GlobalSCAPE® Server COM API version 5.2.6

User Guide

globalSCAPE®

GlobalSCAPE, Inc. (GSB)

	Corporate Headquarters	Andover Office
Address:	4500 Lockhill-Selma Road, Suite 150 San Antonio, TX (USA) 78249	200 Brickstone Square, Suite 104 Andover, MA (USA) 01810
Sales:	(210) 308-8267	(978) 474-9116
Sales (Toll Free):	(800) 290-5054	(800) 474-0116
Technical Support:	(210) 366-3993	(978) 474-9116

Web Support: http://www.globalscape.com/support/

September 16, 2008

Table of Contents

Using the COM Application Programming Interface

You can interact with GlobalSCAPE's server products directly from your own custom applications using any COM-enabled programming language such as Visual Basic (VB), Java, or C++, with the integrated development environment (IDE) of your choice. To create a new script file or program, you need to have some familiarity with programming concepts and, ideally, some experience with a COM-enabled programming language.

This guide is intended as a supplement to the EFT Server/Secure FTP Server user guides. For an understanding of the concepts in this supplement, please refer to those guides.

The SFTPCOMInterface.CIServer Class

The **SFTPCOMInterface.CIServer** class is the class that an application can use directly. To start using the SFTP COM interface, a user application should create the **SFTPCOMInterface.CIServer** class object, and then apply the methods and properties in the object.

For example: `Set SFTPServer = CreateObject("SFTPCOMInterface.CIServer")`

CIServer implements the ICIServer interface.

Available Interface Methods and Properties

- Server Interface Methods and Server Interface Properties- Use the **ICIServer** interface to access and manage the server and its attributes.
- Multiple Sites Interface Methods - Use the **ICISites** interface to list all Sites, to add Sites, and to find individual Sites on the server. It can be obtained by using the **Sites** method of the ICIServer interface.
- Single-Site Interface Methods and Single-Site Interface Properties - Use the ICISite interface to manage individual Sites on the server. It can be obtained by using the Item and **SiteByID** methods of the ICISites interface.

  For example: `Set Site = Sites.Item(0)`
- Permissions Interface Properties - Use the IPermission interface to set and manage folder permissions. Use the **ICISite::GetFolderPermissions** method to get folder permissions.
- Client Settings Interface Methods and Client Setting Interface Properties - Use the ICIClientSettings interface to manage client user or client group settings. Use the **ICISite::GetGroupSettings** method to access the interface.
- Command Settings Interface Methods and Command Settings Interface Properties - Use the ICICommandSettings interface to create or manage Site custom commands. Use the **ICISite::GetCommandSettings** method to access the interface.
- Certificate Interface Properties - Use the ICICertInfo interface to obtain information about certificates. Use the **ICISite::GetTrustedCertificateInfo** method or the **ICISite::GetPendingCertificateInfo** method to access the interface.
- Mail Action Parameters Interface - (EFT Server 5.2 and later) Use the ICIMailActionParams interface to make changes to the **MailAction** settings, such as email subject, email body, and to addresses, CC addresses, and BCC addresses.
- Cleanup Action Parameters Interface - (EFT Server 5.2 and later) Use the ICICleanupActionParams interface to make changes to the Cleanup in Folder Event Action settings.
- OpenPGP Action Interface (ICIPgpActionParams) - (EFT Server 5.2 and later) Use the ICIPgpActionParams interface to make changes to the OpenPGP Event Action settings.

- Event Rule Interfaces - (EFT Server 5.2 and later) Use the Event Rule interfaces to manage Event Rules and their Events, Actions, and Conditions.

For script examples, see Sample Script and Setting Permissions.

List of Methods, Properties, Enumerators, and Constants

The tables below list each of the methods, properties, enumerators, and constants used in Secure FTP Server/EFT Server. The **Applicable Version** column indicates in which product and version number the method/property/enumeration/constant is available. In most cases, the method, property, enumeration, or constant applies to both products, all versions.

Server Interface Properties

Property	How It's Used	Applicable Version
AdminPort	Editing the Administrator Port	All versions
AllowClearCommandChannel	Allowing a Clear Command Channel on the Server	EFT Server 5.0.1 and later
AllowRemoteAdministration	Allowing Remote Administration	All versions
AllowUnprotectedDataChannel	Allowing an Unprotected Data Channel on the Server	EFT Server 5.0.1 and later
ARMDatabaseName	Editing the ARM Database Name	Secure FTP Server 3.3
ARMPassword	Editing the ARM Password	Secure FTP Server 3.3
ARMServerName	Editing the ARM Server Name	Secure FTP Server 3.3
ARMUserName	Editing the ARM Username	Secure FTP Server 3.3
CertificateFilePath	Editing the File Path for Certificates	All versions
CipherList	Editing the Cipher List	EFT Server 5.0.1 and later
ConnectedUsersNumber	Retrieving the Number of Connected Users	EFT Server 3.5.1 and later
EnableARM	Enabling Auditing and Reporting	All versions
IPAccessAllowedDefault	Finding or Setting the Default IP Access Mode	All versions
KeyFilePath	Editing the File Path for Private Key Files	All versions
ListenIP	Editing the Server Administrator IP Address	All versions
LogPath	Editing the Server Log Path	All versions
LogRotation	Rotating Logs	All versions
LogType	Editing the Server Log Type	All versions
PassPhrase	Editing Private Key Passphrases	All versions
SMTPLogin	Editing the Email User Name	All versions
SMTPPassword	Editing the Email Password	All versions
SMTPPort	Editing the Mail Server Port	All versions
SMTPRecipientAddr	Choosing Email Receiver Addresses	All versions
SMTPRecipientName	Choosing Email Receiver Names	All versions
SMTPSenderAddr	Choosing the From Address in Server Emails	All versions
SMTPSenderName	Choosing the Server Email Sender Name	All versions
SMTPServer	Editing the Mail Server Address	All versions
SSLVersionMask	Retrieving or Setting the SSL Version Mask	EFT Server 5.0.1
UseSSLForAdministration	Requiring SSL for Remote Administration	All versions

Server Interface Methods

Method	How It's Used	Applicable Version
AddIPAccessRule	Adding an IP Mask	All versions
ApplyChanges	Applying Changes to the Server	EFT Server 5.0.1+
AreLingeringTransactions	Determining if Lingering Transactions Exist	EFT Server 5.2 and later
ChangeAdminPassword	Changing the Administrator Password	EFT Server 4.3.4
Close	Closing the Administrator Connection to the Server	All versions
Connect	Connecting to the Server as an Administrator	All versions
GetAdminIPs	Retrieving a List of Administrator IP Addresses	EFT Server 4.3.4
GetAllowedMasks	Retrieving Allowed IP Address Masks	All versions
GetDeniedMasks	Retrieving Denied IP Address Masks	All versions
GetLocalIP	Retrieving the Local IP Address	All versions
GetLocalTime	Retrieving the Local Server Time	All versions
GetServerAdminCount	Retrieving the Number of Server Administrator Accounts on the Server	EFT Server 4.3.4
IsDBConnected	Determining if Server is Connected to the ARM Database	EFT Server 5.2 and later
RefreshSettings	Refreshing Server Settings	All versions
RemoveIPAccessRule	Removing an IP Address Mask	All versions
RemoveServerAdminAccount	Removing a Server Administrator Account	EFT Server 4.3.4
Sites	Retrieving a List of Server Sites	All versions

Multiple Sites Interface Methods

Method	How It's Used	Applicable Version
Add	Adding a Site	All versions
AddADSite	Adding an Active Directory Authenticated Site	All versions
AddLDAPSite	Adding an LDAP Site	EFT Server 4.3.4
AddLocalSite	Adding a Local Site	All versions
AddNTLMSite	Adding an NT-Authenticated Site	All versions
AddODBCSite	Adding an ODBC-Authenticated Site	All versions
Count	Retrieving the Number of Sites	All versions
Item	Retrieving a Site's COM Interface by Site Number	All versions
SiteByID	Retrieving a Site's COM Interface by Site ID	All versions

Single-Site Interface Properties

Property	How It's Used	Applicable Version
AssignFullPermissionsForHomeFolder	Setting the Grant Full Permissions Option	All versions
AutoBanIPsPermanently	Banning IP Addresses	All versions
AutoCreateHomeFolder	Setting the Auto Create Home Folder Option	All versions
BanIPOnDOS	Banning Connections from Specific IP Addresses	All versions
BlockAntiTimeOut	Blocking Anti-Timeout Schemes	All versions
BlockSiteToSite	Blocking Site-to-Site FTP	All versions
ConnectMessage	Editing the Server Connection Message	All versions
DisconnectOnDOS	Limiting Consecutive Invalid Commands	All versions
DOSLimit	Managing the Consecutive Invalid Commands Limit	All versions
EnablePortRange	Enabling a Range of Ports for PASV Connections	All versions
ExitMessage	Managing a Site's Exit Message	All versions
HasMaxConcurrentLogins	Retrieving Maximum Concurrent Logins	Secure FTP Server
HasMaxConnectionsPerAccount	Retrieving or Setting Number of Connections per User to a Site	All versions
HasMaxIPPerAccount	Limiting Connections from the Same IP Address	All versions
HasMaxSpeed	Enabling or Disabling a Site's Maximum Transfer Speed	All versions
HasMaxUsers	Limiting Concurrent Connections to a Site	All versions
ID	Retrieving a Site's ID	All versions
InvalidAttemptsPeriod	Viewing Invalid Attempts Period for a Site	EFT Server 5.1.1 and later
IPAccessAllowedDefault	Setting IP Addresses that have Access to a Site	All versions
IsStarted	Determining if a Site is Started	All versions
LimitLoginAttempts	Enabling Allowed Login Attempt Limit	EFT Server 5.1.1 and later
LockoutNotDisable	Enabling Account Lockout	EFT Server 5.1.1 and later
LockoutPeriod	Viewing the Lockout Period for a Site	EFT Server 5.1.1 and later
MaxConcurrentConnections	Retrieving or Setting Maximum Concurrent Connections	All versions
MaxConnectionsFromSameIP	Managing Concurrent Connections Allowed from the Same IP Address	All versions
MaxConcurrentLogins	Setting Maximum Concurrent Logins	Secure Server, All versions
MaxConnectionsPerUser	Retrieving or Setting a Site's User Connection Limit	All versions
MaxInvalidLoginAttempts	Viewing the Number of Invalid Login Attempts Allowed for a Site	EFT Server 5.1.1 and later

Property	How It's Used	Applicable Version
MaxTransferSpeed	Retrieving or Setting Maximum Transfer Speed	All versions
Name	Retrieving a Site's Name	All versions
ODBCSettings	Retrieving ARM ODBC Settings	EFT Server 4.3.4
PASVListenIP	Determining the IP Address for PASV Connections	All versions
PASVPortMax	Determining the High End of a PASV Mode Port Range	All versions
PASVPortMin	Determining the Low End of a PASV Mode Port Range	All versions
UserLimitMessage	Managing the User Limit Message	All versions
VFSFilter	Retrieving and Modifying a List of Banned File Types	All versions

Single-Site Interface Methods

Method	How It's Used	Applicable Version
AddCerficateToTrusted	Adding a Pending Certificate to the Trusted List	All versions
AddIPAccessRule	Adding an IP Mask to a Site	All versions
AddUserToPermissionGroup	Adding a User to a Permission Group	All versions
AssignEventRuleCustomCommandWorkingFolder	Assigning the Event Rule Custom Command Working Folder	EFT Server 4.3.4
CancelTransfer	Cancelling an HTTPS Transfer	EFT Server 4.3.4
ChangeUserPassword	Changing a User's Password	All versions
CreateCommand	Creating a New Custom Command	All versions
CreateComplexPassword	Creating a Complex Password for a User	EFT Server 5.0.1 and later
CreatePermissionGroup	Creating a Permissions Group	All versions
CreatePhysicalFolder	Creating a Physical Folder	All versions
CreateSettingsLevel	Creating a User Settings Level	All versions
CreateUser and CreateUserEx	Creating a User	All versions
CreateVirtualFolder	Creating a Virtual Folder	All versions
EventRules	Retrieving a List of Event Rules	EFT Server 5.2 and later
ExportTrustedCertificate	Retrieving a Certificate's Data String	All versions
ForceSynchronizeUserDatabase	Synchronizing the User Database	All versions
GetAllowedMasks	Retrieving a List of Allowed IP Masks	All versions
GetAuthManagerID	Identifying the Authentication Manager for a Site	All versions

Method	How It's Used	Applicable Version
GetAwaitingCertsPath	Retrieving the Path to Pending Certificates	All versions
GetBlankPermission	Creating Blank Permissions	All versions
GetCertFilePath	Retrieving the File Path for Certificates	All versions
GetCheckClientCert	Verifying if Client Certificates are Required	All versions
GetCommands	Retrieving a List of Custom Commands	All versions
GetCommandSettings	Retrieving a Custom Command's Settings	All versions
GetConnectedCount	Retrieving the Number of Connected Users	All versions
GetConnectedUsers	Retrieving a List of Connected Users	EFT Server 4.3.4
GetDeniedMasks	Retrieving a List of Denied IP Address Masks	All versions
GetDownloadCount	Retrieving the Number of Active Downloads	All versions
GetDownloadSpeed	Retrieving a Site's Download Speed	All versions
GetFolderList	Retrieving a List of Folders	All versions
GetFolderPermissions	Retrieving a List of Folder Permissions	All versions
GetFTPAccess	Verifying if FTP Access is Enabled for a Site	All versions
GetHTTPAccess	Verifying if HTTP is Enabled for a Site	EFT Server 4.3.4
GetHTTPPort	Retrieving the HTTP Port for a Site	EFT Server 4.3.4
GetHTTPSAccess	Verifying if HTTPS is Enabled for a Site	EFT Server 4.3.4
GetHTTPSPort	Retrieving the HTTPS Port for a Site	EFT Server 4.3.4
GetIP	Retrieving a Site's IP Address	All versions
GetKeyFilePath	Retrieving the File Path for Private Keys	All versions
GetPassPhrase	Retrieving Private Key Passphrases	All versions
GetPendingCertificateInfo	Retrieving Information for a Pending Certificate	All versions
GetPendingCertificates	Retrieving a List of Pending Certificates	All versions
GetPermissionGroupList	Retrieving a List of Users of Specified Permission Groups	All versions
GetPermissionGroups	Retrieving a List of Permission Groups on a Site	All versions
GetPermissionGroupsOfUser	Retrieving a List of a User's Permission Groups	All versions
GetPhysicalPath	Retrieving the Physical Path to a Virtual Folder	EFT Server 5.2.5
GetPort	Retrieving a Site's Port Number	All versions

Method	How It's Used	Applicable Version
GetRootFolder	Retrieving a Site's Root Folder	All versions
GetSettingsLevels	Retrieving a List of User Settings Levels	All versions
GetSettingsLevelSettings	Retrieving a List of a Settings in a User Settings Level	All versions
GetSettingsLevelUsers	Retrieving a List of Users in a User Settings Level	All versions
GetSSHKeyFilePath	Retrieving the SFTP (SSH) Certificate File Path	EFT Server 4.3.4
GetSSLAuth	Verifying if Explicit SSL is Enabled for a Site	All versions
GetSSLImp	Verifying if Implicit SSL is Enabled for a Site	All versions
GetStartTime	Retrieving the Site Start Time	All versions
GetTrustedCertificateInfo	Retrieving Information for a Trusted Certificate	All versions
GetTrustedCertificates	Retrieving a List of Trusted Certificates	All versions
GetTrustedCertsPath	Retrieving the Path to Trusted Certificates	All versions
GetUploadCount	Retrieving the Number of Active Uploads	All versions
GetUploadSpeed	Retrieving a Site's Upload Speed	All versions
GetUsers	Retrieving a List of Users	All versions
GetUserSettings	Retrieving a User's Settings	All versions
ImportTrustedCertificate	Saving a Trusted Certificate to a File	All versions
KickUser	Forcing a User to Log Off of the Site	EFT Server 4.3.4
MoveUserToSettingsLevel	Moving a User to a User Settings Level	All versions
RemapVirtualFolder	Remapping a Virtual Folder Path	EFT Server 4.3.4
Remove	Deleting a Site	All versions
RemoveCommand	Deleting a Custom Command	All versions
RemoveFolder	Deleting a Folder	All versions
RemoveIPAccessRule	Removing an IP Address Mask from a Site	All versions
RemovedPendingCertificate	Deleting a Pending Certificate	All versions
RemovePermission	Removing Folder Permissions	All versions
RemovePermissionGroup	Deleting a Permission Group	All versions
RemoveSettingsLevel	Deleting a User Settings Level	All versions
RemoveTrustedCertificate	Deleting a Trusted Certificate	All versions
RemoveUser	Deleting a User	All versions

Method	How It's Used	Applicable Version
RemoveUserFromPermissionGroup	Deleting a User from a Permission Group	All versions
RenameFolder	Renaming a Folder	All versions
RenamePermissionGroup	Renaming a Permission Group	All versions
RenameSettingsLevel	Renaming a Settings Level	All versions
RenameUser	Renaming a User	All versions
SetCertFilePath	Changing the Certificate File Path	All versions
SetCheckClientCert	Requiring User Certificates on Implicit SSL Sites	All versions
SetFTPAccess	Enabling FTP Access to a Site	All versions
SetHTTPAccess	Enabling HTTP Access	EFT Server 4.3.4
SetHTTPPort	Specifying the HTTP Port	EFT Server 4.3.4
SetHTTPSAccess	Enabling HTTPS Access	EFT Server 4.3.4
SetHTTPSPort	Specifying the HTTPS Port	EFT Server 4.3.4
SetIP	Specifying a Site's IP Address	All versions
SetKeyFilePath	Changing the Private Key File Path	All versions
SetPassPhrase	Changing a Private Key Passphrase	All versions
SetPermission	Setting Folder Permissions	All versions
SetPort	Specifying a Site's Port Number	All versions
SetRootFolder	Specifying the Site's Root Folder	All versions
SetSSHKeyFilePath	Specifying the SFTP (SSH) Certificate File Path	EFT Server 4.3.4
SetSSLAuth	Allowing Explicit SSL Access to a Site	All versions
SetSSLImp	Allowing Implicit SSL Access to a Site	All versions
Start	Starting a Site	All versions
Stop	Stopping Site	All versions
ValidatePassword	Validating a User's Password	EFT Server 3.5.1 and later

Permissions Interface Properties

Property	How It's Used	Applicable Version
Client	Retrieving a Client or Permission Group Name	All versions
DirCreate	Granting Folder Creation Permission	All versions
DirDelete	Granting Folder Deletion Permission	All versions
DirList	Granting Permission to View a List of Folder Contents	All versions
DirShowHidden	Showing Hidden Files	All versions
DirShowInList	Showing or Hiding Folders	All versions

Property	How It's Used	Applicable Version
DirShowReadOnly	Showing Read-Only Files	All versions
FileAppend	Granting Append File Permission	All versions
FileDelete	Granting Delete Permission	All versions
FileDownload	Granting Download Permission	All versions
FileRename	Granting Rename Permission	All versions
FileUpload	Granting Upload Permission	All versions
Folder	Retrieving a Folder Name	All versions

Client Settings Interface Properties

Property	How It's Used	Applicable Version
Comment	Viewing User Properties - Comments	All versions
Custom	Viewing User Properties - Custom1, Custom2, Custom3	EFT Server v5.0.1 and later
Email	Viewing User Properties - Email	All versions
Fax	Viewing User Properties - Fax	All versions
FullName	Viewing User Properties - Full Name	All versions
InvalidAttemptsPeriod	Viewing Invalid Attempts Period for a User	EFT Server 5.1.1 and later
IPAccessAllowedDefault	IP addresses with which a user is allowed to connect to the Server	EFT Server 5.1.1 and later
LockoutPeriod	Viewing a User's Lockout Period	EFT Server 5.1.1 and later
MaxInvalidLoginAttempts	Viewing the Number of Invalid Login Attempts Allowed for a Client	EFT Server 5.1.1 and later
Pager	Viewing User Properties - Pager	All versions
Phone	Viewing User Properties - Phone	All versions

Client Settings Interface Methods

Method	How It's Used	Applicable Version
AddIPAccessRule	Adding an IP Access IP Mask for a Client	EFT Server 5.1.1 and later
GetAllowAnyPwd	Determining if Users Can Create Any Password	All versions
GetAllowedMasks	Retrieving Allowed IP Masks for a User	EFT Server 5.1.1 and later
GetAllowMODEZ	Determining if ModeZ is Allowed	All versions
GetAllowNoop	Determining if the NOOP Command is Allowed	All versions
GetAllowXCRC	Determining if the XCRC Command is Allowed	All versions

Method	How It's Used	Applicable Version
GetAnonymousLogin	Viewing Anonymous Logins	All versions
GetAppletEnabled	Determining Web Transfer Client Access	EFT Server 3.5.1 and later
GetChangePwd	Determining if Users are Allowed to Change their Passwords	All versions
GetClearFTP	Determining if Plain FTP Access is Allowed	All versions
GetClearHTTP	Determining if Plain HTTP Access is Allowed	EFT Server 4.3.4 and EFT Server 5.1
GetDeniedMasks	Retrieving Denied IP Mask for a User	EFT Server 5.1.1 and later
GetDescription	Retrieving a User's Description	All versions
GetDownloadsPerSession	Retrieving a Download per Session Limit	All versions
GetEnableAccount	Determining if a User Account or User Settings Level is Enabled	All versions
GetEnableDiskQuota	Determining if an Account has a Disk Quota	All versions
GetEnableTimeOut	Determining if a User Can be Timed Out	All versions
GetExpirationDate	Determining the Expiration Date for a User Account	All versions
GetExpirationDateAsVariant	Determining the Expiration Date for a User Account	EFT Server 4.3.4 and later
GetHasDownloadsPerSession	Determining if a User has a Download per Session Limit	All versions
GetHasMaxDownloadSize	Determining if a User has a Download Size Limit	All versions
GetHasMaxIPs	Determining for an IP Connection Limit	All versions
GetHasMaxSpeed	Determining if a User has a Transfer Speed Limit	All versions
GetHasMaxUploadSize	Determining if a User has an Upload Size Limit	All versions
GetHasMaxUsers	Determining if the Number of Concurrent Connections is Limited for Users	All versions
GetHasPwdRetries	Determining Failed Password Limit	All versions
GetHasUploadsPerSession	Determining if a User has an Upload per Session Limit	All versions
GetHomeDir	Determining if Users Can Have a Home Folder	All versions
GetHomeDirIsRoot	Determining if the Account Home Folder is the Default Root Folder	All versions
GetHomeDirString	Retrieving Users' Home Folders	All versions
GetHomeIP	Determining if Users are Restricted to a Specific IP Address	All versions

Method	How It's Used	Applicable Version
GetHomeIPString	Retrieving Users' Home IP Address	All versions
GetIncorrectPasswordAttempts	Determining the Number of Failed Password Attempts	All versions
GetLimitLoginAttempts	Retrieving Number of Login Attempts Allowed	EFT Server 5.1.1 and later
GetLockoutNotDisable	Viewing Whether Account Lockout is Enabled for a User	EFT Server 5.1.1 and later
GetLoginMsg	Determining How Users' Login Message is Defined	All versions
GetLoginMsgString	Retrieving the Login Message	All versions
GetMaxDownloadSize	Retrieving a User's Download Size Limit	All versions
GetMaxIPs	Determining Number of Connections Allowed from the Same IP Address	All versions
GetMaxSpace	Determining the Disk Quota Size	All versions
GetMaxSpeed	Determining the Maximum Allowed Transfer Speed	All versions
GetMaxUploadSize	Retrieving a User's Upload Size Limit	All versions
GetMaxUsers	Determining the Maximum Concurrent Connections Allowed per User	All versions
GetPwdRetries	Retrieving the Failed Password Limit	All versions
GetSFTP	Viewing if SFTP Access is Enabled for a Client	All versions
GetSFTPAuthenticationType	Identifying the SFTP Authentication Type	All versions
GetSSHKeyID	Retrieving the SFTP (SSH) Certificate ID	All versions
GetSSL	Determining if SSL Access is Allowed	All versions
GetSSLAuthenticationType	Identifying the SSL Authentication Type	EFT Server 5.1 and later
GetSSLKeyID	Identifying the SSL Key ID	EFT Server 5.1 and later
GetTimeOut	Determining How Long a Connection can be Inactive	All versions
GetUploadsPerSession	Retrieving a User's Upload per Session Limit	All versions
GetUsedSpace	Determining How Much Disk Space a User has Used	All versions
RemoveIPAccessRule	Removing an IP Access Mask for User	EFT Server 5.1.1 and later
SetAllowAnyPwd	Allowing Users to Create Any Password	All versions
SetAllowMODEZ	Allowing MODE Z Compression	All versions
SetAllowNoop	Allowing the NOOP Command	All versions

Method	How It's Used	Applicable Version
SetAllowXCRC	Allowing the XCRC Command	All versions
SetAnonymousLogin	Allowing or Prohibiting Anonymous Logins	All versions
SetAppletEnabled	Specifying Web Transfer Client Access	EFT Server 3.5.1 and later
SetChangePwd	Allowing Users to Change their Passwords	All versions
SetClearFTP	Allowing Users Plain FTP Connections	All versions
SetClearHTTP	Allowing Users to Connect Using Clear HTTP	EFT Server 4.3.4 and EFT Server 5.1
SetDescription	Specifying a User Description	All versions
SetDownloadsPerSession	Specifying the Maximum Number of Downloads a User is Permitted per Session	All versions
SetEnableAccount	Enabling a User Account or User Settings Level	All versions
SetEnableDiskQuota	Limiting a User's Disk Space	All versions
SetEnableTimeOut	Enabling Connection Timeout	All versions
SetExpirationDate	Setting the Expiration Date for a User Account	All versions
SetHasDownloadsPerSession	Enabling a User's Downloads-per-Session Limit	All versions
SetHasMaxDownloadSize	Enabling a User's Download Size Limit	All versions
SetHasMaxIPs	Enabling an IP Address Connection Limit	All versions
SetHasMaxSpeed	Enabling the Transfer Speed Limit	All versions
SetHasMaxUploadSize	Enabling a User's Upload Size Limit	All versions
SetHasMaxUsers	Enabling a User's Connection Limit	All versions
SetHasPwdRetries	Limiting Failed Password Attempts	All versions
SetHasUploadsPerSession	Enabling a User's Uploads-per-Session Limit	All versions
SetHomeDir	Allowing Users to Have a Home Folder	All versions
SetHomeDirIsRoot	Setting the Default Root Folder	All versions
SetHomeDirString	Specifying the Path to Users' Home Folders	All versions
SetHomeIP	Restricting Users to a Specific IP Address	All versions
SetHomeIPString	Setting Users' Home IP Address	All versions
SetIncorrectPasswordAttempts	Specifying the Number of Incorrect Password Attempts	All versions
SetLimitLoginAttempts	Specifying Number of Login Attempts Allowed	EFT Server 5.1.1 and later

Method	How It's Used	Applicable Version
SetLockoutNotDisable	Enabling Account Lockout for a User	EFT Server 5.1.1 and later
SetLoginMsg	Specifying the Login Message Used	All versions
SetLoginMsgString	Creating a Login Message	All versions
SetMaxDownloadSize	Specifying the Maximum File Size a User is Permitted to Download	All versions
SetMaxIPs	Specifying the Maximum Connections for IP Addresses	All versions
SetMaxSpace	Specifying a User's Disk Quota	All versions
SetMaxSpeed	Specifying the Maximum Allowed Transfer Speed	All versions
SetMaxUploadSize	Specifying the Maximum File Size a User is Permitted to Upload	All versions
SetMaxUsers	Specifying the Maximum Connections Allowed per User	All versions
SetPwdRetries	Specifying the Failed Password Limit	All versions
SetSFTP	Allowing SFTP Access for a Client	All versions
SetSftpAuthenticationType	Specifying the SFTP Authentication Type for the Client	All versions
SetSSHKeyID	Specifying the SFTP (SSH) Certificate ID	All versions
SetSSL	Allowing Users SSL Connections	All versions
SetSSLAuthenticationType	Specifying the SSL Authentication Type	EFT Server 5.1
SetSSLKeyID	Specifying the SSL Key ID	EFT Server 5.1
SetTimeOut	Specifying the Timeout Value	All versions
SetUploadsPerSession	Specifying the Maximum Number of Uploads a User is Permitted per Session	All versions

Command Settings Interface Properties

Property	How It's Used	Applicable Version
Description	Retrieving or Changing the Description of a Custom Command	All versions
EnableProcessTimeOut	Enabling a Time Limit for a Custom Command	All versions
Executable	Retrieving or Changing the Path to the Executable of a Custom Command	All versions
IsEnabled	Enabling a Custom Command	All versions
MinNumOfParams	Requiring a Minimum Number of Parameters for Custom Commands	All versions
MinNumOfParamsMsg	Defining or Changing a Message for an Invalid Number of	All versions

Property	How It's Used	Applicable Version
	Command Parameters	
Name	Retrieving or Changing the Name of a Custom Command	All versions
Parameters	Retrieving or Changing Custom Command Parameters	All versions
ProcessTimeOut	Specifying the Time Limit for a Custom Command	All versions
RedirectOutputToClient	Redirecting Command Output to Clients	All versions
RedirectOutputToLog	Redirecting Command Output to a Log	All versions
RequireParams	Viewing or Requiring Parameters for Custom Commands	All versions

Command Settings Interface Methods

Method	How It's Used	Applicable Version
AddUserPermission	Listing Users Allowed to Use a Command	All versions
GetUserPermissions	Retrieving a List of Users Allowed to Use a Custom Command	All versions
RemoveUserPermission	Prohibiting Users from Using a Custom Command	All versions

Command Action Parameters Interface Properties

Property	How it's used	Applicable Version
Command	Retrieving or Changing Command to Execute	EFT Server 5.2 and later
Parameters	Retrieving or Changing Parameters for Command	EFT Server 5.2 and later
WorkingFolder	Retrieving or Changing Working Folder for Command	EFT Server 5.2 and later

Certificate Information Interface Properties

Property	How It's Used	Applicable Version
Description	Retrieving a Certificate Description	All versions
ID	Retrieving a Certificate ID	All versions
IssuerCName	Retrieving a Certificate Issuer's Common Name	All versions
IssuerCountry	Retrieving a Certificate Issuer's Country	All versions
IssuerOneLine	Retrieving a Certificate Issuers Information	All versions
IssuerOrg	Retrieving a Certificate Issuer's Organization	All versions
IssuerUnit	Retrieving a Certificate Issuer's Unit	All versions
NotAfter	Retrieving a Certificate's Expiration Date	All versions
NotBefore	Retrieving a Certificate's Start Date	All versions
SubjectCName	Retrieving a Certificate Subject's Common Name	All versions
SubjectCountry	Retrieving Certificate Subject's Country	All versions
SubjectOneLine	Retrieving a Certificate Subject's Information	All versions
SubjectOrg	Retrieving a Certificate Subject's Organization	All versions

Property	How It's Used	Applicable Version
SubjectUnit	Retrieving a Certificate Subject's Unit	All versions

Mail Action Parameters Interface Properties

Property	How it's used	Applicable Version
TOAddresses	Retrieving or Changing Message Recipients	EFT Server 5.2 and later
CCAddresses		EFT Server 5.2 and later
BCCAddresses		EFT Server 5.2 and later
CopyToClient	Determining whether to CC Message to Client Associated with Event	EFT Server 5.2 and later
Subject	Retrieving or Changing Message Subject	EFT Server 5.2 and later
Body	Retrieving or Changing Message Body	EFT Server 5.2 and later

Cleanup Action Parameters Interface Properties

Property	How it's used	Applicable Version
DaysToKeepFiles	Retrieving or Changing the Period to Keep Files before Removing	EFT Server 5.2 and later
ExcludeFileMask	Specifying whether to Remove or Exclude Files from Cleanup	EFT Server 5.2 and later
FileMask	Retrieving or Changing Files to Remove	EFT Server 5.2 and later
Folder	Retrieving or Changing Folder to Cleanup	EFT Server 5.2 and later
Recursive	Specifying whether to Cleanup All Subfolders Recursively	EFT Server 5.2 and later

OpenPGP Action Interface Properties

Property	How it's used	Applicable Version
FilePath	Retrieving or Changing File Path for OpenPGP Event Action	EFT Server 5.2 and later
KeyIDs	Retrieving or Changing Keys to Encrypt/Decrypt Data	EFT Server 5.2 and later
Operation	Determining Operation for PGP Event Action	EFT Server 5.2 and later
PassPhrase	Retrieving or Changing the Passphrase for Signing or Decryption Key	EFT Server 5.2 and later
Sign	Determining whether to Sign Encrypted Data	EFT Server 5.2 and later
SignKeyID	Retrieving or Changing Key to Sign Encrypted Data	EFT Server 5.2 and later

Event Rule Interfaces

ActionStatement Interface Methods and Properties

Property	How it's used	Applicable Version
type	Action Statement Interface	EFT Server 5.2 and later
Action		
FailSection		

ActionStatements Interface Methods

Method	How it's used	Applicable Version
Add	Action Statements Interface	EFT Server 5.2 and later
Count		
Delete		
Item		

CompoundCondition Interface Methods and Properties

Property	How it's used	Applicable Version
Operator	Compound Condition Interface	EFT Server 5.2 and later
Add		
Count		
Delete		
Item		

Download Action Parameters Interface (ICIDownloadActionParams)

Property	How it's used	Applicable Version
AutoLogin		EFT Server 5.25
DeleteSourceFile		
Host		
LocalPath		
Password		
Port		
PrivateKeyPassword		
PrivateKeyPath		
Protocol		
PublicKeyPath		
RemotePath		
User		

Upload Action Parameters Interface (ICIUploadActionParams)

Property	How it's used	Applicable Version
AutoLogin		EFT Server 5.25
DeleteSourceFile		
Host		
LocalPath		
Password		
Port		
PrivateKeyPassword		
PrivateKeyPath		
Protocol		
PublicKeyPath		
RemotePath		
User		

EventAction Interface Properties

Property	How it's used	Applicable Version
Params	Event Action Interface	EFT Server 5.2 and later
type		

EventActions Interface Methods

Method	How it's used	Applicable Version
Add	Event Actions Interface Methods	EFT Server 5.2 and later
Count		
Delete		
Item		

EventRule Interface Methods and Properties

Property	How it's used	Applicable Version
Params	Event Rule Interface	EFT Server 5.2 and later
AddActionStatement		
AddIfStatement		
DeleteStatement		
Statement		
StatementsCount		

EventRules Interface Methods

Method	How it's used	Applicable Version
Add	Event Rules Interface	EFT Server 5.2 and later
Count		
Delete		
Find		
Item		

EventRuleParams Interface Properties

Property	How it's used	Applicable Version
Description	Event Rule Parameters Interface	EFT Server 5.2 and later
Enabled		
Name		

FolderMonitorEventRuleParams Interface Properties

Property	How it's used	Applicable Version
CheckHealth	Folder Monitor Event Rule Parameters Interface	EFT Server 5.2 and later
CheckHealthInterval		
Description		
Enabled		
ForcedlyDisabled		EFT Server 5.2 and prior (removed in 5.2.5)
IncludeSubfolders		
Name		
Path		

IfStatement Interface Methods and Properties

Property	How it's used	Applicable Version
type	If Statement Interface	EFT Server 5.2 and later
Condition		
ElseSection		
IfSection		

ReportActionParams Interface Properties

Property	How it's used	Applicable Version
CustomDate	Report Action Parameters Interface	EFT Server 5.2 and later
DateFormat		
FilterAndOr		
FilterField1		
FilterField2		

Property	How it's used	Applicable Version
FilterOperator1		
FilterOperator2		
FilterValue1		
FilterValue2		
FromDate		
Name		
OptionalParameters		
Path		
ReportFileFormat		
ToDate		

SimpleCondition Interface Properties

Property	How it's used	Applicable Version
Not	Simple Condition Interface	EFT Server 5.2 and later
Operator		
Property		
Value		

StopActionParams Interface Properties

Property	How it's used	Applicable Version
Action	Stop Action Parameters Interface	EFT Server 5.2 and later
Enabled		

TransferActionParams Interface Properties (ICITransferActionParams)

Property	How it's used	Applicable Version
AutoLogin	Transfer Action Parameters Interface	EFT Server 5.2 and later
DeleteSourceFile		EFT Server 5.2.5 and later
Host		EFT Server 5.2 and later
Key		EFT Server 5.2 and later
KeyPass		EFT Server 5.2 and later
LocalPath		EFT Server 5.2 and later
Operation		EFT Server 5.2 and later
Password		EFT Server 5.2 and later
Port		EFT Server 5.2 and later
Protocol		EFT Server 5.2 and later
PubKey		EFT Server 5.2 and later
RemotePath		EFT Server 5.2 and later
User		EFT Server 5.2 and later

EventRuleStatement Interface Property

Property	How it's used	Applicable Version
type	Event Rule Statement Interface	EFT Server 5.2 and later

Enumerators and Constants

The following Enumerators and Constants are used in Event Rule Interfaces.

ConditionOperator

Constant	How it's used	Applicable Version
Contains	Contains (0x08) – Contains	EFT Server 5.2 and later
Equals	Equals (0x01) – Equals	EFT Server 5.2 and later
Less	Less (0x02) – Less	EFT Server 5.2 and later
LessOrEquals	LessOrEquals (0x04) – Less or equal	EFT Server 5.2 and later
Match	Match (0x10) – Match (for file name matching with template such as ***.exe**)	EFT Server 5.2 and later
MemberOf	MemberOf (0x20) – Member of	EFT Server 5.2 and later
OneOf	OneOf (0x40) – One of	EFT Server 5.2 and later
StartsWith	StartsWith (0x80) – Starts with	EFT Server 5.2 and later

EventActionType

Constant	How it's used	Applicable Version
CleanupAction	CleanupAction (0x80) – Cleanup in folder	EFT Server 5.2 and later
CommandAction	CommandAction (0x01) – Execute command	EFT Server 5.2 and later
DownloadAction	DownloadAction (0x08) – Download file	EFT Server 5.2 and later
MailAction	MailAction (0x02) – Send E-mail	EFT Server 5.2 and later
PGPAction	PGPAction (0x20) – PGP action	EFT Server 5.2 and later
ReportAction	ReportAction (0x100) – Generate report	EFT Server 5.2 and later
StopAction	StopAction (0x40) – Stop rule execution	EFT Server 5.2 and later
UploadAction	UploadAction - Upload file	EFT Server 5.2.5 and later

Event Properties

The following properties are used in the Event Rule interfaces.

Category	Property	How it's used	Applicable Version
General properties:	Time (1) – Current time	Event Properties	EFT Server 5.2 and later
	Name (2) – Event name		
	Reason (3) – Event reason		
	TimeStamp (4) – Current time		
	DateStamp (5) – Current date		
	MonitorFolderStatus (6) – Folder monitoring status		
	EventName (7) – Event name		
Server Properties:	ServerRunning (1000) – Whether server is running	Event Properties	EFT Server 5.2 and later
	ServerLogOldName (1001) – Old log file name		
	ServerLogNewName (1002) – New log file name		
	ServerLogOldPath (1003) – Old log file path		
	ServerLogNewPath (1004) – New log file path		
	ServerLogTime (1005) – Log type		
	ServerLogFolder (1006) – Log folder		
	ServerServerNodeName (1007) – Name of the node server is running on		
Site Properties:	SiteRunning (2000) – Whether site is started	Event Properties	EFT Server 5.2 and later
	SiteName (2001) – Site name		
	SiteAccountManagementURL (2002) – PCI account management URL		
Connection Properties:	LocalIP (3000) – Server IP	Event Properties	EFT Server 5.2 and later
	RemoteIP (3001) – Remote peer IP		
	LocalPort (3002) – Server port		
	RemotePort (3003) – Remote peer port		
	Protocol (3004) – Connection protocol		
	WebTransferClientConnection (3005) – Web-Transfer Client connection		
Client Properties:	ClientLogin (4000) – Client's login name	Event Properties	EFT Server 5.2 and later
	ClientPassword (4001) – Client's password		
	ClientAccessGroup (4002) – Whether client belongs to one of the permission groups		
	ClientEnabled (4003) – Is client's account enabled		
	ClientSettingsLevel (4004) – Client's settings template		
	ClientFullName (4005) – Client's full name		
	ClientDescription (4006) – Client's description		
	ClientComment (4007) – Comment to client's account		
(Client Properties, cont'd)	ClientEMail (4008) – Client's e-mail		

Category	Property	How it's used	Applicable Version
	ClientPhone (4009) – Client's phone number		
	ClientPager (4010) – Client's pager		
	ClientFax (4011) – Client's fax		
	ClientHomeFolder (4012) – Client's home folder		
	ClientHomeFolderIsRoot (4013) – Is client's home folder root for them		
	ClientQuotaMax (4014) – Client's disk quota (max)		
	ClientQuotaUsed (4015) – Client's disk quota (currently used)		
	ClientInvalidLoginAttempts (4016) – Client's invalid login attempt count		
	ClientCanChangePassword (4017) – Has client permission to change their password		
	ClientIP (4018) – Client's IP		
	ClientSSLAllowed (4019) – Has client permission to use SSL encryption		
	ClientFTPAllowed (4020) – Has client permission to connect via FTP		
	ClientSFTPAllowed (4021) – Has client permission to connect via SFTP		
	CClientLastLogin (4022) – Client's last login timestamp		
	ClientPasswordExpiration (4023) – Client's password expiration date		
	ClientMustResetPasswordAtFirstLogin (4024) – Must client change their password on		
	initial login		
	ClientAccountExpirationDate (4025) – Client's account expiration date		
	ClientAccountLocked (4026) – Is client's account locked out		
	ClientCustomField1 (4027) – Client's account custom field #1		
	ClientCustomField2 (4028) – Client's account custom field #2		
	ClientCustomField3 (4029) – Client's account custom field #3		
File System Properties:	VirtualPath (5000) – File virtual path	Event Properties	EFT Server 5.2 and later
	PhysicalPath (5001) – File physical path		
	DestinationVirtualPath (5002) – Destination file virtual path		
	DestinationPhysicalPath (5003) – Destination file physical path		
	FolderName (5004) – Folder		
	FileName (5005) – File name		
	DestinationFolderName (5006) – Destination folder		

Category	Property	How it's used	Applicable Version
(File System Properties, cont'd)	DestinationFileName (5007) – Destination file name		
	FolderOperation (5008) – Folder operation		
	FileCreationDate (5009) – File creation date		
	FileCreationTime (5010) – File creation time		
	FileSize (5011) – File size		
	FileCRC (5012) – File CRC		
	ReportPath (5013) – Report file path		
	ReportContent (5014) – Report content		
	ReportFileName (5015) – Report file name		

EventRuleStatementType

Constant	How it's used	Applicable Version
ActionStatement	ActionStatement (0) – Action statement	EFT Server 5.2 and later
IfStatement	IfStatement (1) – Conditional statement	EFT Server 5.2 and later

Event Type

Category	Event	How it's used	Applicable Version
Server Events:	MonitorFolder	Enumerators and Constants	EFT Server 5.2 and later
	OnMonitorFolderFailed		
	OnLogRotate		
	OnServiceStarted		
	OnServiceStopped		
	OnTimer		
Site Events:	OnSiteStarted	Enumerators and Constants	EFT Server 5.2 and later
	OnSiteStopped		
Connection Events:	OnClientConnected	Enumerators and Constants	EFT Server 5.2 and later
	OnClientConnectonFailed		
	OnClientDisconnected		
Client Events:	OnClientCreated	Enumerators and Constants	EFT Server 5.2 and later
	OnClientDisabled		
	OnClientQuotaExceeded		
	OnClientLoggedOut		
	OnClientLoggedIn		
	OnClientLoginFailed		
	OnClientPasswordChanged		
File System Events:	OnFileDeleted	Enumerators and Constants	EFT Server 5.2 and later
	OnFileUploaded		
	BeforeFileDownload		
	OnFileDownloaded		
	OnFileRenamed		
	OnFolderCreated		
	OnFolderDeleted		
	OnUploadFailed		
	OnDownloadFailed		
	OnChangeFolder		
	OnFileMoved		
	OnVerifiedUploadSuccess		
	OnVerifiedUploadFailure		
	OnVerifiedDownloadSuccess		
	OnVerifiedDownloadFailure		

LogicalOperator

Constant	How it's used	Applicable Version
LogicalAnd	LogicalOr (0) – OR	EFT Server 5.2 and later
LogicalOr	LogicalAnd (1) – AND	EFT Server 5.2 and later

Network Protocol

Constant	How it's used	Applicable Version
ProtocolFTP		EFT Server 5.2.5
ProtocolFTPS		EFT Server 5.2.5
ProtocolFTPSAuthTLS		EFT Server 5.2.5
ProtocolFTPSExpl		EFT Server 5.2.5
ProtocolHTTP		EFT Server 5.2.5
ProtocolHTTPS		EFT Server 5.2.5
ProtocolLocal		EFT Server 5.2.5
ProtocolSFTP		EFT Server 5.2.5

PGP Operation

Constant	How it's used	Applicable Version
Encrypt	Determining Operation for PGP Event Action (Operation)	EFT Server 5.2 and later
Decrypt		

Recurrence

Constant	How it's used	Applicable Version
Custom	Custom (0) – Hourly	EFT Server 5.2 and later
Daily	Daily (1) – Daily	EFT Server 5.2 and later
Weekly	Weekly (2) – Weekly	EFT Server 5.2 and later
Monthly	Monthly (3) – Monthly	EFT Server 5.2 and later
Yearly	Yearly (4) – Yearly	EFT Server 5.2 and later
OneTime	OneTime (5) – Trigger only once	EFT Server 5.2 and later

SFTP AdvBool

Constant	How it's used	Applicable Version
abFalse	(0)	EFT Server 5.2 and later
abTrue	(1)	EFT Server 5.2 and later
abInherited	(2)	EFT Server 5.2 and later

StopType

Constant	How it's used	Applicable Version
StopRule	(2)	EFT Server 5.2.5
StopEvent	(4)	EFT Server 5.2.5
StopeRuleEvent	(6)	EFT Server 5.2.5

Sample Script: Retrieve List of Users

The VB Script below connects to the server, retrieves a list of users and publishes them to an Excel file. To try it, save the script below with a .vbs extension and then run it from the command line. Be sure to first edit the "txt" values based on your server's settings. txtServer is your server's IP address; txtPort is your server' listening port ; txtUserName and txtPassword are the administrator's username/password pair.

```
'
' FILE: CreateUserListSpreadSheet
' CREATED: 13 OCT 2004 GTH
' PURPOSE: List the users of a site and create an excel spreadsheet.
'
Set SFTPServer = WScript.CreateObject("SFTPCOMInterface.CIServer")
CRLF = (Chr(13)& Chr(10))
   txtServer = "192.168.134.142"
   txtPort =  "1000"
   txtUserName = "admin"
   txtPassword = "admin"
' On Error Resume Next
SFTPServer.Connect txtServer, txtPort, txtUserName, txtPassword
If Err.Number <> 0 Then
   WScript.Echo "Error connecting to '" & txtServer & ":" & txtPort & "' -- " &
err.Description & " [" & CStr(err.Number) & "]", vbInformation, "Error"
   WScript.Quite(255)
Else
   WScript.Echo "Connected to " & txtServer
End If
set Sites=SFTPServer.Sites
Set oExcel = WScript.CreateObject("Excel.Application")
oExcel.visible = true
Set oWorkbook = oExcel.Workbooks.Add
For i = 0 to SFTPServer.Sites.Count-1
   set theSite=Sites.Item(i)
   Set theSheet = oWorkbook.Worksheets.add
   theSheet.name = theSite.Name
   theSheet.Cells(1, 1) = "Users:"
   arUsers = theSite.GetUsers()
   For j = LBound(arUsers) to UBound(arUsers)
      theSheet.Cells((j+2), 1) = arUsers(j)
   Next
   theSheet.Columns("A:A").EntireColumn.Autofit
Next
Set oExcel = nothing
SFTPServer.Close
Set theSite = nothing
Set SFTPServer = nothing
```

Sample Script: Setting Permissions

This sample VBA script creates a user, creates some virtual folders, and sets various permissions. Before testing, be sure to edit the script for your connection parameters (Admin ID, Password, Port, Server).

```
``````````````````````````````````````

Private Sub Command1_Click()

Set oSFTPServer = CreateObject("SFTPCOMInterface.CIServer") 'instantiate EFT/GSFTPS
COM Object

'==
'   Initialize connection parameters
'==

Dim sAdminID, sAdminPassword, sAdminServer, oSites, oSite

sAdminID = "admin"
sAdminPassword = "test"
sAdminPort = "1100" '1100 for EFT or 1000 for Secure FTP Server by default
sAdminServer = "localhost" 'enter the IP address for the admin connection to the
server

'==
'   Initialize the user parameters
'==
Dim sUsrID, sUsrPasswd, sUsrDesc

sUsrID = "roslin"
sUsrPasswd = "test"
sUsrDesc = "Trading Partner Account"
sGroup = "Trading"

'==
'   Connect to the server
'==

On Error Resume Next
oSFTPServer.Connect sAdminServer, sAdminPort, sAdminID, sAdminPassword
If Err.Number <> 0 Then
    Set oSFTPServer = Nothing
    MsgBox "Could not connect to the server with the specified parameters.",
vbCritical, vbOKOnly, "Error Connecting to Server"
    Exit Sub
End If

'==
'Obtain a handle to all sites and then to the first site in this example
'==

Set oSites = oSFTPServer.Sites
Set oSite = oSites.Item(0)

'==
'   Conditionally create the user
'==
Dim arUsers() As Variant
Dim bUserExist As Boolean

arUsers = oSite.GetUsers()
```
```

```
j = 0
For j = LBound(arUsers) To UBound(arUsers)
    If arUsers(j) = sUsrID Then
        bUserExist = True
    End If
Next

If bUserExist = False Then
Call oSite.CreateuserEx(sUsrID, sUsrPasswd, 0, sUsrDesc, sUsrID, True, False)
End If

'=====================================================================
'    Create some virtual folders and point those to phyisical ones
'=====================================================================

'note: The physical folders must exist on the drive!
Call oSite.CreateVirtualFolder("/Usr/" & sUsrID & "/INBD",
"D://EDITPS//EDITPS//TradingPartners//" & sUsrID & "//INBD//")
Call oSite.CreateVirtualFolder("/Usr/" & sUsrID & "/OUTBD",
"D://EDITPS//EDITPS//TradingPartners//" & sUsrID & "//OUTBD//")
Call oSite.CreateVirtualFolder("/Usr/" & sUsrID & "/LOG",
"D://EDITPS//EDITPS//TradingPartners//" & sUsrID & "//LOG//")


'=====================================================================
'           Add this user to a group
'=====================================================================

'first see if group exists
Dim arGroup() As Variant
Dim bGroupExist As Boolean

arGroup = oSite.GetPermissionGroups()
j = 0
bGroupExist = False
For j = LBound(arGroup) To UBound(arGroup)
    If arGroup(j) = sGroup Then
        bGroupExist = True
    End If
Next

'now check to see if that user exists in the group
Dim arPermGroupOfUser() As Variant
Dim bExistsInGroup As Boolean
arPermGroupOfUser = oSite.GetPermissionGroupsOfUser(sUsrID)
h = 0
bExistsInGroup = False
For h = LBound(arPermGroupOfUser) To UBound(arPermGroupOfUser)
    If arPermGroupOfUser(h) = sGroup Then
        bExistsInGroup = True
    End If
Next

'create the group if it doesn't exist
If bGroupExist = False Then
    Call oSite.CreatePermissionGroup(sGroup)
End If

'add the user to the group if it doesn't exist
If bExistsInGroup = False Then
    Call oSite.AddUserToPermissionGroup(sUsrID, sGroup)
End If
```

```
'=========================================================================
'             Disable non-secure FTP Access for this user
'=========================================================================
Dim oUsrSettings As Variant
oUsrSettings = oSite.GetUserSettings(sUsrID)
oUsrSettings.SetClearFTP (0)
oUsrSettings.SetSSL (1)

'=========================================================================
'           Assign folder permissions for this user
'=========================================================================

Dim aIN, aOUT, aLo, sUsrDir

sUsrDir = "\Usr\" & sUsrID & "\INBD\"
Set aIN = oSite.GetBlankPermission(sUsrDir, sUsrID)
aIN.DirShowInList = True
aIN.DirList = True
aIN.FileUpload = True
Call oSite.SetPermission(aIN)

sUsrDir = "\Usr\" & sUsrID & "\OUTBD\"
Set aOUT = oSite.GetBlankPermission(sUsrDir, sUsrID)
aOUT.DirShowInList = True
aOUT.DirList = True
aOUT.FileDelete = True
aOUT.FileDownload = True
Call oSite.SetPermission(aOUT)

sUsrDir = "\Usr\" & sUsrID & "\LOG\"
Set aLo = oSite.GetBlankPermission(sUsrDir, sUsrID)
aLo.DirShowInList = True
aLo.DirList = True
aLo.FileDownload = True
Call oSite.SetPermission(aLo)


'=========================================================================
'          Apply changes and close connection to the server
'=========================================================================

Call oSFTPServer.ApplyChanges
Call oSFTPServer.Close
End Sub
```

Sample Script: Retrieving the Physical Path to a Virtual Folder

The sample script below is an example of using the GetPhysicalPath method to retrieve the physical path to a virtual folder.

```
Dim SFTPServer
Dim Sites, Site, aUsers
Dim CRLF
CRLF = (Chr(13)& Chr(10))
Set SFTPServer = CreateObject("SFTPCOMInterface.CIServer")
'Get Server address and administrative login
   txtServer = inputbox("Host Address" & CRLF & CRLF & "Enter the host address of
the server you want to connect to." & CRLF & _
               "Use 'localhost' if you are connecting to the server on the local
machine.","EFT COM Query")
   txtPort =  "1100"
   txtUserName = inputbox("Admin Username" & CRLF & CRLF & "Enter the same username
used for the GlobalSCAPE Enhanced File Transfer Server GUI Interface - a.k.a. your
Administrative Login","EFT COM Query")
   txtPassword = inputbox("Admin Password" & CRLF & CRLF & "Enter the same password
used for the GlobalSCAPE Enhanced File Transfer Server GUI Interface - a.k.a. your
Administrative Password","EFT COM Query")
SFTPServer.Connect txtServer, txtPort, txtUserName, txtPassword
set Sites=SFTPServer.Sites
set site = Sites.Item(0)
txtFolder = inputbox("Enter a VFS path" & CRLF & CRLF & "for root you can use /"&
CRLF&"You do not need a trailing slash on the VFS path","EFT COM Query")
msgbox "physical path for " & txtFolder & " is:" & CRLF & CRLF &
site.GetPhysicalPath(txtFolder)
SFTPServer.Close
set Site = nothing
set Sites = nothing
set SFTPServer = nothing
```

Server Interface Properties (ICIServer)

The ICIServer interface allows you to manage the server and access the server's attributes.

Editing the Server Administrator IP Address (ListenIP)

Use the ICIServer interface **ListenIP** property to retrieve or modify the server administrator's IP address.

Signature:

```
HRESULT ListenIP([out, retval] long *pVal);
HRESULT ListenIP([in] long newVal);
```

Editing the Administrator Port (AdminPort)

Use the ICIServer interface **AdminPort** property to retrieve or modify the port used for remote administration connections to administer the Server

Signature:

```
HRESULT AllowClearCommandChannel([out, retval] VARIANT_BOOL *pVal);
HRESULT AllowClearCommandChannel([in] VARIANT_BOOL newVal);
```

Example:

```
oServer.AllowClearCommandChannel = true
```

You must restart the server to apply any change to the server administrator port.

Allowing a Clear Command Channel on the Server (AllowClearCommandChannel)

This method is available in EFT Server 5.0 and later.

Use the ICIServer interface **AllowClearCommandChannel** property to allow or prohibit a clear command channel on the server.

Signature:

```
HRESULT AllowClearCommandChannel([out, retval] VARIANT_BOOL *pVal);
HRESULT AllowClearCommandChannel([in] VARIANT_BOOL newVal);
```

Example:

```
oServer.AllowClearCommandChannel = true
```

Allowing an Unprotected Data Channel on the Server (AllowUnprotectedDataChannel)

This method is available in EFT Server 5.0 and later.

Use the ICIServer interface **AllowUnprotectedDataChannel** property to allow or prohibit an unprotected data channel on the server for the FTP Protocol.

Signature:

```
HRESULT AllowUnprotectedDataChannel([out, retval] VARIANT_BOOL *pVal);
HRESULT AllowUnprotectedDataChannel([in] VARIANT_BOOL newVal);
```

Example:

```
oServer. AllowUnprotectedDataChannel = true
```

Allowing Remote Administration (AllowRemoteAdministration)

Use the ICIServer interface **AllowRemoteAdministration** property to allow or prohibit remote administration of the server.

Signature:

```
HRESULT AllowRemoteAdministration([out, retval] VARIANT_BOOL *pVal);
HRESULT AllowRemoteAdministration([in] VARIANT_BOOL newVal);
```

Editing the ARM Database Name (ARMDatabaseName)

Use the ICIServer interface **ARMDatabaseName** property to retrieve or set the Auditing and Reporting database name.

Signature:

```
HRESULT ARMDatabaseName([out, retval] BSTR *pVal);
HRESULT ARMDatabaseName([in] BSTR newVal);
```

Example:

```
Dim strDBName: strDBName = oServer.ARMDatabaseName
WScript.Echo "ARM Database: " & strDBName
```

Enabling Auditing and Reporting (EnableARM)

Use the ICIServer interface **EnableARM** property to enable or disable Auditing and Reporting, or evaluate status of Auditing and Reporting.

Signature:

```
HRESULT EnableARM([out, retval] VARIANT_BOOL *pVal);
HRESULT EnableARM([in] VARIANT_BOOL newVal);
```

Examples:

```
Dim bARMStatus: bARMStatus = oServer.EnableARM
```

To set the state:

```
oServer.EnableARM = false
```

Editing the ARM Server Name (ARMServerName)

Use the ICIServer interface **ARMServerName** property to retrieve the name of the server on which the Auditing and Reporting database is located.

Signature:

```
HRESULT ARMServerName([out, retval] BSTR *pVal);
HRESULT ARMServerName([in] BSTR newVal);
```

Example:

```
Dim strServerName: strServerName = oServer.ARMServerName
oServer. ARMServerName = "localhost\GlobalSCAPE"
```

Editing the ARM Username (ARMUserName)

Use the ICIServer interface **ARMUserName** property to retrieve or set the username used to connect to the Auditing and Reporting database.

Signature:

```
HRESULT ARMServerName([out, retval] BSTR *pVal);
HRESULT ARMServerName([in] BSTR newVal);
```

Example:

```
Dim strUserName : strUserName = oServer.ARMUserName
oServer. ARMUserName = "admin"
```

Editing the ARM Password (ARMPassword)

Use the ICIServer interface **ARMPassword** property to retrieve or set the password for logging in to the ARM Database.

Signature:

```
HRESULT ARMPassword([out, retval] BSTR *pVal);
HRESULT ARMPassword([in] BSTR newVal);
```

Example:

```
Dim strNewPassword: strNewPassword = "newpassword"
oServer.ARMPassword = strNewPassword
```

Choosing Email Receiver Addresses (SMTPRecipientAddr)

Use the ICIServer interface **SMTPRecipientAddr** property to retrieve or modify the email address of the recipient notifications sent by the Server.

Signature:

```
HRESULT SMTPRecipientAddr([out, retval] BSTR *pVal);
HRESULT SMTPRecipientAddr([in] BSTR newVal);
```

Choosing Email Receiver Names (SMTPRecipientName)

Use the ICIServer interface **SMTPRecipientName** property to retrieve or modify the administrator email name for email notifications sent by the Server.

Signature:

```
HRESULT SMTPRecipientName([out, retval] BSTR *pVal);
HRESULT SMTPRecipientName([in] BSTR newVal);
```

Choosing the From Address in Server Emails (SMTPSenderAddr)

Use the ICIServer interface **SMTPSenderAddr** property to retrieve or modify the administrator email address for email notifications sent by the Server.

Signature:

```
HRESULT SMTPSenderAddr([out, retval] BSTR *pVal);
HRESULT SMTPSenderAddr([in] BSTR newVal);
```

Choosing the Server Email Sender Name (SMTPSenderName)

Use the ICIServer interface **SMTPSenderName** property to retrieve or modify the **From** name used in email notifications sent by the Server.

Signature:

```
HRESULT SMTPSenderName([out, retval] BSTR *pVal);
HRESULT SMTPSenderName([in] BSTR newVal);
```

Editing the Mail Server Address (SMTPServer)

Use the ICIServer interface **SMTPServer** property to retrieve or modify the address for the email server used by the Server for sending event notifications.

Signature:

```
HRESULT SMTPServer([out, retval] BSTR *pVal);
HRESULT SMTPServer([in] BSTR newVal);
```

Editing the Mail Server Port (SMTPPort)

Use the ICIServer interface **SMTPPort** property to retrieve or modify the port number for connection to the SMTP server used by the Server for event notifications.

Signature:

```
HRESULT SMTPPort([out, retval] long *pVal);
HRESULT SMTPPort([in] long newVal);
```

Editing the Email User Name (SMTPLogin)

Use the ICIServer interface **SMTPLogin** property to retrieve or modify the user name for the email server used by the Server for event notifications, if authentication is required.

Signature:

```
HRESULT SMTPLogin([out, retval] BSTR *pVal);
HRESULT SMTPLogin([in] BSTR newVal);
```

Editing the Email Password (SMTPPassword)

Use the ICIServer interface **SMTPPassword** property to retrieve or modify the password for the email server used by the Server for event notifications, if authentication is required.

Signature:

```
HRESULT SMTPPassword([out, retval] BSTR *pVal);
HRESULT SMTPPassword([in] BSTR newVal);
```

Editing the File Path for Certificates (CertificateFilePath)

Use the ICIServer interface **CertificateFilePath** property to retrieve or modify the file path to the server's certificates.

Signature:

```
HRESULT CipherList([out, retval] BSTR *pVal);
HRESULT CipherList([in] BSTR newVal);
```

CipherList must be delimited by a colon (:).

Examples:

```
Dim strCipherList: strCipherList = oServer.CipherList
"AES256-SHA:CAMELLIA256-SHA:DES-CBC3-SHA:AES128-SHA:IDEA-CBC-SHA:RC4-MD5"
```

Refer to the OpenSSL Documentation for the format of these strings:
http://www.openssl.org/docs/apps/ciphers.html#CIPHER_LIST_FORMAT

Editing the File Path for Private Key Files (KeyFilePath)

Use the ICIServer interface **KeyFilePath** property to retrieve or modify the file path for the server's private key files.

Signature:

```
HRESULT KeyFilePath([out, retval] BSTR *pVal);
HRESULT KeyFilePath([in] BSTR newVal);
```

Editing Private Key Passphrases (PassPhrase)

Use the ICIServer interface **PassPhrase** property to retrieve or modify any certificate private key pass phrases.

Signature:

```
HRESULT PassPhrase([out, retval] BSTR *pVal);
HRESULT PassPhrase([in] BSTR newVal);
```

Editing the Server Log Path (LogPath)

Use the ICIServer interface **LogPath** property to retrieve or set the file path to the server's log. You must restart the server to complete any changes to the server's log path.

Signature:

```
HRESULT LogPath([out, retval] BSTR *pVal);
HRESULT LogPath([in] BSTR newVal);
```

Editing the Server Log Type (LogType)

Use the ICIServer interface **LogType** property to retrieve or set the server's log format. You can choose no log, or you can choose to generate a log in one of three formats.

Signature:

```
HRESULT LogType([out, retval] int *pVal);
HRESULT LogType([in] int newVal);
```

0 = W3C

1 = IIS

2 = NCSA

3 = No logging

You must restart the server for any changes to the log format to take effect.

Rotating Logs (LogRotation)

Use the ICIServer interface **LogRotation** property to retrieve or set the amount of time before a new log is started.

You can choose from no rotation or three different lengths of time.

Signature:

```
HRESULT LogRotation([out, retval] int *pVal);
HRESULT LogRotation([in] int newVal);
```

0 = Never

1 = Daily

2 = Weekly

3 = Monthly

You must restart the server to complete any change to the log rotation period.

Finding or Setting the Default IP Access Mode (IPAccessAllowedDefault)

Use the ICIServer interface **IPAccessAllowedDefault** property to find or set the default IP Access restriction list mode.

Signature:

```
HRESULT IPAccessAllowedDefault([out, retval] VARIANT_BOOL *pVal);
HRESULT IPAccessAllowedDefault([in] VARIANT_BOOL newVal);
```

True = All IP addresses are allowed except those specified

False = All IP addresses are denied except those specified

Retrieving or Setting the SSL Version Mask (SSLVersionMask)

Use the ICIServer interface **SSLVersionMask** property to retrieve or set the SSL version mask.

This property is available in EFT Server 5.0 and later.

Signature:

```
HRESULT SSLVersionMask([out, retval] int *pVal);
HRESULT SSLVersionMask([in] int newVal);
```

Example:

```
Dim iVersionMask: iVersionMask = oServer.SSLVersionMask
WScript.Echo "Version Mask: " & iVersionMask
```

Requiring SSL for Remote Administration (UseSSLForAdministration)

Use the ICIServer interface **UseSSLForAdministration** property to enable or disable SSL for the administration socket connection.

SSL.DLL must be deployed alongside SFTPCOMInterface.DLL on the remote machine.

Signature:

```
HRESULT UseSSLForAdministration([out, retval] VARIANT_BOOL *pVal);
HRESULT UseSSLForAdministration([in] VARIANT_BOOL newVal);
```

*You must restart the server to apply changes made with the **UseSSLForAdministration** property.*

Editing the Cipher List (CipherList)

Use the ICIServer interface **CipherList** property to retrieve or modify the cipher list.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT CipherList([out, retval] BSTR *pVal);
HRESULT CipherList([in] BSTR newVal);
```

Example:

```
Dim strCipherList: strCipherList = oServer.CipherList
```

CipherList must be delimited by colon (':').

For example,

```
"AES256-SHA:CAMELLIA256-SHA:DES-CBC3-SHA:AES128-SHA:IDEA-CBC-SHA:RC4-MD5"
```

Retrieving the Number of Connected Users (ConnectedUsersNumber)

Use the ICIServer interface **ConnectedUsersNumber** property to retrieve the number of users connected to the server.

This method is available in EFT Server 3.5 and later.

Signature:

```
HRESULT ConnectedUsersNumber([out, retval] BSTR *pVal);
```

Example:

```
Dim iConnectedUses: iConnectedUsers = oServer.ConnectedUsersNumber
```

ConnectedUsersNumber *is a read-only property.*

The ICIServer interface allows you to manage the server and access the server's attributes.

Retrieving a List of Server Sites (Sites)

Use the ICIServer Interface **Sites** method to access the list of Sites and open the ICISites interface. You can subsequently use the ICISites **Item** method to retrieve a pointer to a particular Site. This method returns a pointer to an ICISites interface.

Signature:

```
HRESULT Sites([out, retval] ICISites** prop);
```

Example in VB:

```
Dim Server As CIServer 'assumes the SFTPCOMInterface Type Library is referenced by
the VB IDE
Dim Site As CISite 'instantiates the ISite interface
Dim Sites As CISites 'instantiates the ISites interface
'insert connection routines here
'no retrieve the list of sites and then set a pointer to the first one
Set Sites = Server.Sites 'now retrieve the list of sites in the server
Set Site = Sites.Item(0)
'continue rest of code...
```

Adding an IP Mask on the Server (AddIPAccessRule)

Use the ICIServer interface **AddIPAccessRule** method to add an allowed or denied IP mask to the server.

Signature:

```
HRESULT AddIPAccessRule(
                [in] BSTR bstrMask,
                [in] VARIANT_BOOL bAllow);
```

Applying Changes to the Server (ApplyChanges)

Use the ICIServer interface **ApplyChanges** method to confirm and set changes you have made. (This method is available in Secure FTP Server, and EFT Server 4.34 and prior. As of EFT Server 5.0, any "Set" or other "Write" method applies the change made immediately.)

Signature: `HRESULT ApplyChanges();`

Refreshing Server Settings (RefreshSettings)

Use the ICIServer interface **RefreshSettings** method to bring the COM object up to date with the latest set of data from the Server to which it is connected.

If a call to any COM method throws an "MX Error: 52 (0x00000034)", this means that the COM object needs to invoke **RefreshSettings**.

Signature:

```
HRESULT RefreshSettings();
```

Determining if Transactions are Audited to a File instead of the Database (AreLingeringTransactions)

This method is available in EFT Server 5.2 and later.

Use the **ICIServer** interface **AreLingeringTransactions** method to determine whether there is socket session/Event Rule/etc that is audited to a file instead of the database.

Signature

```
HRESULT AreLingeringTransactions([out,retval] VARIANT_BOOL * pVal);
```

Example

```
If server. AreLingeringTransactions then
      ' Do something
End if
```

CChanging the Administrator Password (ChangeAdminPassword)

This method is available in EFT Server 4.3.4 and later.

Use the ICIServer interface **ChangeAdminPassword** method to change the password for an administrator account in the EFT Server.

Parameters:

```
AdminUser - the name of the delegated administration account that will have a new
password
Password - the new password value
```

Example:

```
oServer.ChangeAdminPassword( "subadmin", "newpassword")
```

Closing the Administrator Connection to the Server (Close)

Use the ICIServer interface **Close** method to close the COM administrative connection to the server.

Signature:

```
HRESULT Close();
```

Connecting to the Server as the Administrator (Connect)

Use the ICIServer interface **Connect** method to connect to the local or remote server as the administrator.

Signature:

```
HRESULT Connect([in] BSTR Host, [in] UINT nPort, [in] BSTR Login, [in] BSTR
Password);
```

VB Example:

```
Dim Server As CIServer 'assumes the SFTPCOMInterface Type Library is referenced by
the VB IDE
Dim Site As CISite 'instantiates the ISite interface
Dim Sites As CISites 'instantiates the ISites interface
txtServer = "127.0.0.1" 'local IP address of server
txtPort = "1000" 'admin port
```

```
txtUserName = "Admin" 'admin username
txtPassword = "mypass" 'admin password
On Error Resume Next 'verify it could connect ok
Server.Connect txtServer, txtPort, txtUserName, txtPassword 'now connect using
supplied information
If Err.Number <> 0 Then
  MsgBox "Error connecting to '" & txtServer & ":" & txtPort & "' -- " &
Err.Description & " [" & CStr(Err.Number) & "]", vbInformation, "Error" 'all this
on one line!
End If
Set Sites = Server.Sites 'now retrieve the list of sites in the server
Set Site = Sites.Item(0) 'Instantiate the first and only, site (note, this script
assumes only one site present)
'continue rest of code...
```

Retrieving a List of Administrator IPs (GetAdminIPs)

This method is available in EFT Server 4.3.4 and later.

Use the ICIServer interface **GetAdminIPs** to retrieve a string list of IP addresses for the machine on which the EFT Server is running. The IP addresses can be used as LISTENING IP addresses for administration or protocol services.

Example:

```
Dim arIPS : arIPS = oServer.GetAdminIPs()
for i = LBound(arIPS) to UBound(arIPS)
   WScript.Echo CStr(i) & " - " & arIPS(i)
next
```

Retrieving Allowed IP Masks (GetAllowedMasks)

Use the ICIServer interface **GetAllowedMasks** method to retrieve a list of allowed IP masks for the server.

Signature:

```
HRESULT GetAllowedMasks([out, retval] VARIANT *aMasks);
```

Retrieving Denied IP Masks (GetDeniedMasks)

Use the ICIServer interface **GetDeniedMasks** method to retrieve a list of all prohibited IP masks for the server.

Signature:

```
HRESULT GetDeniedMasks([out, retval] VARIANT *aMasks);
```

Retrieving the Local IP Address (GetLocalIP)

Use the ICIServer interface **GetLocalIP** method to find the local IP address.

Signature:

```
HRESULT GetLocalIP(
                [in] long nIP,
                [out, retval] BSTR *prop);
```

Retrieving the Local Server Time (GetLocalTime)

Use the ICIServer interface **GetLocalTime** method to retrieve the time of day on the server.

Signature:

```
HRESULT GetLocalTime([out, retval] BSTR *pTime);
```

Determining if Server is Connected to the ARM Database (IsDBConnected)

This method is available in EFT Server 5.2 and later.

Use the ICIServer interface **IsDBConnected** method to determine whether the ARM database is connected

Signature

```
HRESULT IsDBConnected([out,retval] VARIANT_BOOL * pVal);
```

Example

```
If server.IsDBConnected then
      ' Do something
End if
```

Removing an IP Mask (RemoveIPAccessRule)

Use the ICIServer interface **RemoveIPAccessRule** method to delete an allowed or denied IP mask.

Signature:

```
HRESULT RemoveIPAccessRule(
                [in] BSTR bstrMask,
                [in] VARIANT_BOOL bAllow);
```

Removing a Server Administrator Account (RemoveServerAdminAccount)

This method is available in EFT Server 4.3.4 and later.

Use the ICIServer interface **RemoveServerAdminAccount** to remove an administrator account from the server.

Signature:

```
HRESULT RemoveServerAdminAccount(BSTR bstAdminUser);
```

Example:

```
oServer.RemoveServerAdminAccount("subadmin");
```

Retrieving the Number of Server Administrator Accounts on the Server (GetServerAdminCount)

This method is available in EFT Server 4.3.4 and later.

Use the ICIServer interface **GetServerAdminCount** to get a count of active administrator accounts on the server.

Example:

```
Dim i : i  = oServer.GetServerAdminCount()
```

Multiple Sites Interface Methods (ICISites)

The ICISites interface represents the list of all Sites on the server. Obtain it by using the **Sites** method of the ICIServer interface.

```
Set Sites = SFTPServer.Sites
```

The methods below are used to add additional Sites to the Server.

Retrieving a Site's COM Interface by Site Number (Item)

Use the ICISites interface **Item** method and an integer to retrieve the ICISite interface for a specific Site.

Example:

```
Item(
    [in] VARIANT* Index,
    [out, retval] ICISite** prop);
```

*The **Index** value must be between 0 and (Count-1).*

Example:

```
Set Sites = Server.Sites 'Retrieve list of sites
Set Site = Sites.Item(0) 'Instantiate the first one
```

Retrieving a Site's COM Interface by Site ID (SiteByID)

Use the ICISites interface **SiteByID** method to retrieve the ICISite interface.

Signature:

```
HRESULT SiteByID(
          [in] long ID,
          [out, retval] ICISite** prop);
```

Retrieving the Number of Sites (Count)

Use the ICISites interface **Count** method to retrieve the number of Sites you have in the Server.

Signature:

```
HRESULT Count([out, retval] long* prop);
```

Adding a Site (Add)

Use the ICISites interface **Add** method to create a new Site and retrieve the Site's ICISite interface.

Signature:

```
HRESULT Add(
         [in] BSTR bstrName,
         [in] BSTR bstrRootFolder,
         [in] int nAMID,
         [in] BSTR bstrAMDB,
         [in] VARIANT_BOOL bEncryptPasswords,
         [in] int nIP,
         [in] UINT nPort,
         [in] VARIANT_BOOL bCreateHomeFolder,
         [in] VARIANT_BOOL bPermHomeFolder,
         [in] VARIANT_BOOL bAutoStart,
```

```
[out, retval] ICISite** prop);
```

Parameters	Definition
bstrName	New site name
bstrRootFolder	Site root folder, as a full path for example, "c:\inetpub\eftroot\new site"
nAMID	Authentication manager identifier; this must be a number between 0 and 3 (inclusive), as follows: 0 : GlobalSCAPE EFT Server Authentication (proprietary flat file with an ".aud" extension) 1 : NTLM/Active Directory Authentication 2 : ODBC Authentication 3 : LDAP Authentication
bstrAMDB	Authentication manager database configuration string. This is specific to the Authentication Manager chosen by the nAMID parameter. NT: "NTLM=1;(DC=<<domain controller name>>;)(GROUP=<<group name>>;)" AD: ";(DC=<<domain controller name>>;)(GROUP=<<group name>>;)" ODBC: "<<dsn connection string>>" LOCAL: "<<path to AUD file>>" LDAP: "LDAPDB=<<path to AUD file>>;LDAPSERVER=<<host>>;LDAPPORT=<<port>>; BASEDN="<<base dn for users>>;USERFILTER=<<user query filter>>;USERATTRIBUTE=<<logon attribute>>;SSL={0|1};(USERNAME=<<ldap simple bind username>>;PASSWORD=<<ldap simple bind password>>;)"
bEncryptPasswords	TRUE if database should contain encrypted passwords. This applies only to ODBC Authentication sites, and determines if password in the database are to be considered one-way hashed passwords or stored as plain-text in the columns (useful if you have external processes manipulating rows in the database).
nIP	Index of the site IP address. A "0" value means "all incoming." A number greater than 0 refers to the index in the IP Address list for all addresses on the machine.
nPort	Port number of the site
bCreateHomeFolder	TRUE means that a home folder for new site users should be created
bPermHomeFolder	TRUE means that new site users should be granted full permissions in their home folders
bAutoStart	TRUE means that the site should start automatically

Adding a Local Site (AddLocalSite)

Use the ICISites interface **AddLocalSite** method to create a new unauthenticated Site and retrieve the Site's ICISite interface.

Signature:

```
HRESULT AddLocalSite(
        [in] BSTR bstrName,
        [in] BSTR bstrRootFolder,
        [in] BSTR bstrFilePath,
        [in] int nIP,
        [in] UINT nPort,
        [in] VARIANT_BOOL bCreateHomeFolder,
        [in] VARIANT_BOOL bPermHomeFolder,
        [in] VARIANT_BOOL bAutoStart,
        [out, retval] ICISite** prop);
```

Parameters	Definition
bstrName	New site name
bstrRootFolder	Site root folder
bstrFilePath	Path to the file that will contain user data
nIP	Index of the site IP address
nPort	Port number of the site
bCreateHomeFolder	TRUE means that a home folder for new site users should be created
bPermHomeFolder	TRUE means that new site users should be granted full permissions in their home folders
bAutoStart	TRUE means that the site should start automatically

Adding an Active Directory Authenticated Site (AddADSite)

Use the ICISites interface **AddADSite** method to add a Site that requires Active Directory authentication to the Server and retrieve the Site's COM interface.

Signature:

```
HRESULT AddADSite(
        [in] BSTR bstrName,
        [in] BSTR bstrRootFolder,
        [in] BSTR bstrDomainContext,
        [in] BSTR bstrAllowGroup,
        [in] int nIP,
        [in] UINT nPort,
        [in] VARIANT_BOOL bCreateHomeFolder,
        [in] VARIANT_BOOL bPermHomeFolder,
        [in] VARIANT_BOOL bAutoStart,
        [out, retval] ICISite** prop);
```

Parameters	Definition
bstrName	New site name
bstrRootFolder	Site root folder
bstrDomainName	Default domain context for Active Directory
bstrAllowGroup	Group name
nIP	Index of the site IP address
nPort	Port number of the site
bCreateHomeFolder	TRUE means that a home folder for new site users should be created
bPermHomeFolder	TRUE means that new site users should be granted full permissions in their home folders
bAutoStart	TRUE means that the site should start automatically

Adding an NT Authenticated Site (AddNTLMSite)

Use the ICISites interface **AddNTLMSite** method to add a Site that requires NTLM authentication to the Server, and retrieve the Site's COM interface.

Signature:

```
HRESULT AddNTLMSite(
        [in] BSTR bstrName,
        [in] BSTR bstrRootFolder,
        [in] BSTR bstrDomainName,
        [in] BSTR bstrAllowGroup,
        [in] int nIP,
        [in] UINT nPort,
        [in] VARIANT_BOOL bCreateHomeFolder,
        [in] VARIANT_BOOL bPermHomeFolder,
        [in] VARIANT_BOOL bAutoStart,
        [out, retval] ICISite** prop);
```

Parameters	Definition
bstrName	New site name
bstrRootFolder	Site root folder
bstrDomainName	NT domain name
bstrAllowGroup	NT group name
nIP	Index of the site IP address
nPort	Port number of the site
bCreateHomeFolder	TRUE means that a home folder for new site users should be created
bPermHomeFolder	TRUE means that new site users should be granted full permissions in their home folders
bAutoStart	TRUE means that the site should start automatically

Adding an ODBC-Authenticated Site (AddODBCSite)

Use the ICISites interface **AddODBCSite** method to add a Site that uses an ODBC database for user authentication to the Server and retrieve the Site's COM interface.

Signature:

```
HRESULT AddODBCSite(
        [in] BSTR bstrName,
        [in] BSTR bstrRootFolder,
        [in] BSTR bstrDSN,
        [in] VARIANT_BOOL bEncryptPasswords,
        [in] int nIP,
        [in] UINT nPort,
        [in] VARIANT_BOOL bCreateHomeFolder,
        [in] VARIANT_BOOL bPermHomeFolder,
        [in] VARIANT_BOOL bAutoStart,
        [out, retval] ICISite** prop);
```

Parameters	Definition
bstrName	New site name
bstrRootFolder	Site root folder
bstrDSN	Database DSN
bEncryptPasswords	TRUE if the database should contain encrypted passwords
nIP	Index of the site IP address
nPort	Port number of the site
bCreateHomeFolder	TRUE means that a home folder for new site users should be created
bPermHomeFolder	TRUE means that new site users should be granted full permissions in their home folders
bAutoStart	TRUE means that the site should start automatically

Adding an LDAP Site (AddLDAPSite)

Use the ICISites interface **AddLDAPSite** method to add a Site that uses LDAP.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT AddLDAPSite(
      [in] BSTR bstrName,
      [in] BSTR bstrRootFolder,
      [in,optional] BSTR bstrDomainContext,
      [in,optional] BSTR bstrAllowGroup,
      [in] int nIP,
      [in] long nPort,
      [in] VARIANT_BOOL bCreateHomeFolder,
      [in] VARIANT_BOOL bPermHomeFolder,
      [in] VARIANT_BOOL bAutoStart,
      [out, retval] ICISite** prop);
```

Parameters	Definition
bstrName	New site name
bstrRootFolder	Site root folder
bstrDomainContext	LDAP domain components, e.g. DNS domain name
bstrAllowGroup	LDAP group name
nIP	Index of the site IP address
nPort	Port number of the site
bCreateHomeFolder	TRUE means that a home folder for new site users should be created
bPermHomeFolder	TRUE means that new site users should be granted full permissions in their home folders
bAutoStart	TRUE means that the site should start automatically

You can also call Add(), setting "3" as the nAMID parameter to create an LDAP site.

Configuration Notes

The "bstrDomainContext" string that is passed in must be composed properly to define the parameters of the LDAP site. The syntax is the same for both methods: a semicolon separated list of name/value pairs that defines the LDAP options.

Example:

```
LDAPDB={path to AUD file};LDAPSERVER={IP or Hostname for LDAP
server};LDAPPORT={port on LDAP server};BASEDN={base dn for
users};USERFILTER={filter for LDAP query to return users};USERATTRIBUTE={what user
attribute to use as client login};TYPE={0 for anonymous bind, 1 for simple
bind};USERNAME={cn of the user to bind as for querying users};PASSWORD={password
for that user};SSL={0 for no SSL, 1 for SSL}
```

Empty values can be blank, but the name must exist, such as for an anonymous bind "USERNAME=;"

Example:

Connect to a server using the LDAP protocol. Note that "USERNAME=" and "PASSWORD=" must be configured appropriately for your login, and an appropriate path for "LDAPDB=" should be specified.

```
LDAPDB=d:\program
files\foostation\eft\LDAP.aud;LDAPSERVER=199.199.99.99;LDAPPORT=389;BASEDN=cn=users
,dc=forest,dc=intranet,dc=fs;USERFILTER=(objectClass=person);USERATTRIBUTE=SamAccou
ntName;TYPE=0;USERNAME=cn=user,cn=users,dc=forest,dc=intranet,dc=fs;PASSWORD=secret
;SSL=0
```

Single-Site Interface Properties (ICISite)

The ICISite interface represents an FTP/HTTP Site. Use this interface to manage individual Sites and to access their attributes. The interface can be obtained by using the **Item** and **SiteByID** methods of the ICISites interface.

Signature:

```
Set Site = Sites.Item(0)
```

Site Options

The properties below are used to retrieve information about or change a Site's settings.

Blocking Site-to-Site FTP (BlockSiteToSite)

Use the ICISite interface **BlockSiteToSite** property to check if site-to-site transfers are blocked, and to allow or prohibit site-to-site transfers.

Signature:

```
HRESULT BlockSiteToSite([out, retval] VARIANT_BOOL *pVal);
HRESULT BlockSiteToSite([in] VARIANT_BOOL newVal);
```

True = Blocked

False = Allowed

Blocking Anti-Timeout Schemes (BlockAntiTimeOut)

Use the ICISite interface **BlockAntiTimeOut** property to check if the site is blocking anti-time-out traffic, or to block or allow anti-time-out traffic from clients.

Signature:

```
HRESULT BlockAntiTimeOut([out, retval] VARIANT_BOOL *pVal);
HRESULT BlockAntiTimeOut([in] VARIANT_BOOL newVal);
```

True = Blocked

False = Allowed

Retrieving and Modifying a List of Banned File Types (VFSFilter)

Use the ICISite interface **VFSFilter** property to retrieve or edit the list of file types banned from the site.

Signature:

```
HRESULT VFSFilter([out, retval] BSTR *pVal);
HRESULT VFSFilter([in] BSTR newVal);
```

Editing the Server Connection Message (ConnectMessage)

Use the ICISite interface **ConnectMessage** property to retrieve or set the connection message sent to clients when users connect to the server.

Signature:

```
HRESULT ConnectMessage([out, retval] BSTR *pVal);
HRESULT ConnectMessage([in] BSTR newVal);
```

Managing a Site's Exit Message (ExitMessage)

Use the ICISite interface **ExitMessage** property to retrieve or edit the exit message shown when users disconnect.

Signature:

```
HRESULT ExitMessage([out, retval] BSTR *pVal);
HRESULT ExitMessage([in] BSTR newVal);
```

Managing the User Limit Message (UserLimitMessage)

Use the ICISite interface **UserLimitMessage** property to retrieve or edit the text of the message shown when the site's user limit is reached.

Signature:

```
HRESULT UserLimitMessage([out, retval] BSTR *pVal);
HRESULT UserLimitMessage([in] BSTR newVal);
```

Determining if a Site is Started (IsStarted)

Use the ICISite interface **IsStarted** property to check if a site is active and available.

Signature:

```
HRESULT IsStarted([out, retval] BOOL *pVal);
```

True = Started

False = Stopped

Retrieving a Site's ID (ID)

Use the ICISite interface **ID** property to retrieve the Site Identifier. This returns a read-only version of the ID.

Signature:

```
HRESULT ID([out, retval] long *pVal);
```

Retrieving a Site's Name (Name)

Use the ICISite interface **Name** property to retrieve the name of the site. This returns a read-only version of the name.

Signature:

```
HRESULT Name([out, retval] BSTR *pVal);
```

Retrieving ARM ODBC Settings (ODBCSettings)

Use the ICISite interface **ODBCSettings** property to manage the database connection settings for an ODBC Authentication Site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
STDMETHOD(get_ODBCSettings)(/*[out, retval]*/ BSTR *pVal);
STDMETHOD(put_ODBCSettings)(/*[in]*/ BSTR newVal);
```

User Options

The properties below are used to retrieve information about or change a Site's user options. For more user settings, see User Settings in Client Settings Interface Methods.

- Setting the Grant Full Permissions Option (AssignFullPermissionsForHomeFolder)
- Setting the Auto Create Home Folder Option (AutoCreateHomeFolder)

Setting the Grant Full Permissions Option (AssignFullPermissionsForHomeFolder)

Use the ICISite interface **AssignFullPermissionsForHomeFolder** property to check if new users will have full permissions in their home folders, and to either grant or deny users full permissions to their home folders.

Signature:

```
HRESULT AssignFullPermissionsForHomeFolder([out, retval] VARIANT_BOOL *pVal);
HRESULT AssignFullPermissionsForHomeFolder([in] VARIANT_BOOL newVal);
```

True = Granted

False = Denied

*The **AutoCreateHomeFolder** property must be set to **TRUE** before you can use this property.*

If you do not grant users full permissions to their home folders, they will inherit permissions from their groups.

Setting the Auto Create Home Folder Option (AutoCreateHomeFolder)

Use the ICISite interface **AutoCreateHomeFolder** property to check if the Server will automatically create a home folder for users, and to either enable or disable the automatic folder creation.

Signature:

```
HRESULT AutoCreateHomeFolder([out, retval] VARIANT_BOOL *pVal);
HRESULT AutoCreateHomeFolder([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Connection Limits

The properties below are used to retrieve information about or change a Site's connection limits.

Enabling or Disabling a Site's Maximum Transfer Speed (HasMaxSpeed)

Use the ICISite interface **HasMaxSpeed** property to determine if the site has a maximum transfer speed limit. Also use this property to enable or disable a transfer speed limit.

Signature:

```
HRESULT HasMaxSpeed([out, retval] VARIANT_BOOL *pVal);
HRESULT HasMaxSpeed([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Retrieving or Setting Maximum Transfer Speed (MaxTransferSpeed)

Use the ICISite interface **MaxTransferSpeed** property to retrieve or set the maximum transfer speed allowed on a site.

Signature:

```
HRESULT MaxTransferSpeed([out, retval] long *pVal);
HRESULT MaxTransferSpeed([in] long newVal);
```

Enabling the Concurrent Connections Limit (HasMaxUsers)

Use the ICISite interface **HasMaxUsers** property to determine if the site has a maximum concurrent users limit. Also use this property to enable or disable the concurrent connections limit.

Signature:

```
HRESULT HasMaxUsers([out, retval] VARIANT_BOOL *pVal);
HRESULT HasMaxUsers([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Retrieving or Setting Maximum Concurrent Connections (MaxConcurrentConnections)

Use the ICISite interface **MaxConcurrentConnections** property to retrieve or set the maximum number of concurrent socket connections allowed on a Site.

Signature:

```
HRESULT MaxConcurrentConnections([out, retval] long *pVal);
HRESULT MaxConcurrentConnections([in] long newVal);
```

Retrieving or Setting Number of Connections per User to a Site (HasMaxConnectionsPerAccount)

Use the ICISite interface **HasMaxConnectionsPerAccount** property to determine if the site limits the number of socket connections per user. Also use this property to enable or disable the user connection limit.

Signature:

```
HRESULT HasMaxConnectionsPerAccount([out, retval] VARIANT_BOOL *pVal);
HRESULT HasMaxConnectionsPerAccount([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Retrieving or Setting a Site's User Connection Limit (MaxConnectionsPerUser)

Use the ICISite interface **MaxConnectionsPerUser** property to retrieve or set the maximum number of concurrent socket connections a user can make to a site.

Signature:

```
HRESULT MaxConnectionsPerUser([out, retval] long *pVal);
HRESULT MaxConnectionsPerUser([in] long newVal);
```

Retrieving Maximum Concurrent Logins (HasMaxConcurrentLogins)

Currently only available in Secure FTP Server.

Use the ICISite interface **HasMaxConcurrentLogins** property to return a BOOLEAN value when read indicating whether the Site has the "Max Concurrent Logins" feature enabled. You can set the value by assigning a BOOLEAN value.

Signature:

```
HRESULT HasMaxConcurrentLogins([out, retval] BOOL *pVal);
HRESULT HasMaxConcurrentLogins([in] BOOL newVal);
```

Setting Maximum Concurrent Logins (MaxConcurrentLogins)

Currently only available in Secure FTP Server.

Use the ICISite interface **MaxConcurrentLogins** property to return a LONG value when read indicating the maximum number of concurrent logins allowed on the Site. You can set this value by assigning a LONG value.

```
HRESULT MaxConcurrentLogins(??);
```

Enabling Allowed Login Attempt Limit (LimitLoginAttempts)

Use the ICISite interface **LimitLoginAttempts** property to view whether a login attempt limit is enabled.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT LimitLoginAttempts([out, retval] VARIANT_BOOL *pVal);
HRESULT LimitLoginAttempts([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Enabling Account Lockout (LockoutNotDisable)

Use the ICISite interface **LockoutNotDisable** property to view whether account lockout is enabled.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT LockoutNotDisable([out, retval] VARIANT_BOOL *pVal);
HRESULT LockoutNotDisable([in] VARIANT_BOOL newVal);
```

True = Lockout is not disabled

False = Lockout is disabled

Viewing the Lockout Period for a Site (LockoutPeriod)

Use the ICISiteSettings Interface **LockoutPeriod** property to view a Site's lockout period.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT LockoutPeriod([in] BSTR bstrLogin);
```

Values are 30, 60, or 90 minutes

Viewing Invalid Attempts Period for a Site (InvalidAttemptsPeriod)

This method is available in EFT Server 5.1.1.and later.

Use the ICISiteSettings Interface **InvalidAttemptsPeriod** property to set/retrieve length of time for "Lock out / disable account after *N* invalid login attempts during time period" option.

Signature

```
HRESULT InvalidAttemptsPeriod([out, retval] long *pVal);
HRESULT InvalidAttemptsPeriod([in] long newVal);
```

Example

```
Site.InvalidAttemptsPeriod = 30
```

Viewing the Number of Invalid Login Attempts Allowed for a Site (MaxInvalidLoginAttempts)

Use the ICISite interface **MaxInvalidLoginAttempts** property to view the maximum number of invalid login attempts allowed to a Site.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT MaxInvalidLoginAttempts([out, retval] long *pVal);
HRESULT MaxInvalidLoginAttempts([in] long newVal);
```

IP Addresses

The properties below are used to retrieve information about or change a Site's IP address settings.

Limiting Connections from the Same IP Address (HasMaxIPPerAccount)

Use the ICISite interface **HasMaxIPPerAccount** property to determine if the site limits the number of concurrent connections from one IP address. Also use this property to enable or disable a limit of concurrent connections from one IP address.

Signature:

```
HRESULT HasMaxIPPerAccount([out, retval] VARIANT_BOOL *pVal);
HRESULT HasMaxIPPerAccount([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Managing Concurrent Connections Allowed from the Same IP Address (MaxConnectionsFromSameIP)

Use the ICISite interface **MaxConnectionsFromSameIP** property to retrieve or set the maximum number of concurrent connections allowed from one IP address to a site.

Signature:

```
HRESULT MaxConnectionsFromSameIP([out, retval] long *pVal);
HRESULT MaxConnectionsFromSameIP([in] long newVal);
```

Limiting Consecutive Invalid Commands (DisconnectOnDOS)

Use the ICISite interface **DisconnectOnDOS** property to determine if the Site will disconnect after a certain number consecutive invalid commands. Also use this property to enable or disable such disconnections.

Signature:

```
HRESULT DisconnectOnDOS([out, retval] VARIANT_BOOL *pVal);
HRESULT DisconnectOnDOS([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Setting the Number of Invalid Commands Allowed (DOSLimit)

Use the ICISite interface **DOSLimit** property to retrieve or set the number of consecutive invalid commands that will cause the Site to disconnect a user.

Signature:

```
HRESULT DOSLimit([out, retval] long *pVal);
HRESULT DOSLimit([in] long newVal);
```

Banning Connections from Specific IP Addresses (BanIPOnDOS)

Use the ICISite interface **BanIPOnDOS** property to determine if the server will automatically ban connections from an IP address that has sent too many consecutive invalid commands. Use the same property to enable or disable automatic bans.

Signature:

```
HRESULT BanIPOnDOS([out, retval] VARIANT_BOOL *pVal);
HRESULT BanIPOnDOS([in] VARIANT_BOOL newVal);
```

True = Enabled

False = Disabled

Banning IP Addresses (AutoBanIPsPermanently)

Use the ICISite interface **AutoBanIPsPermanently** property to determine if the server will permanently or temporarily ban connections from an IP address that has sent too many consecutive invalid commands.

Signature:

```
HRESULT AutoBanIPsPermanently([out, retval] VARIANT_BOOL *pVal);
HRESULT AutoBanIPsPermanently([in] VARIANT_BOOL newVal);
```

True = Permanently

False = Temporarily

Enabling a Range of Ports for PASV Connections (EnablePortRange)

Use the ICISite interface **EnablePortRange** property to determine if a specific IP address and port ranges have been assigned for PASV connections to the Site. Use the same property to enable or disable the use of a specific IP address and port ranges.

Signature:

```
HRESULT EnablePortRange([out, retval] VARIANT_BOOL *pVal);
HRESULT EnablePortRange([in] VARIANT_BOOL newVal);
```

Determining the IP Address for PASV Connections (PASVListenIP)

Use the ICISite interface **PASVListenIP** property to determine or set the IP address where the Site listens for PASV mode connections.

Signature:

```
HRESULT PASVListenIP([out, retval] long *pVal);
HRESULT PASVListenIP([in] long newVal);
```

Determining the Low End of a PASV Mode Port Range (PASVPortMin)

Use the ICISite interface **PASVPortMin** property to determine or set the lowest port in a range where a Site listens for PASV connections.

Signature:

```
HRESULT PASVPortMin([out, retval] long *pVal);
HRESULT PASVPortMin([in] long newVal);
```

Setting IP Addresses that have Access to a Site (IPAccessAllowedDefault)

Use the ICISite interface **IPAccessAllowedDefault** property to determine which IP addresses are denied or allowed by default. Use the same property to deny or allow a range of IP addresses based on masks you choose with the **AddIPAccessRule** method.

Signature:

```
HRESULT IPAccessAllowedDefault([out, retval] VARIANT_BOOL *pVal);
HRESULT IPAccessAllowedDefault([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Denied

Determining the High End of a PASV Mode Port Range (PASVPortMax)

Use the ICISite interface **PASVPortMax** property to determine or set the highest port in a range where a Site listens for PASV connections.

Signature:

```
HRESULT PASVPortMax([out, retval] long *pVal);
HRESULT PASVPortMax([in] long newVal);
```

Single-Site Interface Methods (ICISite)

The ICISite interface represents an FTP/HTTP Site. Use this interface to manage individual Sites and to access their attributes.

The interface can be obtained by using the **Item** and **SiteByID** methods of the ICISites interface.

Signature:

```
Set Site = Sites.Item(0)
```

Starting a Site (Start)

Use the ICISite interface **Start** method to start an individual site (start listening for incoming connections).

Signature:

```
HRESULT Start();
```

Stopping Site (Stop)

Use the ICISite interface **Stop** method to stop an individual Site from listening for incoming connections.

Signature:

```
HRESULT Stop();
```

Deleting a Site (Remove)

Use the ICISite interface **Remove** method to delete a Site from the Server. After a Site is removed with the COM interface, the object is no longer valid.

Signature:

```
HRESULT Remove();
```

Cancelling an HTTPS Transfer (CancelTransfer)

Use the ICISite interface **CancelTransfer** method to close all HTTP sockets that upload/download (including POST upload) a file specified by the path parameter. Note that this is not effective for FTP or SFTP.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT CancelTransfer(
          [in] BSTR bstrPath,
          [out, retval] VARIANT_BOOL *pSuccess);
```

Retrieving a List of Event Rules (EventRules)

This method is available in EFT Server 5.2 and later.

Use the **ICISite** interface **EventRules** method to retrieve the list of Event Rules of a certain type (e.g., all Timer Event Rules).

Signature

```
HRESULT EventRules([in] EventType type, [out, retval] IDispatch** ppdispRules);
```

Example

```
Set timerRules = site.EventRules (&H1001) 'Retrieve all the Timer event rules
```

Site Information

The methods below are used to retrieve information about or change a Site's settings.

- Retrieving the Number of Connected Users (GetConnectedCount)
- Identifying the Authentication Manager for a Site (GetAuthManagerID)
- Retrieving a Site's Root Folder (GetRootFolder)
- Specifying the Site's Root Folder (SetRootFolder)
- Retrieving a Site's IP Address (GetIP)
- Specifying a Site's IP Address (SetIP)
- Retrieving a Site's Port Number (GetPort)
- Specifying a Site's Port Number (SetPort)
- Retrieving a Site's Download Speed (GetDownloadSpeed)
- Retrieving a Site's Upload Speed (GetUploadSpeed)
- Retrieving the Number of Active Downloads (GetDownloadCount)
- Retrieving the Number of Active Uploads (GetUploadCount)
- Retrieving the Site Start Time (GetStartTime)
- Editing the Server Connection Message (ConnectMessage)

Retrieving the Number of Connected Users (GetConnectedCount)

Use the ICISite interface **GetConnectedCount** method to retrieve the number of clients currently connected to the server through any protocol engine.

Signature:

```
HRESULT GetConnectedCount([out, retval] long* prop);
```

Identifying the Authentication Manager for a Site (GetAuthManagerID)

Use the ICISite Interface **GetAuthManagerID** method to identify or retrieve the authentication manager ID of a site. The authentication manager ID is the type of authentication used (GS Auth, NT/AD, or ODBC).

Signature:

```
HRESULT GetAuthManagerID([out, retval] long *prop);
```

Retrieving a Site's Root Folder (GetRootFolder)

Use the ICISite interface **GetRootFolder** method to retrieve the root folder for an individual site.

Signature:

```
HRESULT GetRootFolder([out, retval] BSTR *prop);
```

Specifying the Site's Root Folder (SetRootFolder)

Use the ICISite interface **SetRootFolder** method to designate the root folder for an individual site. If the site has to be restarted for the change to take effect, the **SetRootFolder** method will return **TRUE**.

Signature:

```
HRESULT SetRootFolder(
          [in] BSTR newVal,
          [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving a Site's IP Address (GetIP)

Use the ICISite interface **GetIP** method to retrieve the IP address where a site listens for incoming connections.

Signature:

```
HRESULT GetIP([out, retval] long *prop);
```

Specifying a Site's IP Address (SetIP)

Use the ICISite interface **SetIP** method to set the IP address on which a Site listens for incoming connections. If the Site must be restarted after changing IP address, the **SetIP** method returns **TRUE**.

Signature:

```
HRESULT SetIP(
          [in] long newVal,
          [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving a Site's Port Number (GetPort)

Use the ICISite interface **GetPort** method to retrieve the port number a site uses for connections (default for FTP connections).

Signature:

```
HRESULT GetPORT([out, retval] UINT *prop);
```

Specifying a Site's Port Number (SetPort)

Use the ICISite interface **SetPort** method to set a Site's default Port used for FTP connections. If the Site must be restarted after changing the port, the **SetPort** method returns **TRUE**.

Signature:

```
HRESULT SetPort(
          [in] UINT newVal,
          [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving a Site's Download Speed (GetDownloadSpeed)

Use the ICISite interface **GetDownloadSpeed** method to retrieve the download speed of a Site.

Signature:

```
HRESULT GetDownloadSpeed([out, retval] ULONG *pSpeed);
```

Retrieving a Site's Upload Speed (GetUploadSpeed)

Use the ICISite interface **GetUploadSpeed** method to retrieve the upload speed of a site.

Signature:

```
HRESULT GetUploadSpeed([out, retval] ULONG *pSpeed);
```

Retrieving the Number of Active Downloads (GetDownloadCount)

Use the ICISite interface **GetDownloadCount** method to retrieve the number of current active downloads.

Signature:

```
HRESULT GetDownloadCount([out, retval] UINT *pCnt);
```

Retrieving the Number of Active Uploads (GetUploadCount)

Use the ICISite interface **GetUploadCount** method to retrieve the number of current active uploads.

Signature:

```
HRESULT GetUploadCount([out, retval] UINT *pCnt);
```

Retrieving the Site Start Time (GetStartTime)

Use the ICISite interface **GetStartTime** method to retrieve the time the Site was started.

Signature:

```
HRESULT GetStartTime([out, retval] BSTR *bstrStartTime);
```

Editing the Server Connection Message (ConnectMessage)

Use the ICISite interface **ConnectMessage** property to retrieve or set the connection message sent to clients when users connect to the server.

Signature:

```
HRESULT ConnectMessage([out, retval] BSTR *pVal);
HRESULT ConnectMessage([in] BSTR newVal);
```

Site Protocol Settings

The methods below are used to retrieve information about or change a Site's protocol settings.

Verifying FTP is Enabled for a Site

Use the ICISite interface **GetFTPAccess** method to verify that FTP access is enabled for a site.

Signature:

```
HRESULT GetFTPAccess([out, retval] VARIANT_BOOL *prop);
```

Value	FTP Access
True	Access enabled
False	Access disabled

Enabling FTP Access to a Site (SetFTPAccess)

Use the ICISite interface **SetFTPAccess** method to enable or disable FTP access for a Site. If the site must be restarted for the change to take effect, the **SetFTPAccess** method returns **TRUE**.

Signature:

```
HRESULT SetFTPAccess(
            [in] VARIANT_BOOL newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Verifying if HTTP is Enabled for a Site (GetHTTPAccess)

Use the ICISite interface **GetHTTPAccess** method to verify whether HTTP access is enabled for a site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetHTTPAccess([out, retval] VARIANT_BOOL *prop);
```

True = Access enabled

False = Access disabled

Enabling HTTP Access (SetHTTPAccess)

Use the ICISite interface **SetHTTPAccess** method to configure the HTTP engine ON or OFF for this site based on the parameter you pass in.

This method is available in EFT Server 4.3.4 and later.

Retrieving the HTTP Port for a Site (GetHTTPPort)

Use the ICISite interface **GetHTTPPort** method to retrieve the HTTP port number on which the HTTP engine is listening for a Site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetHTTPPort([out, retval] VARIANT_BOOL *prop);
```

Specifying the HTTP Port (SetHTTPPort)

Use the ICISite interface **SetHTTPPort** method to configure the HTTP port for this Site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT SetHTTPPort(long newVal, [returns] VARIANT_BOOL bNeedRestart);
```

Verifying if HTTPS is Enabled for a Site (GetHTTPSAccess)

Use the ICISite interface **GetHTTPSAccess** method to verify whether HTTPs access is enabled for a site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetHTTPSAccess([out, retval] VARIANT_BOOL *prop);
```

True = Access enabled

False = Access disabled

Enabling HTTPS Access (SetHTTPSAccess)

Use the ICISite interface **SetHTTPSAccess** method to configure the HTTPS engine ON or OFF for this site based on the parameter you pass in.

This method is available in EFT Server 4.3.4 and later.

Retrieving the HTTPS Port for a Site (GetHTTPSPort)

Use the ICISite interface **GetHTTPSPort** method to retrieve the HTTPS port number on which the HTTPS engine is listening for a Site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetHTTPSPort([out, retval] VARIANT_BOOL *prop);
```

Specifying the HTTPS Port (SetHTTPSPort)

Use the ICISite interface **SetHTTPSPort** method to configure the HTTPS Port for this site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT SetHTTPSPort(long newVal, [returns] VARIANT_BOOL bNeedRestart);
```

Verifying if Explicit SSL is Enabled for a Site (GetSSLAuth)

Use the ICISite interface **GetSSLAuth** method to verify that explicit SSL is enabled for a site.

Signature:

```
HRESULT GetSSLAuth([out, retval] VARIANT_BOOL *prop);
```

True = Enabled

False = Disabled

Allowing Explicit SSL Access to a Site (SetSSLAuth)

Use the ICISite interface **SetSSLAuth** method to enable or disable explicit SSL for a site. If the site must be restarted for the change to take effect, the **SetSSLAuth** method returns **TRUE**.

Signature:

```
HRESULT SetSSLAuth(
            [in] VARIANT_BOOL newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Verifying if Implicit SSL is Enabled for a Site (GetSSLImp)

Use the ICISite interface **GetSSLImp** method to verify that implicit SSL is enabled for a site.

Signature:

```
HRESULT GetSSLAuth([out, retval] VARIANT_BOOL *prop);
```

True = Enabled

False = Disabled

Allowing Implicit SSL Access to a Site (SetSSLImp)

Use the ICISite interface **SetSSLImp** method to enable or disable implicit SSL for a site. If the site must be restarted for the change to take effect, the **SetSSLImp** method returns **TRUE**.

Signature:

```
HRESULT SetSSLImp(
            [in] VARIANT_BOOL newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving the SFTP (SSH) Certificate File Path (GetSSHKeyFilePath)

User theICISite interface **GetSSHKeyFilePath** method to retrieve the SSH Certificate file path.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetSSHKeyFilePath([out, retval] VARIANT_BOOL *prop);
```

Specifying the SFTP (SSH) Certificate File Path (SetSSHKeyFilePath)

Use the ICISite interface **SetSSHKeyFilePath** method to change the SSH Certificate file path. If the site must be restarted for the change to take effect, **SetSSHKeyFilePath** method returns TRUE.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT SetSSHKeyFilePath(
[in] VARIANT_BOOL newVal,
[out, retval] VARIANT_BOOL *pNeedRestart);
```

Example:

```
Dim bNeedRestart: bNeedRestart = oSite.SetSSHKeyFilePath("C:\Keys\Key.pvk")
If bNeedRestart Then
      Call oSite.Stop()
      Call oSite.Start()
End If
```

Site Certificates and Keys

The methods below are used to retrieve information about or change a Site's certificates and keys.

Verifying if Client Certificates are Required (GetCheckClientCert)

Use the ICISite interface **GetCheckClientCert** method to determine if the server requires certificates from users connecting via implicit SSL. If the server requires certificates, the **GetCheckClientCert** method returns a value of **TRUE**.

Signature:

```
HRESULT GetCheckClientCert([out, retval] VARIANT_BOOL *prop);
```

True = Required

False = Not required

Requiring User Certificates on Implicit SSL Sites (SetCheckClientCert)

Use the ICISite interface **SetCheckClientCert** method to enable or disable the requirement of certificates on implicit SSL sites. If you must restart the server for the change to take effect the **SetCheckClientCert** method returns a value of **TRUE**.

Signature:

```
HRESULT SetCheckClientCert(
            [in] VARIANT_BOOL newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving the File Path for Certificates (GetCertFilePath)

Use the ICISite interface **GetCertFilePath** method to retrieve the file path for SSL certificates.

Signature:

```
HRESULT GetCertFilePath([out, retval] BSTR *prop);
```

Changing the Certificate File Path (SetCertFilePath)

Use the ICISite interface **SetCertFilePath** method to change the file path to server certificates.

Signature:

```
HRESULT SetCertFilePath(
            [in] BSTR newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving the File Path for Private Keys (GetKeyFilePath)

Use the ICISite interface **GetKeyFilePath** method to retrieve the file path for private keys.

Signature:

```
HRESULT GetKeyFilePath([out, retval] BSTR *prop);
```

Changing the Private Key File Path (SetKeyFilePath)

Use the ICISite interface **SetKeyFilePath** method to change the file path to private keys. If you must restart the server for the changes to take effect, the **SetKeyFilePath** method returns a value of **TRUE**.

Signature:

```
HRESULT SetKeyFilePath(
            [in] BSTR newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving Private Key Passphrases (GetPassPhrase)

Use the ICISite interface **GetPassPhrase** method to retrieve private key passphrases.

Signature:

```
HRESULT GetPassPhrase([out, retval] BSTR *prop);
```

Changing a Private Key Passphrase (SetPassPhrase)

Use the ICISite interface **SetPassPhrase** method to change a private key pass phrase. If you must restart the server for the changes to take effect, the **SetPassPhrase** method returns a value of **TRUE**.

Signature:

```
HRESULT SetPassPhrase(
            [in] BSTR newVal,
            [out, retval] VARIANT_BOOL *pNeedRestart);
```

Retrieving the SFTP (SSH) Certificate File Path (GetSSHKeyFilePath)

User theICISite interface **GetSSHKeyFilePath** method to retrieve the SSH Certificate file path.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetSSHKeyFilePath([out, retval] VARIANT_BOOL *prop);
```

Specifying the SFTP (SSH) Certificate File Path (SetSSHKeyFilePath)

Use the ICISite interface **SetSSHKeyFilePath** method to change the SSH Certificate file path. If the site must be restarted for the change to take effect, **SetSSHKeyFilePath** method returns TRUE.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT SetSSHKeyFilePath(
[in] VARIANT_BOOL newVal,
[out, retval] VARIANT_BOOL *pNeedRestart);
```

Example:

```
Dim bNeedRestart: bNeedRestart = oSite.SetSSHKeyFilePath("C:\Keys\Key.pvk")
If bNeedRestart Then
      Call oSite.Stop()
      Call oSite.Start()
End If
```

Retrieving the Path to Trusted Certificates (GetTrustedCertsPath)

Use the ICISite interface **GetTrustedCertsPath** method to retrieve the file path to the server's trusted certificates.

Signature:

```
HRESULT GetTrustedCertsPath([out, retval] BSTR *prop);
```

Retrieving the Path to Pending Certificates (GetAwaitingCertsPath)

Use the ICISite interface **GetAwaitingCertsPath** method to retrieve the file path to the server's trusted certificates.

Signature:

```
HRESULT GetAwaitingCertsPath([out, retval] BSTR *prop);
```

Retrieving Information for a Trusted Certificate (GetTrustedCertificateInfo)

Use the ICISite interface **GetTrustedCertificateInfo** method and a trusted certificate's ID to retrieve the trusted certificate's information from the server. This method will open the ICICertInfo interface.

Signature:

```
HRESULT GetTrustedCertificateInfo(
                [in] long lID,
                [out, retval] ICICertInfo **prop);
```

Retrieving a List of Trusted Certificates (GetTrustedCertificates)

Use the ICISite interface **GetTrustedCertificates** method to retrieve a list of trusted certificates.

Signature:

```
HRESULT GetTrustedCertificates([out, retval] VARIANT *aCerts);
```

Retrieving a List of Pending Certificates (GetPendingCertificates)

Use the ICISite interface **GetPendingCertificates** method to retrieve a list of untrusted, pending certificates.

Signature:

```
HRESULT GetPendingCertificates([out, retval] VARIANT *aCerts);
```

Retrieving Information for a Pending Certificate (GetPendingCertificateInfo)

Use the ICISite interface **GetPendingCertificateInfo** method and a pending certificate's ID to retrieve the pending certificate's information from the server. This method will open the ICICertInfo interface.

Signature:

```
HRESULT GetPendingCertificateInfo(
                [in] long lID,
                [out, retval] ICICertInfo **prop);
```

Retrieving a Certificate's Data String (ExportTrustedCertificate)

Use the ICISite interface **ExportTrustedCertificate** method and a trusted certificate's ID to retrieve the actual certificate data string from the server.

Signature:

```
HRESULT ExportTrustedCertificate(
                [in] long lID,
                [out, retval] BSTR *bstrCertData);
```

Saving a Trusted Certificate to a File (ImportTrustedCertificate)

Use the ICISite interface **ImportTrustedCertificate** method, to save a trusted certificate to a file on the server.

Signature:

```
HRESULT ImportTrustedCertificate([in] BSTR bstrCertPath);
```

Deleting a Trusted Certificate (RemoveTrustedCertificate)

Use the ICISite interface **RemoveTrustedCertificate** method to delete a trusted certificate from the server.

Signature:

```
HRESULT RemoveTrustedCertificate([in] long lID);
```

Deleting a Pending Certificate (RemovedPendingCertificate)

Use the ICISite interface **RemovePendingCertificate** method to delete a pending certificate from the server.

Signature:

```
HRESULT RemovePendingCertificate([in] long lID);
```

Adding a Pending Certificate to the Trusted List (AddCerficateToTrusted)

Use the ICISite interface **AddCertificateToTrusted** method to move a pending certificate to the list of trusted certificates.

Signature:

```
HRESULT AddCertificateToTrusted([in] long lID);
```

Site Folders

The methods below are used to manage a Site's folders.

Retrieving a List of Folders (GetFolderList)

Use the ICISite interface **GetFolderList** method and a folder name to retrieve a list of all the folder's physical and virtual subfolders. Each subfolder will be followed by a carriage return (**0D0A**).

Signature:

```
HRESULT GetFolderList(
             [in] BSTR bstrFolderAlias,
             [out, retval] BSTR *prop);
```

If a listed subfolder also has folders within it, the subfolder name will have a double-quote mark at the end. If you need to use the subfolder name in your script or program, parse out the double quote mark.

Retrieving a List of Folder Permissions (GetFolderPermissions)

Use the ICISite interface **GetFolderPermissions** method to retrieve a list of all folder permissions, including inherited permissions.

Signature:

```
HRESULT GetFolderPermissions(
[in] BSTR bstrFolder,
[out, retval] VARIANT *aPermissions);
```

Example:

```
Dim arPermissions: arPermissions = oSite.GetFolderPermissions("/usr/test")
For iCount = LBound(arPermissions) To UBound(arPermissions)
      Set oPermission = arPermissions(iCount)
      WScript.Echo oPermission.Folder & " - " & oPermission.Client
Next
```

Setting Folder Permissions (SetPermission)

Use the ICISite interface **SetPermission** method to add folder permissions or to change existing permissions.

Signature:

```
HRESULT SetPermission (
                     [in]IPermission *pPermission,
                     [in, optional] VARIANT_BOOL bRemoveOtherPermissions);
```

Optional Parameter

If you want to set permissions on a folder for a user and remove all other permissions, pass TRUE to the second parameter of setPermission(). This will REMOVE all other permissions from a folder (that is, break the inheritance).

For example,

```
set perm = Site.GetBlankPermission( "/usr/foo", "foo" )
perm.FileDelete = True
perm.FileUpload = True
perm.FileDownload = False
Site.SetPermission( perm, True )
```

Removing Folder Permissions (RemovePermission)

Use the ICISite interface **RemovePermission** method, a folder name, and a client name to delete permissions from folders.

Signature:

```
HRESULT RemovePermission(
                [in] BSTR bstrFolder,
                [in] BSTR bstrClient);
```

Creating Blank Permissions (GetBlankPermission)

Use the ICISite interface **GetBlankPermission** method, a folder name, and a client name to create a new set of permissions.

Signature:

```
HRESULT GetBlankPermission(
                [in] BSTR bstrFolder,
                [in] BSTR bstrClient,
                [out, retval] IPermission **pPermission);
```

*After you have defined the permissions you must use the **SetPermisson** method to apply the changes you have made.*

Creating a Physical Folder (CreatePhysicalFolder)

Use the ICISite interface **CreatePhysicalFolder** method and a relative path to create a new physical folder on the server.

Signature:

```
HRESULT CreatePhysicalFolder([in] BSTR bstrNewFolder);
```

Use forward slashes instead of back slashes.

Example in VB:

```
Call Object.CreatePhysicalFolder("/folderpath/")
'another example:
MySite.CreatePhysicalFolder("/Usr/jsmith/")
'Note that backslashes "\" will not work:
MySite.CreatePhysicalFolder("\Usr\jsmith\") <--will fail!
```

Creating a Virtual Folder (CreateVirtualFolder)

Use the ICISite interface **CreateVirtualFolder** method with relative path and a physical target path to create a new virtual folder on a site.

Signature:

```
HRESULT CreateVirtualFolder(/*[in]*/ BSTR bstrNewFolder,
/*[in]*/ BSTR bstrTarget );
```

Example:

```
Call oSite.CreateVirtualFolder("/pub/temp/", "C:\Temp\")
```

Deleting a Folder (RemoveFolder)

Use the ICISite interface **RemoveFolder** method to delete a folder from a site.

Signature:

```
HRESULT RemoveFolder([in] BSTR bstrFolder);
```

Renaming a Folder (RenameFolder)

Use the ICISite interface **RenameFolder** method to change a folder's name.

Signature:

```
HRESULT RenameFolder(
             [in] BSTR bstrSrcFolder,
             [in] BSTR bstrDstFolder);
```

Remapping a Virtual Folder Path (RemapVirtualFolder)

Use the ICISite interface **RemapVirtualFolder** method to update an existing virtual folder path to point to a new physical folder.

This method is available in EFT Server 4.3.4 and later.

Examples:

```
var bResult = oSite.RemapVirtualFolder( "/usr/name", \\\\filer01\\home\\name")
var bResult = oSite.RemapVirtualFolder "/MyVirtualFolder", "c:\testfolder"
```

Site Permission Groups

The methods below are used to manage a Site's Permission Groups.

Creating a Permissions Group (CreatePermissionGroup)

Use the ICISite interface **CreatePermissionGroup** method to add a new permission group to a site.

Signature:

```
HRESULT CreatePermissionGroup([in] BSTR bstrName);
```

Permission Groups control user access to files and folders. Settings Levels control user access (bandwidth resources, banned file times, disk quota, etc.) to the server's resources.

Retrieving a List of Permission Groups on a Site (GetPermissionGroups)

Use the ICISite interface **GetPermissionGroups** method to retrieve a list of permission groups on a site.

Signature:

```
HRESULT GetPermissionGroups([out, retval] VARIANT *aGroups);
```

Permission Groups control user access to files and folders. Settings Levels control user access (bandwidth resources, banned file times, disk quota, etc.) to the server's resources.

Retrieving a List of Users of Specified Permission Groups (GetPermissionGroupList)

Use the ICISite interface **GetPermissionGroupList** method to retrieve a list of users of specified permission groups.

Signature:

```
HRESULT GetPermissionGroupList(
/*[in]*/ BSTR bstrGroup,
/*[out, retval]*/ VARIANT *aUsers);
```

Example:

```
Dim arUsers: arUsers = oSite.GetPermissionGroupList("Administrative")
For iCount = LBound(arUsers) To UBound(arUsers)
     WScript.Echo arUsers(iCount)
Next
```

Deleting a Permission Group (RemovePermissionGroup)

Use the ICISite interface **RemovePermissionGroup** method to delete a permission group from a site.

Signature:

```
HRESULT RemovePermissionGroup([in] BSTR bstrName);
```

Renaming a Permission Group (RenamePermissionGroup)

Use the ICISite interface **RenamePermissionGroup** method to change the name of a permission group.

Signature:

```
HRESULT RenamePermissionGroup(
                    [in] BSTR bstrOldName,
                    [in] BSTR bstrNewName);
```

Adding a User to a Permission Group (AddUserToPermissionGroup)

Use the ICISite interface **AddUserToPermissionGroup** method to add a user to a permission group.

Signature:

```
HRESULT AddUserToPermissionGroup(
                    [in] BSTR bstrUser,
                    [in] BSTR bstrGroup);
```

Deleting a User from a Permission Group (RemoveUserFromPermissionGroup)

Use the ICISite interface **RemoveUserFromPermissionGroup** method to delete a user from a permission group.

Signature:

```
HRESULT RemoveUserFromPermissionGroup(
                    [in] BSTR bstrUser,
                    [in] BSTR bstrGroup);
```

Retrieving a List of a User's Permission Groups (GetPermissionGroupsOfUser)

Use the ICISite interface **GetPermissionGroupsOfUser** method to list all of a user's permission groups.

Signature:

```
HRESULT GetPermissionGroupsOfUser(
                [in] BSTR bstrUser,
                [out, retval] VARIANT *aGroups);
```

Site Users and Settings Levels

The methods below are used to manage a Site's Users and Settings Levels.

Creating a User (CreateUser and CreateUserEx)

Use the ICISite interface **CreateUser** method or the **CreateUserEx** method to create a new user on a site. The **CreateUser** method creates users with the parameters marked in bold text in the **Parameters** table set at default values. The **CreateUserEx** method allows you to set values for all parameters.

Signature:

```
HRESULT CreateUser(
                [in] BSTR bstrLogin,
                [in] BSTR bstrPwd,
                [in] int nPwdType,
                [in] BSTR bstrDescription);

HRESULT CreateUserEx(
                [in] BSTR bstrLogin,
                [in] BSTR bstrPwd,
                [in] int nPwdType,
                [in] BSTR bstrDescription,
                [in] BSTR bstrFullName,
                [in] VARIANT_BOOL bCreateHomeFolder,
                [in] VARIANT_BOOL bFullPermissionsForHomeFolder,
                [in, optional] BSTR bstrSettingsLevel);
```

*The **CreateUser[Ex]** method will fail if a simple password is provided and enforce complex passwords is enabled in EFT Server. For example, the following will fail if enforce complex passwords is enabled in EFT Server:*

```
oSite.CreateUser "tuser1", "testpassword", 0 "New User"
```

For the method to work, you must do one of the following:

- *Disable enforcement of complex passwords.*
- *Provide a complex password. For example:*

  ```
  oSite.CreateUser "test", "$1$Bn3YdWKv$.BS3Qb2UQTV4SmV1JNN.w/", o, "New User"
  ```

- *Call the **CreateComplexPassword** method and use the resulting string in the CreateUser[Ex] method. For example (in pseudo code):*

  ```
  strUserPass = oUserSettings.CreateComplexPassword()
  oSite.CreateUser "test", strUserPass, 0, "New User"
  ```

Parameters Table

Parameter	Input
bstrLogin	User name
bstrPwd	User password
nPwdType	Password type (see Password Table)
bstrDescription	Description of user
bstrFullName	User's full name
bCreateHomeFolder	TRUE if user home folder should be created and assigned
bFullPermissionsForHomeFolder	TRUE if the user should have administrator rights for their home folder
bstrSettingsLevel	User's settings level name or ID.

Password Table

Value	Password type
0	Regular (hashed)
1	Anonymous (regular with empty password)
2	Anonymous (require email as password)
3	OTP-MD4
4	OTP-MD5
5	Literal (no hash) - For use when importing shadow passwords from a *nix password file. The format must follow: `$1$*$*` For example, user: test, password: test would be: site.ChangeUserPassword "test", "1Bn3YdWKv$.BS3Qb2UQTV4SmV1JNN.w/", 5

DES-format passwords are not supported. These are password formats without the salt defined, that is, without the 1.

Retrieving a List of Users (GetUsers)

Use the ICISite interface **GetUsers** method to retrieve a list of users on a site.

Signature:

```
HRESULT GetUsers([out, retval] VARIANT *aUsers);
```

Deleting a User (RemoveUser)

Use the ICISite interface **RemoveUser** method to delete a user from a site.

Signature:

```
HRESULT RemoveUser([in] BSTR bstrLogin);
```

Renaming a User (RenameUser)

Use the ICISite interface **RenameUser** method to change a user's name.

Signature:

```
HRESULT RenameUser(
                [in] BSTR bstrOldName,
                [in] BSTR bstrNewName);
```

Retrieving a List of User Settings Levels (GetSettingsLevels)

Use the ICISite interface **GetSettingsLevels** method to retrieve a list of Settings Levels on a site.

Signature:

```
HRESULT GetSettingsLevels([out, retval] VARIANT *aGroups);
```

Settings Levels control user access (bandwidth resources, banned file times, disk quota, etc.) to the server's resources. Permission Groups control user access to files and folders.

Retrieving a List of Users in a User Settings Level (GetSettingsLevelUsers)

Use the ICISite interface **GetSettingsLevelUsers** method to retrieve a list of users assigned to a Settings Level.

Signature:

```
HRESULT GetSettingsLevelUsers(
                [in] BSTR bstrGroup,
                [out, retval] VARIANT *aUsers);
```

Settings Levels control user access (bandwidth resources, banned file times, disk quota, etc.) to the server's resources. Permission Groups control user access to files and folders.

Creating a User Settings Level (CreateSettingsLevel)

Use the ICISite interface **CreateSettingsLevel** method to add a new Settings Level to a Site.

Signature:

```
HRESULT CreateSettingsLevel(
                [in] BSTR bstrName,
                [in] BSTR bstrDescription);
```

Settings Levels control user access (bandwidth resources, banned file times, disk quota, etc.) to the server's resources. Permission Groups control user access to files and folders.

Deleting a User Settings Level (RemoveSettingsLevel)

Use the ICISite interface **RemoveSettingsLevel** method to delete a Settings Level from a site.

Signature:

```
HRESULT RemoveSettingsLevel([in] BSTR bstrName);
```

Renaming a Settings Level (RenameSettingsLevel)

Use the ICISite interface **RenameSettingsLevel** method to rename a Settings Level.

Signature:

```
HRESULT RenameSettingsLevel(
                [in] BSTR bstrOldName,
                [in] BSTR bstrNewName);
```

Moving a User to a User Settings Level (MoveUserToSettingsLevel)

Use the ICISite interface **MoveUserToSettingsLevel** method to move a user to a new Settings Level.

Signature:

```
HRESULT MoveUserToSettingsLevel(
                [in] BSTR bstrUser,
                [in] BSTR bstrGroup);
```

Changing a User's Password (ChangeUserPassword)

Use the ICISite interface **ChangeUserPassword** method to change a user's password and password type.

Signature:

```
HRESULT ChangeUserPassword([in] BSTR bstrUser,
                [in] BSTR bstrPwd,
                [in] int nPwdType);
```

Password Table

Value	Password type
0	Regular (hashed)
1	Anonymous (regular with empty password)
2	Anonymous (require email as password)
3	OTP-MD4
4	OTP-MD5
5	Literal (no hash) - For use when importing shadow passwords from a *nix password file. The format must follow: `$1$*$*` For example, user: test, password: test would be: `site.ChangeUserPassword "test", "$1$Bn3YdWKv$.BS3Qb2UQTV4SmV1JNN.w/", 5`

DES-format passwords are not supported. These are password formats without the salt defined (that is, without the 1).

Validating a User's Password (ValidatePassword)

Use the ICISite interface **ValidatePassword** method to validate a user's password.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT ValidatePassword(
[in] BSTR bstrUser,
[in] BSTR bstrPwd,
[out, retval] VARIANT_BOOL *bValid);
```

Example:

```
If oSite.ValidatePassword( "test", "pass") Then
     WScript.Echo "Valid password"
End If
```

Creating a Complex Password for a User (CreateComplexPassword)

Use the ICISite interface **CreateComplexPassword** method to generate a complex password that meets the complexity criteria specified for the site.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT CreateComplexPassword([in, optional] BSTR bstrUserName, [out, retval], BSTR
*bstrPassword);
```

Example:

The following example generates a password that adheres to the complexity policy enforced for an existing user:

```
Dim strNewPassword
strNewPassword = oUserSettings.CreateComplexPassword("test")
WScript.Echo "New Password: " strNewPassword
Call oUserSettings.ChangeUserPassword("test", strNewPassword, 1)
```

This example illustrates how to obtain a password from the Server that meets the complexity requirements for the Setting Level that is configured as the default for the site, which is useful when creating new users:

```
Dim strNewPassword
strNewPassword = oUserSettings.CreateComplexPassword()
WScript.Echo "New Password: " strNewPassword
Call oUserSettings.CreateUser("test", strNewPassword, 0, "New User Description")
```

Retrieving a List of Connected Users (GetConnectedUsers)

Use the ICISite interface **GetConnectedUsers** method to retrieve the a list of users currently connected to a site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetConnectedUsers([out, retval] VARIANT *user);
```

Example:

```
dim arConnectedUsers: oSite.GetConnectedUsers()
For iCount = LBound(arConnectedUsers) To UBound(arConnectedUsers)
     WScript.Echo arConnectedUsers(iCount)
Next
```

Retrieving a List of Settings in a User Settings Level (GetSettingsLevelSettings)

Use the ICISite interface **GetSettingsLevelSettings** method to retrieve a list of settings in a Settings Level.

Signature:

```
HRESULT GetSettingsLevelSettings(
                [in] BSTR bstrGroup,
                [out, retval] ICIClientSettings **prop);
```

Settings Levels control user access to the Server. Permission Groups control user access to files and folders.

Retrieving a User's Settings (GetUserSettings)

Use the ICISite interface **GetUserSettings** method to retrieve a list of a user's settings.

In versions 5.2.5 and later, GetUserSettings is case insensitive. That is, in many non-windows environments "KMarsh" and "kmarsh" are two different usernames and case matters. With this method, "KMarsh", "KMARSH", and "kmarsh" are all the same user account.

Signature:

```
HRESULT GetUserSettings(
                [in] BSTR bstrUser,
                [out, retval] ICIClientSettings **prop);
```

Example:

```
Site.GetUserSettings("kmarsh")
```

Refreshing the User Database (ForceSynchronizeUserDatabase)

Use the ICISite interface **ForceSynchronizeUserDatabase** method to synchronize with the user database. This will reload the user list and settings from the authentication store.

Signature:

```
HRESULT ForceSynchronizeUserDatabase();
```

Forcing a User to Log Off of the Site (KickUser)

Use the ICISite interface **KickUser** method to force a user to log off/disconnect from a Site.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT KickUser(
/*[in]*/ long nUserID,
/*[out, retval]*/ VARIANT_BOOL *pSuccess)
```

Example:

```
Dim bResult: bResult = oSite.KickUser(1)
If bResult Then
      WScript.Echo "User disconnected successfully."
Else
      WScript.Echo "Failed to disconnect the user."
End If
```

Retrieving the Physical Path to a Virtual Folder

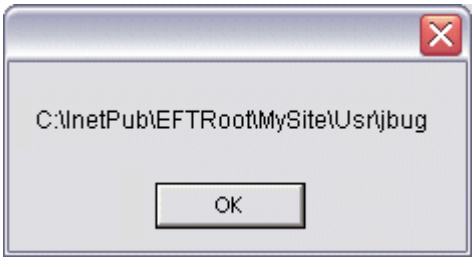
Use the ICISite interface **GetPhysicalPath** method to retrieve the physical path linked to a virtual folder.

This method is available in EFT Server 5.2.5 and later.

Example:

```
Msgbox Site.GetPhysicalPath("/Usr")
```

Returns a message box displaying the physical path, such as:



Site Custom Commands

The methods below are used to manage a Site's Custom Commands.

Retrieving a List of Custom Commands (GetCommands)

Use the ICISite interface **GetCommands** method to retrieve a list of custom commands.

Signature:

```
HRESULT GetCommands([out, retval] VARIANT *aCommands);
```

Retrieving a Custom Command's Settings (GetCommandSettings)

Use the ICISite interface **GetCommandSettings** method to retrieve a list of a custom commands settings and to access the ICICommandSettings interface.

Signature:

```
HRESULT GetCommandSettings(
                [in] BSTR bstrCommand,
                [out, retval] ICICommandSettings **prop);
```

Creating a New Custom Command (CreateCommand)

Use the ICISite interface **CreateCommand** method to create a new custom command for a site and to open the ICICommandSettings interface.

Signature:

```
HRESULT CreateCommand(
                [in] BSTR bstrName,
                [out, retval] ICICommandSettings **prop);
```

Assigning the Event Rule Custom Command Working Folder (AssignEventRuleCustomCommandWorkingFolder)

Use the ICISite interface **AssignEventRuleCustomCommandWorkingFolder** method to update all instances of a given custom command in the event rule list to use a specified working folder.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT AssignEventRuleCustomCommandWorkingFolder (
[in] BSTR bstrCustomCommandName,
[in] BSTR bstrWorkingFolder);
```

Example:

```
Dim strCommandName: strCommandName = "RunScript"
Dim strNewWorkingFolder: strNewWorkingFolder = "C:\Script\folder"
oSite.AssignEventRuleCustomCommandWorkingFolder(strCommandName,
strNewWorkingFolder)
```

Deleting a Custom Command (RemoveCommand)

Use the ICISite interface **RemoveCommand** method to delete a custom command from a Site.

Signature:

```
HRESULT RemoveCommand([in] BSTR bstrName);
```

Site IP Masks

The methods below are used to manage a Site's IP masks.

Adding an IP Mask to a Site (AddIPAccessRule)

Use the ICISite interface **AddIPAccessRule** method to add an allowed or denied IP mask to a site.

Signature:

```
HRESULT AddIPAccessRule(
                [in] BSTR bstrMask,
                [in] VARIANT_BOOL bAllow);
```

Removing an IP Mask from a Site (RemoveIPAccessRule)

Use the ICISite interface **RemoveIPAccessRule** method to delete an allowed or denied IP mask from a site.

Signature:

```
HRESULT RemoveIPAccessRule(
                [in] BSTR bstrMask,
                [in] VARIANT_BOOL bAllow);
```

Retrieving a List of Allowed IP Masks (GetAllowedMasks)

Use the ICISite interface **GetAllowedMasks** method to retrieve a list of allowed IP masks for a site.

Signature:

```
HRESULT GetAllowedMasks([out, retval] VARIANT *aMasks);
```

Retrieving a List of Denied IP Masks (GetDeniedMasks)

Use the ICISite interface **GetDeniedMasks** method to retrieve a list of IP masks prohibited for a site.

Signature:

```
HRESULT GetDeniedMasks([out, retval] VARIANT *aMasks);
```

Permission Interface Properties (ICIPermission)

The ICIPermissions interface represents the permissions for folders. Obtain it by using the GetFolderPermissions method of the ICISite interface. You can also use the GetBlankPermission method to start with no permissions.

Example:

```
aPerm = s.GetFolderPermissions("\pub\")
Set p = s.GetBlankPermission("\pub\", "Administrative")
```

Available Properties

- Retrieving a Folder Name (Folder)
- Retrieving a Client or Permission Group Name (Client)
- Granting Upload Permission (FileUpload)
- Granting Delete Permission (FileDelete)
- Granting Rename Permission (FileRename)
- Granting Append File Permission (FileAppend)
- Granting Download Permission (FileDownload)
- Granting Folder Creation Permission (DirCreate)
- Granting Folder Deletion Permission (DirDelete)
- Granting Permission to View a List of Folder Contents (DirList)
- Showing Hidden Files (DirShowHidden)
- Showing Read-Only Files (DirShowReadOnly)
- Showing or Hiding Folders (DirShowInList)

*Changing properties in the IPermission interface has no effect until you call the ICISite **SetPermission** method.*

Retrieving a Folder Name (Folder)

Use the IPermission interface **Folder** property to retrieve a folder's name.

Signature:

```
HRESULT Folder([out, retval] BSTR *pVal);
```

Retrieving a Client or Permission Group Name (Client)

Use the IPermission interface **Client** property to retrieve a client's name or a permission group's name.

Signature:

```
HRESULT Client([out, retval] BSTR *pVal);
```

Granting Upload Permission (FileUpload)

Use the IPermission interface **FileUpload** property to determine if group or client has permission to upload to a folder, and to allow or deny uploading permission.

Signature:

```
HRESULT FileUpload([out, retval] VARIANT_BOOL *pVal);
HRESULT FileUpload([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Delete Permission (FileDelete)

Use the IPermission interface **FileDelete** property to determine if a group or client has permissions to delete files and folders from a folder, and to allow or deny deletions.

Signature:

```
HRESULT FileDelete([out, retval] VARIANT_BOOL *pVal);
HRESULT FileDelete([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Rename Permission (FileRename)

Use the IPermission interface **FileRename** property to determine if a group or client has permissions to rename files and folders in a folder, and to allow or deny renaming permission.

Signature:

```
HRESULT FileRename([out, retval] VARIANT_BOOL *pVal);
HRESULT FileRename([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Append File Permission (FileAppend)

Use the IPermission interface **FileAppend** property to determine if a group or client has permissions to append files in a folder, and to allow or deny appending permission.

Signature:

```
HRESULT FileAppend([out, retval] VARIANT_BOOL *pVal);
HRESULT FileAppend([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Download Permission (FileDownload)

Use the IPermission interface **FileDownload** property to determine if group or client has permission to download from a folder, and to allow or deny downloading permisson.

Signature:

```
HRESULT FileDownload([out, retval] VARIANT_BOOL *pVal);
HRESULT FileDownload([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Folder Creation Permission (DirCreate)

Use the IPermission interface **DirCreate** property to determine if a group or client has permissions to create folders within a folder, and to allow or deny folder creation permission.

Signature:

```
HRESULT DirCreate([out, retval] VARIANT_BOOL *pVal);
HRESULT DirCreate([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Folder Deletion Permission (DirDelete)

Use the IPermission interface **DirDelete** property to determine if a group or client has permissions to delete folders within a folder, and to allow or prohibit folder deletion.

Signature:

```
HRESULT DirDelete([out, retval] VARIANT_BOOL *pVal);
HRESULT DirDelete([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Granting Permission to View a List of Folder Contents (DirList)

Use the IPermission interface **Dirlist** property to determine if a group or client has permissions to see a list of folder contents, and to allow or deny permission to see folder lists.

Signature:

```
HRESULT DirList([out, retval] VARIANT_BOOL *pVal);
HRESULT DirList([in] VARIANT_BOOL newVal);
```

True = Allowed

False = Prohibited

Showing Hidden Files (DirShowHidden)

Use the IPermission interface **DirShowHidden** property to determine if a group or client can see hidden files, and to display or hide hidden files.

Signature:

```
HRESULT DirShowHidden([out, retval] VARIANT_BOOL *pVal);
HRESULT DirShowHidden([in] VARIANT_BOOL newVal);
```

True = Displayed

False = Hidden

You must allow folder listing before you can choose to hide or display hidden files.

Showing Read-Only Files (DirShowReadOnly)

Use the IPermission interface **DirShowReadOnly** property to determine if a group or client can see read-only files, and to display or hide read-only files.

Signature:

```
HRESULT DirShowReadOnly([out, retval] VARIANT_BOOL *pVal);
HRESULT DirShowReadOnly([in] VARIANT_BOOL newVal);
```

True = Displayed

False = Hidden

You must allow folder listing before you can choose to hide or display read-only files.

Showing or Hiding Folders (DirShowInList)

Use the IPermission interface **DirShowInList** property to determine if a group or client can see a folder, and to display or hide the folder.

Signature:

```
HRESULT DirShowInList([out, retval] VARIANT_BOOL *pVal);
HRESULT DirShowInList([in] VARIANT_BOOL newVal);
```

True = Displayed

False = Hidden

Client Setting Interface Properties (ICIClientSettings)

The **ClientSetttings** interface properties displays the user properties defined for a user in EFT Server.

This method is available in EFT Server 5.0 and later.

Viewing User Properties - (FullName)

Use the ICIClientSettings Interface **FullName** property to view a user's full name.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT FullName([in] BSTR bstrLogin);
```

Viewing User Properties - (Phone)

Use the ICIClientSettings Interface **Phone** property to view a user's telephone number.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT Phone([in] BSTR bstrLogin);
```

Viewing User Properties - (Email)

Use the ICIClientSettings Interface **Email** property to view a user's email address.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT Email([in] BSTR bstrLogin);
```

Viewing User Properties - (Fax)

Use the ICIClientSettings Interface **Fax** property to view a user's fax number.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT Fax([in] BSTR bstrLogin);
```

Viewing User Properties - (Pager)

Use the ICIClientSettings Interface **Pager** property to view a user's pager number.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT pager([in] BSTR bstrLogin);
```

Viewing User Properties - (Comment)

Use the ICIClientSettings Interface **Comment** property to view comments for a user object.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT Comment([in] BSTR bstrLogin);
```

Viewing User Properties - (Custom)

Use the ICIClientSettings Interface **Custom1, Custom2, or Custom3** property to view the custom fields of a user object.

This method is available in EFT Server 5.0 and later.

Signature:

```
HRESULT Custom1([in] BSTR bstrLogin);
HRESULT Custom1([in] BSTR bstrLogin);
HRESULT Custom1([in] BSTR bstrLogin);
```

Example

```
// eftTestCustomFields.js
// CREATED: 24 May 2007
// Script to confirm use of EFT5 Custom Fields properties of a CClientSettings in
Javascript through COM API.
//

var args = WScript.Arguments;
if ( args.length < 5 )
{
 WScript.Echo("Usage: eftTestCustomFields <eft host> <eft port> <userid> <password>
<user id>\n");
 WScript.Echo("Example:  cscript eftTestCustomFields.js 192.168.20.101 1100 admin
admin foo\n");
 WScript.Quit(1);
}
var sHost = args(0);
var iPort = args(1);
var sUser = args(2);
var sPass = args(3);
var sAccountName = args(4);

var server = new ActiveXObject("SFTPCOMInterface.CIServer");
WScript.Echo("Connecting...");
server.Connect(sHost, iPort, sUser, sPass);
WScript.Echo("Connected!");
var site = server.Sites().SiteByID(1); // for simplicity, assume first site
var oSettings = site.GetUserSettings(sAccountName);

WScript.Echo("Obtained user settings for user account '" + oSettings.FullName + "'
with home directory: " + oSettings.GetHomeDirString() );
WScript.Echo("Custom1='" + oSettings.Custom1 + "'");
WScript.Echo("Custom2='" + oSettings.Custom2 + "'");
WScript.Echo("Custom3='" + oSettings.Custom3 + "'");

// Now set the custom fields to new values
WScript.Echo("\nSetting new values...");
oSettings.Custom1 = "Random=" + Math.floor((65535*Math.random())) + " [" + (new
Date()).toLocaleString() + "]";
```

```
oSettings.Custom2 = "Random=" + Math.floor((65535*Math.random())) + " [" + (new
Date()).toLocaleString() + "]";
oSettings.Custom3 = "Random=" + Math.floor((65535*Math.random())) + " [" + (new
Date()).toLocaleString() + "]";

server.ApplyChanges();

// Now REQUERY to CONFIRM the custom fields are the new values
WScript.Echo("Requerying values...");
var oSettings = site.GetUserSettings(sAccountName);
WScript.Echo("\nObtained NEW user settings for user account '" + oSettings.FullName
+ "' with home directory: " + oSettings.GetHomeDirString() );
WScript.Echo("Custom1='" + oSettings.Custom1 + "'");
WScript.Echo("Custom2='" + oSettings.Custom2 + "'");
WScript.Echo("Custom3='" + oSettings.Custom3 + "'");

server.Close();

WScript.Echo("\nDone");
WScript.Quit(0);
```

Viewing Invalid Attempts Period for a Client (InvalidAttemptsPeriod)

Use the ICIClientSettings Interface **InvalidAttemptsPeriod** property to view the length of time during which invalid login attempts are counted for a client.


This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT InvalidAttemptsPeriod([in] BSTR bstrLogin);
```

Viewing Allowed IP Addresses (IPAccessAllowedDefault)

Use the ICIClientSettings Interface **IPAccessAllowedDefault** property to view the IP addresses with which a user is allowed to connect to the server.


This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT IPAccessAllowedDefault([in] BSTR bstrLogin);
```

Viewing a User's Lockout Period (LockoutPeriod)

Use the ICIClientSettings Interface **LockoutPeriod** property to view the length of time a user's account is locked out after invalid login attempts.


This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT LockoutPeriod([in] BSTR bstrLogin);
```
Values are 30, 60, or 90 minutes

Viewing the Number of Invalid Login Attempts Allowed for a Client (MaxInvalidLoginAttempts)

Use the ICIClient interface **MaxInvalidLoginAttempts** property to view the maximum number of invalid login attempts a client can make to a Site.


This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT MaxInvalidLoginAttempts([out, retval] long *pVal);
HRESULT MaxInvalidLoginAttempts([in] long newVal);
```

Client Settings Interface Methods (ICIClientSettings)

Use GetUserSettings or GetSettingsLevelSettings methods in the ICISite interface to manage user account or settings level (template) settings.

Example

```
Set oUser = oSite.GetUserSettings(aUsr(j))
Set 0SettingsLevel = oSite.GetSettingsLevelSettings("Default Settings")
```

Once you obtain a handle to the user or settings level settings, you can Get or Set permissions for that user or template.

When you call methods or change properties of the ICIClientSettings interface, you must call ICIServer's ApplyChanges method in order for the changes to take effect.

SFTPAdvBool Data Type

Some methods of the ICIClientSettings interface use the **SFTPAdvBool** data type. It is an enumeration data type and it is similar to VARIANT_BOOL, but it has one additional value: **abInherited**. When a user setting is given the **abInherited** value, it means the user setting is inherited from the user's Settings Level.

SFTPAdvBool Data Type Definition

```
typedef enum
{
    abFalse = 0,
    abTrue = 1,
    abInherited = -2
} SFTPAdvBool;
```

When you change a user account or Settings Level with the ICIClientSettings interface, you must call ICIServer's ApplyChanges method for the changes to take effect.

Available Client Settings Interface Methods:

- User Account Settings
- Password Settings
- FTP Security Settings
- Transfer and Connection Settings
- Disk Quotas

User Account Settings

The methods below are used to retrieve information about or change user settings.

Determining if the Account Home Folder is the Default Root Folder (GetHomeDirIsRoot)

Use the ICIClientSettings interface **GetHomeDirIsRoot** method to check if a user account home folder is set as the user's default root folder.

Signature:

```
HRESULT GetHomeDirIsRoot(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = No

1 = Yes

Specifying the Default Root Folder (SetHomeDirIsRoot)

Use the ICIClientSettings interface **SetHomeDirIsRoot** method to set a user's home folder as the user's default root folder.

Signature:

```
HRESULT SetHomeDirIsRoot([in] SFTPAdvBool val);
```

0 = Not default

1 = Is default

-2 = Inherited from User Settings Level

Determining if Users Can Have a Home Folder (GetHomeDir)

Use the ICIClientSettings interface **GetHomeDir** method to find the whether the user or User Settings Level can have a home folder.

Signature:

```
HRESULT GetHomeDir(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Allowing Users to Have a Home Folder (SetHomeDir)

Use the ICIClientSettings interface **SetHomeDir** method to allow users to have a home folder. Use the same method to prohibit users from having a home folder.

Signature:

```
HRESULT SetHomeDir([in] long val);
```

Determining the Expiration Date for a User Account (GetExpirationDate)

Use the ICIClientSettings interface **GetExpirationDate** method to determine the expiration date for a particular user account; set with **SetExpirationDate**.

Signature:

```
HRESULT GetExpirationdate (
           [out] VARIANT *dDate,
           [out, retval] VARIANT_BOOL *pVal);
```

dDate results in a string value, i.e. "4/29/05"

Example:

```
Dim strUser: strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
dtAccExpDate =  oUserSettings.GetExpirationDateAsVariant()
      WScript.Echo ("dtAccExpDate = "  & dtAccExpDate)
```

Specifying the Expiration Date for a User Account (SetExpirationDate)

Use the ICIClientSettings interface **SetExpirationDate** method to set the expiration date for a particular user account. Use **GetExpirationDate** to retrieve it. When passing values to COM methods, VARIANTS are preferred to STRING values.

Signature:

```
HRESULT SetExpirationdate (
      [in] VARIANT *dDate,
      [in] VARIANT_BOOL *bEnable);
```

This example, written in PHP 5, uses a VARIANT rather than a STRING to set the expiration date:

```
$expiredate = "05/15/2005";
$enableexpiration = 1;
$vtdate = new VARIANT($expiredate, VT_DATE);
$UserSettings->SetExpirationDate($expiredate, $enableexpiration);
```

Determining the Expiration Date for a User Account (GetExpirationDateAsVariant)

Use the ICIClientSettings interface **GetExpirationDateAsVariant** method to determine the expiration date for a particular user account, set with SetExpirationDate.

This method is available in EFT Server 4.3.4 and later.

Signature:

```
HRESULT GetExpirationDateAsVariant(
[out, retval] VARIANT *dDate);
dDate results in a string value, i.e. "4/29/05"
```

Example:

```
Dim strUser: strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
dtAccExpDate =  oUserSettings.GetExpirationDateAsVariant()
WScript.Echo ("dtAccExpDate = "  & dtAccExpDate)
```

Determining How Users' Login Message is Defined (GetLoginMsg)

Use the ICIClientSettings interface **GetLoginMsg** method to find the whether the server uses the default log in message, adds a custom string to the default message, replaces the default message with a custom string, or if the server uses no login message.

Signature:

```
HRESULT GetLoginMsg(
                  [out, optional] VARIANT_BOOL *pInherited,
                  [out, retval] long *pVal);
```

0 = Use default

-1 = Add to default

-2 = Replace default

-3 = None

Specifying the Login Message Used (SetLoginMsg)

Use the ICIClientSettings interface **SetLoginMsg** method to choose whether the server uses the default log in message, adds a custom string to the default message, replaces the default message with a custom string, or if the server uses no log in message.

Signature:

```
HRESULT SetLoginMsg([in] long val);
```

0 = Use default

-1 = Add to default

-2 = Replace default

-3 = None

Retrieving the Login Message (GetLoginMsgString)

Use the ICIClientSettings interface **GetLoginMsgString** method to retrieve a login message.

Signature:

```
HRESULT GetLoginMsgString([out, retval] BSTR *prop);
```

Creating a Login Message (SetLoginMsgString)

Use the ICIClientSettings interface **SetLoginMsgString** method to create a login message.

Signature:

```
HRESULT SetLoginMsgString([in] BSTR bstrVal);
```

Viewing Anonymous Logins (GetAnonymousLogin)

Use the ICIClientSettings interface **GetAnonymousLogin** method to retrieve a list of anonymous logins.

Signature:

```
HRESULT GetAnonymousLogin(
[out] VARIANT_BOOL *pInherited,
[out, retval]  VARIANT_BOOL *pVal);
Values:
0 = Prohibited
1 = Allowed
```

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
bUserAnmLogin = oUserSettings.GetAnonymousLogin()
      WScript.Echo ("bUserAnmLogin= " & bUserAnmLogin)
```

Allowing or Prohibiting Anonymous Logins (SetAnonymousLogin)

Use the ICIClientSettings interface **SetAnonymousLogin** method to allow or prohibit anonymous logins.

Signature:

```
HRESULT SetAnonymousLogin(
[in]  SFTPAdvBool val );
Values:
0 = Prohibited
1 = Allowed
-2 = Inherited
```

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
oUserSettings.SetAnonymousLogin( True )
```

Retrieving Users' Home Folders (GetHomeDirString)

Use the ICIClientSettings interface **GetHomeDirString** method to retrieve the path to a user or User Settings Level home folder.

Signature:

```
HRESULT GetHomeDirString([out, retval] BSTR *prop);
```

Specifying the Path to Users' Home Folders (SetHomeDirString)

Use the ICIClientSettings interface **SetHomeDirString** method to set the path to a user or User Settings Level home folder.

Signature:

```
HRESULT SetHomeDirString([in] BSTR bstrVal);
```

Determining if Users are Restricted to a Specific IP Address (GetHomeIP)

Use the ICIClientSettings interface **GetHomeIP** method to determine whether a user or Settings Level is restricted to connections on one specific IP address.

Signature:

```
HRESULT GetHomeIP(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Restricting Users to a Specific IP Address (SetHomeIP)

Use the ICIClientSettings interface **SetHomeIP** method to restrict a user or User Settings Level to connections on one specific IP address.

Signature:

```
HRESULT SetHomeIP([in] long val);
```

Retrieving Users' Home IP Address (GetHomeIPString)

Use the ICIClientSettings interface **GetHomeIPString** method to retrieve the IP address where the user or User Settings Level must make connections.

Signature:

```
HRESULT GetHomeIPString([out, retval] BSTR *prop);
```

Specifying Users' Home IP Address (SetHomeIPString)

Use the ICIClientSettings interface **SetHomeIPString** method to designate the IP address where the user or Settings Level must make connections.

Signature:

```
HRESULT SetHomeIPString([in] BSTR bstrVal);
```

Retrieving a User's Description (GetDescription)

Use the ICIClientSettings interface **GetDescription** method to retrieve the description for a user or User Settings Level.

Signature:

```
HRESULT GetDescription([out, retval] BSTR *prop);
```

Specifying a User Description (SetDescription)

Use the ICIClientSettings interface **SetDescription** method to enter the description for a user or Settings Level.

Signature:

```
HRESULT SetDescription([in] BSTR bstrVal);
```

Determining Web Transfer Client Access (GetAppletEnabled)

Use the ICIClientSettings interface **GetAppletEnabled** method to set EFT Web Transfer client access setting for a user or User Settings Level.

This method is available in EFT Server 3.5 and later.

Example:

```
GetAppletEnabled([in]SFTPAdvBool val);
```

Example:

Setting the enabled/disabled state:

```
slSettings = GetSettingsLevelSettings("Default Settings")
slSettings.GetAppletEnabled( True )
```

Specifying Web Transfer Client access (SetAppletEnabled)

Use the ICIClientSettings interface **SetAppletEnabled** method to specify EFT Web Transfer Client applet access setting for a user or User Settings Level.

This method is available in EFT Server 3.5 and later.

Signature:

```
SetAppletEnabled(
      [out, optional]VARIANT_BOOL *pInherited,[out, retval] VARIANT_BOOL
      *pVal);
```

Example:

Setting the state:

```
slSettings = SetSettingsLevelSettings("Default Settings")
bEnabled = slSettings.SetAppletEnabled( bInherited )
```

Determining if a User Account or User Settings Level is Enabled (GetEnableAccount)

Use the ICIClientSettings interface **GetEnableAccount** method to check if a user account or Settings level is enabled.

Signature:

```
HRESULT GetEnableAccount(
            [out, optional] VARIANT_BOOL *pInherited,
            [out, retval] VARIANT_BOOL *pVal);
```

0 = Disabled

1 = Enabled

Enabling a User Account or User Settings Level (SetEnableAccount)

Use the ICIClientSettings interface **SetEnableAccount** method to enable a user account or Settings Level.

Signature:

```
HRESULT SetEnableAccount([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

You need to obtain a handle to the specific user or settings level from ICISite's **GetUserSettings** method or ICISite's **GetSettingsLevelSettings** method before you can perform an action upon that user or settings level. Below is a sample code snippet using PHP that demonstrates this technique:

```
<?php
// first create server object
$Server = new COM("SFTPCOMInterface.CIServer") or die("Unable to
instantiate Server");
// connect to server
$Server->Connect("localhost",1000,"admin","admin");
// get handle to list of sites
$Sites = $Server->Sites();
// chose your site. On most one-site systems this will be "0"
$MySite = $Sites->Item(0);
// Pull the settings for the user that you want.
$Settings = $MySite->GetUserSettings("juser");
// enable or disable or inherit.
// Enable is 1. Disable is 0. Inherit is -2
// Notice that you apply this to the settings for the user that
you just pulled
$Settings->SetEnableAccount(1);
// Be sure to apply the settings or else nothing will really
change
$Server->ApplyChanges();
// close the connection.
$Server->Close();
?>
```

Password Settings

The methods below are used to retrieve information about or change user password settings.

Determining if Users are Allowed to Change their Passwords (GetChangePwd)

Use the ICIClientSettings interface **GetChangePwd** method to check if users can change their passwords.

Signature:

```
HRESULT GetChangePwd(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Allowing Users to Change their Passwords (SetChangePwd)

Use the ICIClientSettings interface **SetChangePwd** method to allow a user to change passwords. Use the same method to prohibit the user from changing passwords.

Signature:

```
HRESULT SetChangePwd([in] SFTPAdvBool val);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Determining if Users Can Create Any Password (GetAllowAnyPwd)

Use the ICIClientSettings interface **GetAllowAnyPwd** method to check if a user may create any password (i.e., complex passwords not required).

Signature:

```
HRESULT GetAllowAnyPwd(
[out, optional] VARIANT_BOOL *pInherited,
[out, retval] VARIANT_BOOL *pVal);
Values:
0 = Prohibited
1 = Allowed
```

Example:

```
Dim strUser: strUser = "test"
Set oUserSettings = oSite.GetUserSettings(strUser)
 bUserPwd = oUserSettings.GetAllowAnyPwd()
     WScript.Echo ("bUserPwd= " & bUserPwd)
```

Allowing Users to Create Any Password (SetAllowAnyPwd)

Use the ICIClientSettings interface **SetAllowAnyPwd** method to specify whether a user to create any password (i.e., complex passwords not required).

Signature:

```
HRESULT SetAllowAnyPwd(
[in] SFTPAdvBool val);
Values:
0 = Prohibited
1 = Allowed
-2 = Inherited
```

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
oUserSettings.SetAllowAnyPwd ( True )
```

Determining the Number of Failed Password Attempts (GetIncorrectPasswordAttempts)

Use the ICIClientSettings interface **GetIncorrectPasswordAttempts** method to show how many times a user has tried to connect with incorrect log in information.

Signature:

```
HRESULT GetIncorrectPasswordAttempts(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Number of Incorrect Password Attempts (SetIncorrectPasswordAttempts)

Use the ICIClientSettings interface **SetIncorrectPasswordAttempts** method to set the number of times a user can try to connect with incorrect log in information.

Signature:

```
HRESULT SetIncorrectPasswordAttempts(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] long *pVal);
```

Retrieving the Maximum Number of Failed Login Attempts Allowed per User (GetPwdRetries)

Use the ICIClientSettings interface **GetPwdRetries** method to determine the number of times a user can try to connect with an incorrect password.

Signature:

```
HRESULT GetPwdRetries(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Failed Password Limit (SetPwdRetries)

Use the ICIClientSettings interface **SetPwdRetries** method to choose the maximum number of times a user can try to connect to the server with incorrect log in information.

Signature:

```
HRESULT SetPwdRetries([in] long val);
```

Determining Failed Password Limit (GetHasPwdRetries)

Use the ICIClientSettings interface **GetHasPwdRetries** method to determine if a user account will be disabled after trying too many bad passwords in a row.

Signature:

```
HRESULT GetHasPwdRetries(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] VARIANT_BOOL *pVal);
```

0 = Unlimited

1 = Limited

Limiting Failed Password Attempts (SetHasPwdRetries)

Use the ICIClientSettings interface **SetHasPwdRetries** method to disable an account if a user enters bad passwords too many times in a row. Use the same method to allow an unlimited number of bad passwords.

Signature:

```
HRESULT SetHasPwdRetries([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

FTP Security Settings

The methods below are used to retrieve information about or change allowed commands for a user.

Determining if the NoOP Command is Allowed (GetAllowNoop)

Use the ICIClientSettings interface **GetAllowNoop** method to check if a user is allowed to use the NOOP command to keep a connection open.

Signature:

```
HRESULT GetAllowNoop(
            [out, optional] VARIANT_BOOL *pInherited,
            [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Allowing the NOOP Command (SetAllowNoop)

Use the ICIClientSettings interface **SetAllowNoop** method to allow a user to send the NOOP command to keep a connection open. Use the same method to prohibit the user from sending the NOOP command.

Signature:

```
HRESULT SetAllowNoop([in] SFTPAdvBool val);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Determining if the XCRC Command is Allowed (GetAllowXCRC)

Use the ICIClientSettings interface **GetAllowXCRC** method to check if a user is allowed to send the XCRC command to confirm successful transfer.

Signature:

```
HRESULT GetAllowXCRC(
           [out, optional] VARIANT_BOOL *pInherited,
           [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Allowing the XCRC Command (SetAllowXCRC)

Use the ICIClientSettings interface **SetAllowXCRC** method to allow a user to send the XCRC command to confirm transfers. Use the same method to prohibit the user from sending the XCRC command.

Signature:

```
HRESULT SetAllowXCRC([in] SFTPAdvBool val);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Determining if ModeZ is Allowed (GetAllowMODEZ)

Use the ICIClientSettings interface **GetAllowMODEZ** method to determine if "Allow MODE Z" is enabled for the Site.

You may also pass in an optional BOOLEAN variable as a parameter to this function; when the function returns this optional parameter contains the INHERITED status of this feature (true means inherited from server; false means set locally at the site level).

Signature:

```
HRESULT GetAllowMODEZ([out, optional] VARIANT_BOOL *pInherited, [out, retval]
VARIANT_BOOL *pVal);
```

Allowing MODE Z Compression (SetAllowMODEZ)

Use the ICIClientSettings interface **SetAllowMODEZ** method to set the "Allow Mode Z" feature of a Site ON or OFF.

Signature:

```
HRESULT SetAllowMODEZ([in] SFTPAdvBool val);
```

Transfer and Connection Settings

The methods below are used to retrieve information about or change a user's transfer limits.

Determining if Plain HTTP Access is Allowed (GetClearHTTP)

Use the ICIClientSettings interface **GetClearHTTP** method to determine if a user account can make connections using unsecured HTTP.

This method is available in EFT Server 4.3.4 and 5.1 and later.

Signature:

```
HRESULT GetClearHTTP(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibited

1 = Allowed

Allowing Users to Connect Using Clear HTTP (SetClearHTTP)

Use the ICIClientSettings interface **SetClearHTTP** method to allow a user account to make connections using unsecured HTTP.

This method is available in EFT Server 4.3.4, 5.1, and later.

Signature:

```
HRESULT SetClearHTTP(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibited

1 = Allowed

Determining if Plain FTP Access is Allowed (GetClearFTP)

Use the ICIClientSettings interface **GetClearFTP** method to determine if a user account can make connections using unsecured FTP.

Signature:

```
HRESULT GetClearFTP(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibited

1 = Allowed

Allowing Users Plain FTP Connections (SetClearFTP)

Use the ICIClientSettings interface **SetClearFTP** method to allow a user to connect using unsecure FTP. Use the same method to prohibit the user from making unsecure FTP connections.

Signature:

```
HRESULT SetClearFTP([in] SFTPAdvBool val);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Viewing if SFTP Access is Enabled for a Client (GetSFTP)

Use the CIClientSettings interface **GetSFTP** method to view whether access is enabled for a client.

Signature:

```
HRESULT GetSFTP(
      [out] VARIANT_BOOL *pInherited,
      [out, retval]  VARIANT_BOOL *pVal);
```

True = Access enabled

False = Access disabled

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
bIsFTPEnabled = oUserSettings.GetSFTP()
      WScript.Echo ("bIsFTPEnabled="& bIsFTPEnabled)
```

Allowing SFTP Access for a Client (SetSFTP)

Use the ICIClientSettings interface **SetSFTP** method to enable or disable FTP access for a client. The site must be restarted for the change to take effect; the **SetSFTP** method returns **TRUE**.

Signature:

```
HRESULT SetsFTP(
      [in] SFTPAdvBool val);
```

0 = Prohibited

1 = Allowed

-2 = Inherited

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
oUserSettings.SetSFTP ( True )
```

Identifying the SFTP Authentication Type (GetSFTPAuthenticationType)

Use the ICIClientSettings interface **GetSFTPAuthenticationType** method to identify or retrieve the SFTP authentication type for the client.

Signature:

```
HRESULT GetSFTPAuthenticationType(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] BSTR *pVal);
```

Example:

```
Dim strUser:strUser = "test"
Set oUserSettings = oSite.GetUserSettings(strUser)
Dim strSFTPEnabled: strSFTPEnabled=false
oUserSettings.GetSFTPAuthenticationType(strSFTPEnabled)
Dim strType: strType = oUserSettings.GetSFTPAuthenticationType()
      WScript.Echo "SFTP Enabled: " & strSFTPEnabled
      WSCript.Echo "SFTP Password type: " & strType
```

Specifying the SFTP Authentication Type for the Client (SetSFTPAuthenticationType)

Use the ICIClientSettings interface **SetSFTPAuthenticationType** method to set the authentication type for the client.

Signature:

```
HRESULT SetSFTPAuthenticationType(
      [in] BSTR val);
```

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
oUserSettings.SetSFTPAuthenticationType ("Password")
```

Determining if SSL Access is Allowed (GetSSL)

Use the ICIClientSettings interface **GetSSL** method to determine if a user account can make connections using SSL over FTP.

Signature:

```
HRESULT GetSSL(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Prohibited

1 = Allowed

Allowing Users SSL Connections (SetSSL)

Use the ICIClientSettings interface **SetSSL** method to allow a user to connect using SSL over FTP. Use the same method to prohibit the user from making SSL connections.

Signature:

```
HRESULT SetSSL([in] SFTPAdvBool val);
```

0 = Prohibit

1 = Allow

-2 = Inherit

Identifying the SSL Authentication Type (GetSSLAuthenticationType)

Use the ICIClientSettings interface **GetSSLAuthenticationType** method to identify or retrieve the SSL authentication type for the client.

This method is available in EFT Server 5.1 and later.

Signature:

```
HRESULT GetSSLAuthenticationType(
[out, optional] VARIANT_BOOL *pInherited,
[out, retval] BSTR *pVal);
```

Example:

```
Dim strUser:strUser = "test"
Set oUserSettings = oSite.GetUserSettings(strUser)
Dim strSSLEnabled: strSSLEnabled=false
oUserSettings.GetSSLAuthenticationType(strSSLEnabled)
Dim strType: strType = oUserSettings.GetSSLAuthenticationType()
      WScript.Echo "SSL Enabled: " & strSSLEnabled
      WSCript.Echo "SSL Password type: " & strType
```

Specifying the SSL Authentication Type (SetSSLAuthenticationType)

Use the ICIClientSettings interface **SetSSLAuthenticationType** method to specify the SSL authentication type for the client.

This method is available in EFT Server 5.1 and later.

Signature:

```
HRESULT SetSSLAuthenticationType(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] BSTR *pVal);
```

Example:

```
Dim strUser:strUser = "test"
Set oUserSettings = oSite.SetUserSettings(strUser)
Dim strSSLEnabled: strSSLEnabled=false
oUserSettings.SetSSLAuthenticationType(strSSLEnabled)
Dim strType: strType = oUserSettings.SetSSLAuthenticationType()
      WScript.Echo "SSL Enabled: " & strSSLEnabled
      WSCript.Echo "SSL Password type: " & strType
```

Identifying the SSL Key ID (GetSSLKeyID)

Use the ICIClientSettings interface **GetSSLKeyID** method to identify or retrieve the SSL key ID for the client.

This method is available in EFT Server 5.1 and later.

Signature:

```
HRESULT GetSSLKeyID(
[out, optional] VARIANT_BOOL *pInherited,
[out, retval] BSTR *pVal);
```

Example:

```
Dim strUser:strUser = "test"
Set oUserSettings = oSite.GetUserSettings(strUser)
Dim strSSLEnabled: strSSLEnabled=false
oUserSettings.GetSSLKeyID(strSSLEnabled)
Dim strType: strType = oUserSettings.GetSSLKeyID()
      WScript.Echo "SSL Enabled: " & strSSLEnabled
      WSCript.Echo "SSL Password type: " & strType
```

Specifying the SSL Key ID (SetSSLKeyID)

Use the ICIClientSettings interface **SetSSLKeyID** method to specify the SSL Key ID used by a client.

This method is available in EFT Server 5.1 and later.

Signature:

```
HRESULT SetSSLKeyID(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] BSTR *pVal);
```

Example:

```
Dim strUser:strUser = "test"
Set oUserSettings = oSite.GetUserSettings(strUser)
Dim strSSLEnabled: strSSLEnabled=false
oUserSettings.SetSSLKeyID(strSSLEnabled)
Dim strType: strType = oUserSettings.SetSSLKeyID()
      WScript.Echo "SSL Enabled: " & strSSLEnabled
      WScript.Echo "SSL Password type: " & strType
```

Retrieving the SFTP (SSH) Certificate ID (GetSSHKeyID)

Use the ICIClientSettings interface **GetSSHKeyID** method to view the SFTP (SSH) certificate ID for the client.

Signature:

```
HRESULT GetSSHKeyID(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] long *pVal);
```

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
Dim iSSHKeyID: iSSHKeyID = oUserSettings.GetSSHKeyID()
      WScript.Echo (iSSHKeyID)
```

Specifying the SFTP (SSH) Certificate ID (SetSSHKeyID)

Use the ICIClientSettings interface **SetSSHKeyID** method to set the SFTP (SSH) certificate ID for the client.

Signature:

```
HRESULT SetSSHKeyID(
       [in] long val);
```

Example:

```
Dim strUser:strUser = "test"
set oUserSettings = oSite.GetUserSettings(strUser)
oUserSettings.SetSSHKeyID(1)
```

Determining if a User has a Transfer Speed Limit (GetHasMaxSpeed)

Use the ICIClientSettings interface **GetHasMaxSpeed** method to check if a user account has a maximum allowed transfer speed.

Signature:

```
HRESULT GetHasMaxSpeed(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Disabled

1 = Enabled

Enabling the Transfer Speed Limit (SetHasMaxSpeed)

Use the ICIClientSettings interface **SetHasMaxSpeed** method to limit a user to a maximum transfer speed. Use the same method to turn off the limit.

Signature:

```
HRESULT SetHasMaxSpeed([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

Determining the Maximum Allowed Transfer Speed (GetMaxSpeed)

Use the ICIClientSettings interface **GetMaxSpeed** method to determine, in kilobytes per second, the maximum transfer speed allowed for an account or Settings Level.

Signature:

```
HRESULT GetMaxSpeed(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] long *pVal);
```

Specifying the Maximum Allowed Transfer Speed (SetMaxSpeed)

Use the ICIClientSettings interface **SetMaxSpeed** method to the speed at which an account or Settings Level can transfer file. The number you enter represents kilobytes per second.

Signature:

```
HRESULT SetMaxSpeed([in] long val);
```

Determining if a User has a Download Size Limit (GetHasMaxDownloadSize)

Use the ICIClientSettings interface **GetHasMaxDownloadSize** method to determine if there is a maximum file size a user is allowed to download.

Signature:

```
HRESULT GetHasMaxDownloadSize(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Enabling a User's Download Size Limit (SetHasMaxDownloadSize)

Use the ICIClientSettings interface **SetHasMaxDownloadSize** method to limit a user to a maximum download size.

Signature:

```
HRESULT SetHasMaxDownloadSize([in] SFTPAdvBool val);
```

Retrieving a User's Download Size Limit (GetMaxDownloadSize)

Use the ICIClientSettings interface **GetMaxDownloadSize** method to determine the maximum file size, in kilobytes, a user is allowed to download.

Signature:

```
HRESULT GetMaxDownloadSize(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Maximum File Size a User is Permitted to Download (SetMaxDownloadSize)

Use the ICIClientSettings interface **SetMaxDownloadSize** method to choose the maximum file size, in kilobytes, a user is permitted download.

Signature:

```
HRESULT SetMaxDownloadSize([in] long val);
```

Determining if a User has a Download per Session Limit (GetHasDownloadsPerSession)

Use the ICIClientSettings interface **GetHasDownloadsPerSession** method to determine if there is a maximum number of files a user is allowed to download per session.

Signature:

```
HRESULT GetHasDownloadsPerSession(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Enabling a User's Downloads-per-Session Limit (SetHasDownloadsPerSession)

Use the ICIClientSettings interface **SetHasDownloadsPerSession** method to limit a user to a maximum number of downloads per session.

Signature:

```
HRESULT SetHasDownloadsPerSession([in] SFTPAdvBool val);
```

Retrieving a Download per Session Limit (GetDownloadsPerSession)

Use the ICIClientSettings interface **GetDownloadsPerSession** method to determine the maximum number of downloads a user is permitted per session.

Signature:

```
HRESULT GetDownloadsPerSession(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Maximum Number of Downloads a User is Permitted per Session (SetDownloadsPerSession)

Use the ICIClientSettings interface **SetDownloadsPerSession** method to choose the maximum number of downloads a user is permitted per session.

Signature:

```
HRESULT SetDownloadsPerSession([in] long val);
```

Determining if a User has an Upload Size Limit (GetHasMaxUploadSize)

Use the ICIClientSettings interface **GetHasMaxUploadSize** method to determine if there is a maximum file size a user is allowed to upload.

Signature:

```
HRESULT GetHasMaxUploadSize(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Enabling a User's Upload Size Limit (SetHasMaxUploadSize)

Use the ICIClientSettings interface **SetHasMaxUploadSize** method to limit a user to a maximum upload size.

Signature:

```
HRESULT SetHasMaxUploadSize([in] SFTPAdvBool val);
```

Determining if a User has an Upload per Session Limit (GetHasUploadsPerSession)

Use the ICIClientSettings interface **GetHasUploadsPerSession** method to determine if there is a maximum number of files a user is allowed to upload per session.

Signature:

```
HRESULT GetHasUploadsPerSession(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Enabling a User's Uploads-per-Session Limit (SetHasUploadsPerSession)

Use the ICIClientSettings interface **SetHasUploadsPerSession** method to limit a user to a maximum number of uploads per session.

Signature:

```
HRESULT SetHasUploadsPerSession([in] SFTPAdvBool val);
```

Retrieving a User's Upload Size Limit (GetMaxUploadSize)

Use the ICIClientSettings interface **GetMaxUploadSize** method to determine the maximum file size, in kilobytes, a user is allowed to upload.

Signature:

```
HRESULT GetMaxUploadSize(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Maximum File Size a User is Permitted to Upload (SetMaxUploadSize)

Use the ICIClientSettings interface **SetMaxUploadSize** method to choose the maximum file size, in kilobytes, a user is permitted upload.

Signature:

```
HRESULT SetMaxUploadSize([in] long val);
```

Retrieving a User's Upload per Session Limit (GetUploadsPerSession)

Use the ICIClientSettings interface **GetUploadsPerSession** method to determine the maximum number of uploads a user is permitted per session.

Signature:

```
HRESULT GetUploadsPerSession(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Maximum Number of Uploads a User is Permitted per Session (SetUploadsPerSession)

Use the ICIClientSettings interface **SetUploadsPerSession** method to choose the maximum number of uploads a user is permitted per session.

Signature:

```
HRESULT SetUploadsPerSession([in] long val);
```

Viewing Whether Account Lockout is Enabled for a User (GetLockoutNotDisable)

Use the ICIClientSettings interface **GetLockoutNotDisable** method to determine whether account lockout is enabled for a user.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT GetLockoutNotDisable(
[out] VARIANT_BOOL *pInherited,
[out, retval]  VARIANT_BOOL *pVal);
Values
```

True = Lockout is disabled
False = Lockout is not disabled

Enabling Account Lockout for a User (SetLockoutNotDisable)

Use the ICIClientSettings interface **SetLockoutNotDisable** method to enable account lockout for a user.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT GetLockoutNotDisable(
[out] VARIANT_BOOL *pInherited,
[out, retval]  VARIANT_BOOL *pVal);
Values
```

True = Lockout is not disabled
False = Lockout is disabled

Retrieving Denied IP Mask for a User (GetDeniedMasks)

Use the ICIClientSettings interface **GetDeniedMasks** method to view the IP mask from which the user is not allowed to connect.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT GetDeniedMasks(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Retrieving Allowed IP Masks for a User (GetAllowedMasks)

Use the ICIClientSettings interface **GetAllowedMasks** method to view the IP mask from which the user is allowed to connect.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT GetAllowedMasks(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Adding an IP Access IP Mask for a Client (AddIPAccessRule)

Use the ICIClientSettings interface **AddIPAccessRule** method to add the allowed or denied IP mask from which the client is allowed to connect.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT AddIPAcessRule([in] long val);
```

Removing an IP Access Mask for User (RemoveIPAccessRule)

Use the ICIClientSettings interface **RemoveIPAccessRule** method to remove the allowed or denied IP mask from which the user is allowed to connect.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT RemoveIPAcessRule([in] long val);
```

Retrieving Number of Login Attempts Allowed (GetLimitLoginAttempts)

Use the ICIClientSettings interface **GetLimitLoginAttempts** method to view the number of login attempts the user is allowed.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT GetLimitLoginAttempts(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Unlimited

1 = Limited

Specifying Number of Login Attempts Allowed (SetLimitLoginAttempts)

Use the ICIClientSettings interface **SetLimitLoginAttempts** method to set the number of login attempts the user is allowed.

This method is available in EFT Server 5.1.1 and later.

Signature:

```
HRESULT SetLimitLoginAttempts([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

Determining Number of Connections Allowed from the Same IP Address (GetMaxIPs)

Use the ICIClientSettings interface **GetMaxIPs** method to determine the number of concurrent connections allowed from the same IP address.

Signature:

```
HRESULT GetMaxIPs(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying the Maximum Connections for IP Addresses (SetMaxIPs)

Use the ICIClientSettings interface **SetMaxIPs** method to choose the number of concurrent connections allowed from the same IP address.

Signature:

```
HRESULT SetMaxIPs([in] long val);
```

Determining for an IP Connection Limit (GetHasMaxIPs)

Use the ICIClientSettings interface **GetHasMaxIPs** method to check if IP addresses have a maximum number of allowed concurrent connections.

Signature:

```
HRESULT GetHasMaxIPs(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] VARIANT_BOOL *pVal);
```

0 = Unlimited

1 = Limited

Enabling an IP Address Connection Limit (SetHasMaxIPs)

Use the ICIClientSettings interface **SetHasMaxIPs** method to limit the maximum number of concurrent connections from the same IP address. Use the same method to turn off the limit.

Signature:

```
HRESULT SetHasMaxIPs([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

Determining the Maximum Concurrent Connections Allowed per User (GetMaxUsers)

Use the ICIClientSettings interface **GetMaxUsers** method to determine the maximum number of concurrent connections allowed for an account or Settings Level.

Signature:

```
HRESULT GetMaxUsers(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] long *pVal);
```

Specifying the Maximum Connections Allowed per User (SetMaxUsers)

Use the ICIClientSettings interface **SetMaxUsers** method to choose the maximum number of concurrent connections a user can make, either for an individual account or through a Settings Level.

Signature:

```
HRESULT SetMaxUsers([in] long val);
```

Determining if the Number of Concurrent Connections is Limited for Users (GetHasMaxUsers)

Use the ICIClientSettings interface **GetHasMaxUsers** method to check if a user account has a maximum number of allowed concurrent connections.

Signature:

```
HRESULT GetHasMaxUsers(
      [out, optional] VARIANT_BOOL *pInherited,
      [out, retval] VARIANT_BOOL *pVal);
```

0 = Unlimited

1 = Limited

Enabling a User's Connection Limit (SetHasMaxUsers)

Use the ICIClientSettings interface **SetHasMaxUsers** method to limit a user to a maximum number of concurrent connections. Use the same method to turn off the limit.

Signature:

```
HRESULT SetHasMaxUsers([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

Determining How Long a Connection can be Inactive (GetTimeOut)

Use the ICIClientSettings interface **GetTimeOut** method to find the number of seconds the server will allow a user's connection to be inactive before closing the connection.

Signature:

```
HRESULT GetTimeOut(
                  [out, optional] VARIANT_BOOL *pInherited,
                  [out, retval] long *pVal);
```

Specifying the Timeout Value (SetTimeOut)

Use the IICIClientSettings interface **SetTimeOut** method to choose the number of seconds the server will wait before closing a user's inactive connection.

Signature:

```
HRESULT SetTimeOut([in] long val);
```

Determining if a User Can be Timed Out (GetEnableTimeOut)

Use the ICIClientSettings interface **GetEnableTimeOut** method to check if a user will be timed out after their connection is inactive too long.

Signature:

```
HRESULT GetEnableTimeOut(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] VARIANT_BOOL *pVal);
```

0 = No

1 = Yes

Enabling Connection Timeout (SetEnableTimeOut)

Use the ICIClientSettings interface **SetEnableTimeOut** method to enable the timeout feature for a user or Settings Level. Use the same method to disable the timeout feature.

Signature:

```
HRESULT SetEnableTimeOut([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-2 = Inherit

Disk Quotas

The methods below are used to retrieve information about or change a user's disk quotas.

Determining if an Account has a Disk Quota (GetEnableDiskQuota)

Use the ICIClientSettings interface **GetEnableDiskQuota** method to determine if a user account's home folder space is restricted.

Signature:

```
HRESULT GetEnableDiskQuota(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] VARIANT_BOOL *pVal);
```

0 = Disabled

1 = Enabled

Limiting a User's Disk Space (SetEnableDiskQuota)

Use the ICIClientSettings interface **SetEnableDiskQuota** method to limit the amount of space a user can use in a home folder. Use the same method to turn off the limit.

Signature:

```
HRESULT SetEnableDiskQuota([in] SFTPAdvBool val);
```

0 = Disable

1 = Enable

-**2** = Inherit

Determining the Disk Quota Size (GetMaxSpace)

Use the ICIClientSettings interface **GetMaxSpace** method to determine the amount of space a user can use in a home folder. The number returned represents kilobytes.

Signature:

```
HRESULT GetMaxSpace(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Specifying a User's Disk Quota (SetMaxSpace)

Use the ICIClientSettings interface **SetMaxSpace** method to choose the amount of space, in kilobytes, a user can use in a home folder.

Signature:

```
HRESULT SetMaxSpace([in] long val);
```

Determining How Much Disk Space a User has Used (GetUsedSpace)

Use the ICIClientSettings interface **GetUsedSpace** method to determine the amount of disk space, in kilobytes, a user has used in a home folder.

Signature:

```
HRESULT GetUsedSpace(
                [out, optional] VARIANT_BOOL *pInherited,
                [out, retval] long *pVal);
```

Command Settings Interface Properties (ICICommandSettings)

The ICICommandSettings interface allows you to make changes to the settings of site custom commands. Access the ICICommandSettings interface with the ICISite **GetCommandSettings** method.

Example

```
Set cs = s.GetCommandSettings(aCommands(i))
```

When you change a custom command with the ICICommandSettings interface, you must call ICIServer's ApplyChanges method in order for the changes to take effect.

Retrieving or Changing the Name of a Custom Command (Name)

Use the ICICommandSettings interface **Name** property to retrieve the name of a custom command. You can also use the property to change the name.

Signature:

```
HRESULT Name([out, retval] BSTR *pVal);
HRESULT Name([in] BSTR newVal);
```

Retrieving or Changing the Description of a Custom Command (Description)

Use the ICICommandSettings interface **Description** property to retrieve the description of a custom command. You can also use the property to change the description.

Signature:

```
HRESULT Description([out, retval] BSTR *pVal);
HRESULT Description([in] BSTR newVal);
```

Retrieving or Changing the Path to the Executable of a Custom Command (Executable)

Use the ICICommandSettings interface **Executable** property to retrieve or change the path to a custom command's executable program.

Signature:

```
HRESULT Executable([out, retval] BSTR *pVal);
HRESULT Executable([in] BSTR newVal);
```

Enabling a Custom Command (IsEnabled)

Use the ICICommandSettings interface **IsEnabled** property to check if a custom command is available, and to either enable or disable a custom command.

Signature:

```
HRESULT IsEnabled([out, retval] VARIANT_BOOL *pVal);
HRESULT IsEnabled([in] VARIANT_BOOL newVal);
```

Retrieving or Changing Custom Command Parameters (Parameters)

Use the ICICommandSettings interface **Parameters** property to retrieve or change a custom command's parameters.

Signature:

```
HRESULT Parameters([out, retval] BSTR *pVal);
HRESULT Parameters([in] BSTR newVal);
```

Viewing or Requiring Parameters for Custom Commands (RequireParams)

Use the ICICommandSettings interface **RequireParams** property to check if a custom command must have parameters. Use the same property to require or parameters, or to allow custom commands without parameters.

Signature:

```
HRESULT RequireParams([out, retval] VARIANT_BOOL *pVal);
HRESULT RequireParams([in] VARIANT_BOOL newVal);
```

Requiring a Minimum Number of Parameters for Custom Commands (MinNumOfParams)

Use the ICICommandSettings interface **MinNumOfParams** property to check if a custom command must have a minimum number of parameters. Use the same property to set the minimum number of required parameters. You can use any integer from 0 to 10.

Signature:

```
HRESULT MinNumOfParams([out, retval] long *pVal);
HRESULT MinNumOfParams([in] long newVal);
```

Defining or Changing a Message for an Invalid Number of Command Parameters (MinNumOfParamsMsg)

Use the ICICommandSettings interface **MinNumOfParamsMsg** property to check or change the message sent when the wrong number of parameters is entered for a command.

Signature:

```
HRESULT MinNumOfParamsMsg([out, retval] BSTR *pVal);
HRESULT MinNumOfParamsMsg([in] BSTR newVal);
```

Redirecting Command Output to Clients (RedirectOutputToClient)

Use the ICICommandSettings interface **RedirectOutputToClient** property to check if output is sent to users. Also use the property to send output to users.

Signature:

```
HRESULT RedirectOutputToClient([out, retval] VARIANT_BOOL *pVal);
HRESULT RedirectOutputToClient([in] VARIANT_BOOL newVal);
```

Redirecting Command Output to a Log (RedirectOutputToLog)

Use the ICICommandSettings interface **RedirectOutputToLog** property to check if output is sent to a log. Also use the property to send output to a log.

Signature:

```
HRESULT RedirectOutputToLog([out, retval] VARIANT_BOOL *pVal);
HRESULT RedirectOutputToLog([in] VARIANT_BOOL newVal);
```

Enabling a Time Limit for a Custom Command (EnableProcessTimeOut)

Use the ICICommandSettings interface **EnableProcessTimeOut** property to check if a custom command has a time limit, and to either enable or disable the time limit.

Signature:

```
HRESULT EnableProcessTimeOut([out, retval] VARIANT_BOOL *pVal);
HRESULT EnableProcessTimeOut([in] VARIANT_BOOL newVal);
```

Specifying the Time Limit for a Custom Command (ProcessTimeOut)

Use the ICICommandSettings interface **ProcessTimeOut** property to check how long a custom command time limit will be, and to set a time limit, in seconds.

Signature:

```
HRESULT ProcessTimeOut([out, retval] long *pVal);
HRESULT ProcessTimeOut([in] long newVal);
```

Command Settings Interface Methods (ICICommandSettings)

The ICICommandSettings interface allows you to make changes to the settings of site custom commands. Access the ICICommandSettings interface with the ICISite GetCommandSettings method.

Example

```
Set cs = s.GetCommandSettings(aCommands(i))
```

When you change a custom command with the ICICommandSettings interface, you must call ICIServer's ApplyChanges method in order for the changes to take effect.

Retrieving a List of Users Allowed to Use a Custom Command (GetUserPermissions)

Use the ICICommandSettings interface **GetUserPermissions** method to retrieve a list of users or Settings Levels that have permission to use a custom command.

Signature:

```
HRESULT GetUserPermissions([out, retval] VARIANT *aUsers);
```

Listing Users Allowed to Use a Command (AddUserPermission)

Use the ICICommandSettings interface **AddUserPermission** method to add a user or Settings Levels to the list of those allowed to use a specific command.

Signature:

```
HRESULT AddUserPermission([in] BSTR bstrUser);
```

Prohibiting Users from Using a Custom Command (RemoveUserPermission)

Use the ICICommandSettings interface **RemoveUserPermission** method to remove a user or Settings Level from the list of those that have permission to use a custom command.

Signature:

```
HRESULT RemoveUserPermission([in] BSTR bstrUser);
```

Command Action Parameters Interface (ICICommandActionParams)

This interface is available in EFT Server 5.2 and later.

The **ICICommandActionParams** interface allows you to make changes to the settings of Execute Command Event Action. Access the **ICICommandActionParams** interface with **ICIEventAction Params** property (when its Type property returns **CommandAction**).

Example:

```
If action.Type = 1 then 'CommandAction
      Set commandParams = action.Params
EndIf
```

Retrieving or Changing Command to Execute (Command)

This property is available in EFT Server 5.2 and later.

Use the **ICICommandActionParams** interface **Command** property to retrieve or change the command to execute.

Signature:

```
HRESULT Command([out, retval] BSTR *pVal);
HRESULT Command([in] BSTR newVal);
```

Example:

```
commandParams.Command = "My_command"
```

Retrieving or Changing Parameters for Command (Parameters)

This property is available in EFT Server 5.2 and later.

Use the **ICICommandActionParams** interface **Parameters** property to retrieve or change the parameters for a command.

Signature:

```
HRESULT Parameters([out, retval] BSTR *pVal);
HRESULT Parameters([in] BSTR newVal);
```

Example:

```
commandParams.Parameters = "D:\MyFolder\MyScript.vbs –A -B"
```

Retrieving or Changing Working Folder for Command (WorkingFolder)

This property is available in EFT Server 5.2 and later.

Use the **ICICommandActionParams** interface **WorkingFolder** property to retrieve or change the working folder for the command.

Signature:

```
HRESULT WorkingFolder([out, retval] BSTR *pVal);
HRESULT WorkingFolder([in] BSTR newVal);
```

Example:

```
commandParams. WorkingFolder = "D:\MyFolder"
```

Certificate Information Interface Properties (ICICertInfo)

The ICICertInfo interface allows you to retrieve information about certificates. Access the ICICertInfo interface with either the ICISite GetTrustedCertificateInfo method or the ICISite GetPendingCertificateInfo method.

All properties of the ICICertInfo are read only.

Examples

```
Set ci = s.GetTrustedCertificateInfo(aCerts(i))
Set ci = s.GetPendingCertificateInfo(aCerts(i))
```

Retrieving a Certificate ID (ID)

Use the ICICertInfo interface **ID** property to retrieve the ID of a certificate.

Signature:

```
HRESULT ID([out, retval] long *pVal);
```

Retrieving a Certificate Description (Description)

Use the ICICertInfo interface **Description** property to retrieve the description of a certificate.

Signature:

```
HRESULT Description([out, retval] BSTR *pVal);
```

Retrieving a Certificate's Start Date (NotBefore)

Use the ICICertInfo interface **NotBefore** property to retrieve the first date the certificate became valid.

Signature:

```
HRESULT NotBefore([out, retval] BSTR *pVal);
```

Retrieving a Certificate's Expiration Date (NotAfter)

Use the ICICertInfo interface **NotAfter** property to retrieve the certificate's expiration date.

Signature:

```
HRESULT NotAfter([out, retval] BSTR *pVal);
```

Certificate Issuer

The properties below are used to retrieve information about a certificate issuer.

Retrieving a Certificate Issuers Information (IssuerOneLine)

Use the ICICertInfo interface **IssuerOneLine** property to find who issued a certificate, their email address, location, name and organization. The **IssuerOneLine** property will return all the information in one comma delimited string.

Signature:

```
HRESULT IssuerOneLine([out, retval] BSTR *pVal);
```

Retrieving a Certificate Issuer's Unit (IssuerUnit)

Use the ICICertInfo interface **IssuerUnit** property to retrieve the unit of a certificate's issuer.

Signature:

```
HRESULT IssuerUnit([out, retval] BSTR *pVal);
```

Retrieving a Certificate Issuer's Organization (IssuerOrg)

Use the ICICertInfo interface **IssuerOrg** property to learn the organization that issued a certificate.

Signature:

```
HRESULT IssuerOrg([out, retval] BSTR *pVal);
```

Retrieving a Certificate Issuer's Common Name (IssuerCName)

Use the ICICertInfo interface **IssuerCName** property to retrieve the issuer's common name on a certificate.

Signature:

```
HRESULT IssuerCName([out, retval] BSTR *pVal);
```

Retrieving a Certificate Issuer's Country (IssuerCountry)

Use the ICICertInfo interface **IssuerCountry** property to find the country of a certificate's issuer.

Signature:

```
HRESULT IssuerCountry([out, retval] BSTR *pVal);
```

Certificate Subject

The properties below are used to retrieve a certificate subject's information.

Retrieving a Certificate Subject's Information (SubjectOneLine)

Use the ICICertInfo interface **SubjectOneLine** property to retrieve a certificate subject's name, their location, organization and other information. The **SubjectOneLine** property will return all the information in one comma delimited string.

Signature:

```
HRESULT SubjectOneLine([out, retval] BSTR *pVal);
```

Retrieving a Certificate Subject's Unit (SubjectUnit)

Use the ICICertInfo interface **SubjectUnit** property to retrieve a certificate subject's unit.

Signature:

```
HRESULT SubjectUnit([out, retval] BSTR *pVal);
```

Retrieving a Certificate Subject's Organization (SubjectOrg)

Use the ICICertInfo interface **SubjectOrg** property to learn the organization that is the subject of a certificate.

Signature:

```
HRESULT SubjectOrg([out, retval] BSTR *pVal);
```

Retrieving Certificate Subject's Country (SubjectCountry)

Use the ICICertInfo interface **SubjectCountry** property to retrieve the country of a certificate's subject.

Signature:

```
HRESULT SubjectCountry([out, retval] BSTR *pVal);
```

Retrieving a Certificate Subject's Common Name (SubjectCName)

Use the ICICertInfo interface **SubjectCName** property to retrieve the certificate subject's common name.

Signature:

```
HRESULT SubjectCName([out, retval] BSTR *pVal);
```

Mail Action Parameters Interface (ICIMailActionParams)

This interface is available in EFT Server 5.2 and later.

The **ICIMailActionParams** interface allows you to make changes to the **MailAction** settings. Access the **ICIMailActionParams** interface with **ICIEventActionParams** property (when its **Type** property returns **MailAction**).

Example:

```
If action.Type = 2 then 'MailAction
      Set mailParams = action.Params
EndIf
```

Retrieving or Changing Message Recipients

These properties are available in EFT Server 5.2 and later.

Use the **ICIMailActionParams** interface **TOAddresses**, **CCAddresses**, and **BCCAddresses** properties to retrieve or change recipients in the To, CC, and BCC fields of a message. Separate multiple addresses with a semicolon.

Signature:

```
HRESULT TOAddresses([out, retval] BSTR *pVal);
HRESULT TOAddresses([in] BSTR newVal);
HRESULT CCAddresses([out, retval] BSTR *pVal);
HRESULT CCAddresses([in] BSTR newVal);
HRESULT BCCAddresses([out, retval] BSTR *pVal);
HRESULT BCCAddresses([in] BSTR newVal);
```

Example:

```
mailParams.CCAddresses = "rto@globalscape.com;foo@bar.com;"
```

Retrieving or Changing Message Subject

This property is available in EFT Server 5.2 and later.

Use the **ICIMailActionParams** interface **Subject** property to retrieve or change the subject of an e-mail message.

Signature:

```
HRESULT Subject([out, retval] BSTR *pVal);
HRESULT Subject([in] BSTR newVal);
```

Example:

```
mailParams.Subject = "Important! Need to talk"
```

Retrieving or Changing the Message Body

This property is available in EFT Server 5.2 and later.

Use the **ICIMailActionParams** interface **Body** property to retrieve or change the body of an e-mail message.

Signature:

```
HRESULT Body([out, retval] BSTR *pVal);
HRESULT Body([in] BSTR newVal);
```

Example:

```
mailParams.Body = "<HTML>This message was sent to you automatically by the
GlobalSCAPE EFT Server</HTML>"
```

Determining whether to CC Message to Client Associated with Event

This property is available in EFT Server 5.2 and later.

Use the **ICIMailActionParams** interface **CopyToClient** to determine whether to add the client associated with an event to a message's CC list.

Signature:

```
HRESULT CopyToClient([out, retval] VARIANT_BOOL *pVal);
HRESULT CopyToClient([in] VARIANT_BOOL newVal);
```

Example:

```
mailParams. CopyToClient = True
```

Cleanup Action Parameters Interface (ICICleanupActionParams)

This interface is available in EFT Server 5.2 and later.

The **ICICleanupActionParams** interface allows you to make changes to the Cleanup in Folder Event Action settings. Access the **ICICleanupActionParams** interface with **ICIEventActionParams** property (when its **Type** property returns **CleanupAction**).

Example:

```
If action.Type = 128 then 'CleanupAction
      Set cleanupParams = action.Params
EndIf
```

Retrieving or Changing the Period to Keep Files before Removing

This property is available in EFT Server 5.2 and later.

Use the **ICICleanupActionParams** interface **DaysToKeepFiles** property to retrieve or change the number of days to keep files prior to cleanup.

Signature:

```
HRESULT DaysToKeepFiles([out, retval] long *pVal);
HRESULT DaysToKeepFiles([in] long newVal);
```

Example:

```
cleanupParams.DaysToKeepFiles = 14 'Remove files older than 2 weeks
```

Specifying whether to Remove or Exclude Files from Cleanup (ExcludeFileMask)

This property is available in EFT Server 5.2 and later.

Use the **ICICleanupActionParams** interface **ExcludeFileMask** to check/specify whether the **FileMask** property specifies files to remove or exclude from cleanup (e.g., remove all the files except for those matching **FileMask**).

Signature:

```
HRESULT ExcludeFileMask([out, retval] VARIANT_BOOL *pVal);
HRESULT ExcludeFileMask ([in] VARIANT_BOOL newVal);
```

Example:

```
cleanupParams.ExcludeFileMask = False 'Remove only files matching FileMask during
cleanup
```

Retrieving or Changing Files to Remove (FileMask)

This property is available in EFT Server 5.2 and later.

Use the **ICICleanupActionParams** interface **FileMask** property to retrieve or change the files to remove or keep, depending on **ExcludeFileMask** property value.

Signature:

```
HRESULT FileMask([out, retval] BSTR *pVal);
HRESULT FileMask([in] BSTR newVal);
```

Example:

```
cleanupParams.FileMask = "*.txt"
```

Retrieving or Changing Folder to Cleanup (Folder)

This property is available in EFT Server 5.2 and later.

Use the **ICICleanupActionParams** interface **Folder** property to retrieve or change the folder to cleanup in.

Signature:

```
HRESULT Folder([out, retval] BSTR *pVal);
HRESULT Folder([in] BSTR newVal);
```

Example:

```
cleanupParams.Folder = "C:\EFT\Folder_to_cleanup"
```

Specifying whether to Cleanup All Subfolders Recursively (Recursive)

This property is available in EFT Server 5.2 and later.

Use the **ICICleanupActionParams** interface **Recursive** to specify whether to cleanup all subfolders recursively.

Signature:

```
HRESULT Recursive([out, retval] VARIANT_BOOL *pVal);
HRESULT Recursive([in] VARIANT_BOOL newVal);
```

Example:

```
mailParams.Recursive = False
```

OpenPGP Action Interface (ICIPgpActionParams)

This interface is available in EFT Server 5.2 and later.

The **ICIPgpActionParams** interface allows you to make changes to the OpenPGP Event Action settings. Access the **ICIPgpActionParams** interface with the **ICIEventAction Params** property (when its **Type** property returns **PGPAction**).

Example:

```
If action.Type = 32 then 'PGPAction
      Set pgpParams = action.Params
EndIf
```

Retrieving or Changing File Path for OpenPGP Event Action (FilePath)

This property is available in EFT Server 5.2 and later.

Use the **ICIPgpActionParams** interface **FilePath** property to retrieve the path to file(s) to manipulate with for OpenPGP Event action. You can also use the property to change the file(s) path.

Signature:

```
HRESULT FilePath([out, retval] BSTR *pVal);
HRESULT FilePath([in] BSTR newVal);
```

Example:

```
pgpParams.FilePath = "C:\MyFolder\*.*"
```

Retrieving or Changing Keys to Encrypt/Decrypt Data

This property is available in EFT Server 5.2 and later.

Use the **ICIPgpActionParams** interface **KeyIDs** property to retrieve the list of OpenPGP key IDs for encrypting/decrypting keys. You can also use the property to change the keys used to encrypt/decrypt.

Signature:

```
HRESULT KeyIDs([out, retval] SAFEARRAY(BSTR) *pVal);
HRESULT KeyIDs([in] SAFEARRAY(BSTR) newVal);
```

Example:

```
pgpParams.KeyIDs = Array("0x067194CB", "0x4567890F")
```

Determining Operation for PGP Event Action

This property is available in EFT Server 5.2 and later.

Use the **ICIPgpActionParams** interface **Operation** property to verify or specify whether an OpenPGP Event Action will encrypt or decrypt data.

Signature:

```
HRESULT Operation([out, retval] PGPOperation *pVal);
HRESULT Operation([in] PGPOperation newVal);
```

Example:

```
pgpParams.Operation = 0 'Encrypt
```

Retrieving or Changing the Passphrase for Signing or Decryption Key (Passphrase)

This property is available in EFT Server 5.2 and later.

Use the **ICIPgpActionParams** interface **Passphrase** property to retrieve the passphrase for the key to sign or decrypt data (only for Decrypt operation or for Encrypt operation with signing). You can also use the property to change the passphrase.

Signature:

```
HRESULT Passphrase([out, retval] BSTR *pVal);
HRESULT Passphrase([in] BSTR newVal);
```

Example:

```
pgpParams.Passphrase = "My_very_smart_passphrase"
```

Determining whether to Sign Encrypted Data (Sign)

This property is available in EFT Server 5.2 and later.

Use the **ICIPgpActionParams** interface **Sign** property to verify or specify whether encrypted data will be additionally signed (for **Encrypt** operation only).

Signature:

```
HRESULT Sign([out, retval] VARIANT_BOOL *pVal);
HRESULT Sign([in] VARIANT_BOOL newVal);
```

Example:

```
pgpParams.Sign = True
```

Retrieving or Changing Key to Sign Encrypted Data (SignKeyID)

This property is available in EFT Server 5.2 and later.

Use the **ICIPgpActionParams** interface **SignKeyID** property to retrieve the OpenPGP key ID for the signing key (for **Encrypt** operation only). You can also use the property to change the key used to sign.

Signature:

```
HRESULT SignKeyID([out, retval] BSTR *pVal);
HRESULT SignKeyID([in] BSTR newVal);
```

Example:

```
pgpParams.SignKeyID = "0x071894C0"
```

Event Rule Interfaces

The interfaces, methods, properties, constants, and enumerators below are used for managing Event Rules and their Events, Actions, and Conditions.

*Many of the interfaces below appear to be the same, but some have an "s" and some do not (e.g., ICIEventRule and ICIEventRule**s**; ICIEventAction and ICIEventAction**s**).*

Event Rules Interface

This interface is available in EFT Server 5.2 and later.

The **ICIEventRules** interface allows you to make changes to the Event Rule settings using the following methods:

```
HRESULT Add([in]long lIndex, [in] IDispatch* pdispParams, [out, retval]

HRESULT Count([out, retval] long* plCount);

HRESULT Delete([in] long lIndex); IDispatch** ppdispNewRule);

HRESULT Find([in] BSTR strName, [out, retval] IDispatch **ppdisp);

HRESULT Item([in] long lIndex, [out, retval] IDispatch **ppdispRule);
```

Event Rule Interface

This interface is available in EFT Server 5.2 and later.

The **ICIEventRule** interface allows you to make changes to the Event Rule settings using the following methods and properties:

```
HRESULT AddActionStatement([in] long lIndex, [in] EventActionType type, [in]
IDispatch* pdispParams, [out, retval] IDispatch** ppdispActionStatement);

HRESULT AddIfStatement([in] long lIndex, [in] long lPropertyID, [in]
ConditionOperator op, [in] VARIANT varConditionValue, [out, retval] IDispatch**
ppdispIfStatement);

HRESULT DeleteStatement([in] long lIndex);

HRESULT Params([in] IDispatch* newVal);
HRESULT Params([out, retval] IDispatch* *pVal);

HRESULT Statement([in] long lIndex, [out, retval] IDispatch** ppdispStatement);

HRESULT StatementsCount([out, retval] long* plCount);
```

Event Rule Statement Interface

This interface is available in EFT Server 5.2 and later.

The **ICIEventRuleStatement** interface allows you to make changes to the Event Rule settings using the **Type** property:

```
HRESULT Type([out, retval] EventRuleStatementType *pVal);
```

Event Rule Parameters Interface

This interface is available in EFT Server 5.2 and later.

The **ICIEventRuleParams** interface allows you to make changes to the Event Rule Condition settings using the following properties:

```
HRESULT Description([out, retval] BSTR *pVal);

HRESULT Description([in] BSTR newVal);

HRESULT Enabled([out, retval] VARIANT_BOOL *pVal);

HRESULT Enabled([in] VARIANT_BOOL newVal);

HRESULT Name([out, retval] BSTR *pVal);

HRESULT Name([in] BSTR newVal);
```

If Statement Interface

This interface is available in EFT Server 5.2 and later.

The **ICIIfStatement** interface allows you to make changes to the Event Rule settings using the following methods and properties:

```
HRESULT Condition([out, retval] IDispatch** ppdispCompoundCondition);

HRESULT ElseSection([out, retval] IDispatch** ppdispActionStatements);

HRESULT IfSection([out, retval] IDispatch** ppdispActionStatements);
```

Stop Action Parameters Interface

This interface is available in EFT Server 5.2 and later.

The **ICIStopActionParams** interface allows you to enable/disable the **Stop Processing** Action in Events.

```
HRESULT Enabled([out, retval] VARIANT_BOOL *pVal);

HRESULT Enabled([in] VARIANT_BOOL newVal);
```

Report Action Parameters Interface

This interface is available in EFT Server 5.2 and later.

The **ICIReportActionParams** interface allows you to make changes to transfer-related Events, such as when a file is uploaded or downloaded, using the following properties:

```
HRESULT CustomDate([out, retval] long *pVal);
HRESULT CustomDate([in] long newVal);

HRESULT DateFormat([out, retval] long *pVal);
HRESULT DateFormat([in] long newVal);

HRESULT FilterAndOr([out, retval] long *pVal);
HRESULT FilterAndOr([in] long newVal);

HRESULT FilterField1([out, retval] BSTR *pVal);
```

```
HRESULT FilterField1([in] BSTR newVal);

HRESULT FilterField2([out, retval] BSTR *pVal);
HRESULT FilterField2([in] BSTR newVal);

HRESULT FilterOperator1([out, retval] long *pVal);
HRESULT FilterOperator1([in] long newVal);

HRESULT FilterOperator2([out, retval] long *pVal);
HRESULT FilterOperator2([in] long newVal);

HRESULT FilterValue1([out, retval] BSTR *pVal);
HRESULT FilterValue1([in] BSTR newVal);

HRESULT FilterValue2([out, retval] BSTR *pVal);
HRESULT FilterValue2([in] BSTR newVal);

HRESULT FromDate([out, retval] VARIANT *pVal);
HRESULT FromDate([in] VARIANT newVal);

HRESULT Name([out, retval] BSTR *pVal);
HRESULT Name([in] BSTR newVal);

HRESULT OptionalParameters([out, retval] BSTR *pVal);
HRESULT OptionalParameters([in] BSTR newVal);

HRESULT Path([out, retval] BSTR *pVal);
HRESULT Path([in] BSTR newVal);

HRESULT ReportFileFormat([out, retval] long *pVal);
HRESULT ReportFileFormat([in] long newVal);

HRESULT ToDate([out, retval] VARIANT *pVal);
HRESULT ToDate([in] VARIANT newVal);
```

Event Action Interface

This interface is available in EFT Server 5.2 and later.

The **ICIEventAction** interface allows you to make changes to the Event Rule Action settings using the following properties:

```
HRESULT Type([out, retval] EventActionType *pVal);

HRESULT Params([out, retval] IDispatch** ppdispParams);
HRESULT Params([in] IDispatch* pdispParams);
```

Event Actions Interface Methods

This interface is available in EFT Server 5.2 and later.

The **ICIEventActions** interface allows you to make changes to the Event Rule Action settings using the following methods:

```
HRESULT Add([in] long lIndex, [in] EventActionType type, [in] IDispatch*
pdispParams, [out, retval] IDispatch** ppdispAction);
HRESULT Count([out, retval] long* plCount);
HRESULT Delete([in] long lIndex);
HRESULT Item([in] long lIndex, [out, retval] IDispatch** ppdispAction);
```

Action Statement Interface

This interface is available in EFT Server 5.2 and later.

The **ICIActionStatement** interface allows you to make changes to the Event Rule settings using the following methods and properties:

```
HRESULT FailSection([out, retval] IDispatch** ppdispActions);
HRESULT Action([out, retval] IDispatch** ppdispAction);
```

Action Statements Interface

This interface is available in EFT Server 5.2 and later.

The **ICIActionStatements** interface allows you to make changes to the Event Rule settings using the following methods:

```
HRESULT Add([in] long lIndex, [in] EventActionType type, [in] IDispatch*
pdispActionParams, [out, retval] IDispatch** ppdispActionStatement);
HRESULT Count([out, retval] long* plCount);
HRESULT Delete([in] long lIndex);
HRESULT Item([in] long lIndex, [out, retval] IDispatch** ppdispActionStatement);
```

Simple Condition Interface

This interface is available in EFT Server 5.2 and later.

The **ICISimpleCondition** interface allows you to make changes to the Event Rule Condition settings using the following properties:

```
HRESULT Not([out, retval] VARIANT_BOOL *pVal);
HRESULT Not([in] VARIANT_BOOL newVal);
HRESULT Operator([out, retval] ConditionOperator *pVal);
HRESULT Operator([in] ConditionOperator newVal);
HRESULT Property([out, retval] long *pVal);
HRESULT Value([out, retval] VARIANT *pVal);
HRESULT Value([in] VARIANT newVal);
```

Compound Condition Interface

This interface is available in EFT Server 5.2 and later.

The **ICICompoundCondition** interface allows you to make changes to the Event Rule Condition settings using the following methods and properties:

```
HRESULT Add([in] long lPropertyID, [in] ConditionOperator op, [in] VARIANT
varValue, [out, retval] IDispatch** ppdispSimpleCondition);
HRESULT Count([out, retval] long *plCount);
HRESULT Delete([in] long lIndex);
HRESULT Item([in] long lIndex, [out, retval] IDispatch** ppdispSimpleCondition);
HRESULT Operator([out, retval] LogicalOperator *pVal);
HRESULT Operator([in] LogicalOperator newVal);
```

Transfer Action Parameters Interface

This interface is available in EFT Server 5.2 and later.

The **ICITransferActionParams** interface allows you to make changes to transfer-related Events, such as when a file is uploaded or downloaded, using the following properties:

```
HRESULT AutoLogin([out, retval] VARIANT_BOOL *pVal);
HRESULT AutoLogin([in] VARIANT_BOOL newVal);
HRESULT Host([out, retval] BSTR *pVal);
HRESULT Host([in] BSTR newVal);
HRESULT Key([out, retval] BSTR *pVal);
HRESULT Key([in] BSTR newVal);
HRESULT KeyPass([out, retval] BSTR *pVal);
HRESULT KeyPass([in] BSTR newVal);
HRESULT LocalPath([out, retval] BSTR *pVal);
HRESULT LocalPath([in] BSTR newVal);
HRESULT Operation([out, retval] long *pVal);
HRESULT Operation([in] long newVal);
HRESULT Password([out, retval] BSTR *pVal);
HRESULT Password([in] BSTR newVal);
HRESULT Port([out, retval] long *pVal);
HRESULT Port([in] long newVal);
HRESULT Protocol([out, retval] long *pVal);
HRESULT Protocol([in] long newVal);
HRESULT PubKey([out, retval] BSTR *pVal);
HRESULT PubKey([in] BSTR newVal);
HRESULT RemotePath([out, retval] BSTR *pVal);
HRESULT RemotePath([in] BSTR newVal);
HRESULT User([out, retval] BSTR *pVal);
HRESULT User([in] BSTR newVal);
```

Folder Monitor Event Rule Parameters Interface

This interface is available in EFT Server 5.2 and later.

The **ICIFolderMonitorEventRuleParams** interface allows you to make changes to the Event Rule settings using the following properties:

```
HRESULT CheckHealth([out, retval] VARIANT_BOOL *pVal);
HRESULT CheckHealth([in] VARIANT_BOOL newVal);
HRESULT CheckHealthInterval([out, retval] long *pVal);
HRESULT CheckHealthInterval([in] long newVal);
HRESULT Enabled([out, retval] VARIANT_BOOL *pVal);
HRESULT Enabled([in] VARIANT_BOOL newVal);
HRESULT ForcedlyDisabled([out, retval] VARIANT_BOOL *pVal);
HRESULT ForcedlyDisabled([in] VARIANT_BOOL newVal);
HRESULT IncludeSubfolders([out, retval] VARIANT_BOOL *pVal);
HRESULT IncludeSubfolders([in] VARIANT_BOOL newVal);
HRESULT Name([out, retval] BSTR *pVal);
HRESULT Name([in] BSTR newVal);
HRESULT Path([out, retval] BSTR *pVal);
HRESULT Path([in] BSTR newVal);
```

Timer Event Rule Parameters Interface

This interface is available in EFT Server 5.2 and later.

The **ICITimerEventRuleParameters** interface allows you to make changes to Timer Event Rule settings using the following properties:

```
HRESULT DayOfMonth([out, retval] long *pVal);
HRESULT DayOfMonth([in] long newVal);
HRESULT DayIndex([out, retval] long *pVal);
HRESULT DayIndex([in] long newVal);
HRESULT DayPeriod([out, retval] long *pVal);
HRESULT DayPeriod([in] long newVal);
HRESULT FixedDate([out, retval] VARIANT_BOOL *pVal);
HRESULT FixedDate([in] VARIANT_BOOL newVal);
HRESULT LastPremature([out, retval] VARIANT_BOOL *pVal);
HRESULT LastTime([out, retval] VARIANT *pVal);
HRESULT MonthIndex([out, retval] long *pVal);
HRESULT MonthIndex([in] long newVal);
HRESULT MonthPeriod([out, retval] long *pVal);
HRESULT MonthPeriod([in] long newVal);
HRESULT Recurrence([out, retval] Recurrence *pVal);
HRESULT Recurrence([in] Recurrence newVal);
HRESULT TimeStart([out, retval] VARIANT *pVal);
HRESULT TimeStart([in] VARIANT newVal);
HRESULT WeekDayIndex([out, retval] long *pVal);
HRESULT WeekDayIndex([in] long newVal);
HRESULT WeekPeriod([out, retval] long *pVal);
HRESULT WeekPeriod([in] long newVal);
```

Event Properties

These properties are available in EFT Server 5.2 and later.

General properties:

- `Time (1)` – Current time
- `Name (2)` – Event name
- `Reason (3)` – Event reason
- `TimeStamp (4)` – Current time
- `DateStamp (5)` – Current date
- `MonitorFolderStatus (6)` – Folder monitoring status
- `EventName (7)` – Event name

Server Properties:

- `ServerRunning (1000)` – Whether server is running
- `ServerLogOldName (1001)` – Old log file name
- `ServerLogNewName (1002)` – New log file name
- `ServerLogOldPath (1003)` – Old log file path
- `ServerLogNewPath (1004)` – New log file path
- `ServerLogTime (1005)` – Log type

- `ServerLogFolder (1006)` – Log folder
- `ServerServerNodeName (1007)` – Name of the node server is running on

Site Properties:

- `SiteRunning (2000)` – Whether site is started
- `SiteName (2001)` – Site name
- `SiteAccountManagementURL (2002)` – PCI account management URL

Connection Properties:

- `LocalIP (3000)` – Server IP
- `RemoteIP (3001)` – Remote peer IP
- `LocalPort (3002)` – Server port
- `RemotePort (3003)` – Remote peer port
- `Protocol (3004)` – Connection protocol
- `WebTransferClientConnection (3005)` – Web-Transfer Client connection

Client Properties:

- `ClientLogin (4000)` – Client's login name
- `ClientPassword (4001)` – Client's password
- `ClientAccessGroup (4002)` – Whether client belongs to one of the permission groups
- `ClientEnabled (4003)` – Is client's account enabled
- `ClientSettingsLevel (4004)` – Client's settings template
- `ClientFullName (4005)` – Client's full name
- `ClientDescription (4006)` – Client's description
- `ClientComment (4007)` – Comment to client's account
- `ClientEMail (4008)` – Client's e-mail
- `ClientPhone (4009)` – Client's phone number
- `ClientPager (4010)` – Client's pager
- `ClientFax (4011)` – Client's fax
- `ClientHomeFolder (4012)` – Client's home folder
- `ClientHomeFolderIsRoot (4013)` – Is client's home folder root for them
- `ClientQuotaMax (4014)` – Client's disk quota (max)
- `ClientQuotaUsed (4015)` – Client's disk quota (currently used)
- `ClientInvalidLoginAttempts (4016)` – Client's invalid login attempt count
- `ClientCanChangePassword (4017)` – Has client permission to change their password
- `ClientIP (4018)` – Client's IP
- `ClientSSLAllowed (4019)` – Has client permission to use SSL encryption
- `ClientFTPAllowed (4020)` – Has client permission to connect via FTP
- `ClientSFTPAllowed (4021)` – Has client permission to connect via SFTP

- `CClientLastLogin (4022)` – Client's last login timestamp
- `ClientPasswordExpiration (4023)` – Client's password expiration date
- `ClientMustResetPasswordAtFirstLogin (4024)` – Must client change their password on initial login
- `ClientAccountExpirationDate (4025)` – Client's account expiration date
- `ClientAccountLocked (4026)` – Is client's account locked out
- `ClientCustomField1 (4027)` – Client's account custom field #1
- `ClientCustomField2 (4028)` – Client's account custom field #2
- `ClientCustomField3 (4029)` – Client's account custom field #3

File System Properties:

- `VirtualPath (5000)` – File virtual path
- `PhysicalPath (5001)` – File physical path
- `DestinationVirtualPath (5002)` – Destination file virtual path
- `DestinationPhysicalPath (5003)` – Destination file physical path
- `FolderName (5004)` – Folder
- `FileName (5005)` – File name
- `DestinationFolderName (5006)` – Destination folder
- `DestinationFileName (5007)` – Destination file name
- `FolderOperation (5008)` – Folder operation
- `FileCreationDate (5009)` – File creation date
- `FileCreationTime (5010)` – File creation time
- `FileSize (5011)` – File size
- `FileCRC (5012)` – File CRC
- `ReportPath (5013)` – Report file path
- `ReportContent (5014)` – Report content
- `ReportFileName (5015)` – Report file name

Enumerators and Constants

These enumerators and constants are available in EFT Server 5.2 and later.

The following enumerators and constants are used in the Event Rule interfaces.

- EventType - Type of Event trigger
- EventRuleStatementType - Action and Condition statement
- EventActionType - Type of Event Action
- ConditionOperator - Operators used in simple Condition
- LogicalOperator - Operators combining simple Conditions
- Recurrence - Frequency of timer triggering
- PGPOperation - PGP decrypt/encrypt operations

EventType

Server Events

```
OnTimer (0x1001) – Timer
OnServer_LogRotate (0x1002) – Log rotated
OnServer_ServiceStop (0x1003) – Service stopped
OnServer_Service_Start (0x1004) – Service started
Monitor_Folder (0x1005) – Folder monitoring
OnMonitor_Folder_Failed (0x1006) – Folder monitoring failed
```

Site Events

```
OnSiteStarted (0x2001) – Site started
OnSiteStopped (0x2002) – Site stopped
```

Connection Events

```
Connection_Connect (0x3001) – Client connected
Connection_ConnectFailed (0x3002) – Client connection failed
OnClientDisconnected (0x3003) – Client disconnected
```

Client Events

```
OnClientDisabled (0x4001) – Client disabled
OnClientQuotaExceeded (0x4002) – Client's disk quota exceeded
OnClientLoggedOut (0x4003) – Client logged out
OnClientLoggedIn (0x4004) – Client logged in
OnClientLoginFailed (0x4005) – Client login failed
OnClientPasswordChanged (0x4006) – Client's password is changed
OnClientCreated (0x4007) – New client created
```

File System Events

```
OnFileDeleted (0x5001) – File deleted
OnFileUpload (0x5002) – File uploaded
BeforeFileDownload (0x5003) – Before file download
OnFileDownload (0x5004) – File downloaded
OnFileRenamed (0x5005) – File renamed
OnFolderCreated (0x5006) – New folder created
OnFolderDeleted (0x5007) – Folder deleted
OnUploadFailed (0x5009) – File upload failed
OnDownloadFailed (0x500A) – File download failed
OnChangeFolder (0x500B) – Client changed their current folder
OnFileMoved (0x500C) – File moved
OnVerifiedUploadSuccess (0x500D) – Verified file upload succeeded
OnVerifiedUploadFailure (0x500E) – Verified file upload failed
OnVerifiedDownloadSuccess (0x500F) – Verifies file download succeeded
OnVerifiedDownloadFailure (0x5010) – Verified file download failed
```

EventRuleStatementType

```
ActionStatement (0) – Action statement
IfStatement (1) – Conditional statement
```

EventActionType

```
CommandAction (0x01) – Execute command
MailAction (0x02) – Send E-mail
DownloadAction (0x08) – Download file
UploadAction - Upload file
PGPAction (0x20) – PGP action
StopAction (0x40) – Stop rule execution
CleanupAction (0x80) – Cleanup in folder
ReportAction (0x100) – Generate report
```

ConditionOperator

```
Equals (0x01) – Equals
Less (0x02) – Less
LessOrEquals (0x04) – Less or equal
Contains (0x08) – Contains
Match (0x10) – Match (for file name matching with template such as '*.exe' etc)
MemberOf (0x20) – Member of
OneOf (0x40) – One of
StartsWith (0x80) – Starts with
```

LogicalOperator

```
LogicalOr (0) – OR
LogicalAnd (1) – AND
```

Recurrence

```
Custom (0) – Hourly
Daily (1) – Daily
Weekly (2) – Weekly
Monthly (3) – Monthly
Yearly (4) – Yearly
OneTime (5) – Trigger only once
```

PGPOperation

```
Encrypt (0) – Encrypt
Decrypt (1) – Decrypt
```

Refer to Determining Operation for PGP Event Action (Operation).