenta	light magenta	13
yellow	yellow	14
white	white	15
nocolor	nocolor	16

## Attributes

<b>Attribute</b>	<b>DDE String Values</b>	<b>OLE Decimal Value</b>	<b>OLE Hex Value</b>
normal	normal	0	0x0
bold	bold	1	0x01
blink	blink	2	0x02
underline	underline	4	0x04
reverse	reverse	8	0x08
protected	protected	16	0x10
hidden	hidden	128	0x20

# Configuring Telnet Using OLE

## 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 Property and Method Syntax corresponding to the object you want to use.

## 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.

## 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.

## 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.

**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.

**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
- EmulationObject Property
- Title Object
- ConnectionStatus Property
- Page Object

### Methods

- LoadSetupFile Method
- ConnectTelnet Method
- Visible BOOL Method
- Fullname Method
- SendStringToTerminal Method
- GetEventStatus Method
- StartDataSave Method
- StopDataSave Method
- ConnectRlogin Method
- Disconnect Method
- Name Method
- SendString Method
- LookForString Method
- RemoveEvent Method
- ReadData Method

## EmulationType Property

Returns the current emulation type. The following list details the types Telnet supports and the values for each type:

- ANSI-BBS—1
- SCOANSI—2
- VT220—3
- VT320—4
- WYSE50—5
- WYSE60—6

**Syntax** EmulationType

**Parameters** This property has no parameters.

**VarType** light red

## ConnectionStatus Property

Returns the current connection state of the Telnet application. The following list details the three states Telnet can return and the values for each type:

- Connected—2
- Connecting—1
- Disconnecting—0

**Syntax** ConnectionStatus

**Parameters** This property has no parameters.

**VarType** Short

## EmulationObject Property

This property is an object reference to the Emulation object. The following list details the possible object types:

**Syntax** EmulationObject

**Parameters** This property has no parameters.

**VarType** LPDISPATCH

## LoadSetupFile Method

Loads a Telnet settings file. If you do not specify a path, the script uses the USER directory.

**Syntax** LoadSetupFile(*str*)

**Parameters** *str*—is the name of the settings file you want to load.

**VarType** BOOL

## ConnectRlogin Method

Establishes a connection using the RLOGIN protocol. Use the ConnectStatus property to determine if the connection is successful.

**Syntax** ConnectRlogin(*str host*, *str port*)

*str host*—is the name of the host to which you want to connect.

*str port*—is the port you want to use to establish the connection.

**Return Type** BOOL

## ConnectTelnet Method

Establishes a connection using the TELNET protocol. Use the ConnectStatus property to determine if the connection is successful.

<b>Syntax</b>	ConnectTelnet(str host, str port)
<b>Parameters</b>	<i>str host</i> —is the name of the host to which you want to connect. <i>str port</i> —is the port you want to use to establish the connection.
<b>Return Type</b>	BOOL

## Disconnect Method

Disconnects you from your current host session. Use the ConnectStatus property to determine if the disconnection is successful.

<b>Syntax</b>	Disconnect ()
<b>Parameters</b>	This method has no parameters.
<b>Return Type</b>	BOOL

## Visible BOOL Method

Shows or hides the Telnet window.

<b>Syntax</b>	Visible BOOL
<b>Parameters</b>	This method has no parameters.
<b>Return Type</b>	True=visible False=hide

## Name Method

Returns the Telnet application file name. This is a read only command. Use this method if you have multiple applications running to determine from where the Telnet session is running.

**Syntax** Name

**Parameters** This method has no parameters.

**Return Type** BSTR

## Fullname Method

Returns the Telnet application path and file name. This is a read only commands. Use this method if you have multiple applications running to determine from where the Telnet session is running.

**Syntax** Fullname

**Parameters** This method has no parameters.

**Return Type** BSTR

## SendString Method

Sends the character string to the host as it appears in the STR *str* argument.

**Syntax** SendString(STR *str*)

**Parameters** *STR str*—is any character string consisting of letters, numbers, and symbols.

**Return Type** BOOL

## SendStringToTerminal Method

Sends the character string to a terminal as if sent by the host. This item sends the string as it appears in the STR str argument.

<b>Syntax</b>	SendStringToTerminal (STR str)
<b>Parameters</b>	STR str—is any character string consisting of letters, numbers, and symbols.
<b>Return Type</b>	BOOL

## LookForString Method

Looks for a string (as it displays in the STR str argument) in the incoming data stream from the host. This method creates a new item with the same name as STR. To find the result of the search, use the event number returned, and call GetEventStatus using this number. To delete the event, use this same number with RemoveEvent.

<b>Syntax</b>	LookForString (STR str)
<b>Parameters</b>	STR str—is any character string consisting of letters, numbers, and symbols.
<b>Return Type</b>	SHORT EVENT—representing a number that is initially false.

## GetEventStatus Method

Returns the status of an event. To remove the event, use the event number returned with RemoveEvent.

<b>Syntax</b>	GetEventStatus ( <i>short event</i> )
<b>Parameters</b>	<i>short event</i> —is the number between 0 and 65535 returned in the LookForString method.
<b>Return Type</b>	BOOL

## RemoveEvent Method

Removes the event for which you are looking.

<b>Syntax</b>	RemoveEvent ( <i>short event</i> )
<b>Parameters</b>	<i>short event</i> —is the number between 0 and 65535 returned in the LookForString method.
<b>Return Type</b>	VOID

## StartDataSave Method

Starts saving the data stream for the ReadData method to read.

<b>Syntax</b>	StartDataSave ()
<b>Parameters</b>	This method has no parameters.
<b>Return Type</b>	VOID

## ReadData Method

Reads data that you captured using the StartDataSave method.

<b>Syntax</b>	ReadData
<b>Parameters</b>	This method has no parameters.
<b>Return Type</b>	STRING

## StopDataSave Method

Stops saving the data stream to ReadData.

<b>Syntax</b>	StopDataSave
<b>Parameters</b>	This method has no parameters.
<b>Return Type</b>	VOID

## 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

## ***Generic Properties and Methods***

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

### Properties

- Interpret Property
- BackSpaceDelete Property
- NewLine Property
- LocalEcho Property
- DisplayLines Property
- SmoothScroll Property
- History Property
- CaptureMode Property
- CurrentRow Property
- Autowrap Property
- WarningBell Property
- TermStringId Property
- Online Property
- DisplayWidth80 Property
- JumpScrollNumber Property
- HistorySize Property
- CaptureDestination Property
- CurrentCol Property

## Methods

- ReadKeyboardFile Method
- SetAttributeColor Method
- SetAttributeUsage Method

## Interpret Property

Interprets or displays control codes. The default is TRUE.

**Syntax**              Interpret

**Parameters**        This property has no parameters.

**VarType**            BOOL

## Autowrap Property

Determines if lines wrap or not. The default is FALSE.

**Syntax**              AutoWrap

**Parameters**        This property has no parameters.

**VarType**            BOOL

## BackSpaceDelete Property

Determines if the backspace key sends a delete command or a backspace command. TRUE sends delete, and FALSE sends backspace.

**Syntax**              BackSpaceDelete

**Parameters**        This property has no parameters.

**VarType**            BOOL

## WarningBell Property

Determines if a bell rings or not. The default is TRUE.

**Syntax**            WarningBell

**Parameters**        This property has no parameters.

**VarType**            BOOL

## NewLine Property

Determines if the Return key sends a CRLF command. True sends a CRLF command, and FALSE sends a CR only. The default is TRUE.

**Syntax**            NewLine

**Parameters**        This property has no parameters.

**VarType**            BOOL

## TermStringId Property

Determines the string ID of a Terminal. The string is different for each terminal type.

**Syntax**            TermStringId

**Parameters**        This property has no parameters.

**VarType**            STR

## LocalEcho Property

Determines if typed characters are displayed locally by the client terminal. The default is FALSE.

**Syntax**              LocalEcho

**Parameters**        This property has no parameters.

**VarType**            BOOL

## Online Property

Determines if typed characters are forwarded to the host, or just reflected on the client terminal. The default is TRUE.

**Syntax**              Online

**Parameters**        This property has no parameters.

**VarType**            BOOL

## DisplayLines Property

Determines the number of lines used in the terminal's display (range or 1-100). The default number of lines is 24.

**Syntax**              DisplayLines

**Parameters**        This property has no parameters.

**VarType**            SHORT

## DisplayWidth80 Property

Determines the width, in columns, of the terminal's display. TRUE is 80 columns, and FALSE is 132 columns. The default is TRUE.

**Syntax**                  `DisplayWidth80`

**Parameters**              This property has no parameters.

**VarType**                `BOOL`

## SmoothScroll Property

Determines how the terminal scrolls through data. TRUE scrolls smoothly through the data, and FALSE jump scrolls through it. The default is FALSE.

**Syntax**                  `SmoothScroll`

**Parameters**              This property has no parameters.

**VarType**                `BOOL`

## JumpScrollNumber Property

Determines the number of lines that are jumped when scrolling through data. The value can be between 2-10.

**Syntax**                  `JumpScrollNumber`

**Parameters**              This property has no parameters.

**VarType**                `SHORT`

## History Property

Determines if the scroll back history buffer is used. TRUE keeps the scroll back buffer, and FALSE does not. The default is TRUE.

**Syntax** History

**Parameters** This property has no parameters.

**VarType** BOOL

## HistorySize Property

Determines the size of the scroll back history buffer. The number can be between 1-1000. The default is 200.

**Syntax** HistorySize

**Parameters** This property has no parameters.

**VarType** SHORT

## CaptureMode Property

Determines if the incoming data is captured to a file. The values are as follows:

- 0=off
- 1=on
- 2=on with host control

**Syntax** CaptureMode

**Parameters** This property has no parameters.

**VarType** SHORT

## CaptureDestination Property

Determines the destination file of the captured output.

**Syntax**      CaptureDestination

**Parameters**      This property has no parameters.

**VarType**      STR

## CurrentRow Property

Determines the current row on the page.

**Syntax**      CurrentRow

**Parameters**      This property has no parameters.

**VarType**      SHORT

## CurrentCol Property

Determines the current column on the page.

**Syntax**      CurrentCol

**Parameters**      This property has no parameters.

**VarType**      SHORT

## ReadKeyboardFile Method

Reads a keyboard file to change the mapping of the PC key combinations needed to forward host keys to the host.

**Note:** The HKeyMap program must create these keyboard files. This program can be started only in Telnet. The HKeyMap program is not DDE enabled. As a result, this item can read only existing keyboard files.

<b>Syntax</b>	ReadKeyboardFile( <i>STR filename</i> )
<b>Parameters</b>	<i>STR filename</i> —is the name and path of the keyboard file you want to read.
<b>VarType</b>	BOOL

## SetAttributeColor Method

Sets the foreground and background color for an attribute.

**Note:** Attributes can be ORed together.

<b>Syntax</b>	SetAttributeColor( <i>unsigned char attr, unsigned char fgcolor, unsighned char bgcolor</i> )
<b>Parameters</b>	<i>unsigned char attr</i> —is the attribute for which you want to set the color. <i>unsigned char fgcolor</i> —is the foreground color you want to set. <i>unsigned char bgcolor</i> —is the background color you want to set.
<b>Return Type</b>	VOID

## SetAttributeUsage Method

Sets the display for the attribute you specify. You can change the foreground color or the actual screen attribute characteristic. For example, you can change all bold characters to display underlined, or use any other valid attribute. The screen updates to reflect this change.

**Note:** Attributes can not be ORed together.

<b>Syntax</b>	SetAttributeUsage( <i>unsigned char attr, realAttrib, color</i> )
<b>Parameters</b>	<i>unsigned char attr</i> —is the name of the attribute. <i>realAttrib</i> —is the attribute characteristic you want to set. <i>color</i> —is the foreground color you want to set for the attribute.
<b>Return Type</b>	VOID

## ***VT220 and VT320 Properties***

The properties listed below are available for either the VT220 or VT320 emulation modes.

### **VT220**

- UFLock Property
- UDKLock Property
- EmulationMode Property
- NationalSet Property
- Multinational Property

### **VT320**

- UserPrefCharSetISO Property
- StatusLine Property

## UFLock Property

Determines if user features are locked or not. If TRUE, the user features are locked, and if FALSE, the user features are not locked.

**Syntax** UfLock

**Parameters** This property has no parameters.

**VarType** BOOL

## UDKfLock Property

Determines if the user-defined keys are locked or not. If TRUE, the user-defined keys are locked, and if FALSE, the user-defined keys are not locked.

**Syntax** UDKfLock

**Parameters** This property has no parameters.

**VarType** BOOL

## EmulationMode Property

Determines the emulation mode. The following list details the possible emulation modes and their values:

- VT52—0
- VT100—1
- VT200 7bit—2
- VT200 8bit—3

**Syntax** EmulationMode

**Parameters** This property has no parameters.

**VarType** SHORT

## NationalSet Property

Determines which National Character Set is used. The following list details the available National Character Set modes and their values:

- North American—0
- UK—1
- Dutch—2
- Finish—3
- French—4
- French Canadian—5
- German—6
- Italian—7
- Norwegian/Danish—8
- Portuguese—9
- Spanish—10
- Swedish—11
- Swiss—12

**Syntax**              NationalSet

**Parameters**        This property has no parameters.

**VarType**          SHORT

## Multinational Property

Determines if DEC multinational characters are used. If TRUE, DEC characters are used, and if FALSE, only the National Character Set is used.

**Note:** If this value is TRUE and UserPrefCharSetISO is TRUE, then ISO Latin 2 is the character set used.

<b>Syntax</b>	Multinational
<b>Parameters</b>	This property has no parameters.
<b>VarType</b>	BOOL

## UserPrefCharSetISO Property

Determines which character set is used. If TRUE, ISO LATIN 1 is used, and if FALSE, DEC multinational is used.

<b>Syntax</b>	UserPrefCharSetISO
<b>Parameters</b>	This property has no parameters.
<b>VarType</b>	BOOL

## StatusLine Property

Determines how terminal information displays. The following list details the possible StatusLine settings and their values:

- Off—0
- On—1
- host writable—2

<b>Syntax</b>	StatusLine
<b>Parameters</b>	This property has no parameters.
<b>VarType</b>	SHORT

## ***WYSE50 and WYSE60 Properties***

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

- ApplicationMode Property
- BlockEndUSCR Property
- AutoScroll Property
- ActivePage Property
- EditModeLocal Property
- AutoPageMode Property
- CommMode Property
- ActivePageSize Property

### **ApplicationMode Property**

Determines if keys are in application mode. TRUE means the keys are in application mode, and FALSE means they are not. The default mode is FALSE.

**Syntax**              ApplicationMode

**Parameters**        This property has no parameters.

**VarType**            BOOL

### **EditModeLocal Property**

Determines if a remote file can be edited on the client terminal. If TRUE, the client is working in local mode and can edit the file, and if FALSE, it is not in local mode and cannot edit the file. The default mode is FALSE.

**Syntax**              EditModeLocal

**Parameters**        This property has no parameters.

**VarType**            BOOL

## BlockEndUSCR Property

Determines if the USCR is used as the block terminator. If TRUE, USCR is used, and if FALSE, it is not.

**Syntax** BlockEndUSCR

**Parameters** This property has no parameters.

**VarType** BOOL

## AutoPageMode Property

Determines if the client terminal moves to the next page. If TRUE, the client terminal moves to the next page automatically; if FALSE, it does not.

**Syntax** AutoPageMode

**Parameters** This property has no parameters.

**VarType** BOOL

## AutoScroll Property

Determines if the page scrolls so that new text appears on a new bottom line. If TRUE, text automatically scrolls; if FALSE, text wraps to the top of the page.

**Syntax** AutoScroll

**Parameters** This property has no parameters.

**VarType** BOOL

## CommMode Property

Determines the current communication mode. The following list details the possible communication modes and their values:

- Full Duplex—0
- Half Duplex—1
- Block—2

**Syntax**                    CommMode

**Parameters**                This property has no parameters.

**VarType**                    SHORT

## ActivePage Property

Determines the number of Wyse pages that are active. You can have any number of active pages between 0 and 3.

**Note:** Changing this value changes the state of PageObject to correspond to this new active page. As a result, PageObject is the only current active page in WYSE50 or WYSE60 mode.

**Syntax**                    ActivePage

**Parameters**                This property has no parameters.

**VarType**                    SHORT

## ActivePageSize Property

Determines the page size of the active page. You can have active page size in the range 0 to100.

**Note:** Each page can have a different size.

**Syntax**              ActivePageSize

**Parameters**          This property has no parameters.

**VarType**              SHORT

## 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
- GetCharacterAttrib Method
- GetCharacterBGColor Method
- SetCharacterAttrib Method
- SetCharacterBGColor Method
- GetCharacterString Method
- GetCharacterFGColor Method
- SetCharacter Method
- SetCharacterFGColor Method

## FGColor Property

Determines the current foreground color given to all new characters.  
Default is Nocolor.

**Syntax** FGColor

**Parameters** This property has no parameters.

**VarType** SHORT

## BGColor Property

Determines the current background color given to all new characters.  
Default is Nocolor.

**Syntax** BGColor

**Parameters** This property has no parameters.

**VarType** SHORT

## Attribute Property

Determines the current attribute given to all new characters. Attributes can be ORed together to generate a whole attribute.

**Syntax** Attribute

**Parameters** This property has no parameters.

**VarType** SHORT

## GetCharacter Method

Returns the character from the r, c coordinates. For example GetCharacter(2,75) returns the character from the second row and the 75th column.

<b>Syntax</b>	<code>GetCharacter(short r, short c)</code>
<b>Parameters</b>	<i>short r</i> —is the number of the row where the character is located. <i>short c</i> —is the number of the column where the character is located.
<b>VarType</b>	UNSIGNED CHAR

## GetCharacterString Method

Returns the characters from the r, c coordinate pair. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, StrBuf=Telnet.GetCharString (1,1,24,80) returns all characters between the r, c coordinates of (1,1) and (24,80).

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

<b>Syntax</b>	<code>GetCharString(short r, short c, short r, short c)</code>
<b>Parameters</b>	<i>short r short c</i> —is the row and column where the string selection begins. <i>short r short c</i> —is the row and column where the string selection finishes.
<b>VarType</b>	UNSIGNED CHAR

## GetCharacterAttrib Method

Returns the attribute of a character. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, GetCharacterAttrib(1,1) returns the attribute of the character between the r, c coordinates of (1,1).

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

**Syntax**`GetCharAttrib(short r, short c)`

*short r*—is the number of the row where the character is located.

**Parameters**

*short c*—is the number of the column where the character is located.

**VarType**`UNSIGNED CHAR`

## GetCharacterFGColor Method

Returns the foreground color of a character. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, GetCharacterFGColor(1,1) returns the foreground color of the character between the r, c coordinates of (1,1).

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

**Syntax**`GetCharacterFGColor(short r, short c)`

*short r*—is the number of the row where the character is located.

**Parameters**

*short c*—is the number of the column where the character is located.

**VarType**`UNSIGNED CHAR`

## GetCharacterBColor Method

Returns the background color of a character. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, GetCharacterBColor(1,1) returns the background color of the character between the r, c coordinates of (1,1).

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

**Syntax**

```
GetCharacterBColor(short r, short c)
```

**Parameters**

*short r*—is the number of the row where the character is located.

*short c*—is the number of the column where the character is located.

**VarType**

UNSIGNED CHAR

## SetCharacter Method

Changes the character at the r, c coordinates. Changes the character at the r, c coordinates. For example, SetCharacter(1,1,"q") changes the character between the r, c coordinates of (1,1) to "q".

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

**Syntax**

```
SetCharacter(short r, short c unsigned char ch)
```

*short r*—is the number of the row where the character is located.

**Parameters**

*short c*—is the number of the column where the character is located.

*unsigned char ch*—is the character you want to set.

**VarType**

VOID

## SetCharacterAttrib Method

Sets a character attribute at the r, c coordinates. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, SetCharacterAttrib(1,1,"0x40") changes the character attribute between the r, c coordinates of (1,1) to "0x40"--a private attribute.

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

### Syntax

```
SetCharacterAttrib(short r, short c unsigned char attrib)
```

*short r*—is the number of the row where the character is located.

### Parameters

*short c*—is the number of the column where the character is located.

*unsigned char attrib*—is the character attribute you want to set.

### VarType

VOID

## SetCharacterFGColor Method

Sets the foreground color of the character at the r, c coordinates. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, SetCharacterFGColor(1,1,7) changes the character's foreground color between the r, c coordinates of (1,1) to 7--a dark white color.

**Note:** To access the history buffer, specify a number between 0 and -HistorySize+1.

<b>Syntax</b>	<code>SetCharacterFGColor(short r, short c unsigned char color)</code>
<b>Parameters</b>	<p><i>short r</i>—is the number of the row where the character is located.</p> <p><i>short c</i>—is the number of the column where the character is located.</p> <p><i>unsigned char color</i>—is the character color you want to set.</p>
<b>VarType</b>	VOID

## SetCharacterBGColor Method

Sets the background color of the character at the *r, c* coordinates. The top left corner of the screen has a value of *r,c*=1,1, and the bottom right is *r,c*=24,80, or *r,c*=24,132 depending upon the current column width. For example, SetCharacterBGColor(1,1,7) changes the character's background color between the *r, c* coordinates of (1,1) to 7--a dark white color.

<b>Syntax</b>	<code>SetCharacterBGColor(short r, short c unsigned char color)</code>
<b>Parameters</b>	<p><i>short r</i>—is the number of the row where the character is located.</p> <p><i>short c</i>—is the number of the column where the character is located.</p> <p><i>unsigned char color</i>—is the character color you want to set.</p>
<b>VarType</b>	VOID

## 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 snippet 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
```

## **Appendix**

---

### **Accessibility and Technical Support**



## 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.

<b>Hummingbird Ltd. 1 Sparks Avenue, Toronto, Ontario, Canada M2H 2W1</b>		
	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:	support@hummingbird.com	
FTP:	ftp.hummingbird.com	
Web Support:	support.hummingbird.com/customer	
Web Site:	www.hummingbird.com www.connectivity.hummingbird.com	



---

# Index

## Numerics

- 1KPacket ..... 632  
3270/5250 special sequences ..... 191

## A

- accessibility, general ..... 1531  
Account function ..... 1429  
Account property ..... 1442  
acPassword property ..... 833  
ActionOnExist CfgVT property ..... 106  
Activate method ..... 26  
ActivePage property ..... 1520  
ActivePageSize property ..... 1521  
AddFeature ..... 208  
AddFormFeed ..... 722  
AddOIAToCapture property ..... 269  
Advise commands ..... 1396  
Advise message ..... 1394  
ALA Cfg3270 property ..... 168  
ALADisplay Mode Cfg3270 property ..... 168  
ALAInputMode Cfg3270 property ..... 168  
ALAKeyboardProfileName  
    Cfg3270 method ..... 168  
Allow Connect Physical ..... 1387  
AllowAIDKeyRepeat ..... 419  
AllowClose property ..... 27  
AllowDiac ..... 422  
AllowEmuTracing property ..... 288  
AllowErrorRestart property ..... 271  
AllowUpdates property ..... 27  
Alphanumeric ..... 1324  
AlternateScreen ..... 344

- AlwaysAutoskip ..... 483  
AlwaysAutoSkip Cfg3270 property ..... 106  
AlwaysPromptForHostName property ..... 835  
Answerback CfgVT property ..... 107  
AnswerBack property ..... 1061  
API commands ..... 1410  
APIs ..... 4  
APL (IHEProfileGraphics) ..... 406  
APL (IOhioOIA) ..... 1325  
APLInputMode property ..... 1063  
Append ..... 553  
AppendFile function ..... 1417  
AppendFile method ..... 1459  
Application Name field ..... 1396  
Application object methods and properties .. 10  
Application property ..... 76, 1469  
ApplicationMode property ..... 1518  
Area methods ..... 28  
Area object methods and properties ..... 76  
AreaCode ..... 837  
AreaCode CfgVT property ..... 107  
ASCII ..... 555  
AsciiToHost method ..... 956  
ASCIIType setting ..... 1420  
Assignment group ..... 169  
AttnFormat ..... 233  
AttentionFormat ..... 1204  
AttnFormat ..... 273  
Attr property ..... 92  
Attribute ..... 1340  
Attribute property ..... 1522  
attributes ..... 1494

Auto Sync.....	1387	Button property .....	452
Auto Unload.....	1385	Bytes property .....	30
AutoCC.....	557	<b>C</b>	
AutoClearMonitor.....	559	CanChangeScreen .....	1298
AutoClearXferMonitor Cfg3270 and CfgVTproperty .....	108	CanChangeScreen property.....	1067
AutoCopy .....	485	Capture property.....	30, 317
AutoCopyKeepSelection .....	487	CaptureDestination property .....	1512
AutoCopySelectedText Cfgxxxx property ..	108	CaptureMode .....	234, 1069
AutoEndMacroName property.....	839	CaptureMode property .....	1511
AutoMacro Cfgxxxx property .....	109	CaptureOIA property .....	31
AutoMacroName.....	841	CellCopyMode .....	1071
automation objects .....	1433	CellDelimited .....	235
HclFtp.Engine .....	1433	CellDelimited (IHEProfileEdit).....	491
AutoPageMode property.....	1519	Cfg3270 methods and properties.....	99
AutoRunMacroDelayTime property .....	843	Cfg3270 object .....	103
AutoScroll property.....	1519	Cfg5250 methods and properties.....	99
AutoSignOn property.....	845	Cfg5250 object .....	104
AutoUnlockKeyboard Cfg3270 property....	109	CfgVT methods and properties .....	99
AutoWrap CfgVT property.....	110	CfgVT object .....	105
AutoWrap property.....	1169, 1507	ChangeHostDir function .....	1412
<b>B</b>		ChangeHostDir method .....	1452
Backspace key interpretation.....	234	changing a title .....	1527
BackSpaceDelete property.....	1507	CharacteBGColorAt item .....	1493
BackspaceKeyInterpretation property .....	847	CharacterAt item.....	1492
Behaviour.....	261, 262	CharacterAttributeAt item.....	1492
BellMargin .....	489	CharacterFGColorAt item .....	1493
BellMargin Cfgxxxx property .....	110	CharacterSet .....	347
BFPress methods .....	29	CharacterSpacing .....	709
BFStatus property .....	29	CharSet.....	1206
BGColor property.....	1522	ChooseTerminalFont.....	196
BigToolbar property .....	450	ClearAllTabStops Cfgxxxx method .....	112
BinaryType setting.....	1421	ClearAllTabStops method .....	482
BitMode CfgVT property .....	111	ClearPassword.....	275
BlinkToItalic .....	669	ClearScreenOnSizeChange property .....	168
BlinkToItalic Cfgxxxx property.....	111	ClearSel method .....	958
BlkSiz .....	561	clients.....	1410
BlockEndUSCR property .....	1519	source code .....	1410
BlockSelect.....	446	ClipFormatBitmap .....	493
Bottom property .....	78	ClipFormatCSV .....	495
BSIsDel CfgVT property .....	112	ClipFormatHE .....	497
BufRows.....	1065	ClipFormatPasteLink.....	499
		ClipFormatRTF.....	501

ClipFormatText .....	503	
Close method.....	31	
CloseAll method.....	22	
CloseSession .....	1284	
CMS-specific options on upload.....	189	
CodePage .....	1208	
coding OLE Automation.....	1433	
Color .....	261, 1358	
Color property .....	276	
ColorDisplay Cfg5250 property .....	113	
colors.....	1494	
Column .....	1355	
Columns .....	1319	
Columns property .....	32	
ColumnSeparators.....	672	
COM objects.....	207, 208 relationship .....	208
command syntax .....	1398	
Receive File.....	1398	
commands	1396, 1397, 1399, 1400, 1406, 1410	
Advise .....	1396	
API.....	1410	
Execute .....	1397	
Hummingbird Basic Language.....	1410	
Poke.....	1399	
Request.....	1400	
Send File .....	1398	
System Topic.....	1406	
CommCheckCode.....	1326	
CommMode property .....	1520	
CompressBlankLinesInScrollbar CfgVT property .....	113	
ConcealAnswerback CfgVT property .....	114	
configuration tips .....	1391	
ConfigurationResource .....	1300	
Connect (EHLLAPI) .....	1368	
Connect (IHETerminal).....	196	
Connect (IOhioSession).....	1294	
Connect method.....	32	
Connect Presentation Space (1).....	1365	
Connect to Host method.....	1433	
ConnectBy .....	236	
ConnectBy (IHEParser) .....	1072	
ConnectBy (IHEProfileConnection) .....	849	
ConnectBy (IHETerminal) .....	199	
ConnectBy property.....	33	
Connected (IHETerminal) .....	200	
Connected (IHETransport) .....	1210	
Connected property .....	1445	
ConnectErrorStatus property .....	33, 1075	
connecting to.....	1433	
hosts .....	1433	
connecting to Telnet with .....	1473, 1496	
DDE .....	1473	
OLE.....	1496	
Connection property .....	278	
connections .....	1474	
establishing .....	1474	
ConnectionStatus.....	1212	
ConnectionStatus item .....	1479	
ConnectionStatus property .....	1500	
ConnectPS (1).....	1365	
ConnectRC property.....	34, 1077	
ConnectRlogin item .....	1480	
ConnectRlogin method .....	1501	
ConnectTelnet item .....	1480	
ConnectTelnet method .....	1502	
ConnectTimeout Cfgxxxx property .....	114	
ConnectToHost function .....	1413	
ConnectToHost method.....	1455	
Constant group .....	170	
ConStatus.....	237	
conversation .....	1403	
HostExplorer .....	1403	
Microsoft Excel .....	1403	
Word for Windows .....	1403	
conversation topics .....	1474	
Conversion group .....	170	
Convert Nulls .....	1388	
Convert Nulls Cfg3270 property .....	115	
ConvertNulls (IHEParser) .....	1079	
ConvertNulls (IHEProfileEdit) .....	505	
ConvertPosToRowCol.....	1369	
ConvertRowColToPos.....	1369	
Copy method .....	79	
CopyFieldToString.....	1370	

CopyOIA .....	1370	CurrentCol property .....	1512
CopyPS .....	1371	CurrentFGColor item .....	1491
CopyPSToString.....	1371	CurrentHost property .....	11
CopyStringToField .....	1372	CurrentKeyboard .....	424
CopyStringToPS.....	1372	CurrentLanguage .....	470
Count (IOhioFields).....	1336	CurrentRow property .....	1512
Count (IOhioSessions).....	1290	Cursor (IHEProfile) .....	279
Count property .....	22, 1469	Cursor (IOhioScreen) .....	1319
Country CfgVT property.....	115	Cursor example .....	1400
Country property.....	851	Cursor property .....	35
CountryCode .....	853	CursorKeyMode CfgVT property.....	116
CountryID .....	855	CursorMode .....	700
CreateOhioPosition .....	1354	CursorMode Cfgxxxx property .....	117
CreateValidFATFiles setting.....	1422	CursorPosition item .....	1484
creating .....	17, 1380, 1403, 1410, 1433	CursorRC method .....	35
DDE keystroke mnemonics .....	1403	CursorSelectMode Cfgxxxx property .....	168
FTP Session object .....	1433	CursorType .....	702
OLE scripts .....	1433	CursorType Cfgxxxx property .....	117
scripts .....	1410	CustomModel property .....	364
sessions .....	17	CustomModelCols property .....	366
special key strings.....	1380	CustomModelRows property .....	368
CRLF .....	563	CustomTransferTable property .....	565
CRTToCRLF CfgVT property.....	116	Cut method .....	80
CSVCopyEmptyFields property.....	507	CutChar .....	511
CSVTrimFields property .....	509	Cutmode .....	237
CTRLZStatus setting .....	1427	Cutmode (IHEParser) .....	1081
CurrentAttribute item .....	1491	Cutmode (IHEProfileEdit) .....	513
CurrentBGCOLOR item .....	1491		

**D**

data type.....	1363
HEPARSER_FEATURE .....	227
HEPARSER_VALUE.....	232
HETRANSPORT_FEATURE.....	232
HOSTEX_ATN_FORMAT .....	233
HOSTEX_BACKSPACE_KEY_	
INTERPRETATION .....	234
HOSTEX_CAPTURE_MODE.....	234
HOSTEX_CELL_DELIMITED .....	235
HOSTEX_CON_STATUS.....	237
HOSTEX_CONNECT_BY.....	236
HOSTEX_CUT_MODE.....	237
HOSTEX_DEVICE_TYPE.....	238
HOSTEX_ENCRYPTED .....	238
HOSTEX_ENTER_KEY_	
INTERPRETATION .....	239
HOSTEX_FIELD_ATTR_	
REPLACEMENT .....	239
HOSTEX_FUNCTION_KEY.....	240
HOSTEX_GRAPHICS_CELLSIZE....	241
HOSTEX_GRAPHICS_CURSOR_	
TYPE .....	241
HOSTEX_GRAPHICS_MODEL.....	242
HOSTEX_HOTSPOT_DISPLAY .....	243
HOSTEX_HOTSPOT_MOUSE_	
ACTIVATION.....	243
HOSTEX_INSERT_KEY_STYLE .....	244
HOSTEX_KEYBOARD_	
BUFFER_MODE.....	244
HOSTEX_KEYBOARD_TYPE .....	245
HOSTEX_LINEMODE .....	245
HOSTEX_NEXT_FIELD_KEY .....	246
HOSTEX_OIA_DISPLAY .....	247
HOSTEX_PASTE_MODE .....	247
HOSTEX_PRINT_TARGET .....	249
HOSTEX_PRINTFILE_MODE .....	248
HOSTEX_RESIZE_BEHAVIOR.....	249
HOSTEX_SAVE_OPTIONS.....	250
HOSTEX_SECURITY_OPTIONS.....	250
HOSTEX_SELECTION_MODE.....	251
HOSTEX_STATUS_LINE_MODE ....	251
HOSTEX_SWITCHSCREENTYPE ....	252

HOSTEX_TELNETECHO .....	253
HOSTEX_TERM_MODEL .....	254
HOSTEX_TERMINAL_ID.....	253
HOSTEX_TOGGLE_RECEIVE .....	255
HOSTEX_TPRINT_OUTPUT.....	256
HOSTEX_TRANSFER.....	257
HOSTEX_TRANSFER_HOSTTYPE ..	257
HOSTEX_TRANSFER_	
INITIALACTION .....	258
HOSTEX_TRANSFER_	
RECORDFORMAT .....	258
HOSTEX_TRANSFER_TARGET .....	259
HOSTEX_TRANSFER_TYPE .....	259
HOSTOVERWRITE_BEHAVIOUR....	261
OHIO_COLOR.....	1358
OHIO_DIRECTION.....	1358
OHIO_EXTENDED .....	1359
OHIO_EXTENDED_COLOR.....	1359
OHIO_EXTENDED_HILITE .....	1360
OHIO_FIELD.....	1360
OHIO_INPUTINHIBITED.....	1361
OHIO_OWNER.....	1361
OHIO_PLANE .....	1362
OHIO_STATE.....	1362
OHIO_TYPE.....	1363
OHIO_UPDATE.....	1363
OLE_COLOR .....	261
PC_OVERWRITE_BEHAVIOUR .....	262
Data Type group .....	171
data types.....	948, 1175, 1282, 1470
OHIO.....	1357
Parser objects.....	1175
Profile object.....	948
TFirewallType .....	1470
TLowerCaseType.....	1470
Transport objects.....	1282
TTransferType .....	1470

DDE	1393, 1394, 1395, 1402, 1404, 1473, 1474		
Advise message	1394	Device3279 property .....	36
Execute message	1394	DeviceName .....	857
Poke message	1395	DeviceType .....	238
Request message	1395	DeviceType (IHETransport) .....	1216
sample code	1404	dialog .....	175
terminology	1402	methods and statements .....	175
topics	1474	Differences between Hummingbird Basic and	
DDE and OLE	1472	WinWrap Basic .....	169
introducing	1472	DirectToModem .....	859
DDE conversation	1396	DirectToModem CfgVT property .....	120
Application Name field	1396	Disconnect (EHLLAPI) .....	1372
Topic field	1396	Disconnect (IHETerminal) .....	197
DDE group	172	Disconnect (IOhioSession) .....	1295
DDE keystroke mnemonics	1403	Disconnect item .....	1481
DDE sample code	1404	Disconnect method .....	36, 1502
DDE terminology	1402	DisconnectFromHost function .....	1414
DDEServerName	281	DisconnectFromHost method .....	1455
Declaration group	172	Display .....	283
default	1407	Display3DBorder Cfgxxxx property .....	120
locations for user files	1407	DisplayAbortDlg .....	725
Default		DisplayAttr Cfg3270 property .....	121
DownloadPath	567	DisplayBorder .....	911
DefaultHeight CfgVT property	118	DisplayControlCodes CfgVT property .....	121
DefaultProtocol	569	DisplayCrossHairCursor property .....	704
DefaultRecvDir	571	DisplayInOIA property .....	674
DefaultRecvDir CfgVT property	118	DisplayLines property .....	1509
DefaultSystemType property	1451	DisplayPrintDlg .....	727
DefaultUploadPath	573	DisplayRowCol .....	676
DefaultWidth CfgVT property	119	DisplayRowCol Cfgxxxx property .....	122
DefType statement	174	DisplayStyle property .....	818
DeinitFTP function	1411	DisplayUpperCase .....	678
Delete method	81	DisplayUpperCase Cfgxxxx property .....	122
DeleteHostFile function	1417	DisplayWidth80 property .....	1510
DeleteHostFile method	1452	DLLs .....	1364, 1367
DelSession method	168	EHLLAPI .....	1364, 1367
DelTree function	1419	WinHLLAPI .....	1364
DelTree method	1451	DownloadHostName .....	575
DetectChainedIO property	349	DownloadPCFileName .....	577
Dev7171Terminal Cfg3270 property	119	DownloadTranslate property .....	579
development files	1389, 1392	DropExtension property .....	1447
EHLLAPI	1389	DropVersion property .....	1447
WinHLLAPI	1392	Dynamic Data Exchange .....	1393

**E**

EAB property .....	37
EBS files .....	1410
opening.....	1410
printing.....	1410
running.....	1410
.ebs files.....	1410
opening.....	1410
printing.....	1410
running.....	1410
Edit property .....	284
EditModeLocal property .....	1518
EditSessionProperties .....	198
EHLLAPI .....	1364, 1367, 1383
calls .....	1367
Visual Basic interface .....	1383
EHLLAPI and WinHLLAPI support .....	1364
EHLLAPI calls..	1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1382
EHLLAPICConnect .....	1368
EHLLAPICConvertPosToRowCol .....	1369
EHLLAPICConvertRowColToPos .....	1369
EHLLAPICopyFieldToString .....	1370
EHLLAPICopyOIA .....	1370
EHLLAPICopyPS .....	1371
EHLLAPICopyPSToString .....	1371
EHLLAPICopyStringToField .....	1372
EHLLAPICopyStringToPS .....	1372
EHLLAPIDDisconnect .....	1372
EHLLAPIFindFieldPosition .....	1373
EHLLAPIGetRowString .....	1373
EHLLAPIGetVersion .....	1374
EHLLAPIPause .....	1374
EHLLAPIQueryCursorLocation .....	1374
EHLLAPIQueryFieldAttribute .....	1375
EHLLAPIQuerySessions .....	1375
EHLLAPIQuerySessionStatus .....	1376
EHLLAPIReceiveFile .....	1376
EHLLAPIRelease .....	1377
EHLLAPIReserve .....	1377
EHLLAPIReset .....	1377
EHLLAPISearchField .....	1378
EHLLAPISearchPS .....	1378
EHLLAPISendFile .....	1379
EHLLAPISendKey .....	1379
EHLLAPISetCursorPosition .....	1380
EHLLAPISetSessionParameters .....	1380
EHLLAPIWait .....	1382
EHLLAPI development files .....	1389
EHLLAPI support .....	1388
VT and NVT modes .....	1388
EHLLAPICConnect .....	1368
EHLLAPICConvertPosToRowCol .....	1369
EHLLAPICConvertRowColToPos .....	1369
EHLLAPICopyFieldToString .....	1370
EHLLAPICopyOIA .....	1370
EHLLAPICopyPS .....	1371
EHLLAPICopyPSToString .....	1371
EHLLAPICopyStringToField .....	1372

EHLLAPICopyStringToPS.....	1372	
EHLLAPIDisconnect.....	1372	
EHLLAPIFindFieldPosition .....	1373	
EHLLAPIGetRowString.....	1373	
EHLLAPIGetVersion.....	1374	
EHLLAPIPause.....	1374	
EHLLAPIQueryCursorPosition .....	1374	
EHLLAPIQueryFieldAttribute .....	1375	
EHLLAPIQuerySessions .....	1375	
EHLLAPIQuerySessionStatus .....	1376	
EHLLAPIReceiveFile .....	1376	
EHLLAPIRelease .....	1377	
EHLLAPIReserve.....	1377	
EHLLAPIReset.....	1377	
EHLLAPISearchField .....	1378	
EHLLAPISearchPS .....	1378	
EHLLAPISendFile .....	1379	
EHLLAPISendKey .....	1379	
EHLLAPISetCursorPosition .....	1380	
EHLLAPISetSessionParameters.....	1380, 1381 valid options.....	1381
EHLLAPITimeout .....	1382	
EMailAddress property.....	1465	
Emulation object .....	1498, 1506	
Emulation options.....	1486	
Emulation Type Topic items.....	1483	
EmulationMode property .....	1515	
EmulationObject property .....	1501	
EmulationType item.....	1478	
EmulationType property.....	1500	
EmuTraceFilename .....	286	
EnableAPI .....	1083	
EnableEMode .....	1218	
EnableEMode property .....	861	
EnableEvents property.....	828	
EnableHLLAPITracing.....	290	
EnableHotspots property .....	821	
EnableInfiniteRetries property .....	863	
EnablePrinterTimeout.....	1085	
EnableSSH property .....	1220	
EnableTracing.....	1222	
EnableTracing property.....	292	
EnableWorkspaceBackgroundBitmap .....	914	
End .....	1342	
Enter key interpretation .....	239	
Entering control sequences .....	194	
EnterKeyInterpretation property.....	865	
EntryAssist.....	515	
EntryAssist Cfgxxxx property .....	123	
envelope types--codes for .....	1198	
Error Handling group .....	176	
establishing connections .....	1474	
Events property .....	293	
examples.....	1400, 1401, 1402, 1406 Cursor.....	1400
file transfer .....	1401	
Fnn.....	1402	
Keystroke .....	1400	
Start Session.....	1406	
exceptions.....	1390	
NVT mode.....	1390	
ExecReturn item.....	1479	
Execute commands .....	1397	
Execute message .....	1394	
execute syntax .....	1473	
ExitAll method .....	12	
ExtAttr property .....	94	
Extended.....	1359	
ExtendedColor .....	1359	
ExtendedHilite .....	1360	
ExtraOptions .....	581	
<b>F</b>		
examples		
Pnnnn .....	1402	
Pnnnnn .....	1402	
Feature.....	227	
FGColor property .....	1522	
field .....	1360, 1396 Application Name .....	1396
Topic.....	1396	
Field object methods and properties .....	90	
FieldAttrReplacement .....	239	
FieldID method.....	37	
Fields .....	1320	
Fields Collection object.....	38	

File group.....	176
file transfer example .....	1401
file transfer options.....	188
FileExistAction.....	583
FileTransfer.....	295
FileXferProtocol CfgVT property .....	123
FindByPosition .....	1333
FindByString.....	1334
FindFieldPosition .....	1373
FindString .....	1304
FindString method .....	959
FirewallPassword property .....	1450
FirewallServerName property .....	1449
FirewallServerPort property .....	1449
FirewallType property .....	1450
FirewallUserName property .....	1449
flags .....	1366
Flow Control group .....	178
Fnn examples.....	1402
FontLarger method.....	38
Fonts .....	296
FontSmaller method.....	39
ForceAltSize .....	351
ForceAltSize Cfg3270 property .....	124
ForceDefaultSystemType property .....	1451
ForceExactSize Cfgxxxx property.....	124
Formats item.....	1476
FTP API .....	1409, 1410
FTP engine.....	1433
FTP OLE API .....	1432
FTP OLE API object .....	1434, 1435, 1437
IHclFtpEngine.....	1434
IHclFtpSession .....	1435
IHclFtpSessions.....	1437
FTP OLE features .....	1433
FTP sessions.....	1433
creating.....	1433
retrieving .....	1433
FTPActive setting.....	1426
FTPAplicationName Cfgxxxx property ....	168
FTPDIR function.....	1425
FTPDIR method.....	1457
FTPPassive setting .....	1426
Fullname .....	1503
FullName property.....	1439
FullScreenMode .....	916
FunctionKey.....	240
functions .....	1389, 1409
NVT mode.....	1389
<b>G</b>	
General options.....	188
generic .....	1486, 1506
options .....	1486
Get method .....	1433
GetCharacter method .....	1523
GetCharacterAttrib method .....	1524
GetCharacterBColor method .....	1525
GetCharacterFColor method.....	1524
GetCharacterString method .....	1523
GetCurrentDir method.....	12
GetCursorPosition method .....	962
GetData (IOhioField) .....	1339
GetData (IOhioScreen).....	1305
GetErrorText function .....	1432
GetEventStatus method .....	1504
GetFeature method (IHEParser) .....	964
GetFeature method (IHETransport) .....	1179
GetFieldAttribute method .....	966
GetFieldCount method.....	969
GetFieldExtAttribute method .....	971
GetFieldIndex method .....	974
GetFieldLength method .....	976
GetFieldPos method .....	978
GetFieldText method .....	980
GetFile function .....	1415
GetFile method .....	1457
GetFilePath method .....	13
GetLocalFilenameStatus setting.....	1428
GetOption item .....	1484
GetOption item values .....	1486
GetOutputText function.....	1430
GetProfileString method.....	13
GetReplyTimeoutValueSetting.....	1424
GetRowString .....	1373
GetScreenText method .....	982

GetSel method .....	985
GetSelectionArea method .....	987
GetStatusString.....	1182
getting started .....	1472
getting started with DDE.....	1473
getting started with OLE.....	1496
GetTransferType setting.....	1422
GetTree function .....	1418
GetTree method .....	1457
GetValue method .....	989
GetVersion.....	1374
GlobalSettingsPath .....	298
Graphics.....	300
GraphicsCellsize .....	241
GraphicsCursorType .....	241
GraphicsCursorType property .....	409
GraphicsModel .....	242
GraphicsModel (IHEParser) .....	1087
GraphicsModel property .....	411
group. 169, 170, 171, 172, 176, 178, 179, 180, 181, 182, 183, 185, 186, 187	
Assignment.....	169
Constant .....	170
Conversion .....	170
Data Type .....	171
DDE.....	172
Declarations .....	172
Error handling.....	176
File .....	176
Flow Control .....	178
Math .....	179
Miscellaneous.....	180
Object .....	181
Operators .....	182
Settings .....	182
String .....	183
TimeDate.....	185
User Dialog.....	186
User Input .....	187
Variable Info .....	187
<b>H</b>	
HclFtp.Engine .....	1433
help .....	1472
HEPAR3270 object .....	950
methods, properties, and data types specific to.....	950
HEPAR3270/5250 objects.....	950
methods, properties, and data types specific to.....	950
HEPAR3270/5250/VT objects .....	953
methods, properties, and data types specific to.....	953
HEPARSER_FEATURE .....	227
HEPARSER_VALUE.....	232
HEPARVT object.....	952
properties and data types specific to ....	952
HETP3270 object .....	1177
properties specific to.....	1177
HETP3270/5250 Objects .....	1177
properties specific to.....	1177
HETP3270/5250/VT objects.....	1178
properties and data types specific to ..	1178
HETP3270/VT objects .....	1178
properties specific to.....	1178
HETP5250 object .....	1177
properties specific to.....	1177
HETPVT object.....	1177
properties and data types specific to ..	1177
HETRANSPORT FEATURE.....	232
Hex3270.hxl sample .....	219
Hex5250.hxl sample .....	223
Hide method .....	39
HideToolbar method .....	39
HighIntensity .....	1343
HighlightText method .....	40
History property.....	1511
HistoryLines .....	1089
HistorySize property .....	1511
HLLAPIName .....	1091
HLLAPITraceFilename property.....	332
home directory .....	1410
Host (IHEProfileFileTransfer).....	585

Host (IHETerminal).....	201
Host Cfgxxxx property .....	125
Host object methods and properties .....	25
host writable status line .....	154
HostAddress property .....	1224
HostBGColor Cfgxxxx method.....	125
HostColor Cfgxxxx method .....	126
HOSTEX_ATN_FORMAT.....	233
HOSTEX_BACKSPACE_KEY_	
INTERPRETATION .....	234
HOSTEX_CAPTURE_MODE .....	234
HOSTEX_CELL_DELIMITED .....	235
HOSTEX_CON_STATUS .....	237
HOSTEX_CONNECT_BY .....	236
HOSTEX_CUT_MODE .....	237
HOSTEX_DEVICE_TYPE.....	238
HOSTEX_ENCRYPTED data type.....	238
HOSTEX_ENTER_KEY_	
INTERPRETATION .....	239
HOSTEX_FIELD_ATTR_	
REPLACEMENT .....	239
HOSTEX_FUNCTION_KEY .....	240
HOSTEX_GRAPHICS_CELLSIZE .....	241
HOSTEX_GRAPHICS_CURSOR_TYPE ..	241
HOSTEX_GRAPHICS_MODEL.....	242
HOSTEX_HOTSPOT_DISPLAY .....	243
HOSTEX_HOTSPOT_MOUSE_	
ACTIVATION.....	243
HOSTEX_INSERT_KEY_STYLE.....	244
HOSTEX_KEYBOARD_	
BUFFER_MODE.....	244
HOSTEX_KEYBOARD_TYPE.....	245
HOSTEX_LINEMODE.....	245
HOSTEX_NEXT_FIELD_KEY.....	246
HOSTEX_OIA_DISPLAY .....	247
HOSTEX_PASTE_MODE.....	247
HOSTEX_PRINT_TARGET .....	249
HOSTEX_PRINTFILE_MODE.....	248
HOSTEX_RESIZE_BEHAVIOR .....	249
HOSTEX_SAVE_OPTIONS .....	250
HOSTEX_SECURITY_OPTIONS .....	250
HOSTEX_SELECTION_MODE.....	251
HOSTEX_STATUS_LINE_MODE.....	251
HOSTEX_SWITCHSCREENTYPE .....	252
HOSTEX_TELNETECHO .....	253
HOSTEX_TERM_MODEL.....	254
HOSTEX_TERMINAL_ID .....	253
HOSTEX_TOGGLE_RECEIVE.....	255
HOSTEX_TPRINT_OUTPUT .....	256
HOSTEX_TRANSFER .....	257
HOSTEX_TRANSFER_HOSTTYPE.....	257
HOSTEX_TRANSFER_	
INITIALACTION .....	258
HOSTEX_TRANSFER_	
RECORDFORMAT .....	258
HOSTEX_TRANSFER_TARGET .....	259
HOSTEX_TRANSFER_TYPE.....	259
hostex.ini .....	17
HostExplorer APIs .....	4
HostExplorer programming .....	2
HostFGColor Cfgxxxx method .....	126
HostFromProfile method .....	14
HostFromShortName method .....	15
HostName (IHEProfileConnection) .....	867
HostName (IHETransport) .....	1226
HostName property .....	753
HOSTOVERWRITE_BEHAVIOUR .....	261
HostPrinting property .....	340
HostResponseTime property.....	1093
hosts .....	1433
connecting to .....	1433
transferring files between .....	1433
Hosts Collection object.....	16
Hosts object methods and properties .....	21
HostScreenPerPage property .....	729
Hosts.FieldsCollection object .....	91
HostToAscii method.....	991
HostToFileAppend method.....	1461
HostToFileTransfer method .....	1460
HostToMultipleFileTransfer method .....	1461
HostToTreeTransfer method .....	1460
HostWritableString.....	1095
hotspot display style .....	243
hotspot mouse activation.....	243
Hotspots property .....	301
Hummingbird accessibility.....	1531

Hummingbird applications.....	1410	IHEParser.GetScreenText.....	982
Hummingbird Basic Workbench .....	1410	IHEParser.GetSel .....	985
Hummingbird Basic .....	1496	IHEParser.GetSelectionArea.....	987
Hummingbird Basic Language .....	1410, 1433	IHEParser.GetValue.....	989
Hummingbird Basic Workbench.....	1410	IHEParser.GraphicsModel .....	1087
HXL .....	219, 223, 227	IHEParser.HistoryLines.....	1089
<b>I</b>		IHEParser.HLLAPIName .....	1091
IHclFtpEngine Object .....	1434	IHEParser.HostResponseTime.....	1093
IHclFtpEngine Object properties.....	1435	IHEParser.HostToAscii .....	991
IHclFtpEnginge Object methods .....	1434	IHEParser.HostWritableString.....	1095
IHclFtpSession Object .....	1435	IHEParser.IsFieldBold .....	993
IHclFtpSession Object methods.....	1436	IHEParser.IsFieldHidden .....	995
IHclFtpSession Object properties .....	1437	IHEParser.IsFieldModified .....	997
IHclFtpSessions Object .....	1437	IHEParser.IsFieldNumeric .....	999
IHclFtpSessions Object methods .....	1437	IHEParser.IsFieldPenSelectable.....	1001
IHclFtpSessions Object properties.....	1438	IHEParser.IsFieldProtected .....	1003
IHEParser.AnswerBack .....	1061	IHEParser.KeyboardLocked .....	1096
IHEParser.APIInputMode .....	1063	IHEParser.ModelColumns .....	1099
IHEParser.AsciiToHost.....	956	IHEParser.ModelRows .....	1101
IHEParser.AutoWrap.....	1169	IHEParser.MoveCursor OnMouseClick .....	1165
IHEParser.BufRows.....	1065	IHEParser.MoveCursorRelative .....	1005
IHEParser.CanChangeScreen .....	1067	IHEParser.NRCID .....	1103
IHEParser.CaptureMode .....	1069	IHEParser.NumericCharacters.....	1105
IHEParser.CellCopyMode .....	1071	IHEParser.NVTMode .....	1107
IHEParser.ClearSel .....	958	IHEParser.OIAString .....	1109
IHEParser.ConnectBy .....	1072	IHEParser.OIAStringW .....	1111
IHEParser.ConnectErrorStatus.....	1075	IHEParser.OnCopyReplace FieldAttributeWith .....	1131
IHEParser.ConnectRC .....	1077	IHEParser.OnPasteField ModeTabCharacter .....	1133
IHEParser.ConvertNulls .....	1079	IHEParser.PasteDataToScreen .....	1007
IHEParser.Cutmode .....	1081	IHEParser.PasteMode .....	1113
IHEParser.EnableAPL .....	1083	IHEParser.PrintByPassWindows.....	1115
IHEParser.EnablePrinterTimeout.....	1085	IHEParser.PrintDisableTranslation.....	1119
IHEParser.FindString.....	959	IHEParser.PrinterDeInitString.....	1117
IHEParser.GetCursorPosition .....	962	IHEParser.PrinterInitString.....	1121
IHEParser.GetFeature .....	964	IHEParser.PrinterTimeout .....	1123
IHEParser.GetFieldAttribute.....	966	IHEParser.PrinterTimeoutValue.....	1125
IHEParser.GetFieldCount .....	969	IHEParser.PrintLFtoCRLF .....	1127
IHEParser.GetFieldExtAttribute .....	971	IHEParser.ProgramSymbols.....	1129
IHEParser.GetFieldIndex .....	974	IHEParser.PutString .....	1009
IHEParser.GetFieldLength .....	976	IHEParser.PutText.....	1011
IHEParser.GetFieldPos.....	978		
IHEParser.GetFieldText .....	980		

IHEParser.ReceiveFile .....	1013	IHEProfile.Capture .....	317
IHEParser.ReplaceSel .....	1015	IHEProfileCapture.SaveAppend .....	768
IHEParser.SaveAppend .....	1135	IHEProfileCapture.SaveConfirm .....	771
IHEParser.SaveFileName .....	1137	IHEProfileCapture.SaveFileName .....	773
IHEParser.ScreenChanged .....	1139	IHEProfileCapture.SaveMode .....	775
IHEParser.ScreenCol .....	1141	IHEProfileCapture.VTCaptureMode .....	777
IHEParser.ScreenRow .....	1143	IHEProfile.ClearPassword .....	275
IHEParser.SelectionMode .....	1145	IHEProfile.Color .....	276
IHEParser.SendAid .....	1017	IHEProfileColor.Schemes .....	550
IHEParser.SendFile .....	1025	IHEProfileColor.SystemColor .....	549
IHEParser.SendKeys .....	1027	IHEProfile.Connection .....	278
IHEParser.SessionName .....	1147	IHEProfileConnection.acPassword .....	833
IHEParser.SetCursorPosition .....	1029	IHEProfileConnection .....	
IHEParser.SetFeature (IHEParser) .....	1031	AlwaysPromptForHostName .....	835
IHEParser.SetFieldText .....	1033	IHEProfileConnection.AreaCode .....	837
IHEParser.SetSel .....	1035	IHEProfileConnection .....	
IHEParser.SetValue .....	1037	AutoEndMacroName .....	839
IHEParser.SoftCharacterSetID .....	1149	IHEProfileConnection .....	
IHEParser.StatusLineMode .....	1151	AutoMacroName .....	841
IHEParser.TerminalID .....	1153	IHEProfileConnection .....	
IHEParser.TerminalModel .....	1155	AutoRunMacroDelayTime .....	843
IHEParser.Text .....	1157	IHEProfileConnection.AutoSignOn .....	845
IHEParser.TransferErrorCode .....	1171	IHEProfileConnection .....	
IHEParser.TransferMode .....	1173	BackspaceKeyInterpretation .....	847
IHEParser.Transport .....	1159	IHEProfileConnection.ConnectBy .....	849
IHEParser.TypeAheadTimeout .....	1161	IHEProfileConnection.Country .....	851
IHEParser.UPSS .....	1167	IHEProfileConnection.CountryCode .....	853
IHEParser.Validate		IHEProfileConnection.CountryID .....	855
NumericFieldData .....	1163	IHEProfileConnection.DeviceName .....	857
IHEParser.WaitConnected .....	1039	IHEProfileConnection .....	
IHEParser.WaitForCursor .....	1041	DirectToModem .....	859
IHEParser.WaitForCursorMove .....	1043	IHEProfileConnection.EnableEMode .....	861
IHEParser.WaitForIO .....	1045	IHEProfileConnection .....	
IHEParser.WaitForString .....	1047	EnableInfiniteRetries .....	863
IHEParser.WaitHostQuiet .....	1050	IHEProfileConnection.Enter	
IHEParser.WaitIdle .....	1052	KeyInterpretation .....	865
IHEParser.WaitPSUpdated .....	1054	IHEProfileConnection.HostName .....	867
IHEParser.WaitXfer .....	1056	IHEProfileConnection.Keyboard	
IHEParser.WriteProtectedText .....	1058	BufferMode .....	869
IHEProfile.AddOIAToCapture .....	269	IHEProfileConnection.LUName .....	871
IHEProfile.AllowEmuTracing .....	288	IHEProfileConnection.Modem .....	873
IHEProfile.AllowErrorRestart .....	271	IHEProfileConnection.ModemID .....	875
IHEProfile.AttnFormat .....	273		

IHEProfileConnection.	
NumberOfRetries .....	877
IHEProfileConnection.Password .....	879
IHEProfileConnection.Port .....	881
IHEProfileConnection.PortList .....	883
IHEProfileConnection.Retry	
DelayTimeBetweenHosts .....	885
IHEProfileConnection.	
ShowDialupDlg .....	887
IHEProfileConnection.SilentConnect .....	889
IHEProfileConnection.	
SYSREQasIACIP .....	891
IHEProfileConnection.TelnetEcho .....	893
IHEProfileConnection.TelnetName .....	895
IHEProfileConnection.Timeout .....	897
IHEProfileConnection.	
UponDisconnect .....	899
IHEProfileConnection.UseDialProp .....	901
IHEProfileConnection.UserName .....	903
IHEProfileConnection.VTDoHost	
WindowSize .....	905
IHEProfileConnection.VTInitiate	
TelnetNegotiation .....	907
IHEProfileConnection.VTLineMode .....	909
IHEProfile.Cursor .....	279
IHEProfileCursor.CursorMode .....	700
IHEProfileCursor.CursorType .....	702
IHEProfileCursor.Display	
CrossHairCursor .....	704
IHEProfileCursor.MoveCursor	
OnMouseClick .....	706
IHEProfile.DDEServerName .....	281
IHEProfile.Display .....	283
IHEProfileDisplay.BlinkToItalic .....	669
IHEProfileDisplay.ColumnSeparators .....	672
IHEProfileDisplay.DisplayInOIA .....	674
IHEProfileDisplay.DisplayRowCol .....	676
IHEProfileDisplay.DisplayUpperCase .....	678
IHEProfileDisplay.ShowNulls .....	680
IHEProfileDisplay.StatusLineMode .....	682
IHEProfileDisplay.VTClearScreen	
OnSizeChange .....	684
IHEProfileDisplay.VTHostWritable	
StatusLine .....	686
IHEProfileDisplay.VTISOColors .....	688
IHEProfileDisplay.	
VTMaxScrollBufferSize .....	690
IHEProfileDisplay.VTResetISOColors .....	692
IHEProfileDisplay.VTSaveAttribs	
InScrollbar .....	694
IHEProfileDisplay.	
VTSaveEraseScreens .....	696
IHEProfileDisplay.VTScrollNoBlanks .....	698
IHEProfile.Edit .....	284
IHEProfileEdit.AlwaysAutoskip .....	483
IHEProfileEdit.AutoCopy .....	485
IHEProfileEdit.AutoCopy	
KeepSelection .....	487
IHEProfileEdit.BellMargin .....	489
IHEProfileEdit.CellDelimited .....	491
IHEProfileEdit.ClearAllTabStops .....	482
IHEProfileEdit.ClipFormatBitmap .....	493
IHEProfileEdit.ClipFormatCSV .....	495
IHEProfileEdit.ClipFormatHE .....	497
IHEProfileEdit.ClipFormatPasteLink .....	499
IHEProfileEdit.ClipFormatRTF .....	501
IHEProfileEdit.ClipFormatText .....	503
IHEProfileEdit.ConvertNulls .....	505
IHEProfileEdit.CSVCopyEmptyFields .....	507
IHEProfileEdit.CSVTrimFields .....	509
IHEProfileEdit.CutChar .....	511
IHEProfileEdit.CutMode .....	513
IHEProfileEdit.EntryAssist .....	515
IHEProfileEdit.InsertResetByAttn .....	517
IHEProfileEdit.LeftMargin .....	519
IHEProfileEdit.MoveCursorAfterPaste .....	521
IHEProfileEdit.MultiLineDelete .....	523
IHEProfileEdit.MultiLineInsert .....	525
IHEProfileEdit.NoLockKeyb .....	527
IHEProfileEdit.NumericCharacters .....	529
IHEProfileEdit.PasteChar .....	531
IHEProfileEdit.PasteMode .....	533
IHEProfileEdit.RemoveTrailingBlank	
OnCopy .....	535

IHEProfileEdit.ResetMDT	
OnEraseInput .....	537
IHEProfileEdit.RespectNumeric .....	539
IHEProfileEdit.RightMargin .....	541
IHEProfileEdit.SmartInsert .....	543
IHEProfileEdit.TabStop .....	482
IHEProfileEdit.WordWrap .....	545
IHEProfileEdit.WordWrapWith	
NewLineReturn .....	547
IHEProfile.EmuTraceFilename .....	286
IHEProfile.EnableHLLAPITracing .....	290
IHEProfile.EnableTracing .....	292
IHEProfile.Events .....	293
IHEProfileEvents.EnableEvents .....	828
IHEProfileEvents.RemoveEvent .....	827
IHEProfileEvents.Schemes .....	830
IHEProfile.FileTransfer .....	295
IHEProfileFileTransfer.1KPacket .....	632
IHEProfileFileTransfer.Append .....	553
IHEProfileFileTransfer.ASCII .....	555
IHEProfileFileTransfer.AutoCC .....	557
IHEProfileFileTransfer.	
AutoClearMonitor .....	559
IHEProfileFileTransfer.BlkSiz .....	561
IHEProfileFileTransfer.CRLF .....	563
IHEProfileFileTransfer.Custom	
TransferTable .....	565
IHEProfileFileTransfer.Default	
DownloadPath .....	567
IHEProfileFileTransfer.	
DefaultProtocol .....	569
IHEProfileFileTransfer.	
DefaultRecvDir .....	571
IHEProfileFileTransfer.	
DefaultUploadPath .....	573
IHEProfileFileTransfer.Download	
HostName .....	575
IHEProfileFileTransfer.Download	
PCFileName .....	577
IHEProfileFileTransfer.Download	
Translate .....	579
IHEProfileFileTransfer.ExtraOptions .....	581
IHEProfileFileTransfer.	
FileExistAction .....	583
IHEProfileFileTransfer.Host .....	585
IHEProfileFileTransfer.	
INDFILEName .....	587
IHEProfileFileTransfer.	
KmBinaryPrefix .....	589
IHEProfileFileTransfer.KmRLE .....	591
IHEProfileFileTransfer.KmTextMode .....	593
IHEProfileFileTransfer.	
KmUseFullPath .....	595
IHEProfileFileTransfer.LrecL .....	597
IHEProfileFileTransfer.QuickMode .....	599
IHEProfileFileTransfer.Recfm .....	601
IHEProfileFileTransfer.Schemes .....	603
IHEProfileFileTransfer.ShowRecvDlg .....	605
IHEProfileFileTransfer.Upload	
HostFileName .....	607
IHEProfileFileTransfer.Upload	
PCFileName .....	609
IHEProfileFileTransfer.	
UploadTranslate .....	611
IHEProfileFileTransfer.User	
DefinedDownload .....	613
IHEProfileFileTransfer.	
UserDefinedUpload .....	615
IHEProfileFileTransfer.VTAuto	
ClearMonitor .....	617
IHEProfileFileTransfer.XferBlockSize .....	619
IHEProfileFileTransfer.XferDest .....	621
IHEProfileFileTransfer.	
XferHostCodePage .....	623
IHEProfileFileTransfer.	
XferPCCodePage .....	626
IHEProfileFileTransfer.XferSource .....	628
IHEProfileFileTransfer.	
XferStartAction .....	630
IHEProfileFileTransfer.	
XmAckTimeout .....	634
IHEProfileFileTransfer.XmCRC .....	636
IHEProfileFileTransfer.	
YmAckTimeout .....	638

IHEProfileFileTransfer.	
YmUseFullPath.....	640
IHEProfileFileTransfer.	
ZmAutoDownload .....	642
IHEProfileFileTransfer.	
ZmCrashRecovery .....	644
IHEProfileFileTransfer.ZmMaxErrors.....	646
IHEProfileFileTransfer.Zm	
OverwriteMngmt.....	648
IHEProfileFileTransfer.	
ZmSlideWindow .....	650
IHEProfileFileTransfer.	
ZmUseFullPath.....	652
IHEProfileFileTransfer.	
ZmWindowSize .....	654
IHEProfile.Fonts.....	296
IHEProfileFonts.CharacterSpacing.....	709
IHEProfileFontsSetFont.....	708
IHEProfileFonts.VariableWidthFont.....	711
IHEProfile.GlobalSettingsPath.....	298
IHEProfile.Graphics .....	300
IHEProfileGraphics.APL.....	406
IHEProfileGraphics.	
GraphicsCursorType .....	409
IHEProfileGraphics.GraphicsModel.....	411
IHEProfileGraphics.LightPen .....	413
IHEProfileGraphics.ProgramSymbols.....	415
IHEProfileGraphics.PSCellSize.....	417
IHEProfile.HLLAPITraceFilename.....	332
IHEProfile.HostPrinting .....	340
IHEProfileHostPrinting.	
VTAutoFormFeed .....	779
IHEProfileHostPrinting.	
VTiPrintMaxCols .....	782
IHEProfileHostPrinting.VTi	
PrintMaxRows.....	784
IHEProfileHostPrinting.	
VTPassThruUPSS.....	786
IHEProfileHostPrinting.VTPrint	
ByPassWindows.....	788
IHEProfileHostPrinting.VTPrint	
DefaultFont.....	790
IHEProfileHostPrinting.VTPrint	
DisableTranslation .....	792
IHEProfileHostPrinting.	
VTPrinterDeinit.....	794
IHEProfileHostPrinting.VTPrinter	
EnableTimeout.....	796
IHEProfileHostPrinting.VTPrinter	
FontName .....	798
IHEProfileHostPrinting.VTPrinterInit .....	800
IHEProfileHostPrinting.VTPrinter	
Timeout.....	802
IHEProfileHostPrinting.VTPrinter	
TimeoutValue .....	804
IHEProfileHostPrinting.VTPrintFile .....	806
IHEProfileHostPrinting.	
VTPrint FileMode .....	808
IHEProfileHostPrinting.VTPrint	
FitFontToPage .....	810
IHEProfileHostPrinting.VTPrintLF	
toCRLF .....	812
IHEProfileHostPrinting.	
VTPrint Target .....	814
IHEProfileHostPrinting.VTUSe	
SpecificPrinter .....	816
IHEProfile.Hotspots .....	301
IHEProfileHotspots.DisplayStyle .....	818
IHEProfileHotspots.EnableHotspots .....	821
IHEProfileHotspots.MouseActivation .....	823
IHEProfileHotspots.Schemes .....	825
IHEProfile.Keyboard .....	303
IHEProfileKeyboard.	
AllowAIDKeyRepeat .....	419
IHEProfileKeyboard.AllowDiac .....	422
IHEProfileKeyboard.	
CurrentKeyboard .....	424
IHEProfileKeyboard.KeyboardType .....	426
IHEProfileKeyboard.	
LockOnAttention .....	428
IHEProfileKeyboard.MapNumLock .....	430
IHEProfileKeyboard.RemapKeypad .....	432
IHEProfileKeyboard.TypeAhead.....	434
IHEProfileKeyboard.	
TypeAheadTimeout .....	436

IHEProfileKeyboard.VTCursor	
KeyApplMode.....	438
IHEProfileKeyboard.VTEnableBreak .....	439
IHEProfileKeyboard.	
VTKeypadApplMode .....	442
IHEProfileKeyboard.	
VTNewLineMode.....	444
IHEProfile.KillMacrosOnSessionExit .....	305
IHEProfile.Load.....	265
IHEProfile.LoadColorScheme .....	266
IHEProfile.LoadFileTransferScheme .....	266
IHEProfile.LoadQuickKeyFile .....	267
IHEProfile.Mouse.....	307
IHEProfileMouse.BlockSelect.....	446
IHEProfileMouse.SelectHilight.....	448
IHEProfile.PCPrint .....	308
IHEProfilePCPrint.PrinterDeinit.....	716
IHEProfilePCPrint.PrinterInit .....	718
IHEProfilePCPrint.PrintMode7171.....	713
IHEProfilePCPrint.TprintMode .....	720
IHEProfile.PrintScreen.....	311
IHEProfilePrintScreen.AddFormFeed.....	722
IHEProfilePrintScreen.	
DisplayAbortDlg.....	725
IHEProfilePrintScreen.	
DisplayPrintDlg .....	727
IHEProfilePrintScreen.	
HostScreenPerPage .....	729
IHEProfilePrintScreen.Print	
BlackAndWhite .....	731
IHEProfilePrintScreen.PrintBorder .....	733
IHEProfilePrintScreen.PrintDocname .....	735
IHEProfilePrintScreen.PrinterFooter .....	737
IHEProfilePrintScreen.PrinterHeader .....	739
IHEProfilePrintScreen.PrintLocation .....	741
IHEProfilePrintScreen.PrintOIA .....	743
IHEProfilePrintScreen.Print	
ReversedColors.....	745
IHEProfilePrintScreen.Print	
ScreenFontName .....	747
IHEProfilePrintScreen.PrintScreen	
FontSize .....	749
IHEProfilePrintScreen.PRTSCRUse	
SpecificPrinter .....	751
IHEProfile.PrintSession .....	310
IHEProfilePrintSession.HostName .....	753
IHEProfilePrintSession.LimitTo	
SingleInstance .....	756
IHEProfilePrintSession.LUName .....	758
IHEProfilePrintSession.LUType .....	760
IHEProfilePrintSession.ProfileName .....	762
IHEProfilePrintSession.StartPrinter.....	764
IHEProfilePrintSession.StopPrinter .....	766
IHEProfile.ProfileName .....	312
IHEProfile.QueryShutdown .....	315
IHEProfile.RecordPortableMacros .....	318
IHEProfile.ReRunAutoMacro .....	320
IHEProfile.Save .....	268
IHEProfile.Security .....	322
IHEProfileSecurity.Kerberos .....	656
IHEProfileSecurity.KerberosAltName .....	659
IHEProfileSecurity.Kerberos	
Encryption.....	661
IHEProfileSecurity.Kerberos	
ForwardTkt .....	663
IHEProfileSecurity.KerberosVersion .....	665
IHEProfileSecurity.SecurityOption .....	667
IHEProfile.SessionWindow .....	323
IHEProfileSessionWindow.	
DisplayBorder .....	911
IHEProfileSessionWindow.EnableWorkspace	
BackgroundBitmap .....	914
IHEProfileSessionWindow.	
FullScreenMode .....	916
IHEProfileSessionWindow.	
KeepFontAspectRatio .....	918
IHEProfileSessionWindow.	
LongName.....	920
IHEProfileSessionWindow.	
PromptOnClose .....	922
IHEProfileSessionWindow.	
ResizeBehavior .....	924
IHEProfileSessionWindow.	
SaveFontOnExit .....	928

IHEProfileSessionWindow.	
SaveProfOnClose.....	926
IHEProfileSessionWindow.	
SnapFrameBack.....	930
IHEProfileSessionWindow.	
SwitchScreenType.....	932
IHEProfileSessionWindow.	
WindowState .....	934
IHEProfileSessionWindow.	
WindowTitle.....	936
IHEProfileSessionWindow.WorkSpace	
BackgroundBitmap .....	938
IHEProfileSessionWindow.Workspace	
BackgroundColor .....	940
IHEProfileSessionWindow.Workspace	
ForegroundColor.....	942
IHEProfile.Sound .....	325
IHEProfileSound.Notify.....	944
IHEProfileSound.Sound .....	946
IHEProfile.Terminal.....	326
IHEProfileTerminal.AlternateScreen.....	344
IHEProfileTerminal.CharacterSet.....	347
IHEProfileTerminal.CustomModel.....	364
IHEProfileTerminal.Custom	
ModelCols .....	366
IHEProfileTerminal.Custom	
ModelRows.....	368
IHEProfileTerminal.DetectChainedIO.....	349
IHEProfileTerminal.ForceAltSize .....	351
IHEProfileTerminal.New3270EAB .....	353
IHEProfileTerminal.NewModel3279.....	355
IHEProfileTerminal.NewModelType .....	357
IHEProfileTerminal.ReplyOEM .....	360
IHEProfileTerminal.ShortName.....	362
IHEProfile.TerminalType .....	328
IHEProfileTerminal.VT8BitMode .....	370
IHEProfileTerminal.VTAnswerback .....	372
IHEProfileTerminal.VTAutoResize.....	374
IHEProfileTerminal.VTBIsDel .....	376
IHEProfileTerminal.	
VTConcealAnswerback.....	378
IHEProfileTerminal.VTDefCols	
PerScreen .....	380
IHEProfileTerminal.VTDefLines	
PerScreen.....	382
IHEProfileTerminal.VTDisplayMode .....	384
IHEProfileTerminal.VTEnableSSH.....	386
IHEProfileTerminal.VTForce8Bit.....	388
IHEProfileTerminal.VTLocalEcho.....	390
IHEProfileTerminal.VTN	
TerminalType.....	392
IHEProfileTerminal.VTO	
VTO.....	395
IHEProfileTerminal.VTSc	
VTSc.....	397
IHEProfileTerminal.VTS	
VTS.....	399
IHEProfileTerminal.VTT	
VTT.....	401
IHEProfileTerminal.VTW	
VTW.....	404
IHEProfile.Toolbar .....	330
IHEProfileToolbar.BigToolbar.....	450
IHEProfileToolbar.Button.....	452
IHEProfileToolbar.MaxBitmaps .....	454
IHEProfileToolbar.NumTools .....	456
IHEProfileToolbar.ShowTips .....	458
IHEProfileToolbar.TBU	
TBU.....	460
IHEProfileToolbar.ToolbarDockType .....	462
IHEProfileToolbar.ToolbarFilename .....	464
IHEProfile.TrackMenu .....	334
IHEProfileTrack.TrackCommands .....	466
IHEProfileTrack.TrackLabels .....	468
IHEProfile.TranslationTable .....	335
IHEProfileTranslationTable.	
CurrentLanguage .....	470
IHEProfile.UserDirectory .....	337
IHEProfile.VTCharSet .....	339
IHEProfileVTCharSet.VTForceNRC .....	472
IHEProfileVTCharSet.VTNRC.....	475
IHEProfileVTCharSet.VTNRCMode .....	477
IHEProfileVTCharSet.VTUPSS .....	479
IHEProfile.WinDDEEnabled .....	342
IHETerminal.ChooseTerminalFont .....	196
IHETerminal.Connect .....	196
IHETerminal.ConnectBy .....	199
IHETerminal.Connected .....	200
IHETerminal.Disconnect .....	197
IHETerminal.EditSessionProperties .....	198
IHETerminal.Host .....	201
IHETerminal.Parser .....	202

IHETerminal.Session .....	203
IHETerminal.SilentConnect .....	203
IHETerminal.TCPPort .....	204
IHETerminal.Transport .....	205
IHETerminal.UserDir .....	206
IHETransport.AddFeature .....	208
IHETransport.AttentionFormat .....	1204
IHETransport.CharSet .....	1206
IHETransport.CodePage .....	1208
IHETransport.Connected .....	1210
IHETransport.ConnectionStatus .....	1212
IHETransport.DeviceType .....	1216
IHETransport.EnableEMode .....	1218
IHETransport.EnableSSH .....	1220
IHETransport.EnableTracing .....	1222
IHETransport.GetFeature .....	1179
IHETransport.GetStatusString .....	1182
IHETransport.HostAddress .....	1224
IHETransportHostName .....	1226
IHETransport.IsEncrypted .....	1228
IHETransport.IsReceiveBlocked .....	1230
IHETransport.Keyboard .....	1232
IHETransport.LineMode .....	1280
IHETransport.LUNameRequested .....	1214
IHETransport.MaxBlockSize .....	1234
IHETransport.MessageQueueLibrary .....	1236
IHETransport.MessageQueueName .....	1238
IHETransport.ModelColumns .....	1239
IHETransport.ModelRows .....	1242
IHETransport.NegotiateNAWS .....	1184
IHETransport.NumberOfRetries .....	1244
IHETransport.Password .....	1246
IHETransport.PerformingTransfer .....	1248
IHETransport.Port .....	1250
IHETransport.PortList .....	1252
IHETransport.RemoveFeature .....	208
IHETransport.Retry DelayTimeBetweenHosts .....	1253
IHETransport.SecurityOption .....	1256
IHETransport.SendData .....	1186
IHETransport.SendFunctionKey .....	1188
IHETransport.SendKeepAlive .....	1190
IHETransport.SessionKeepAlive .....	1258
IHETransport.SetFeature .....	1192
IHETransport.SetHostPrint TransformInfo .....	1194
IHETransport.SetKerberosInfo .....	1198
IHETransport.TelnetEcho .....	1260
IHETransport.TelnetIsLineMode .....	1262
IHETransport.TelnetIsLocalEcho .....	1264
IHETransport.TelnetName .....	1266
IHETransport.TerminalModel .....	1268
IHETransport.TerminalOnline .....	1270
IHETransport.TerminalType .....	1272
IHETransport.TNESession .....	1274
IHETransport.ToggleBlockReceive .....	1200
IHETransport.TraceFilename .....	1276
IHETransport.Username .....	1278
Index property .....	41
INDFILEName .....	587
INI files .....	13, 17
retrieving values from .....	13
.ini files .....	13, 17
retrieving values from .....	13
InitFTP function .....	1410
InitialHostDirectory property .....	1442
InputInhibited .....	1361
InputInhibited (IOhioOIA) .....	1327
InsertKeyStyle .....	244
InsertMode .....	1328
InsertMode property .....	41
InsertResetByAttn .....	517
interface .....	263, 343, 406, 418, 445, 449, 465, 470, 472, 481, 549, 551, 656, 669, 700, 708, 713, 722, 753, 768, 779,

818, 827, 832, 911, 944, 1283, 1289, 1293, 1302, 1324, 1332, 1338, 1353	introducing FTP API .....	1409, 1432
IHEProfileCursor.....	non-OLE.....	1409
OhioField .....	OLE.....	1432
OhioFields.....	introduction .....	1472
OhioManager .....	IOhioField.Attribute .....	1340
OhioOIA .....	IOhioField.End .....	1342
OhioPosition .....	IOhioField.GetData .....	1339
OhioScreen.....	IOhioField.HighIntensity .....	1343
OhioSession .....	IOhioField.Length.....	1344
OhioSessions .....	IOhioField.Modified.....	1345
Profile .....	IOhioField.Normal .....	1346
ProfileCapture.....	IOhioField.Numeric .....	1347
ProfileColor.....	IOhioField.PenSelectable.....	1348
ProfileConnection.....	IOhioField.Protected .....	1349
ProfileDisplay.....	IOhioFields.Count .....	1336
ProfileEdit .....	IOhioFields.FindByPosition .....	1333
ProfileEvents .....	IOhioFields.FindByString .....	1334
ProfileFileTransfer .....	IOhioFields.Item .....	1337
ProfileFonts.....	IOhioFields.Refresh .....	1335
ProfileGraphics .....	IOhioField.Start .....	1350
ProfileHostPrinting.....	IOhioField.String .....	1351
ProfileHotspots .....	IOhioManager.CloseSession.....	1284
ProfileKeyboard .....	IOhioManager.OhioVersion .....	1286
ProfileMouse .....	IOhioManager.OpenSession .....	1284
ProfilePCPrint.....	IOhioManager.Sessions .....	1286
ProfilePrintScreen .....	IOhioManager.VendorName .....	1287
ProfilePrintSession.....	IOhioManager.VendorObject .....	1288
ProfileSecurity .....	IOhioManager.Vendor	
ProfileSessionFactory.....	ProductVersion .....	1288
ProfileSessionWindow.....	IOhioOIA.Alphanumeric .....	1324
ProfileSound .....	IOhioOIA.APL.....	1325
ProfileTerminal .....	IOhioOIA.CommCheckCode .....	1326
ProfileToolbar .....	IOhioOIA.InputInhibited.....	1327
ProfileTrackMenu.....	IOhioOIA.InsertMode .....	1328
ProfileTranslationTable .....	IOhioOIA.MachineCheckCode .....	1328
ProfileVTCharset .....	IOhioOIA.Numeric .....	1329
Interpret property .....	IOhioOIA.Owner .....	1330
introducing .....	IOhioOIA.ProgCheckCode .....	1331
DDE.....	IOhioPosition.Column .....	1355
EHLLAPI and WinHLLAPI.....	IOhioPosition.CreateOhioPosition .....	1354
HostExplorer APIs .....	IOhioPosition.Row .....	1356
HostExplorer programming .....	IOhioScreen.Columns .....	1319
OLE automation .....	IOhioScreen.Cursor .....	1319

IOhioScreen.Fields .....	1320
IOhioScreen.FindString .....	1304
IOhioScreen.GetData .....	1305
IOhioScreen.OIA .....	1321
IOhioScreen.OnCursorMoved .....	1306
IOhioScreen.OnScreenChanged .....	1307
IOhioScreen.OnSizeChanged .....	1309
IOhioScreen.PutString .....	1310
IOhioScreen.Rows .....	1322
IOhioScreen.SendKeys .....	1311
IOhioScreen.String .....	1323
IOhioScreen.WaitForInput .....	1315
IOhioScreen.WaitForString .....	1316
IOhioScreen.WaitIdle .....	1318
IOhioSession.CanChangeScreen .....	1298
IOhioSession.ConfigurationResource .....	1300
IOhioSession.Connect .....	1294
IOhioSession.Disconnect .....	1295
IOhioSession.isConnected .....	1297
IOhioSession.OnSessionChanged .....	1297
IOhioSessions.Count .....	1290
IOhioSession.Screen .....	1300
IOhioSession.SessionName .....	1301
IOhioSession.SessionType .....	1302
IOhioSessions.Item .....	1291
IOhioSessions.Refresh .....	1289
Irma compatibility mode .....	1367
IsBold property .....	95
IsConnected .....	1297
IsConnected property .....	42
IsEncrypted .....	1228
IsFieldBold method .....	993
IsFieldHidden method .....	995
IsFieldModified method .....	997
IsFieldNumeric method .....	999
IsFieldPenSelectable method .....	1001
IsFieldProtected method .....	1003
IsHidden property .....	95
IsModified property .....	96
IsNumeric property .....	96
IsPenSelectable property .....	97
IsProtected property .....	97
IsReceiveBlocked .....	1230
IsXfer property .....	42
Item (IOhioFields) .....	1337
Item (IOhioSessions) .....	1291
Item property .....	22
items .....	1475, 1476, 1478, 1479, 1480, 1481, 1482, 1484, 1485, 1486, 1491, 1492, 1493
CharacterAt .....	1492
CharacterAttributeAt .....	1492
CharacterBColorAt .....	1493
CharacterFGColorAt .....	1493
ConnectionStatus .....	1479
ConnectRlogin .....	1480
ConnectTelnet .....	1480
CurrenFGColor .....	1491
CurrentAttribute .....	1491
CurrentBColor .....	1491
CursorPosition .....	1484
Disconnect .....	1481
EmulationType .....	1478
ExecReturn .....	1479
Formats .....	1476
GetOption .....	1484
LoadSetupFile .....	1480
LookForString .....	1482
ReadKeyboardFile .....	1485
SaveData .....	1479
SendStringToTerminal .....	1481
SenString .....	1481
SetAttributeColor .....	1485
SetAttributeUsage .....	1486
SetOption .....	1484
StartDataSave .....	1482
StopDataSave .....	1482
SysItems .....	1476
TopicItemList .....	1478
topics .....	1475
<b>J</b>	
JumpScrollNumber property .....	1510

**K**

KeepFontAspectRatio property.....	918
Kerberos.....	656
KerberosAltNam.....	659
KerberosEncryption .....	661
KerberosForwardTkt .....	663
KerberosVersion .....	665
KermitBinPrefix CfgVT property.....	127
KermitCompression CfgVT property .....	127
KermitTextMode CfgVT property.....	128
KermitUseFullPath CfgVT property .....	128
key strings .....	1380
Keyboard (IHEProfile) .....	303
Keyboard (IHETransport) .....	1232
keyboard buffer mode .....	244
keyboard mapping.....	1019, 1022
TN3270.....	1019
TN5250.....	1022
Keyboard property.....	43
Keyboard	
BufferMode property.....	869
KeyboardLocked.....	1096
KeyboardProfileName Cfgxxxx method....	129
KeyboardType .....	245
KeyboardType (IHEProfileKeyboard) .....	426
KeypadMode CfgVT property .....	129
Keys method .....	43
Keystroke example.....	1400
KillMacrosOnSessionExit property.....	305
KmBinaryPrefix .....	589
KmRLE .....	591
KmTextMode .....	593
KmUseFullPath .....	595

**L**

language index.....	216, 220, 224, 226
TN3270.....	216
TN5250.....	220
TNVT NRC.....	226
TNVT UPSS .....	224

language-conversion table ..	216, 220, 224, 226
TN3270 .....	216
TN5250 .....	220
TNVT NRC .....	226
TNVT UPSS .....	224
Left property.....	82
LeftMargin.....	519
LeftMargin Cfgxxxx property .....	130
legacy APIs .....	1364
Length.....	1344
Length property .....	98
LightPen property .....	413
LimitToSingleInstance property.....	756
Linemode .....	245
Linemode CfgVT property .....	131
LineMode property .....	1280
LinesInScrollbar CfgVT property.....	130
Load method .....	265
LoadColorScheme method .....	266
LoadFileTransferScheme method.....	266
LoadQuickKeyFile method (Cfgxxxx) .....	45
LoadQuickKeyFile method (IHEProfile) ....	267
LoadSetupFile item .....	1480
LoadSetupFile method .....	1501
LocalDefaultDirectory property .....	1466
LocalDelTree function .....	1420
LocalDelTree method .....	1469
LocalEcho CfgVT property .....	131
LocalEcho property .....	1509
LocalFileNamesCase property .....	1466
LockOnAttention .....	428
LongName.....	920
LongName Cfgxxxx property .....	132
LookForString item.....	1482
LookForString method .....	1504
Lrecl.....	597
LS function .....	1424
LS method .....	1456
LUName.....	871
LUName Cfgxxxx property .....	132
LUName property.....	758
LUNameRequested property .....	1214
LUType property .....	760

**M**

MachineCheckCode .....	1328
MakeHostDir function .....	1411
MAkeHostDir method .....	1462
MapNumLock .....	430
Math group .....	179
MaxBitmaps property .....	454
MaxBlockSize property .....	1234
Maximize method .....	45
MDelete method .....	1462
MessageQueueLibrary .....	1236
MessageQueueName .....	1238
methods .	1432, 1433, 1497, 1501, 1502, 1503, 1504, 1505, 1506, 1512, 1513, 1523, 1524, 1525, 1526, 1527
Activate .....	26
ALAKeyboardProfileName Cfg3270 ...	168
Application object .....	10
Area .....	28
Area object .....	76
BFPress .....	29
Cfg3270, Cfg5250, and CfgVT .....	99
ClearAllTabStops Cfgxxxx .....	112
Close .....	31
CloseAll .....	22
Connect .....	32
Connect to Host .....	1433
Copy .....	79
CursorRC .....	35
Cut .....	80
Delete .....	81
DelSession .....	168
Disconnect .....	36
ExitAll .....	12
Field object .....	90
FieldID .....	37
FontLarger .....	38
FontSmaller .....	39
Get .....	1433
GetCurrentDir .....	12
GetFilePath .....	13
GetProfileString .....	13
Hide .....	39

HideToolbar .....	39
HighlightText .....	40
Host object .....	21, 25
HostBColor Cfgxxxx .....	125
HostColor Cfgxxxx .....	126
HostFGColor Cfgxxxx .....	126
HostFromShortName .....	15
IHEParser.AsciiToHost .....	956
IHEParser.ClearSel .....	958
IHEParser.FindString .....	959
IHEParser.GetCursorPosition .....	962
IHEParser.GetFeature .....	964
IHEParser.GetFieldAttribute .....	966
IHEParser.GetFieldCount .....	969
IHEParser.GetFieldExtAttribute .....	971
IHEParser.GetFieldIndex .....	974
IHEParser.GetFieldLength .....	976
IHEParser.GetFieldPos .....	978
IHEParser.GetFieldText .....	980
IHEParser.GetScreenText .....	982
IHEParser.GetSel .....	985
IHEParser.GetSelectionArea .....	987
IHEParser.GetValue .....	989
IHEParser.HostToAscii .....	991
IHEParser.IsFieldBold .....	993
IHEParser.IsFieldHidden .....	995
IHEParser.IsFieldModified .....	997
IHEParser.IsFieldNumeric .....	999
IHEParser.IsFieldPenSelectable .....	1001
IHEParser.IsFieldProtected .....	1003
IHEParser.MoveCursorPosition .....	1005
IHEParser.PasteDataToScreen .....	1007
IHEParser.PutString .....	1009
IHEParser.PutText .....	1011
IHEParser.ReceiveFile .....	1013
IHEParser.ReplaceSel .....	1015
IHEParser.SendAid .....	1017
IHEParser.SendFile .....	1025
IHEParser.SendKeys .....	1027
IHEParser.SetCursorPosition .....	1029
IHEParser.SetFeature .....	1031
IHEParser.SetFieldText .....	1033
IHEParser.SetSel .....	1035

IHEParser.SetValue.....	1037	IOhioManager.OpenSession.....	1284
IHEParser.WaitConnected .....	1039	IOhioPosition.CreateOhioPosition ...	1354
IHEParser.WaitForCursor.....	1041	IOhioScreen.FindString .....	1304
IHEParser.WaitForCursorMove .....	1043	IOhioScreen.GetData .....	1305
IHEParser.WaitForIO.....	1045	IOhioScreen.OnCursorMoved .....	1306
IHEParser.WaitForString .....	1047	IOhioScreen.OnScreenChanged .....	1307
IHEParser.WaitHostQuiet.....	1050	IOhioScreen.OnSizeChanged .....	1309
IHEParser.WaitIdle.....	1052	IOhioScreen.PutString .....	1310
IHEParser.WaitPSUpdated .....	1054	IOhioScreen.SendKeys .....	1311
IHEParser.WaitXfer.....	1056	IOhioScreen.WaitForInput .....	1315
IHEParser.WriteProtectedText .....	1058	IOhioScreen.WaitForString .....	1316
IHEProfileColor.SystemColor.....	549	IOhioScreen.WaitIdle .....	1318
IHEProfileEdit.ClearAllTabStops.....	482	IOhioSession.Connect.....	1294
IHEProfileEdit.TabStop.....	482	IOhioSession.Disconnect .....	1295
IHEProfileEvents.RemoveEvent.....	827	IOhioSession.isConnected .....	1297
IHEProfileFontsSetFont .....	708	IOhioSession.OnSessionChanged....	1297
IHEProfile.Load .....	265	IOhioSessions.Refresh .....	1289
IHEProfile.LoadColorScheme .....	266	KeyboardProfileName Cfgxxxx.....	129
IHEProfile.LoadFileTransferScheme...	266	Keys.....	43
IHEProfile.LoadQuickKeyFile .....	267	LoadQuickKeyFile .....	45
IHEProfile.Save .....	268	Maximize .....	45
IHETerminal.ChooseTerminalFont....	196	Minimize.....	46
IHETerminal.Connect .....	196	NewSession .....	17
IHETerminalDisconnect .....	197	OhioField interface .....	1338
IHETerminal.EditSessionProperties ...	198	OhioFields interface .....	1332
IHETransport.AddFeature .....	208	OhioManager interface .....	1283
IHETransport.GetFeature.....	1179	OhioOIA interface.....	1324
IHETransport.GetStatusString .....	1182	OhioPosition interface .....	1353
IHETransport.NegotiateNAWS .....	1184	OhioScreen interface .....	1302
IHETransport.RemoveFeature .....	208	OhioSession interface .....	1293
IHETransport.SendData.....	1186	OhioSessions interface .....	1289
IHETransport.SendFunctionKey .....	1188	OLE object .....	9
IHETransport.SendKeepAlive .....	1190	Open .....	23
IHETransport.SetFeature.....	1192	Parser objects .....	955
IHETransport.SetHostPrint TransformInfo .....	1194	Password Cfgxxxx.....	138
IHETransport.SetKerberosInfo .....	1198	Pause .....	48
IHETransport.ToggleBlockReceive...	1200	PrintScreen .....	49
IOhioField.GetData .....	1339	Profile interface.....	263
IOhioFields.FindByPosition .....	1333	ProfilePrintSession interface .....	753
IOhioFields.FindByString .....	1334	ProfileTranslationTable interface.....	470
IOhioFields.Refresh .....	1335	ProtectedText .....	49
IOhioManager.CloseSession.....	1284	PutText .....	51
		ReceiveFile .....	52

renamed or moved.....	210	ModelRows (IHETransport) .....	1242
Restore.....	54	Modem CfgVT property.....	133
Row.....	54	Modem property.....	873
RunCmd.....	55	ModemID .....	875
RunQuickKey.....	56	Modified.....	1345
SaveProfile Cfgxxxx.....	150	modules.....	1367, 1392
SaveQuickKeyFile .....	57	EHLLAPI .....	1367
SaveScreen.....	57	WinHLLAPI .....	1392
SaveScrollbar.....	58	Mouse.....	307
Search .....	58	MouseActivation property.....	823
Select.....	86	MouseToCursor property.....	47
SendFile.....	59	MoveCursorAfterPaste .....	521
SetFont .....	60	MoveCursorOnMouseClicked (IHEParser) .	1165
Show .....	62	MoveCursorOnMouse	
Show Toolbar .....	62	Click (IHEProfileCursor) .....	706
StartSession .....	18	MoveCursorRelative method .....	1005
SystemColor .....	64	MPut function.....	1416
TabStop Cfgxxxx.....	155	MPut method.....	1459
Terminal objects.....	195	MultiLineDelete .....	523
TextRC .....	66	MultiLineDelete Cfgxxxx property.....	133
TrackMenu.....	68	MultiLineInsert .....	525
Transport objects .....	1179	MultiLineInsert Cfgxxx property.....	134
unsupported.....	168, 208	Multinational property .....	1516
Update .....	68	MUSIC-specific options on upload.....	190
User Login.....	1433		
WaitConnected .....	68		
WaitForIO .....	69		
WaitForString .....	70	Name .....	1503
WaitForStringRC .....	71	Name property	
WaitIdle .....	72	IHclFtpEngine .....	1439
WaitPSUpdated .....	73	IHclFtpSession.....	1450
WaitXfer .....	74	NationalSet property.....	1516
Word .....	20	NegotiateNAWS method.....	1184
WriteProfileString.....	20	New3270EAB .....	353
MGet function .....	1415	NewLine property .....	1508
MGet method .....	1459	NewModel3279 .....	355
Minimize method.....	46	NewModelType.....	357
Miscellaneous group.....	180	NewSession method.....	17, 1467
mnemonic keywords .....	1312	NewUserLogIn method .....	1456
Model property.....	46	Next property .....	23
ModelColumns (IHEParser).....	1099	NextFieldKey .....	246
ModelColumns (IHETransport) .....	1239	NoLockKeyb .....	527
ModelRows (IHEParser) .....	1101	non-OLE automation .....	1409
		Normal .....	1346

**N**

Name .....	1503
Name property	
IHclFtpEngine .....	1439
IHclFtpSession.....	1450
NationalSet property.....	1516
NegotiateNAWS method.....	1184
New3270EAB .....	353
NewLine property .....	1508
NewModel3279 .....	355
NewModelType.....	357
NewSession method.....	17, 1467
NewUserLogIn method .....	1456
Next property .....	23
NextFieldKey .....	246
NoLockKeyb .....	527
non-OLE automation .....	1409
Normal .....	1346

Notify .....	944
Notify Cfgxxxx property .....	134
NRCID property .....	1103
NRCSets CfgVT property .....	135
NumberOfRetries property (IHEProfileConnection) .....	877
NumberOfRetries property (IHETransport) .....	1244
Numeric (IOhioField) .....	1347
Numeric (IOhioOIA) .....	1329
NumericCharacters property (IHEParser) .....	1105
NumericCharacters property (IHEProfileEdit) .....	529
NumTools property .....	456
NVT mode exceptions .....	1390
NVT mode functions .....	1389
NVTMode .....	1107
<b>O</b>	
object16, 38, 91, 103, 104, 105, 195, 263, 949, 1176, 1432, 1434, 1435, 1437	
Cfg3270 .....	103
Cfg5250 .....	104
CfgVT .....	105
Fields Collection .....	38
Hosts Collection .....	16
Hosts Fields Collection .....	91
IHclFtpEngine .....	1434
IHclFtpSession .....	1435
IHclFtpSessions .....	1437
Parser .....	949
Profile .....	263
Terminal .....	195
Transport .....	1176
Object group .....	181
objects .....	1498, 1499, 1506, 1521
OHIO ....	1283, 1289, 1293, 1302, 1324, 1332, 1338, 1353, 1357
data types .....	1357
OhioField interface .....	1338
OhioFields interface .....	1332
OhioManager interface .....	1283
OhioOIA interface .....	1324
OhioPosition interface .....	1353
OhioScreen interface .....	1302
OhioSession interface .....	1293
OhioSessions interface .....	1289
OHIO_COLOR .....	1358
OHIO_DIRECTION .....	1358
OHIO_EXTENDED .....	1359
OHIO_EXTENDED_COLOR .....	1359
OHIO_EXTENDED_HILITE .....	1360
OHIO_FIELD .....	1360
OHIO_INPUTINHIBITED .....	1361
OHIO_OWNER .....	1361
OHIO_PLANE .....	1362
OHIO_STATE .....	1362
OHIO_TYPE .....	1363
OHIO_UPDATE .....	1363
OhioField interface .....	1338
OhioFields interface .....	1332
OhioManager interface .....	1283
OhioOIA interface .....	1324
OhioPosition interface .....	1353
OhioScreen interface .....	1302
OhioScreen.SendKeys .....	1312
mnemonic keywords .....	1312
OhioSession interface .....	1293
OhioSessions interface .....	1289
OhioVersion .....	1286
OIA .....	1321
OIA display mode .....	247
OIA property .....	47
OIAString .....	1109
OIAStringW property .....	1111
OIAUpdated property .....	48
OLE .....	1433, 1496
scripts .....	1433
OLE automation .....	8, 1432

OLE automation reference .....	8
OLE objects.....	9, 1432, 1434, 1437, 1498
Ole objects .....	1435
OLE_COLOR .....	261
OnCopyReplace	
FieldAttributeWith property.....	1131
OnCursorMoved .....	1306
OnDisconnect Cfgxxxx property .....	136
Online CfgVT property .....	137
Online property .....	1509
OnPasteFieldModeTab	
Character property .....	1133
OnScreenChanged.....	1307
OnSessionChanged .....	1297
OnSizeChanged .....	1309
Open method.....	23
OpenSession .....	1284
Operators group .....	182
Optimized mode.....	137
OptimizedDisplayMode CfgVT property...	137
options .....	1381, 1486
emulation-specific.....	1486
overview .....	207, 1364, 1472
COM objects .....	207
legacy APIs .....	1364
Owner .....	1361
Owner (IOhioOIA) .....	1330
<b>P</b>	
page object .....	1498
Page object items .....	1521
page topic items .....	1490
paper types--codes for .....	1197
Parent property.....	82, 1439
ParentDir method.....	1461
Parser .....	202
Parser objects .....	949, 955, 1060, 1175
data types.....	1175
methods.....	955
properties .....	1060
Password (IHEProfileConnection) .....	879
Password (IHETransport).....	1246
Password Cfgxxxx method .....	138
Password property .....	1441
Paste property .....	84
PasteChar .....	531
PasteDataToScreen method .....	1007
PasteMode.....	247
PasteMode (IHEParser).....	1113
PasteMode (IHEProfileEdit) .....	533
PASVMode property .....	1444
Pause .....	1374
Pause method.....	48
PC_OVERWRITE_BEHAVIOUR.....	262
PCPrint.....	308
PenSelectable.....	1348
PerformingTransfer .....	1248
per-user files.....	1407
Plane .....	1362
Poke commands.....	1399
Poke message .....	1395
Port (IHEProfileConnection).....	881
Port (IHETransport).....	1250
PortList property (IHEProfileConnection).	883
PortList property (IHETransport).....	1252
Pos property .....	98
print file mode.....	248
print target.....	249
PrintBlackAndWhite .....	731
PrintBorder .....	733
PrintBorder Cfgxxxx property .....	138
PrintByPassWindows property.....	1115
PrintDisableTranslation property.....	1119
PrintDocname .....	735
PrintDocumentName Cfgxxxx property....	139
PrinterDeinit .....	716
PrinterDeInit Cfg3270 property .....	139
PrinterDeInitString.....	1117
PrinterFooter .....	737
PrinterHeader .....	739
PrinterInit.....	718
PrinterInit Cfg3270 property .....	140
PrinterInitString.....	1121
PrinterTimeout .....	1123
PrinterTimeoutValue property.....	1125
PrintFooter Cfgxxxx property.....	140

PrintHeader Cfgxxxx property.....	141	Profile object .....	263, 343, 406, 418, 445, 449, 465, 470, 472, 481, 549, 551, 556, 669, 700, 708, 713, 722, 753, 768, 779, 818, 827, 832, 911, 944, 948
PrintHostFile method.....	1462	data types .....	948
PrintLFtoCRLF property.....	1127	IHEProfileCursor interface .....	700
PrintLocation.....	741	moved properties.....	213
PrintLocation Cfgxxxx property .....	141	Profile interface.....	263
PrintMode7171 property.....	713	ProfileCapture interface .....	768
PrintOIA .....	743	ProfileColor interface .....	549
PrintOIA Cfgxxxx property.....	142	ProfileConnection interface .....	832
PrintReversedColors.....	745	ProfileDisplay interface .....	669
PrintScreen .....	311	ProfileEdit interface .....	481
PrintScreen method.....	49	ProfileEvents interface.....	827
PrintScreenFontName.....	747	ProfileFileTransfer interface.....	551
PrintScreen		ProfileFonts interface .....	708
FontSize .....	749	ProfileGraphics interface.....	406
PrintSession property .....	310	ProfileHostPrinting interface .....	779
PrintWorkingDir function .....	1412	ProfileHotspots interface.....	818
Profile Cfgxxxx property .....	143	ProfileKeyboard interface.....	418
		ProfileMouse interface .....	445
		ProfilePCPrint interface .....	713
		ProfilePrintScreen interface .....	722
		ProfilePrintSession interface .....	753
		ProfileSecurity interface .....	656
		ProfileSessionWindow interface .....	911
		ProfileSound interface .....	944
		ProfileTerminal interface .....	343
		ProfileToolbar interface .....	449
		ProfileTrackMenu interface .....	465
		ProfileTranslationTable interface.....	470
		ProfileVTCharset interface.....	472
		renamed interfaces .....	212
		renamed properties .....	212
		ProfileName .....	312
		ProfileName property .....	762
		ProfileTranslationTable interface .....	470
		ProgCheckCode .....	1331
		ProgramSymbols (IHEParser) .....	1129
		ProgramSymbols (IHEProfileGraphics) .....	415
		PromptOnClose.....	922
		properties1432, 1497, 1500, 1501, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1514,	

1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522	
ActionOnExist CfgVT.....	106
ALA Cfg3270.....	168
ALADisplayMode Cfg3270.....	168
ALAInputMode Cfg3270.....	168
AllowClose .....	27
AllowUpdates .....	27
AlwaysAutoSkip Cfg3270 .....	106
Answerback CfgVT .....	107
Application.....	76
Application object.....	10
Area object .....	76
AreaCode CfgVT.....	107
Attr .....	92
AutoClearXferMonitor Cfg3270 and CfgVT .....	108
AutoCopySelectedText Cfgxxxx .....	108
AutoMacro Cfgxxxx.....	109
AutoUnlockKeyboard Cfg3270 .....	109
AutoWrap CfgVT .....	110
BellMargin Cfgxxxx .....	110
BFStatus.....	29
BitMode CfgVT.....	111
BlinkToItalic Cfgxxxx .....	111
Bottom .....	78
BSIsDel CfgVT .....	112
Bytes .....	30
Capture.....	30
CaptureOIA.....	31
Cfg3270, Cfg5250, and CfgVT .....	99
ClearScreenOnSizeChange CfgVT .....	168
ColorDisplay Cfg525.....	113
Columns.....	32
CompressBlankLines InScrollbar CfgVT .....	113
ConcealAnswerback CfgVT.....	114
ConnectBy.....	33
ConnectErrorStatus .....	33
ConnectRC.....	34
ConnectTimeout Cfgxxxx .....	114
Convert Nulls Cfg3270 .....	115
Count .....	22
Country CfgVT.....	115
CRTToCRLF CfgVT .....	116
CurrentHost .....	11
Cursor.....	35
CursorKeyMode CfgVT .....	116
CursorMode Cfgxxxx .....	117
CursorSelectMode Cfgxxxx.....	168
CursorType Cfgxxxx .....	117
DefaultHeight CfgVT .....	118
DefaultRecvDir CfgVT .....	118
DefaultWidth CfgVT .....	119
Dev7171Terminal Cfg3270 .....	119
Device3279.....	36
DirectToModem CfgVT.....	120
Display3DBorder Cfgxxxx.....	120
DisplayAttr Cfg3270.....	121
DisplayControlCodes CfgVT .....	121
DisplayRowCol Cfgxxxx.....	122
DisplayUpperCase Cfgxxxx.....	122
EAB .....	37
EmulationType .....	1500
EntryAssist Cfgxxxx.....	123
ExtAttr .....	94
Field object.....	90
FileXferProtocol CfgVT .....	123
ForceAltSize Cfg3270 .....	124
ForceExactSize Cfgxxxx.....	124
FTPApplicationname Cfgxxxx .....	168
Host Cfgxxxx .....	125
Host object.....	21, 25
HostFromProfile .....	14
IHEParser.AnswerBack .....	1061
IHEParser.APIInputMode .....	1063
IHEParser.AutoWrap.....	1169
IHEParser.BufRows.....	1065
IHEParser.CanChangeScreen .....	1067
IHEParser.CaptureMode .....	1069
IHEParser.CellCopyMode .....	1071
IHEParser.ConnectBy .....	1072
IHEParser.ConnectErrorStatus.....	1075
IHEParser.ConnectRC .....	1077
IHEParser.ConvertNulls .....	1079
IHEParser.Cutmode.....	1081

IHEParser.EnableAPL.....	1083	IHEParser.TransferErrorCode .....	1171
IHEParser.EnablePrinterTimeout .....	1085	IHEParser.TransferMode .....	1173
IHEParser.GraphicsModel.....	1087	IHEParser.Transport .....	1159
IHEParser.HistoryLines .....	1089	IHEParser.TypeAheadTimeout .....	1161
IHEParser.HLLAPIName .....	1091	IHEParser.UPSS .....	1167
IHEParser.HostResponseTime.....	1093	IHEParser.Validate	
IHEParser.HostWritableString.....	1095	NumericFieldData .....	1163
IHEParser.KeyboardLocked .....	1096	IHEProfile.AddOIAToCapture.....	269
IHEParser.MaxBlockSize .....	1234	IHEProfile.AllowEmuTracing.....	288
IHEParser.ModelColumns .....	1099	IHEProfile.AllowErrorRestart.....	271
IHEParser.ModelRows .....	1101	IHEProfile.AttnFormat .....	273
IHEParser.MoveCursor		IHEProfile.Capture .....	317
OnMouseClick .....	1165	IHEProfileCapture.SaveAppend .....	768
IHEParser.NRCID .....	1103	IHEProfileCapture.SaveConfirm .....	771
IHEParser.NumericCharacters.....	1105	IHEProfileCapture.SaveFileName .....	773
IHEParser.NVTMode .....	1107	IHEProfileCapture.SaveMode .....	775
IHEParser.OIAString .....	1109	IHEProfileCapture.VTCaptureMode ..	777
IHEParser.OIAStringW .....	1111	IHEProfile.ClearPassword .....	275
IHEParser.OnCopyReplace		IHEProfile.Color .....	276
FieldAttributeWith .....	1131	IHEProfileColor.Schemes .....	550
IHEParser.OnPasteField		IHEProfile.Connection .....	278
ModeTabCharacter .....	1133	IHEProfileConnection.acPassword .....	833
IHEParser.PasteMode .....	1113	IHEProfileConnection.	
IHEParser.PrintByPassWindows.....	1115	AlwaysPromptForHostName .....	835
IHEParser.PrintDisableTranslation...	1119	IHEProfileConnection.AreaCode .....	837
IHEParser.PrinterDeInitString.....	1117	IHEProfileConnection.	
IHEParser.PrinterInitString.....	1121	AutoEndMacroName .....	839
IHEParser.PrinterTimeout .....	1123	IHEProfileConnection.	
IHEParser.PrinterTimeoutValue.....	1125	AutoMacroName .....	841
IHEParser.PrintLFtoCRLF .....	1127	IHEProfileConnection.	
IHEParser.ProgramSymbols.....	1129	AutoRunMacroDelayTime .....	843
IHEParser.SaveAppend.....	1135	IHEProfileConnection.AutoSignOn....	845
IHEParser.SaveFileName.....	1137	IHEProfileConnection.	
IHEParser.ScreenChanged .....	1139	BackspaceKeyInterpretation .....	847
IHEParser.ScreenCol .....	1141	IHEProfileConnection.ConnectBy .....	849
IHEParser.ScreenRow.....	1143	IHEProfileConnection.Country.....	851
IHEParser.SelectionMode .....	1145	IHEProfileConnection.CountryCode..	853
IHEParser.SessionName .....	1147	IHEProfileConnection.CountryID .....	855
IHEParser.SoftCharacterSetID .....	1149	IHEProfileConnection.DeviceName ...	857
IHEParser.StatusLineMode .....	1151	IHEProfileConnection.	
IHEParser.TerminalID .....	1153	DirectToModem .....	859
IHEParser.TerminalModel .....	1155	IHEProfileConnection.EnableEMode .	861
IHEParser.Text.....	1157	IHEProfileConnection.	

IHEProfileConnection.EnableInfiniteRetries .....	863
IHEProfileConnection.Enter	
KeyInterpretation .....	865
IHEProfileConnection.HostName .....	867
IHEProfileConnection.Keyboard	
BufferMode.....	869
IHEProfileConnection.LUName.....	871
IHEProfileConnection.Modem .....	873
IHEProfileConnection.ModemID .....	875
IHEProfileConnection.	
NumberOfRetries .....	877
IHEProfileConnection.Password .....	879
IHEProfileConnection.Port.....	881
IHEProfileConnection.PortList.....	883
IHEProfileConnection.Retry	
DelayTimeBetweenHosts .....	885
IHEProfileConnection.	
ShowDialupDlg .....	887
IHEProfileConnection.SilentConnect.	889
IHEProfileConnection.	
SYSREQasIACIP .....	891
IHEProfileConnection.TelnetEcho ....	893
IHEProfileConnection.TelnetName....	895
IHEProfileConnection.Timeout.....	897
IHEProfileConnection.	
UponDisconnect .....	899
IHEProfileConnection.UseDialProp...	901
IHEProfileConnection.UserName .....	903
IHEProfileConnection.VTDoHost	
WindowSize.....	905
IHEProfileConnection.VTInitiate	
TelnetNegotiation.....	907
IHEProfileConnection.VTLineMode..	909
IHEProfile.Cursor .....	279
IHEProfileCursor interface.....	700
IHEProfileCursor.CursorMode.....	700
IHEProfileCursor.CursorType .....	702
IHEProfileCursor.Display	
CrossHairCursor .....	704
IHEProfileCursor.MoveCursor	
OnMouseClick .....	706
IHEProfile.DDEServerName.....	281
IHEProfile.Display .....	283
IHEProfileDisplay.BlinkToItalic .....	669
IHEProfileDisplay.ColumnSeparators.	672
IHEProfileDisplay.DisplayInOIA .....	674
IHEProfileDisplay.DisplayRowCol.....	676
IHEProfileDisplay.DisplayUpperCase.	678
IHEProfileDisplay.ShowNulls.....	680
IHEProfileDisplay.StatusLineMode.....	682
IHEProfileDisplay.VTClearScreen	
OnSizeChange.....	684
IHEProfileDisplay.VTHostWritable	
StatusLine.....	686
IHEProfileDisplay.VTISOColors.....	688
IHEProfileDisplay.	
VTMaxScrollBufferSize .....	690
IHEProfileDisplay.VTResetISOColors	692
IHEProfileDisplay.VTSaveAttribs	
InScrollbar .....	694
IHEProfileDisplay.	
VTSaveEraseScreens .....	696
IHEProfileDisplay.VTScrollNoBlanks.	698
IHEProfile.Edit .....	284
IHEProfileEdit.AlwaysAutoskip .....	483
IHEProfileEdit.AutoCopy .....	485
IHEProfileEdit.AutoCopy	
KeepSelection .....	487
IHEProfileEdit.BellMargin.....	489
IHEProfileEdit.CellDelimited .....	491
IHEProfileEdit.ClipFormatBitmap .....	493
IHEProfileEdit.ClipFormatCSV .....	495
IHEProfileEdit.ClipFormatHE .....	497
IHEProfileEdit.ClipFormatPasteLink..	499
IHEProfileEdit.ClipFormatRTF .....	501
IHEProfileEdit.ClipFormatText.....	503
IHEProfileEdit.ConvertNulls .....	505
IHEProfileEdit.CSVCopyEmptyFields	507
IHEProfileEdit.CSVTrimFields .....	509
IHEProfileEdit.CutChar.....	511
IHEProfileEdit.CutMode .....	513
IHEProfileEdit.EntryAssist.....	515
IHEProfileEdit.InsertResetByAttn .....	517
IHEProfileEdit.LeftMargin.....	519
IHEProfileEdit.MoveCursorAfterPaste	521
IHEProfileEdit.MultiLineDelete .....	523

IHEProfileEdit.MultiLineInsert.....	525
IHEProfileEdit.NoLockKeyb .....	527
IHEProfileEdit.NumericCharacters .....	529
IHEProfileEdit.PasteChar.....	531
IHEProfileEdit.PasteMode .....	533
IHEProfileEdit.RemoveTrailingBlank	
OnCopy .....	535
IHEProfileEdit.ResetMDT	
OnEraseInput .....	537
IHEProfileEdit.RespectNumeric .....	539
IHEProfileEdit.RightMargin .....	541
IHEProfileEdit.SmartInsert .....	543
IHEProfileEdit.WordWrap.....	545
IHEProfileEdit.WordWrapWith	
NewLineReturn .....	547
IHEProfile.EmuTraceFilename .....	286
IHEProfile.EnableHLLAPITracing.....	290
IHEProfile.EnableTracing .....	292
IHEProfile.Events .....	293
IHEProfileEvents.EnableEvents.....	828
IHEProfileEvents.Schemes .....	830
IHEProfile.FileTransfer.....	295
IHEProfileFileTransfer.1KPacket .....	632
IHEProfileFileTransfer.Append .....	553
IHEProfileFileTransfer.ASCII .....	555
IHEProfileFileTransfer.AutoCC.....	557
IHEProfileFileTransfer.	
AutoClearMonitor.....	559
IHEProfileFileTransfer.BlkSiz .....	561
IHEProfileFileTransfer.CRLF .....	563
IHEProfileFileTransfer.Custom .....	
TransferTable.....	565
IHEProfileFileTransfer.Default	
DownloadPath.....	567
IHEProfileFileTransfer.	
DefaultProtocol .....	569
IHEProfileFileTransfer.	
DefaultRecvDir .....	571
IHEProfileFileTransfer.	
DefaultUploadPath.....	573
IHEProfileFileTransfer.Download	
HostName.....	575
IHEProfileFileTransfer.Download	
PCFileName .....	577
IHEProfileFileTransfer.Download	
Translate.....	579
IHEProfileFileTransfer.ExtraOptions ..	581
IHEProfileFileTransfer.	
FileExistAction .....	583
IHEProfileFileTransfer.Host .....	585
IHEProfileFileTransfer.	
INDFILEName.....	587
IHEProfileFileTransfer.	
KmBinaryPrefix .....	589
IHEProfileFileTransfer.KmRLE .....	591
IHEProfileFileTransfer.KmTextMode..	593
IHEProfileFileTransfer.	
KmUseFullPath.....	595
IHEProfileFileTransfer.Lrecl.....	597
IHEProfileFileTransfer.QuickMode .....	599
IHEProfileFileTransfer.Recfm .....	601
IHEProfileFileTransfer.Schemes.....	603
IHEProfileFileTransfer.ShowRecvDlg..	605
IHEProfileFileTransfer.Upload	
HostName .....	607
IHEProfileFileTransfer.Upload	
PCFileName .....	609
IHEProfileFileTransfer.	
UploadTranslate.....	611
IHEProfileFileTransfer.User	
DefinedDownload.....	613
IHEProfileFileTransfer.	
UserDefinedUpload.....	615
IHEProfileFileTransfer.VTAuto	
ClearMonitor .....	617
IHEProfileFileTransfer.XferBlockSize ..	619
IHEProfileFileTransfer.XferDest .....	621
IHEProfileFileTransfer.	
XferHostCodePage .....	623
IHEProfileFileTransfer.	
XferPCCodePage .....	626
IHEProfileFileTransfer.XferSource.....	628
IHEProfileFileTransfer.	
XferStartAction .....	630
IHEProfileFileTransfer.	
XmAckTimeout .....	634

IHEProfileFileTransfer.XmCRC .....	636
IHEProfileFileTransfer.	
YmAckTimeout .....	638
IHEProfileFileTransfer.	
YmUseFullPath.....	640
IHEProfileFileTransfer.	
ZmAutoDownload .....	642
IHEProfileFileTransfer.	
ZmCrashRecovery .....	644
IHEProfileFileTransfer.ZmMaxErrors	646
IHEProfileFileTransfer.Zm	
OverwriteMngmt.....	648
IHEProfileFileTransfer	
.ZmSlideWindow .....	650
IHEProfileFileTransfer.	
ZmUseFullPath.....	652
IHEProfileFileTransfer.	
ZmWindowSize .....	654
IHEProfile.Fonts .....	296
IHEProfileFonts.CharacterSpacing .....	709
IHEProfileFonts.VariableWidthFont ..	711
IHEProfile.GlobalSettingsPath .....	298
IHEProfile.Graphics.....	300
IHEProfileGraphics.APL .....	406
IHEProfileGraphics.	
GraphicsCursorType .....	409
IHEProfileGraphics.GraphicsModel ..	411
IHEProfileGraphics.LightPen.....	413
IHEProfileGraphics.ProgramSymbols	415
IHEProfileGraphics.PSCellSize .....	417
IHEProfile.HLLAPITraceFilename .....	332
IHEProfile.HostPrinting.....	340
IHEProfileHostPrinting.	
VTAutoFormFeed .....	779
IHEProfileHostPrinting.	
VTiPrintMaxCols .....	782
IHEProfileHostPrinting.VTi	
PrintMaxRows.....	784
IHEProfileHostPrinting.	
VTPassThruUPSS.....	786
IHEProfileHostPrinting.VTPrint	
DefaultFont.....	790
IHEProfileHostPrinting.VTPrint	
DisableTranslation .....	792
IHEProfileHostPrinting.	
VTPrinterDeinit .....	794
IHEProfileHostPrinting.VTPrinter	
EnableTimeout.....	796
IHEProfileHostPrinting.VTPrinter	
FontName .....	798
IHEProfileHostPrinting.VTPrinterInit800	
IHEProfileHostPrinting.VTPrinter	
Timeout.....	802
IHEProfileHostPrinting.VTPrinter	
TimeoutValue .....	804
IHEProfileHostPrinting.VTPrintFile...	806
IHEProfileHostPrinting.	
VTPrint FileMode.....	808
IHEProfileHostPrinting.VTPrint	
FitFontToPage .....	810
IHEProfileHostPrinting.VTPrintLF	
toCRLF .....	812
IHEProfileHostPrinting.	
VTPrint Target.....	814
IHEProfileHostPrinting.VTUse	
SpecificPrinter .....	816
IHEProfile.Hotspots .....	301
IHEProfileHotspots.DisplayStyle .....	818
IHEProfileHotspots.EnableHotspots...	821
IHEProfileHotspots.MouseActivation.	823
IHEProfileHotspots.Schemes.....	825
IHEProfile.Keyboard .....	303
IHEProfileKeyboard.	
AllowAIDKeyRepeat .....	419
IHEProfileKeyboard.AllowDiac .....	422
IHEProfileKeyboard.	
CurrentKeyboard .....	424
IHEProfileKeyboard.KeyboardType .....	426
IHEProfileKeyboard.	
LockOnAttention .....	428
IHEProfileKeyboard.MapNumLock....	430
IHEProfileKeyboard.RemapKeypad .....	432
IHEProfileKeyboard.TypeAhead .....	434
IHEProfileKeyboard.	
TypeAheadTimeout .....	436
IHEProfileKeyboard.VTCursor	

IHEProfilePrintSession.KeyApplMode.....	438
IHEProfileKeyboard.VTEnableBreak..	439
IHEProfileKeyboard.	
VTKeypadApplMode .....	442
IHEProfileKeyboard.	
VTNewLineMode.....	444
IHEProfile.KillMacrosOnSessionExit .	305
IHEProfile.Mouse .....	307
IHEProfileMouse.BlockSelect .....	446
IHEProfileMouse.SelectHilight .....	448
IHEProfile.PCPrint .....	308
IHEProfilePCPrint.PrinterDeinit .....	716
IHEProfilePCPrint.PrinterInit .....	718
IHEProfilePCPrint.PrintMode7171 ....	713
IHEProfilePCPrint.TprintMode.....	720
IHEProfile.PrintScreen .....	311
IHEProfilePrintScreen.AddFormFeed	722
IHEProfilePrintScreen.	
DisplayAbortDlg.....	725
IHEProfilePrintScreen.	
DisplayPrintDlg .....	727
IHEProfilePrintScreen.	
HostScreenPerPage .....	729
IHEProfilePrintScreen.Prin	
BlackAndWhite .....	731
IHEProfilePrintScreen.PrintBorder ....	733
IHEProfilePrintScreen.PrintDocname	735
IHEProfilePrintScreen.PrinterFooter..	737
IHEProfilePrintScreen.PrinterHeader.	739
IHEProfilePrintScreen.PrintLocation .	741
IHEProfilePrintScreen.PrintOIA.....	743
IHEProfilePrintScreen.Print	
ReversedColors.....	745
IHEProfilePrintScreen.Print	
ScreenFontName .....	747
IHEProfilePrintScreen.PrintScreenFontP	
ointSize .....	749
IHEProfilePrintScreen.PRTSCRUse	
SpecificPrinter .....	751
IHEProfile.PrintSession .....	310
IHEProfilePrintSession.HostName .....	753
IHEProfilePrintSession.LimitTo	
SingleInstance .....	756
IHEProfilePrintSession.LUName .....	758
IHEProfilePrintSession.LUType .....	760
IHEProfilePrintSession.ProfileNam...	762
IHEProfilePrintSession.StartPrinter .....	764
IHEProfilePrintSession.StopPrinter .....	766
IHEProfile.ProfileNam...	312
IHEProfile.QueryShutdown.....	315
IHEProfile.RecordPortableMacros .....	318
IHEProfile.ReRunAutoMacro .....	320
IHEProfile.Security.....	322
IHEProfileSecurity.Kerberos.....	656
IHEProfileSecurity.KerberosAltName.	659
IHEProfileSecurity.Kerberos	
Encryption.....	661
IHEProfileSecurity.Kerberos	
ForwardTkt .....	663
IHEProfileSecurity.KerberosVersion...	665
IHEProfileSecurity.SecurityOption .....	667
IHEProfile.SessionWindow .....	323
IHEProfileSessionWindow.	
DisplayBorder .....	911
IHEProfileSessionWindow.EnableWorks	
paceBackgroundBitmap.....	914
IHEProfileSessionWindow.	
FullScreenMode .....	916
IHEProfileSessionWindow.	
KeepFontAspectRatio .....	918
IHEProfileSessionWindow.	
LongName.....	920
IHEProfileSessionWindow.	
PromptOnClose .....	922
IHEProfileSessionWindow.	
ResizeBehavior .....	924
IHEProfileSessionWindow.	
SaveFontOnExit .....	928
IHEProfileSessionWindow.	
SaveProfOnClose .....	926
IHEProfileSessionWindow.	
SnapFrameBack .....	930
IHEProfileSessionWindow.	
SwitchScreenType .....	932
IHEProfileSessionWindow.	
WindowState .....	934

IHEProfileSessionWindow.WindowTitle.....	936
IHEProfileSessionWindow.WorkSpace.BackgroundBitmap .....	938
IHEProfileSessionWindow.Workspace.BackgroundColor .....	940
IHEProfileSessionWindow.Workspace.ForegroundColor.....	942
IHEProfile.Sound.....	325
IHEProfileSound.Notify .....	944
IHEProfileSound.Sound .....	946
IHEProfile.Terminal .....	326
IHEProfileTerminal.AlternateScreen ..	344
IHEProfileTerminal.CharacterSet .....	347
IHEProfileTerminal.CustomModel ....	364
IHEProfileTerminal.Custom.ModelCols .....	366
IHEProfileTerminal.Custom.ModelRows.....	368
IHEProfileTerminal.DetectChainedIO	349
IHEProfileTerminal.ForceAltSize.....	351
IHEProfileTerminal.New3270EAB ....	353
IHEProfileTerminal.NewModel3279 ..	355
IHEProfileTerminal.NewModelType..	357
IHEProfileTerminal.ReplyOEM .....	360
IHEProfileTerminal.ShortName .....	362
IHEProfile.TerminalType .....	328
IHEProfileTerminal.VT8BitMode .....	370
IHEProfileTerminal.VTAnswerback ...	372
IHEProfileTerminal.VTAutoResize .....	374
IHEProfileTerminal.VTBSSIsDel.....	376
IHEProfileTerminal.VTConcealAnswerback .....	378
IHEProfileTerminal.VTDefCols.PerScreen .....	380
IHEProfileTerminal.VTDefLines.PerScreen .....	382
IHEProfileTerminal.VTDisplayMode ..	384
IHEProfileTerminal.VTEnableSSH....	386
IHEProfileTerminal.VTForce8Bit .....	388
IHEProfileTerminal.VTLocalEcho.....	390
IHEProfileTerminal.VTNTerminalType .....	392
IHEProfileTerminal.VTOnLine.....	395
IHEProfileTerminal.VTScrollSpeed ....	397
IHEProfileTerminal.VTSmoothScroll ..	399
IHEProfileTerminal.VTTerminalID ....	401
IHEProfileTerminal.VTWrapLine.....	404
IHEProfile.Toolbar .....	330
IHEProfileToolbar.BigToolbar .....	450
IHEProfileToolbar.Button .....	452
IHEProfileToolbar.MaxBitmaps.....	454
IHEProfileToolbar.NumTools .....	456
IHEProfileToolbar.ShowTips .....	458
IHEProfileToolbar.TBUserBitmaps....	460
IHEProfileToolbar.ToolbarDockType	462
IHEProfileToolbar.ToolbarFilename ..	464
IHEProfile.TrackMenu .....	334
IHEProfileTrack.TrackCommands .....	466
IHEProfileTrack.TrackLabels .....	468
IHEProfile.TranslationTable.....	335
IHEProfileTranslationTable.CurrentLanguage .....	470
IHEProfile.UserDirectory .....	337
IHEProfile.VTCharset .....	339
IHEProfileVTCharSet.VTForceNRC ..	472
IHEProfileVTCharset.VTNRC .....	475
IHEProfileVTCharset.VTNRCMode ..	477
IHEProfileVTCharset.VTUPSS .....	479
IHEProfile.WinDDEEnabled .....	342
IHETerminal.ConnectBy .....	199
IHETerminal.Connected.....	200
IHETerminal.Host .....	201
IHETerminal.Parser .....	202
IHETerminal.Session .....	203
IHETerminal.SilentConnect .....	203
IHETerminal.TCPPort.....	204
IHETerminal.Transport .....	205
IHETerminal.UserDir .....	206
IHETransport.AttentionFormat .....	1204
IHETransport.CharSet .....	1206
IHETransport.CodePage .....	1208
IHETransport.Connected .....	1210
IHETransport.ConnectionStatus.....	1212
IHETransport.DeviceType .....	1216
IHETransport.EnableEMode .....	1218

IHETransport.EnableSSH .....	1220	IOhioFields.Count.....	1336
IHETransport.EnableTracing.....	1222	IOhioFields.Item .....	1337
IHETransport.HostAddress.....	1224	IOhioField.Start.....	1350
IHETransportHostName.....	1226	IOhioField.String.....	1351
IHETransport.IsEncrypted .....	1228	IOhioManager.OhioVersion.....	1286
IHETransport.IsReceiveBlocked .....	1230	IOhioManager.Sessions.....	1286
IHETransport.Keyboard.....	1232	IOhioManager.VendorName.....	1287
IHETransport.LineMode.....	1280	IOhioManager.VendorObject.....	1288
IHETransport.LUNameRequested....	1214	IOhioManager.Vendor	
IHETransport.MessageQueueLibrary	1236	ProductVersion .....	1288
IHETransport.MessageQueueName .	1238	IOhioOIA.Alphanumeric.....	1324
IHETransport.ModelColumns.....	1239	IOhioOIA.APL .....	1325
IHETransport.ModelRows .....	1242	IOhioOIA.CommCheckCode.....	1326
IHETransport.NumberOfRetries .....	1244	IOhioOIA.InputInhibited .....	1327
IHETransport.Password .....	1246	IOhioOIA.InsertMode .....	1328
IHETransport.PerformingTransfer ...	1248	IOhioOIA.MachineCheckCode .....	1328
IHETransport.Port.....	1250	IOhioOIA.Numeric.....	1329
IHETransport.PortList.....	1252	IOhioOIA.Owner .....	1330
IHETransport.Retry		IOhioOIA.ProgCheckCode.....	1331
DelayTimeBetweenHosts .....	1253	IOhioPosition.Column .....	1355
IHETransport.SecurityOpton.....	1256	IOhioPosition.Row.....	1356
IHETransport.SessionKeepAlive .....	1258	IOhioScreen.Columns.....	1319
IHETransport.TelnetEcho .....	1260	IOhioScreen.Cursor .....	1319
IHETransport.TelnetIsLineMode.....	1262	IOhioScreen.Fields .....	1320
IHETransport.TelnetIsLocalEcho .....	1264	IOhioScreen.OIA.....	1321
IHETransport.TelnetName .....	1266	IOhioScreen.Rows .....	1322
IHETransport.TerminalModel.....	1268	IOhioScreen.String .....	1323
IHETransport.TerminalOnline .....	1270	IOhioSession.CanChangeScreen .....	1298
IHETransport.TerminalType .....	1272	IOhioSession.ConfigurationResource	1300
IHETransport.TNESession .....	1274	IOhioSessions.Count.....	1290
IHETransport.TraceFilename.....	1276	IOhioSession.Screen .....	1300
IHETransport.Username .....	1278	IOhioSession.SessionName.....	1301
Index.....	41	IOhioSession.SessionType .....	1302
InsertMode.....	41	IOhioSessions.Item .....	1291
IOhioField.Attribute .....	1340	IsBold .....	95
IOhioField.End .....	1342	IsConnected .....	42
IOhioField.HighIntensity .....	1343	IsHidden .....	95
IOhioField.Length.....	1344	IsModified .....	96
IOhioField.Modified .....	1345	IsNumeric .....	96
IOhioField.Normal .....	1346	IsPenSelectable .....	97
IOhioField.Numeric .....	1347	IsProtected .....	97
IOhioField.PenSelectable.....	1348	IsXfer.....	42
IOhioField.Protected .....	1349	Item.....	22

KermitBinPrefix CfgVT .....	127
KermitCompression CfgVT .....	127
KermitTextMode CfgVT .....	128
KermitUseFullPath CfgVT .....	128
Keyboard .....	43
KeypadMode CfgVT .....	129
Left.....	82
LeftMargin Cfgxxxx .....	130
Length.....	98
Linemode CfgVT .....	131
LinesInScrollbar CfgVT .....	130
LocalEcho CfgVT .....	131
LongName Cfgxxxx .....	132
LUName Cfgxxxx.....	132
Model .....	46
Modem CfgVT .....	133
MouseToCursor.....	47
MultiLineDelete Cfgxxxx.....	133
MultiLineInsert Cfgxxx.....	134
Next .....	23
Notify Cfgxxxx .....	134
NRCSet CfgVT.....	135
OhioField interface .....	1338
OhioFields interface .....	1332
OhioManager interface .....	1283
OhioOIA interface .....	1324
OhioPosition interface.....	1353
OhioScreen interface.....	1302
OhioSession interface .....	1293
OhioSessions interface .....	1289
OIA.....	47
OIAUpdated.....	48
OLE object.....	9
OnDisconnect Cfgxxxx.....	136
Online CfgVT.....	137
OptimizedDisplayMode CfgVT .....	137
Parent .....	82
Parser obejcts .....	1060
Paste .....	84
Pos .....	98
PrintBorder Cfgxxxx.....	138
PrintDocumentName Cfgxxxx.....	139
PrinterDeInit Cfg3270 .....	139
PrinterInit Cfg3270 .....	140
PrintFooter Cfgxxxx .....	140
PrintHeader Cfgxxxx.....	141
PrintLocation Cfgxxx .....	141
PrintOIA Cfgxxxx.....	142
Profile Cfgxxxx .....	143
Profile interface.....	263
ProfileCapture interface .....	768
ProfileColor interface .....	549
ProfileConnection interface .....	832
ProfileDisplay interface .....	669
ProfileEdit interface.....	481
ProfileEvents interface .....	827
ProfileFileTransfer interface.....	551
ProfileFonts interface .....	708
ProfileGraphics interface .....	406
ProfileHostPrinting interface .....	779
ProfileHotspots interface.....	818
ProfileKeyboard interface.....	418
ProfileMouse interface .....	445
ProfilePCPrint interface .....	713
ProfilePrintScreen interface .....	722
ProfilePrintSession interface .....	753
ProfileSecurity interface .....	656
ProfileSessionWindow interface .....	911
ProfileSound interface .....	944
ProfileTerminal interface .....	343
ProfileToolbar interface .....	449
ProfileTrackMenu interface .....	465
ProfileTranslationTable interface.....	470
ProfileVTCharset interface.....	472
ProportionalFonts Cfgxxxx.....	143
PSReserved .....	50
PSUpdated .....	50
QueryCloseRequest .....	51
QuickKeyFile .....	52
RawAddFormFeed Cfgxxxx .....	144
RawCaptureMode CfgVT .....	144
renamed or moved .....	210
ReplyOEM Cfg3270.....	145
ReRunAutoMacro Cfgxxx.....	145
RespectNumeric Cfg3270.....	146
Right .....	85

RightMargin Cfgxxxx.....	146
Rows .....	55
SaveAppend Cfgxxxx .....	147
SaveAttrsInScrollback CfgVT .....	147
SaveConfirm Cfgxxxx .....	148
SaveFile Cfgxxxx .....	148
SaveFontOnExit Cfgxxxx.....	149
SaveMode Cfgxxxxx.....	149
SaveProfileOnClose Cfgxxxx .....	150
ShortName .....	61
ShowDialupDlg CfgVT .....	151
ShowHotspots Cfgxxxx.....	151
ShowNulls Cfgxxxx.....	152
ShowPoppad Host .....	168
ShowRecvDialog CfgVT .....	152
SilentConnect .....	63
SmoothScrolling CfgVT .....	153
SmoothScrollSpeed CfgVT .....	153
Sound Cfgxxxxx .....	154
StatusLineMode CfgVT .....	154
TCPPort Cfgxxxx .....	156
TelnetEcho Cfgxxxx .....	156
TelnetName Cfgxxxx .....	157
Terminal objects.....	199
TerminalMode .....	65
Text.....	66, 99
TN3270.....	67
Top .....	87
TPRINTDestination Cfg3270 .....	157
Transport objects .....	1203
Type.....	88
TypeAhead Cfgxxxx .....	158
unsupported.....	168
UPSSet CfgVT .....	158
UseDialProperties CfgVT .....	159
Value .....	89
WindowTitle Cfgxxxx .....	160
WordWrap Cfgxxxx.....	160
XferBlockSize Cfg3270.....	161
XferCount .....	75
XferHostSystem Cfg3270.....	161
XferProgramName Cfg3270 .....	162
XferRC.....	75
XferStartAction Cfg3270 .....	162
XModem16BitCrc CfgVT .....	163
XModemPkt1024 CfgVT .....	163
XModemSendTimeout CfgVT .....	164
YModemSendTimeout CfgVT.....	164
YModemUseFullPath CfgVT .....	165
ZModemAutoDownload CfgVT .....	165
ZModemCrashRecovery CfgVT .....	165
ZModemMaxErr CfgVT .....	166
ZModemOverwrite CfgVT.....	166
ZModemSlidingBytes CfgVT .....	167
ZModemSlidingWin CfgVT .....	167
ZModemUseFullPath CfgV .....	168
property.....	788
IHEProfileHostPrinting.VTPrint ByPassWindows .....	788
ProportionalFonts Cfgxxxx property .....	143
Protected .....	1349
ProtectedText method .....	49
PRTSCRUseSpecificPrinter property.....	751
PSCellSize.....	417
PSReserved property.....	50
PSUpdated property .....	50
PutFile function .....	1416
PutFile method.....	1458
PutString .....	1310
PutString method.....	1009
PutText method (Host) .....	51
PutText method (IHEParser) .....	1011
PutTree function .....	1419
PutTree method .....	1458
<b>Q</b>	
QueryCloseRequest property.....	51
QueryCursorPosition.....	1374
QueryFieldAttribute.....	1375
QuerySessions .....	1375
QuerySessionStatus.....	1376
QueryShutdown .....	315
QuickKeyFile property.....	52
QuickMode .....	599
Quit method.....	1440
QuoteCommand function .....	1431

QuoteCommand method.....	1458
<b>R</b>	
RawAddFormFeed Cfgxxxx property .....	144
RawCaptureMode CfgVT property .....	144
ReadData method.....	1505
ReadKeyboardFile item .....	1485
ReadKeyboardFile method.....	1512
Realistic Normal mode.....	137, 153
Realistic Smooth mode.....	153
Receive File command syntax .....	1398
ReceiveFile .....	1376
ReceiveFile method (Host).....	52
ReceiveFile method (IHEParser).....	1013
Recfm .....	601
RecordPortableMacros property.....	318
Refresh (IOhioFields) .....	1335
Refresh (IOhioSessions) .....	1289
relationship between COM objects .....	208
Release .....	1377
RelpyTimeout property.....	1443
RemapKeypad.....	432
RemoteHelp method .....	1454
RemoveAllSessions method .....	1468
RemoveEvent method .....	827, 1505
RemoveFeature.....	208
RemoveHostDir function.....	1413
RemoveHostDir method.....	1454
RemoveSession method .....	1467
RemoveSessionByName method .....	1468
RemoveSessionByNumber method .....	1468
RemoveTrailingBlank	
OnCopy property .....	535
RenameHostFile function .....	1418
RenameHostFile method .....	1454
ReplaceSel method .....	1015
ReplyOEM .....	360
ReplyOEM Cfg3270 property.....	145
ReplyText method .....	1453
Request commands .....	1400
Request message .....	1395
request syntax .....	1473
ReRunAutoMacro Cfgxxxx property.....	145
ReRunAutoMacro property.....	320
Reserve .....	1377
Reset .....	1377
ResetMDT	
OnEraseInput property.....	537
resize behavior.....	249
ResizeBehavior property .....	924
RespectNumeric.....	539
RespectNumeric Cfg3270 property .....	146
Restore method .....	54
retrieving .....	1433
FTP Sessions collection .....	1433
RetryDelayTimeBetweenHosts property...	1253
RetryDelayTimeBetween	
Hosts property .....	885
return codes.....	1384
Return Extra Session Info .....	1386
Right property .....	85
RightMargin .....	541
RightMargin Cfgxxxx property.....	146
Row .....	1356
Row method.....	54
Rows .....	1322
Rows property .....	55
RunCmd method .....	55
RunQuickKey method .....	56
<b>S</b>	
sample .....	219, 223, 226, 227
Hex3270.hxl.....	219
Hex5250.hxl.....	223
VT_NRC.hxl.....	227
VT_UPS.hxl.....	226
sample scripts .....	1410
samples .....	1404
Save method .....	268
save options .....	250
SaveAppend .....	768
SaveAppend Cfgxxxx property .....	147
SaveAppend property .....	1135
SaveAttrsInScrollbar CfgVT property.....	147
SaveConfirm .....	771
SaveConfirm Cfgxxxx property .....	148

SaveData item .....	1479
SaveFile Cfgxxxx property .....	148
SaveFileName .....	773
SaveFileName property .....	1137
SaveFontOnExit.....	928
SaveFontOnExit Cfgxxxx property .....	149
SaveMode.....	775
SaveMode Cfgxxxxx property .....	149
SaveProfile Cfgxxxx method .....	150
SaveProfileOnClose Cfgxxxx property.....	150
SaveProfOnClose.....	926
SaveQuickKeyFile method .....	57
SaveScreen method.....	57
SaveScrollbar method.....	58
Schemes property (IHEProfileColor) .....	550
Schemes property (IHEProfileEvents) .....	830
Schemes property (IHEProfileFileTransfer).....	603
Schemes property (IHEProfileHotspots)....	825
Screen.....	1300
ScreenChanged property .....	1139
ScreenCol property.....	1141
ScreenRow property .....	1143
scripts .....	1410, 1433
creating.....	1410, 1433
Search method.....	58
SearchField.....	1378
SearchPS .....	1378
Security .....	322
SecurityOption .....	250
SecurityOption (IHEProfileSecurity).....	667
SecurityOption (IHETransport) .....	1256
Select method .....	86
SelectHilight property (IHEProfileMouse). 448	
SelectionMode .....	251
SelectionMode (IHEParser) .....	1145
Send File command syntax.....	1398
SendAccountCommand method .....	1456
SendAid method.....	1017
SendData	
.....	1186
SendFile .....	1379
SendFile method (Host) .....	59
SendFile method (IHEParser) .....	1025
SendFunctionKey.....	1188
SendKeepAlive .....	1190
SendKey.....	1379
SendKeys .....	1311
SendKeys method .....	1027
SendString item.....	1481
SendString method .....	1503
SendStringToTerminal item .....	1481
SendStringToTerminal method.....	1504
ServerBanner method .....	1452
ServerName property.....	1441
ServerPort property.....	1443
Session .....	203
Session methods.....	1433
Connect to Host .....	1433
Get.....	1433
User Login .....	1433
SessionByName property .....	1467
SessionByNumber property.....	1466
SessionKeepAlive .....	1258
SessionName (IHEParser) .....	1147
SessionName (IOhioSession) .....	1301
sessions .....	17, 1286
creating .....	17
Sessions property .....	1440
SessionType .....	1302
SessionType setting .....	1427
SessionWindow property.....	323
SetAttributeColor item .....	1485
SetAttributeColor method.....	1513
SetAttributeUsage item.....	1486
SetAttributeUsage method .....	1513
SetCharacter method .....	1525
SetCharacterAttrib method .....	1526
SetCharacterBColor method .....	1527
SetCharacterFGColor method.....	1526
SetCursorPosition .....	1380
SetCursorPosition method .....	1029
SetFeature method .....	1031
SetFeature method (IHETransport) .....	1192
SetFieldText method.....	1033
SetFont method.....	60, 708
SetHostPrintTransformInfo method.....	1194

SetKerberosInfo method .....	1198
SetLocalFilenamesLowercase setting.....	1428
SetOption item .....	1484
SetOption item values .....	1486
SetReplyTimeoutValue setting.....	1423
SetSel method .....	1035
SetSessionParameters .....	1380, 1381
valid options.....	1381
settings .....	1367, 1409
Settings group .....	182
SetTransferType setting.....	1421
SetValue method .....	1037
ShortName property (Host).....	61
ShortName property (IHEProfileTerminal).....	362
Show method.....	62
ShowDialupDlg .....	887
ShowDialupDlg CfgVT property .....	151
ShowHotspots Cfgxxxx property .....	151
ShowNulls.....	680
ShowNulls Cfgxxxx property .....	152
ShowPoppad Host property .....	168
ShowRecvDialog CfgVT property.....	152
ShowRecvDlg property.....	605
ShowTips property .....	458
ShowToolbar method.....	62
SilentConnect .....	203
SilentConnect property (Host) .....	63
SilentConnect property (IHEProfileConnection) .....	889
SmartInsert property .....	543
SmoothScroll property .....	1510
SmoothScrolling CfgVT property .....	153
SmoothScrollSpeed CfgVT property .....	153
SnapFrameBack .....	930
SoftCharacterSetID.....	1149
Sound .....	946
Sound Cfgxxxxx property.....	154
Sound property .....	325
source code .....	1410
special HLLAPI and WinHLLAPI flags ....	1366
special HLLAPI flags... .....	1385, 1386, 1387, 1388
Allow Connect Physical.....	1387
Auto Start Delay .....	1385
Auto Sync.....	1387
Auto Unload .....	1385
Convert Nulls .....	1388
Return Extra Session Info.....	1386
Start Minimized.....	1385
Update Screen After Copy.....	1388
Yield Wait .....	1386
SSHMode property .....	1445
Start .....	1350
Start Minimized .....	1385
Start Session example.....	1406
StartDataSave item .....	1482
StartDataSave method .....	1505
starting a conversation .....	1403
with HostExplorer .....	1403
with Microsoft Excel.....	1403
with Word for Windows .....	1403
starting Telnet with OLE.....	1496
StartPrinter property.....	764
StartSession method.....	18
statement .....	174
DefType .....	174
status lines .....	154
StatusLine property.....	1517
StatusLineMode .....	251
StatusLineMode (IHEParser) .....	1151
StatusLineMode (IHEProfileDisplay) .....	682
StatusLineMode CfgVT property .....	154
StopDataSave item .....	1482
StopDataSave method .....	1505
StopPrinter property.....	766
String (IOhioField) .....	1351
String (IOhioScreen).....	1323
String group .....	183
StripCtrlZ property .....	1446
StripCtrlZ setting .....	1428
support .....	1364
EHLLAPI .....	1364
WinHLLAPI .....	1364
SwitchScreenType .....	252

SwitchScreenType	
(IHEProfileSessionWindow)	932
syntax	1398, 1473, 1497
execute	1473
request	1473
SysItems item	1476
SYSREQasIACIP	891
System Topic	1405
System Topic commands	1406
System Topic items	1475
SystemColor method (Host)	64
SystemColor method (IHEProfileColor)	549
SystemType function	1430
SystemType method	1453
 <b>T</b>	
TabStop Cfgxxxx method	155
TabStop method	482
TBUserBitmaps property	460
TCPPort	204
TCPPort Cfgxxxx property	156
TDSystem data type	1463
Technical Support	1533
Telnet and DDE	1473
Telnet and OLE	1496
Telnet API	1472
introducing	1472
TelnetEcho	253
TelnetEcho (IHEProfileConnection)	893
TelnetEcho (IHETransport)	1260
TelnetEcho Cfgxxxx property	156
TelnetIsLineMode	1262
TelnetIsLocalEcho	1264
TelnetName (IHEProfileConnection)	895
TelnetName (IHETransport)	1266
TelnetName Cfgxxxx property	157
Terminal	326
Terminal objects	195, 199
methods	195
properties	199
Terminal Style status line	154
Terminal Topic items	1476
TerminalID	253

TerminalID (IHEParser)	1153
TerminalMode property	65
TerminalModel (IHEParser)	1155
TerminalModel (IHETransport)	1268
TerminalOnline property	1270
TerminalType	328
TerminalType property	1272
terminology	1402
TermModel	254
TermStringId property	1508
TestFTP.ebs	1410
Text property (Field)	99
Text property (Host)	66
Text property (IHEParser)	1157
TextRC method	66
TFirewall data type	1464, 1470
TimeDate group	185
Timeout	897
title property	1527
TLocalCase data type	1464
TLowerCase data type	1470
TN3270	1019
keyboard mapping	1019
TN3270 keyboard mapping	1019
TN3270 language conversion table	219
Hex3270.hxl sample	219
TN3270 language-conversion table	216
TN3270 property	67
TN5250	1022
keyboard mapping	1022
TN5250 keyboard mapping	1022
TN5250 language conversion table	223
Hex5250.hxl sample	223
TN5250 language-conversion table	220
TNESession	1274
TNVT NRC language conversion table	227
VT_NRC.hxl sample	227
TNVT NRC language-conversion table	226
TNVT UPSS language conversion table	226
VT_UPSS.hxl sample	226
TNVT UPSS language-conversation table	224
ToggleBlockReceive	1200
ToggleReceive	255

Toolbar property .....	330	Type property .....	88
ToolbarDockType property .....	462	TypeAhead .....	434
ToolbarFilename property .....	464	TypeAhead Cfgxxxx property .....	158
Top property .....	87	TypeAheadTimeout (IHEParser) .....	1161
Topic field .....	1396	TypeAheadTimeout (IHEProfileKeyboard) .....	436
TopicItemList item .....	1478		
topics .....	1474, 1475, 1476, 1483		
Emulation Type.....	1483		
System .....	1475		
Terminal.....	1476		
topics item .....	1475		
TPRINTDestination Cfg3270 property .....	157	UDKLock property .....	1515
TprintMode .....	720	UFLock property .....	1515
TPrintOutput.....	256	unsupported methods and properties .....	168, 208
TraceFilename .....	1276	Update .....	1363
TraceFileName property .....	1448	Update method .....	68
TraceMode property.....	1448	Update Screen After Copy .....	1388
TrackCommands.....	466	UploadHostName .....	607
TrackLabels.....	468	UploadPCFileName .....	609
TrackMenu .....	334	UploadTranslate property .....	611
TrackMenu method .....	68	UponDisconnect .....	899
Transfer.....	257	UPSS property .....	1167
TransferErrorCode property .....	1171	UPSSet CfgVT property .....	158
TransferHosttype.....	257	UseDialProp .....	901
TransferInitialAction .....	258	UseDialProperties CfgVT property .....	159
TransferMode property .....	1173	UseFirewall property .....	1448
TransferRecordformat.....	258	User Dialog group .....	186
transferring files .....	1433	user files .....	1407
between hosts .....	1433	User Input group .....	187
TransferTarget .....	259	User Login method .....	1433
TransferType .....	259	UserDefinedDownload .....	613
TransferType property .....	1444	UserDefinedUpload .....	615
translation tables.....	219, 223, 227	UserDir .....	206
TranslationTable.....	335	UserDirectory .....	337
Transport (IHEParser) .....	1159	UserLogin function .....	1414
Transport (IHETerminal) .....	205	UserLogin method .....	1455
Transport objects.....	1176, 1179, 1203, 1282	Username (IHEProfileConnection) .....	903
data types.....	1282	Username (IHETransport) .....	1278
methods .....	1179	UserName property .....	1441
properties .....	1203	UserPrefCharSetISO property .....	1517
TSO-specific options on upload .....	189	using .....	1472, 1473, 1496
TTransfer data type .....	1463, 1470	DDE .....	1473
		DDE and OLE .....	1472
		OLE with Telnet .....	1496
		using OLE objects .....	1432

## U

UDKLock property .....	1515
UFLock property .....	1515
unsupported methods and properties .....	168, 208
Update .....	1363
Update method .....	68
Update Screen After Copy .....	1388
UploadHostName .....	607
UploadPCFileName .....	609
UploadTranslate property .....	611
UponDisconnect .....	899
UPSS property .....	1167
UPSSet CfgVT property .....	158
UseDialProp .....	901
UseDialProperties CfgVT property .....	159
UseFirewall property .....	1448
User Dialog group .....	186
user files .....	1407
User Input group .....	187
User Login method .....	1433
UserDefinedDownload .....	613
UserDefinedUpload .....	615
UserDir .....	206
UserDirectory .....	337
UserLogin function .....	1414
UserLogin method .....	1455
Username (IHEProfileConnection) .....	903
Username (IHETransport) .....	1278
UserName property .....	1441
UserPrefCharSetISO property .....	1517
using .....	1472, 1473, 1496
DDE .....	1473
DDE and OLE .....	1472
OLE with Telnet .....	1496
using OLE objects .....	1432

**V**

valid colors and attributes ..... 1494  
valid options ..... 1381  
ValidateNumericFieldData ..... 1163  
ValidFATFileCreate setting ..... 1423  
ValidLocalFileNames property ..... 1465  
Value property ..... 89  
values ..... 13  
    retrieving from .ini files ..... 13  
Variable Info group ..... 187  
VariableWidthFont ..... 711  
VB ..... 1383, 1384  
    return codes ..... 1384  
VB interface ..... 1383  
VendorName ..... 1287  
VendorObject ..... 1288  
VendorProductVersion ..... 1288  
Visible BOOL ..... 1502  
Visible property ..... 1440  
Visual Basic ..... 1383, 1384  
    return codes ..... 1384  
Visual Basic interface ..... 1383  
    using ..... 1383  
VT properties ..... 1514  
VT special sequences ..... 193  
VT\_NRC.hxl sample ..... 227  
VT\_UPS.hxl sample ..... 226  
VT220 options ..... 1486  
VT320 options ..... 1486  
VT8BitMode ..... 370  
VTAnswerback ..... 372  
VTAutoClearMonitor property ..... 617  
VTAutoFormFeed property ..... 779  
VTAutoResize property ..... 374  
VTBISIsDel ..... 376  
VTCaptureMode ..... 777  
VTCharset ..... 339  
VTClearScreen  
    OnSizeChange property ..... 684  
VTC concealAnswerback ..... 378  
VTCursorKeyApplMode ..... 438  
VTDefColsPerScreen ..... 380  
VTDefLinesPerScreen ..... 382

VTDisplayMode ..... 384  
VTDoHost  
    WindowSize ..... 905  
VTEnableBreak ..... 439  
VTEnableSSH property ..... 386  
VTForce8Bit ..... 388  
VTForceNRC property ..... 472  
VTHostWritable  
    StatusLine ..... 686  
VTInitiate  
    TelnetNegotiation ..... 907  
VTiPrintMaxCols property ..... 782  
VTiPrintMaxRows property ..... 784  
VTISOCOLORS ..... 688  
VTKeypadApplMode ..... 442  
VTLineMode ..... 909  
VTLocalEcho ..... 390  
VTMaxScrollBufferSize ..... 690  
VTNewLineMode ..... 444  
VTNewTerminalType ..... 392  
VTNRC ..... 475  
VTNRCMode ..... 477  
VTOnLine ..... 395  
VTPassThruUPSS property ..... 786  
VTPrintByPassWindows property ..... 788  
VTPrintDefaultFont property ..... 790  
VTPrintDisableTranslation property ..... 792  
VTPrinterDeinit property ..... 794  
VTPrinterEnableTimeout property ..... 796  
VTPrinterFontName property ..... 798  
VTPrinterInit property ..... 800  
VTPrinterTimeout ..... 802  
VTPrinterTimeoutValue property ..... 804  
VTPrintFile property ..... 806  
VTPrint FileMode property ..... 808  
VTPrintFitFontToPage property ..... 810  
VTPrintLFtoCRLF property ..... 812  
VTPrintTarget property ..... 814  
VTResetISOCOLORS property ..... 692  
VTSaveAttribs  
    InScrollback ..... 694  
VTSaveEraseScreens ..... 696  
VTSscrollNoBlanks ..... 698

VTScrollSpeed .....	397
VTSmoothScroll.....	399
VTTerminalID.....	401
VTUPSS .....	479
VTUseSpecificPrinter.....	816
VTWrapLine.....	404

**W**

wait.....	1382
Wait command.....	1385
WaitConnected method (Host).....	68
WaitConnected method (IHEParser) .....	1039
WaitForCursor method.....	1041
WaitForCursorMove method .....	1043
WaitForInput.....	1315
WaitForIO method (Host).....	69
WaitForIO method (IHEParser).....	1045
WaitForString.....	1316
WaitForString method (Host) .....	70
WaitForString method (IHEParser) .....	1047
WaitForStringRC method.....	71
WaitHostQuiet method .....	1050
WaitIdle .....	1318
WaitIdle method (Host).....	72
WaitIdle method (IHEParser).....	1052
WaitPSUpdated method (Host) .....	73
WaitPSUpdated method (IHEParser) .....	1054
WaitXfer method (Host).....	74
WaitXfer method (IHEParser) .....	1056
WarningBell property.....	1508
WinDDEEnabled.....	342
Window Close (201) .....	1366
Windows Style status line.....	154
WindowState property .....	934
WindowTitle .....	936
WindowTitle Cfgxxxx property .....	160
WinHLLAPI .....	1364
WinHLLAPI development files.....	1392
WinHLLAPI module .....	1392
Word method .....	20
WordWrap .....	545
WordWrap Cfgxxxx property .....	160

**WordWrapWith**

NewLineReturn property.....	547
working with DDE .....	1393
WorkingHostDirectory method .....	1453
WorkSpaceBackgroundBitmap .....	938
WorkspaceBackgroundColor .....	940
WorkspaceForegroundColor.....	942
WriteCtrlZ property.....	1446
WriteCtrlZ setting.....	1429
WriteProfileString method .....	20
WriteProtectedText method .....	1058
WYSE properties.....	1518
WYSE50 options .....	1486
WYSE60 options .....	1486
WyseTerm object .....	1498
WyseTerm object items .....	1499

**X**

XferBlockSize .....	619
XferBlockSize Cfg3270 property.....	161
XferCount property .....	75
XferDest .....	621
XferHostCodePage .....	623
XferHostSystem Cfg3270 property.....	161
XferPCCodePage .....	626
XferProgramName Cfg3270 property .....	162
XferRC property.....	75
XferSource .....	628
XferStartAction .....	630
XferStartAction Cfg3270 property .....	162
XmAckTimeout .....	634
XmCRC .....	636
XModem16BitCrc CfgVT property.....	163
XModemPkt1024 CfgVT property .....	163
XModemSendTimeout CfgVT property....	164

**Y**

Yield Wait.....	1386
YmAckTimeout .....	638
YModemSendTimeout CfgVT property ....	164
YModemUseFullPath CfgVT property .....	165
YmUseFullPath .....	640

**Z**

ZmAutoDownload .....	642	ZModemSlidingBytes CfgVT property .....	167
ZmCrashRecovery .....	644	ZModemSlidingWin CfgVT property.....	167
ZmMaxErrors .....	646	ZModemUseFullPath CfgVT property .....	168
ZModemAutoDownload CfgVT property .	165	ZmOverwriteMngmt .....	648
ZModemCrashRecovery CfgVT property ..	165	ZmSlideWindow .....	650
ZModemMaxErr CfgVT property .....	166	ZmUseFullPath .....	652
ZModemOverwrite CfgVT property.....	166	ZnWindowSize .....	654