IPSWITCH
FILE TRANSFER

WS_FTP Professional 12

Security Guide

CHAPTER 1 Secure File Transfer

CHAPTER 2 Secure Sockets Layer (SSL)

CHAPTER 3 File Transfer Integrity

CHAPTER 4 Secure Shell (SSH)

CHAPTER 5 OpenPGP

CHAPTER 6 Using Firewalls

APPENDIX A FireScript Editor

Index

CHAPTER 1

Secure File Transfer

In This Chapter

Selecting a Secure Transfer Method

The method you use to implement secure file transfer depends on your security goals. The following table can help you choose the best security method for your needs:

	Need to configure Client?	Need to configure Server?	Encrypts Login?	Encrypts Command Channel?	Encrypts File Transfer?	Encrypts Actual File?
SSL	Y	Y	Y	Y	Y	N
SSH	Y	Y	Y	Y	Y	N
OpenPGP	Y	N	N	N	N	Y

Note:With both SSL and SSH, check with the FTP server Administrator to confirm the server type that is running at the address you want to post files to. If you do not know the server type and attempt to make an SSL or SSH connection to a server that does not support the necessary protocols, the connection will fail.

About SSL

SSL (Secure Socket Layer) is a protocol for encrypting and decrypting data sent across direct internet connections. When a client makes an SSL connection with a server, all data sent to and from that server is encoded with a complex mathematical algorithm that makes it difficult to decode anything that is intercepted. For more information, see *Secure Sockets Layer (SSL)* (on page 5).

About SSH

SSH (Secure Shell) is a security protocol that allows you to make a secure connection to a server that has the SSH and SFTP (Secure File Transfer Protocol) protocols installed.

SSH encrypts all communications to and from the client and server. When an SSH connection is made, SFTP is the protocol that is used to perform all tasks on that single secure connection. For more information, see *Secure Shell (SSH)* (on page 17).

About OpenPGP

OpenPGP is a key-based encryption method used to encrypt files so that only their intended recipient can receive and decrypt them. OpenPGP is used widely to secure e-mail communication, but its technology can also be applied to FTP.

OpenPGP works by using two cryptographic keys to secure files. A Public Key is used to encrypt the file so that only its corresponding Private Key can decrypt it. For more information, see *OpenPGP* (on page 21).

Note:Unlike SSL and SSH, OpenPGP is not a type of connection, but a method of encrypting a file prior to uploading it. As such, OpenPGP Mode can be used in conjunction with standard FTP, SSL or SSH connections.

Using Secure Transfer Encryption with OpenPGP

You can protect your files before, during, and after transfer with WS_FTP best-in-class 256-bit AES or other encryption options. This lets you combine the fully-integrated OpenPGP mode to encrypt individual files with secure file transfer over SSL/FTPS and SSH/SFTP connections to safely transfer and store your private and confidential files.

Using FIPS 140-2 Validated Cryptography

About FIPS (WS_FTP Professional)

FIPS (Federal Information Processing Standard) is a standard published by the U. S. National Institute of Standards and Technology (NIST), a non-regulatory agency of the U. S. Department of Commerce. NIST works to establish various standards that the U.S. military and various government agencies must abide by. Therefore, vendors, contractors, and any organization working with the government and military must also comply with these standards where they are required. Additionally, despite the fact that FIPS is a U.S.-developed standard, the Canadian government has similar policies requiring FIPS-validated software.

WS_FTP FIPS mode includes Triple DES, AES, and HMAC SHA-1 encryption of files during transfer. Other modes of encryption are not supported, as specified by FIPS 140-2.

Using FIPS mode (WS_FTP Professional)

When FIPS mode is enabled, WS_FTP encrypts any files sent via SSH, FTP with SSL, or HTTPS with a FIPS 140-2 validated cipher. You can still send unencrypted files or files with a lower-level of encryption via regular FTP.

To activate FIPS Mode:

1 From the Tools menu, select **Options**. The Program Options dialog opens.
2 In the left pane, select FIPS.
3 Select the **Operate cryptographic module in FIPS 140-2 mode** option.

If the option is already selected, but greyed out, this means your system administrator has enabled FIPS Mode by default. In this case, FIPS Mode can be disabled only by the administrator.

4 Restart (close, then re-open) the WS_FTP application.

Any transfers initiated from the WS_FTP application and using FTPS (FTP over SSL), SSH, or HTTPS protocols will be sent using the FIPS 140-2 validated cryptographic module.

Transfers over FTP protocol are still allowed, but they cannot use FIPS mode.

For more information, see *About FIPS* (on page 2).

Checking File Integrity

WS_FTP can be configured to verify file transfer integrity with algorithms that guarantee files are transferred without being tampered with between the source and destination locations. For more information, see *File Transfer Integrity* (on page 13).

Configuring Firewalls

WS_FTP lets you enter information about a particular firewall into a firewall configuration, which you can then use when connecting to an FTP site from behind that firewall. You can configure the firewall once, and then assign that firewall configuration to those sites that require it. For more information, see *Using Firewalls* (on page 29).

CHAPTER 2

Secure Sockets Layer (SSL)

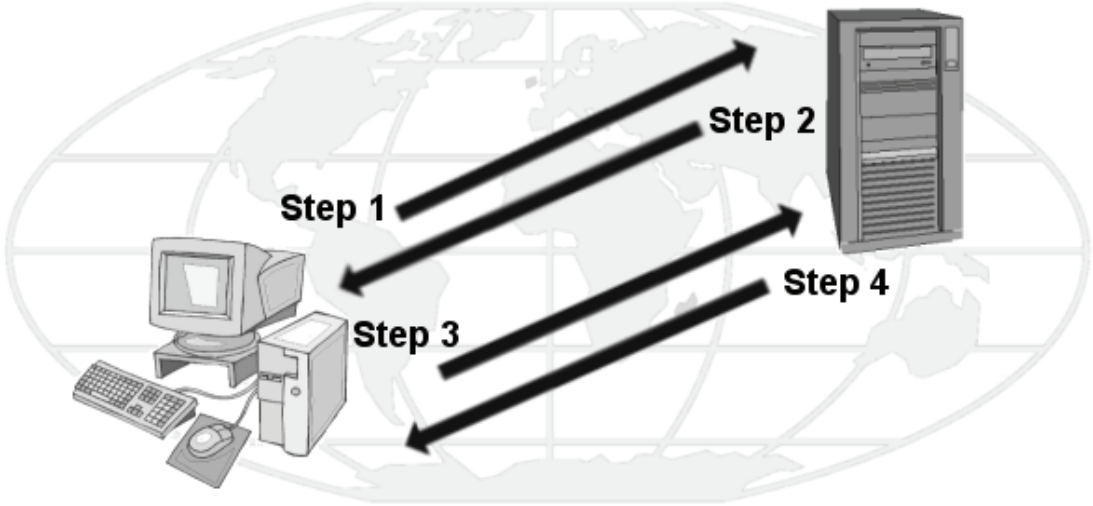
In This Chapter

Overview

SSL (Secure Sockets Layer) can be used in conjunction with FTP to provide increased security over standard FTP. This chapter provides an overview of the SSL protocol and describes how SSL works within WS_FTP.

SSL is a protocol for encrypting and decrypting data sent across direct internet connections. When a client makes an SSL connection with a server, all data sent to and from that server is encoded with a complex mathematical algorithm that makes it difficult to decode anything that is intercepted.

The following is a step-by-step illustration of how SSL works.



Step 1. The client makes the initial connection with the server and requests that an SSL connection be made. If Implicit SSL is used, the initial connection will be encrypted. If Explicit is used, the initial contact will be unencrypted.

Step 2. If the server is properly configured, the server will send to the client its certificate and public key.

Step 3. The client compares the certificate from the server to a trusted authorities database. If the certificate is listed there, it means the client trusts the server and will move to step 4. If the certificate is not listed there, the user must add the certificate to the trusted authorities database before going to step 4.

Step 4. The client uses that public key to encrypt a session key and sends the session key to the server. If the server asks for the client's certificate in Step 2, the client must send it at this point.

Step 5. If the server is set up to receive certificates, it compares the certificate it received with those listed in its trusted authorities database and either accepts or rejects the connection.

If the connection is rejected, a fail message is sent to the client. If the connection is accepted, or if the server is not set up to receive certificates, it decodes the session key from the client with its own private key and sends a success message back to the client, thereby opening a secure data channel.

The key to understanding how SSL works is in understanding the parts that make SSL itself work. The following is a list of these parts and the role each plays.

- **Client**. In this case, the client is Ipswitch WS_FTP Professional.

- **Certificate**. The Certificate file holds the identification information of the client or server. This file is used during connection negotiations to identify the parties involved. In some cases, the client's certificate must be signed by the server's certificate in order to open an SSL connection. Certificate files have the .crt ending.
- **Session Key**. The session key is what both the client and the server use to encrypt data. It is created by the client.
- **Public Key**. The public key is the device with which the client encrypts a session key. It does not exist as a file, but is a by-product of the creation of a certificate and private key. Data encrypted with a public key can only be decrypted by the private key that made it.
- **Private Key**. The private key decrypts the client's session key that is encrypted by a public key. The private key file has the .key ending. Private keys should NEVER be distributed to anyone.
- **Certificate Signing Request**. A certificate signing request is generated each time a certificate is created. This file is used when you need to have your certificate signed. After the Certificate Signing Request file is signed, a new certificate is made and can be used to replace the unsigned certificate.

Why use SSL?

SSL improves on the security of standard FTP by encrypting and securing most aspects of the connection. You can protect your files during transfer with WS_FTP Professional's best-in-class 256-bit AES or other encryption options. With this encryption method, you can transfer over SSL/FTPS connections to maintain the security and privacy of sensitive files.

Note:You cannot use SSL unless the FTP server supports it and has been configured to accept SSL connections. If you would like to use SSL, but your server does not support it, contact your server administrator.

How to make an SSL connection

To make an SSL connection with a server configured for SSL:

1. Create a site profile and select either **FTP/Implicit SSL** or **FTP/SSL (AUTH SSL)** when asked for the server type.
2. After you click **Connect**, Ipswitch WS_FTP Professional tells the server that you want to make an SSL connection. The server then transmits to you an identifying certificate, letting the client know who the server is. If that certificate is already listed in your Trusted Authority database, the connection is made.
3. If that certificate is not listed as a trusted authority, the Non-Trusted Authority dialog box appears.

4 Select the option you need and click **OK**. If the server does not require a certificate to be returned, the secure connection will be established. All data transmitted between you and the server will be encrypted.

If the server you are attempting to make a connection to asks WS_FTP to send back a certificate, follow the direction for Client Certificate Verification.

Client Certificate Verification

If the server you are attempting to make a connection to requires your client to send an identifying certificate back to the server, you must:

1 Configure the site and select either **FTP/Implicit SSL** or **FTP/SSL (AUTH SSL)** when asked for the server type.
2 Create a certificate. Refer to the section *Generating a Certificate* (on page 8) for more information.
3 Send the Certificate Signing Request file to your server administrator.
4 After the server administrator signs the Certificate Signing Request, it will be sent back to you.
5 When you receive the file, follow the directions for *Selecting a Certificate* (on page 9), selecting the new certificate to go in the **Certificate** box.
6 Connect to the server.

Generating a Certificate

To create an SSL certificate:

1 From the main window, select **Tools > Options**. The Program Options dialog appears.
2 Select **Client Certificates**.
3 Click **Create**. The **Create Client SSL Certificate** wizard appears.
4 Enter a name in the **Certificate** box. This will be the name of the certificate that is generated by WS_FTP.
5 Select a date you want the certificate to expire.
6 Enter and then reenter a pass phrase for this certificate. The pass phrase is used to encrypt the private key.

Note:It is important to remember this pass phrase. The pass phrase can be any combination of words, symbols, spaces, or numbers.

7 Click **Next** to continue.

Enter information in all of the Certificate Information boxes:

City/Town. City or town where you are located. (Ex. Augusta)

State/Province. State or Province where you are located. (Ex. Georgia)

Organization. Company or individual user name.

Common Name. This can be either the name of the person creating the certificate or the fully qualified domain name of the server associated with the host.

E-mail. E-mail address of the person the certificate belongs to.

Unit. Name of organizational unit. (Ex. Research and Development)

Country. The country you are in. This must be a valid two letter country code. (Ex. US)

8 After all of the boxes are filled in correctly, click **Next** to continue. If all of the boxes are not filled in, you cannot continue.

9 Review the information on the last dialog and click **Finish** to create the certificate.

If you are creating a certificate to be used by WS_FTP, you should send the certificate signing request (by E-mail) to your server administrator. If they require it, they will sign the certificate and return it to you. After you receive the certificate, you must import it into your certificate database.

Importing a Certificate

To use a certificate sent to you, or one you have generated by Ipswitch WS_FTP Server, you must import the certificate into your certificate database.

To import a certificate:

1 From the main window, select **Tools > Options**. The Program Options dialog appears.

2 Select **Client Certificates**, then click the **Import** button. The Import Certificate wizard appears.

3 Select a certificate, then click **Next**.

4 Select the private key file for that certificate, then click **Next**.

5 Enter the pass phrase that was used to create that certificate, then click **Next**.

6 Enter the name that you want to use to identify the certificate on your certificate list, then click **Next**.

7 Review the information on the final dialog and click **Finish** to add the certificate to the list.

Selecting a Certificate

Certificates are used on the site level, meaning that you must select a certificate for each site profile that you create (you can use the same certificate for all of your sites if you wish).

Certificates are selected on the Site Options: SSL dialog by choosing the certificate name from the **Client certificate** list box. The list box displays all certificates in the Program Options: Client Certificate dialog.

Trusted Authorities

The **Trusted Authorities** dialog stores a list of certificate names that are trusted by the user. To access the list of Trusted Authorities:

1. Click **Tools** > **Options**. The Program Options dialog opens.
2. Click to expand the **SSL** options, then click **Trusted Authorities**. The Trusted Authorities list opens.

Certificate Display

Issued To. Who the certificate was issued to.

Issued By. Who the certificate was signed by.

Expires. Date on which the certificate expires.

Adding a Certificate

To add a certificate to the database:

1. Click **Import** and select the path and file name for the certificate. The **Add Certificate?** dialog box appears.
2. Review the information on that dialog box and click **Yes** to add the certificate to the database.

Exporting a Certificate

To export a certificate from the Trusted Authorities database:

1. Select the certificate you want to copy out of your database.
2. Click **Export**.
3. Select the folder you want to copy the certificate to and enter the name you want to save the certificate file as.
4. Click **OK**.

Removing a Certificate

To remove a certificate:

1. Select the certificate to be removed.
2. Click **Remove**.
3. A warning appears advising you to export the certificate before you remove it. (Removing the certificate deletes the certificate file.)
4. Click **OK** to remove the certificate.

Non-Trusted Certificate

When you connect to a server using the SSL connection option, that server sends you a certificate. If that certificate is not listed on the Trusted Authority tab, or if it was not signed by a certificate on this list, this dialog box appears.

Certificate Information

Issued To. Name of the person or company to whom the certificate belongs.

Issued By. Name of the person or company who signed the certificate.

Active From. The date on which this certificate was activated.

Expires On. The date the displayed certificate will no longer be a valid certificate.

Certificate Options

Allow this connection only. If this option is selected, the connection will be made, but WS_FTP will still not recognize the certificate as a trusted authority. The next time you attempt to connect to this server, this dialog box appears again.

Trust this certificate. If this option is selected, the connection will be made and the certificate will be added to the trusted authority database in the Trusted Authority tab, so future connections can be made without you being prompted.

Do not allow this connection. If this option is selected, the connection will be terminated.

Using a NAT Firewall

When using a NAT (Network Address Translation) firewall, you may encounter problems when trying to use SSL encryption. To fix this, you should configure WS_FTP

and the firewall to allow incoming connections to your PC. WS_FTP needs to tell the server to connect to the external IP address, and the firewall should forward these incoming connections to your PC. You should also limit the number of ports that the firewall opens for these connections. In many cases, this will allow you to use SSL through a NAT firewall.

Note:If you are using Windows XP, you may be able to automatically configure your firewall to open the necessary ports and obtain the external IP address using UPnP. UPnP can be enabled on the Program Options: Firewall dialog.

To configure SSL through a NAT Firewall:

1. Select **Tools > Options**. The Program Options dialog appears.
2. In the left pane, select **Firewall**.
3. Select **Force PORT IP address**, then enter the IP Address of the NAT firewall.
4. Select **Limit local port range**, then enter the **Minimum** and **Maximum** port range.
5. Click **OK**.

CHAPTER 3

File Transfer Integrity

In This Chapter

Overview

WS_FTP can be configured to verify file transfer integrity with algorithms that guarantee files are transferred without being tampered with between the source and destination locations.

This chapter provides an overview of the File Transfer Integrity feature and describes how to use it within WS_FTP.

If supported by the FTP server, WS_FTP can perform a file integrity check on all transferred files. As part of the last step of the transfer, both the FTP client and the FTP server perform a cryptographic hash of the transferred file. If the values agree, both sides "know" that the file transferred is completely identical to the original. The results of the file checks are displayed in the WS_FTP log.

This feature detects data modification so connection hijacking attacks (when an attacker reads, inserts, or modifies files in transit) can be detected in file transfers.

So how does file transfer integrity work?

First, file transfer integrity is enabled on a site-by-site basis be selecting the **Check Transfer Integrity** option in the Site Options > Transfer dialog.

Once enabled for a site, the type of integrity checking done is controlled by the Program Options> File Integrity dialog. WS_FTP will use all the file integrity algorithms that are selected, in the order they are displayed, to check a file after transferring. For each algorithm, WS_FTP will check the response from the server to see if the server supports it (using the FEAT command). If the server returns an error, (usually “500 command not supported”), WS_FTP will try the next algorithm, and remember the ones that the server does not support. Support of a particular file integrity algorithm can be checked by looking for errors in the Connection Log after a file transfer.

WS_FTP will use the following commands to the server for integrity checking: XCRC, XMD5, XSHA1, XSHA256 and XSHA512. Although none of these commands have the weight of Internet standards, one or more of these commands are supported by most of the servers that support integrity checking.

If no file integrity algorithms are supported by the server, then, as a last resort, if the **Use size comparison in Transfer Integrity Checking** option is selected, WS_FTP will check the size of the recently transferred file by sending the SIZE command to the remote server. If the server supports the SIZE command (and not all servers do), WS_FTP will compare the value returned from the server with the size of the local file. Although this is not a guarantee of file integrity, it is a minimum check that the correct number of bytes were transferred.

The algorithm strength also impacts the time it takes to verify file integrity. The higher the algorithm strength, the longer the transfer verification takes.

How to set up File Transfer Integrity Checks

You can enable the File Transfer Integrity checking in the Site Options for individual sites or in the Folder Options (default site options). After you have enabled the options, you need to select the File Integrity Algorithm preferences and priorities.

File integrity algorithms can only be used for binary transfers to and from servers that support the optional FTP commands (for example, SSH and HTTP servers). Check with the FTP Administrator to confirm the algorithms that are supported on the FTP server you want to post files to.

Enabling the File Transfer Integrity option

To enable Transfer Integrity checking for a site:

1. On the toolbar, click **Connect** to open the Site Manager.
2. Select the site you want to enable Transfer Integrity checking for, then click **Edit**.
3. The Site Options dialog appears.
4. Select **Transfer**.
5. Select the **Check Transfer Integrity** checkbox.
6. Click **OK** to close the Site Options dialog.

To enable Transfer Integrity checking in the Folder Options (defaults):

1. On the toolbar, click **Connect** to open the Site Manager.
2. Select **Sites**, then click **Edit**. The Folder Options dialog appears.
3. Select **Transfer**.
4. Select the **Check Transfer Integrity** checkbox.
5. Click **OK** to close the Folder Options dialog.

After setting the **Check Transfer Integrity** in the Folder options, all new sites will check transfer integrity by default. Any existing sites will retain their previous settings; if you want to enable transfer integrity checking for existing sites, you must modify their profiles individually.

Selecting the File Transfer Integrity Algorithm preferences

When you select the File Transfer Integrity Algorithm preferences, you need to select the algorithms and set the priority of the algorithms you want to use. Some FTP servers support the optional file integrity algorithms and FTP commands used to calculate a transferred file's checksum or hash; however, each FTP server varies on the algorithms they support. Check with the FTP Administrator to confirm the algorithms that are supported on the FTP server you want to post files to.

To specify the file integrity algorithm for file transfers:

1. On the tool bar, click **Options** (or select **Tools** > **Options**) to open the Program Options dialog.
2. In the left pane, click **Transfers** > **File Integrity**. The File Integrity dialog opens.
3. Click the File Integrity options you want to use: CRC32, SHA1, MD5, SHA256, and SHA512.
4. Use the up and down arrows to set the preferred algorithm priority for file transfers.
5. Click **Use size comparison in Transfer Integrity Checking** to have the FTP client compare file size as a way to check transfer integrity if the FTP server does not support the selected file integrity algorithms.

CHAPTER 4

Secure Shell (SSH)

In This Chapter

Overview

SSH (Secure Shell) protocol, also known as SFTP (Secure File Transfer Protocol), can be used with FTP to provide improved security over standard FTP. This chapter describes how the SSH protocol is used within Ipswitch WS_FTP Professional.

SSH is a security protocol that lets you make a secure connection to a server that has the SSH and SFTP (Secure File Transfer Protocol) protocols installed.

SSH encrypts all communications to and from the client and server. When an SSH connection is made, SFTP is the protocol that is used to perform all tasks on that single secure connection. For maximum security, Ipswitch WS_FTP Professional uses SSH2 with the SFTP protocol to make SSH transfers.

Whereas FTP servers usually 'listen' on port 21 for connection, SSH servers use port 22.

Note: SSH can be used with a variety of authentication methods. However, WS_FTP only supports simple password authentication, public key authentication, and keyboard interactive authentication.

WS_FTP Professional supports SFTP/SSH2 only.

Why use SSH?

SSH improves on the security of standard FTP by encrypting all data transfer traffic, connection data, and passwords to eliminate eavesdropping, connection hijacking, and other attacks.

You can protect your files during transfers with WS_FTP best-in-class 256-bit AES or other encryption options. With this encryption method, you can transfer over SSH/SFTP connections to maintain the security and privacy of sensitive files.

You cannot use SSH unless the FTP server supports it and has been configured to accept SSH connections. If you would like to use SSH, but your server does not support it, contact your server administrator.

Exporting an SSH Public Key

1. From the main window, select **Tools > Options**. The Program Options dialog appears.
2. Select **SSH > Client Keys**.
3. Click **Export**. The Save As... dialog appears.
4. Enter a filename, then click **Save**.

Making an SSH connection (WS_FTP Professional)

Making an SSH connection requires little additional configuration to new or existing site profiles.

When creating a new site profile through the Connection Wizard, change the Server Type to **SFTP/SSH** when prompted by the wizard.

If editing an existing site profile:

1. In the toolbar, click **Connect** to open the Site Manager.
2. Select the site from the configured sites list, then click **Edit**. The Site Options dialog appears.
3. Select **Advanced.**
4. In **Server type**, select SFTP/SSH. Click **OK**.
5. Select an authentication method:
   - **Password** - If your server uses password authentication, then setup is complete. The next time you log onto this site, SSH will be used to secure the connection.

- **Public Key** - If your server uses public key authentication, select **Advanced > SSH**. Select the correct key pair from **SSH Keypair**. If no key pairs are available, you can *create* (on page 19) one.
- **Keyboard Interactive** - If your server is configured to use keyboard interactive authentication, the server will ask for the user's input. Ipswitch WS_FTP Professional will present a dialog automatically asking the user for a password, answer to a security question, etc., depending on the server's security settings.

Note:If you select Keyboard Interactive authentication, then *manual user input* is required. This means that automated file synchronization processes will require a *user's real-time interaction* (in contrast to the other methods, which automatically access a stored password or key file).

6 Click **OK** to close the Site Options dialog.
7 Click **Connect** to connect to the site.

When you use that profile to make a connection, the client will automatically attempt to make an SFTP/SSH connection on port 22 of that server.

Use the **Tools > Options > SSH** dialog to select which Ciphers/MAC (Message Authentication Code) to be used with an SSH connection.

Generating an SSH key pair (WS_FTP Professional)

You can generate an SSH key pair for use in authenticating an SSH connection to a server. Before you can use any SSH keys you create, however, you must provide the public key to the server administrator to install on the server.

To generate an SSH Key Pair:

1 From the main window, select **Tools > Options**. The Program Options dialog appears.
2 Select **SSH > Client Keys**.
3 Click **Create**. The SSH Client Key Pair Generation wizard appears.
4 Follow the on-screen prompts to complete the wizard.

Importing an SSH Key (WS_FTP Professional)

You can import an SSH public key to use with your SSH server's public key authentication feature.

To import an SSH Public Key:

1 From the main window, select **Tools > Options**. The Program Options dialog appears.

2 Select **SSH > Client Keys**.
3 Click **Import**. Follow the on-screen steps to complete the import.

CHAPTER 5

OpenPGP

In This Chapter

Overview

OpenPGP can be used with FTP to provide improved security over standard FTP. WS_FTP Professional provides the industry's first fully-integrated OpenPGP mode to encrypt individual files to enable secure file management before and after transfer. Only individuals with the appropriate key can decrypt OpenPGP files. The OpenPGP mode also supports AES, 3DES, and other ciphers; offers Signature (key) strength from 1,024 up to 4,096 bit; and provides support for RSA and Diffie-Hellman key types with expirable keys.

You can additionally protect your files before, during, and after transfer with Ipswitch WS_FTP Professional best-in-class 256-bit AES or other encryption options. This lets you combine the fully-integrated OpenPGP mode to encrypt individual files with secure file transfer over SSL/FTPS and SSH/SFTP connections to safely transfer and store your private and confidential files.

This chapter discusses how OpenPGP works within WS_FTP, outlines the steps to transfer a file using OpenPGP encryption, and provides scenarios that illustrate how OpenPGP can solve security problems.

WS_FTP contains software based on standards defined by the OpenPGP Working Group of the Internet Engineering Task Force (IETF) Proposed Standard RFC 2440. Ipswitch WS_FTP Professional can work with OpenPGP, PGP or GPGP keys.

How it works

OpenPGP is a key-based encryption method used to encrypt files so that only their intended recipient can receive and decrypt them. OpenPGP is used widely to secure e-mail communications, but its technology can also be applied to FTP.

OpenPGP works by using two cryptographic keys to secure files. A Public Key is used to encrypt the file so that only its corresponding Private Key can decrypt it.

Unlike SSL and SSH, OpenPGP is not a type of connection, but a method of encrypting a file prior to uploading it. As such, OpenPGP Mode can be used in conjunction with standard FTP, SSL or SSH connections.

The following is a step-by-step illustration of how OpenPGP Mode works with FTP.

Step 1. The file to be uploaded is encrypted using a Public Key that the file's intended recipient has previously provided.

Step 2. The encrypted file is uploaded to the FTP server.

Step 3. The intended recipient retrieves the file from the FTP server.

Step 4. Using the Private Key (which together with the Public Key used to encrypt the file initially comprises the Key Pair), the intended recipient decrypts the file and accesses its contents.

You can also use OpenPGP to encrypt and decrypt files in local storage.

Using OpenPGP mode to encrypt and decrypt files for transfer (WS_FTP Professional)

OpenPGP Mode allows you to encrypt files that you transfer by using OpenPGP keys. Encrypting a file with OpenPGP ensures that only people who have the appropriate key can decrypt the file. When OpenPGP mode is enabled for a remote site, encryption works as follows:

- When you upload a file to the site, if the file is not already encrypted, the file will be encrypted, then transferred. This is true whether the site is a local view or a remote site.

- When you download an encrypted file (a file with a .pgp extension) from a remote site, if you have the appropriate private key, the file will be transferred, then decrypted. The decrytped file will have the ".pgp" extension removed. For example, if the encrypted file was named spreadsheet.xls.pgp, then the decrytped file will be spreadsheet.xls.
- When you download an encrypted file from a local site, a local to local transfer, the encrypted file is transferred as is; it is not automatically decrypted.

Turning on OpenPGP Mode

OpenPGP Mode can be enabled for any remote server connection.

1. Connect to a remote server.
2. Select the remote server's tab.
3. Select **Tools > OpenPGP Mode**, or click **OpenPGP Mode** on the toolbar. The OpenPGP Mode dialog appears.
4. Select the OpenPGP Transfer method you would prefer to use from the options listed:
   - Encrypt the files using a key from your keyring. Select the Encryption keys to use to encrypt the files.
   - Sign the files using your private key as the digital signature. Select the Signing key to use to sign the files. Enter the **Passphrase** of signing key.
   - Select **Encrypt and Sign** to use both options together.
5. Click **OK**. OpenPGP Mode is now enabled for the duration of the connection or until it is disabled.
6. Upload or download files. When you upload a file, it will ask you to choose the OpenPGP key pair to use.

   Your key pairs, those that you have either generated or imported, are shown in **Program Options > OpenPGP > Key Pairs**.

   The uploaded, OpenPGP encrypted files will have the file extension: .pgp. For example, if you upload the file named chapter4.pdf, it will appear as chapter4.pdf.pgp on the remote site folder.

   If you download a file with the .pgp extension, if you have the appropriate private key, the file will be decrypted and verified. The file name will appear with it's original extension.

Turning off OpenPGP Mode

1. While connected to a remote server, select the remote server's tab.
2. Select **Tools > OpenPGP Mode**, or click **OpenPGP Mode** on the toolbar.
3. OpenPGP Mode is now turned off.

Enabling OpenPGP mode for a site by default (WS_FTP Professional)

You can configure a site to enable OpenPGP Mode automatically each time you connect to that site.

Note: OpenPGP Mode cannot be used when uploading files using the Synchronize or Script Utilities.

Note: You must have at least one OpenPGP Key in your keyring before you can configure a site to enable OpenPGP Mode automatically each time you connect. For more information, see *Generating an OpenPGP Key Pair* (on page 25) or *Importing OpenPGP Keys* (on page 26).

This site option can be set independently of the Program Option OpenPGP Mode setting.

To enable OpenPGP Mode for a site by default:

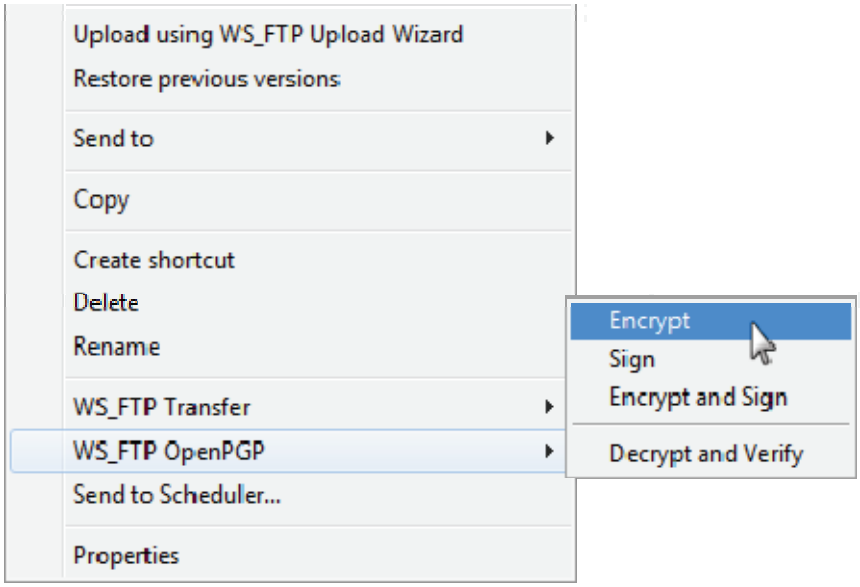
1. In the toolbar, click **Connect** to open the Site Manager.
2. From the list, locate and select the site for which you want to enable OpenPGP Mode automatically upon connecting. Click **Edit**. The Site Options dialog appears.
3. Expand **Advanced** and click **OpenPGP.** The OpenPGP options appear.
4. Select **Use OpenPGP Transfer Mode after connection.**
5. Select the OpenPGP Transfer method you would prefer to use from the options listed:
6. Encrypt the files using a key from your keyring. Select the **Encryption keys** to use to encrypt the files.
7. Sign the files using your private key as the digital signature. Select the **Signing key** to use to sign the files. Enter the **Passphrase** of signing key**.**
8. Select **Encrypt and Sign** to use both options together.
9. Click **OK** to save the settings and close the dialog.

Encrypting and decrypting local files (WS_FTP Professional)

You can access the encryption commands for local files and local transfers in two ways:

- OpenPGP menu on the local pane toolbar.

- Right-click menu on a local folder or file provides the same menu options as above:



Both menus offer the same set of OpenPGP commands:

- **Encrypt** - Select this option to use an encryption key to encrypt the file before transfer.
- **Sign** - Select this option to add your digital signature to the file. By adding your digital signature, you will allow others who download the file and who know and trust you to have more confidence in the file's integrity.
- **Encrypt and Sign** - Select this option to first encrypt the file, then to add your digital signature.
- **Decrypt and Verify** - Select this option to decrypt an OpenPGP encrypted file (identified by file extension .pgp) and verify the digital signature, if one is present.

When encrypting and signing files, you will be asked to select a key pair to use. Your key pairs, those that you have either generated or imported, are shown in **Program Options > OpenPGP > Key Pairs**.

Generating an OpenPGP key pair (WS_FTP Professional)

If you do not already have a personal OpenPGP Key Pair, WS_FTP can help you generate one.

To generate a OpenPGP Key Pair:

1. From the main window, select **Tools > Options**. The Program Options dialog appears.
2. Select **OpenPGP > Keys**.
3. Click **Create**. The OpenPGP Key Generation Wizard appears.

4 Follow the on-screen directions to complete the key creation process.

Related Topics

Importing OpenPGP Keys (on page 26)

Exporting OpenPGP Keys (on page 26)

Importing OpenPGP keys (WS_FTP Professional)

In order to encrypt files for others to open using OpenPGP Mode, you will need to import their OpenPGP keys. You can also import keys and keyrings that you use in another OpenPGP application.

Note: WS_FTP can only access keys and keyrings that exist as standalone files. If the keys or keyrings you wish to import are embedded in an e-mail or in a OpenPGP application, you must export them prior to importing them with WS_FTP.

To import a OpenPGP Key or Keyring:

1 From the main window, select **Tools > Options**. The Program Options dialog appears.
2 Select **OpenPGP > Keys**.
3 Click **Import**. The OpenPGP Key Import Wizard appears.
4 Follow the on-screen directions to complete the import process.

Exporting OpenPGP keys (WS_FTP Professional)

You can export OpenPGP Keys to use with other applications.

To export a OpenPGP Key:

1 From the main window, select **Tools > Options**. The Program Options dialog appears.
2 Select **OpenPGP > Keys**.
3 Select the key you wish to export, then click **Export**. The OpenPGP Key Export Wizard appears
4 Follow the instructions in the wizard to export your keys.

Scenario: Encrypting files for transfer to or from a remote site (WS_FTP Professional)

You can use OpenPGP encryption to secure a transfer from a local site to a remote site. A typical scenario is to secure a transfer that a colleague at a remote site needs to send to you. In this case:

1. You send a public key to colleague via email. You use WS_FTP to *generate* (on page 25) or *export* (on page 26) a key.
2. The colleague detaches the public key (from the email), encrypts and uploads the file using your public key, as follows:
    1. *Imports the key* (on page 26) into WS_FTP.
    2. Connects to the company FTP server.
    3. *Enables OpenPGP Mode* (on page 22) with your key used for encryption.
    4. Uploads the file.
3. You download and decrypt the file using your private key.

    To decrypt an OpenPGP encrypted file (identified by file extension .pgp), right-click the file in WS_FTP and select **OpenPGP > Decrypt and Verify**.

Related Topics

Using OpenPGP Mode to encrypt and decrypt transfers to or from a remote site (on page 22)

Scenario: Encrypting local files and local transfers (on page 27)

Encrypting and decrypting local files (on page 24)

Scenario: Encrypting local files and local transfers (WS_FTP Professional)

You can use OpenPGP encryption to secure files that your store on a local disk, or to secure a transfer from a local folder to another local folder. In this case, the transfers occur on the same machine. Typical scenarios are:

- You have files on your laptop that you want to secure in place as a precaution. You can encrypt the files with your public key, and when you need access, decrypt with your private key.
- You regularly archive files on an external hard drive, so you want to encrypt the files when you copy them to the drive.

Encrypting files for storage:

1. You use WS_FTP to *generate* (on page 25) or *import* (on page 26) a key.

   If you already have an OpenPGP key, you can select to use it for encryption. You can see the available keys in the Program Options > OpenPGP > Keys dialog.

   If you are the only one using the file, you do not need to send the public key to anyone else.
2. You select a file, right-click, then select WS_FTP> OpenPGP > Encrypt.

   The encrypted file is saved with a .pgp extension.

   For information about all the OpenPGP commands, see *Encrypting and decrypting local files* (on page 24).
3. When you want to use the file, you select it, right-click, then select OpenPGP > Decrypt and Verify.

   Your private key is used to decrypt the file. The file must have been encrypted with the associated public key.

Encrypting files for local transfer:

1. Connect to the local site and select the destination folder.
2. *Enable OpenPGP Mode* (on page 22) with your key used for encryption.
3. Upload the file. The file is saved in the destination folder; the file will have a .pgp extension.
4. When you want to use the file, download it, then right-click the file and select OpenPGP > Decrypt and Verify.

Related Topics

Encrypting and decrypting local files (on page 24)

Scenario: Encrypting files for transfer to or from a remote site (on page 27)

CHAPTER 6

Using Firewalls

In This Chapter

Multiple Firewalls

There are several reasons you might want to create more than one firewall configuration. If you use a laptop computer in different locations that have different firewalls, you will want to set up a firewall configuration for each location, so you can switch to the appropriate firewall configuration when you are in each location.

Another reason you might want to set up multiple firewall configurations is that your network could have more than one router connection as a firewall. In this case, you would assign a different firewall configuration to an FTP site depending on which part of the network you are working from.

Furthermore, you might have a number of trusted sites (for example FTP sites owned by your company) for which you would use a different firewall (or no firewall).

Firewall Types

The following table lists all conventional firewall types and the information about each that you will need to enter into WS_FTP.

Type of Firewall	**Information to enter in WS_FTP**
SITE hostname	Host Name (or Address), User Name (ID)
USER after logon	Host Name (or Address), User Name (ID), Password
USER with no logon	Host Name (or Address)
Proxy OPEN	Host Name (or Address)

Type of Firewall	Information to enter in WS_FTP
USER remoteID @remoteHost fireID	Host Name (or Address), User Name (ID), Password
USER fireID@remoteHost	Host Name (or Address), User Name (ID), Password
Transparent	User Name (ID), Password
USER remoteID@fireID @remoteHost	Host Name (or Address), User Name (ID), Password
SOCKS4 and SOCKS5	Host Name (or Address), User Name (ID), Password
HTTP	Host Name (or Address), User Name (ID), Password

Configuring a Firewall

To enter firewall information, you will need to get data about your firewall from your network administrator. For more information, see **Firewall Types** above.

For some router-based firewalls, you will want to use passive mode, in which the data connections are established by the FTP client (WS_FTP) rather than by the FTP site.

To configure a firewall:

1. Select **Tools > Options**.
2. Select the **Firewall** dialog.
3. Click **New**.
4. Enter the following information in the Firewall Properties dialog. For more information on the Firewall Properties, refer to the application Help.
   - **Name** - A descriptive name for this firewall configuration. You can use any name that helps you recall how or where this firewall configuration is used.
   - **Hostname** - The firewall Host Name or IP Address.
   - **UserID** - If needed, enter the UserID for the firewall.
   - **Password** - Enter the password for the firewall. We do not recommend selecting Save Password unless you are the only user of your computer.
   - **Type** - Select the firewall type from the drop-down list.
   - **Port** - Enter a port number.
5. When you click **Finish**, the firewall will be added to the Configured Firewalls list.

You can also assign the firewall configuration to the site, as described in *Using a Configured Firewall* (on page 31) below.

Using a Configured Firewall

After you have configured a firewall, you can then apply the firewall configuration to an FTP site.

1. Click **Connect** to open the Site Manager.
2. Select a site, then click **Edit**.
3. Select **Advanced**.
4. In the **Firewall** list, select a firewall configuration you want to use.

Using UPnP

If you are using Windows XP, you may be able to automatically configure your firewall to open the necessary ports and obtain the external IP address using UPnP.

To enable UPnP:

1. Click **Options** on the toolbar, or select **Tools > Options** from the menu. The Program Options dialog appears.
2. Select **Firewall**.
3. Select **Use UPnP for PORT Commands**.

APPENDIX A

FireScript Editor

What is a FireScript?

This appendix describes the purpose and syntax of the FireScript language and how it is used to make an FTP connection through a firewall.

A FireScript allows you to customize the sequence of commands and responses used to log in to an FTP server. This customization may be necessary if your FTP server requires any non-standard commands to be issued before or after logging in, or if certain types of firewalls are between the client and the server.

FireScripts are written in a custom FireScript language, developed specifically for use by Ipswitch WS_FTP Professional. FireScripts can perform the same functions that WS_FTP uses internally to connect to a host or firewall type. FireScripts, however, let you determine if and when these functions are used. In particular, the FireScript determines when to autodetect the host type, and when to go secure with an SSL connection. The script can choose whether or not to try the XAUTH command, and also whether it is necessary to log in to a user account after sending the user ID and password.

FireScript Components

A FireScript is broken into three sections: **fwsc**, **comment** and **script**. As in a Windows ini file, the name of the section appears alone on a line, in square brackets, followed by the rest of the section.

The **fwsc** section is internally structured with name=value pairs in the same manner as a Windows ini section. It contains identifying information about the script, and indicates what variables will be required by the script.

The **comment** section is free-form text intended for human readers. It is ignored by the script executable.

The **script** section contains the scripts executable portion and conforms to the FireScript syntax.

Below is an example FireScript demonstrating this layout.

[fwsc]

… other values not shown would typically include 'required=' and 'version='

[comment]

This is an example script that connects to an FTP proxy. It is incomplete because many of the commands required to connect have been deleted for clarity. The main purpose is to demonstrate the organization of the FireScript into three sections.

```
[script]
send ("OPEN %HostAddress") {}
tryssl;
send (“USER %HostUserId”)
{
case (300..399) :
```

… most of script not shown due to the size.

The fwsc Section

The **fwsc** section lets you specify information about the script in a manner similar to a Win.ini file. Most of the parameters are present for informational purposes. This includes the **author** and **version** fields. A few of the parameters are used by the script executive in determining whether or not to show the login dialog, and which IP address to use.

The parser recognizes and stores values for the following parameters:

fwsc Parameters	
Parameter	Meaning and Values
author	Informational only. Author of the FireScript.
version	Informational only. Version number of the script file.
verdate	Informational only. Date on which this version was updated.
required	A comma delimited list of fields that must be present for the FireScript to execute. The login dialog is displayed if all required fields are not present, and the **Connect** button is disabled until all required fields have been filled in.
preask	A comma delimited list of fields that are not required but which, if not present, will cause the login dialog to be displayed.
connectto	'firewall' or 'host'. This parameter tells Ipswitch WS_FTP Professional which IP address to use when establishing the connection.

Unrecognized parameters are ignored.

The Comment Section

Use the **comment** section to describe the actions of the FireScript. The FireScript code should be well described, so it will be easier to understand and update later. The FireScript executive ignores the comment section.

You can also insert comments in the script section by using the '//' comment delimiter as in C++ and Java. Any text on a line following the '//' sequence is ignored by the parser.

The Script Section

The **script** section consists of a sequence of statements that send commands to the firewall or to the FTP server. Some of these statements have results, or trigger responses from the firewall or FTP server. There is a simple control structure that allows the script to take different paths of execution, based on these results or responses.

The Connection Sequence

A request for connection to an FTP site comes from user actions in either the Classic or Explorer interface, or by one of the Ipswitch WS_FTP Professional utilities such as Find or Synchronize. Sometimes, additional connections are requested by the Transfer Manager to resume or to speed up transfers. All connections are created by the CreateConnection function in the Ipswitch WS_FTP Professional API.

The connection sequence consists of two stages.

- Stage 1: Establish the connection with either the firewall or the FTP server.
- Stage 2: Send commands to log in and authorize the connected user. It is during this stage that the commands in a FireScript are executed.

The first stage works the same whether Ipswitch WS_FTP Professional is using a FireScript or using one of its internal firewall types. Before executing the script, Ipswitch WS_FTP Professional checks the **fwsc** section for the list of fields marked as **required** and **preask**. If any are missing, it displays the login dialog. If the user fills in all required information and presses **Connect**, Ipswitch WS_FTP Professional then checks the **connectto** field. Depending on this field, it will either establish a connection to the firewall's IP address and port, or to the FTP server's IP address and port. If this field is not present, Ipswitch WS_FTP Professional defaults to the IP address of the firewall, if present.

After the connection is established successfully, and a valid socket is opened, Ipswitch WS_FTP Professional calls the FireScript executive to execute the FireScript. If the Firelogs in correctly and returns success, the CreateConnection function returns the authorized connection to the caller.

The FireScript Language

The FireScript language contains a limited version of elements you may be familiar with if you have written scripts or programs in other languages. It uses variables, declarations, and statements to perform actions and direct program flow. Each of these elements is described in the following sections.

Syntactically, FireScript statements are terminated by semicolons. They may therefore extend across multiple lines, and you can have multiple statements on a line. A string however, may not span lines. The final closing quote must appear on the same line of source code as the opening quote. For example, the code below is valid:

but the following is not:

FireScript Variables

Firescripts work with the login information provided by Ipswitch WS_FTP Professional. This includes at least the user IDs and passwords, the IP address and port of the FTP server, and sometimes the IP address and port of the firewall. These fields are often read from a site profile, an FTP URL, or from the command line. As described before, if some of the required information is missing, the connect sequence presents the login dialog so that the user can enter it interactively. The script executive stores this information in a set of intrinsic variables before beginning execution. In addition there are intrinsic variables that contain the results of the last command issued. These are set by the script executive after such statements are executed.

The syntax for using a variable depends on the statement or expression in which it is used. Below is a list of all the intrinsic variables:

FireScript Intrinsic Variables	
Variable	Meaning and Usage
FwUserId	The user's user ID on the firewall. Some firewalls require users to log in to the firewall before allowing other connections to be made through the firewall.
FwPassword	The user's password on the firewall. Required if the user must log in to the firewall.
FwAccount	Account on the firewall. Required if the user must specify an account on the firewall. Practically unheard of but included in case required.
FwAddress	The IP address of the firewall. Required if the user must connect to the firewall, and have the firewall in turn connect to the FTP server and act as proxy.

HostUserId	The user's ID on the FTP server. Almost always required. Specify 'anonymous' if the user does not have a user ID on the server.
HostPassword	The user's password on the FTP server. Almost always required in conjunction with a user ID. Use your email address as the password when using 'anonymous' for the user ID.
HostAccount	The user's account on the FTP server. To access certain information in some operating systems, FTP servers on those systems require an account to be sent after successful login with user ID and password.
HostAddress	The IP address of the host. The script executive may connect directly to this address, or will send the address to a firewall that will act as a proxy.
LastFtpCode	The 3-digit, numeric code of the last response received from the FTP server or firewall. For example, after a successful login, the LastFtpCode would be 230.
LastReply	The text of the last response from the server. e.g. "230 user logged in"

FireScripts neither need nor use user-defined variables, so there are no variable declarations. Also, since the FireScript cannot directly set the value of one of the intrinsic variables, there is no need for any assignment statements.

String Expansion

Some of the commands and functions in the FireScript language take strings as arguments. To these you may either pass a string variable or a string literal surrounded by double quotes, e.g. “This is a string.” To put a double quote inside a string, preface it with the percent sign '%'. The percent sign '%' is used as an escape character to embed variables and quote characters in strings.

The sequence %% is replaced by a single %.
The sequence %" is replaced by ".
% followed by the name of a variable is replaced by the value of the variable.

For example, the script statement below:

If HostAddress is equal to “ftp.ipswitch.com” when this script is invoked, the command will be expanded to:

the expression,

will be expanded at runtime to:

and the statement

the expanded string sent will be:

Passing a string variable is equivalent to, but faster than passing a string literal that expands the variable.

Example:

is equivalent to but faster than

Function Expressions

Currently the FireScript language does not allow full-blown expressions. It does include two function expressions with some boolean operators for evaluating the state of variables. They are **contains** and **isempty**. The boolean operators supported are **not** and **and**.

The **contains** function takes two strings and returns **true** if the second string is found in the first. The search is case sensitive. Both strings are expanded first.

The **isempty** function takes a string and returns **true** if there are any characters in the string. You can use it to test if a value was specified for one of the intrinsic variables.

The **not** boolean operator reverses the value returned by the function expression.

Example:

If the HostAccount variable contains the value 'usr987i'
isempty (HostAccount) will return false but
not isempty(HostAccount) will evaluate to true.

The **and** boolean operator requires all specified conditions to be true.

Example, If the HostAccount variable contains a value such as 'usr987I'
The last reply from the server is "230 User logged in, please send account"

then the following expression will evaluate to true:

FireScript Statements

The FireScript language includes several types of statements. Statements cause actions to be taken, or direct the flow of execution of the script. The following sections describe the types of statements.

Switch Statements

The **send** statement and the **xauth** statement are called switching statements, because they imply an immediate switch statement based on the server response. The switch statement contains **case** statements very similar to Java and C++ case statements, except the conditions are not constants checked against a single expression.

A switching statement such as **send** and **xauth** is always immediately followed by a set of case statements between curly braces { <case statements> }. The set of case statements may be empty, in which case there is nothing between the curly braces, but the braces must be present.

Example of Switch Statement:

The **send** statement takes a single argument, the string to be sent to the server. The string is expanded before it is sent. The maximum legal length for the expanded string is about 512 bytes, the maximum length of an FTP line. The send command then waits on a response from the server and evaluates the response against the conditions in each of the enclosed case statements.

The **xauth** statement takes no arguments. It examines the welcome banner for an xauth invisibly supplied by Ipswitch WS_FTP server. If it is not connected to Ipswitch WS_FTP server or cannot find the invitation, **xauth** does nothing, and the case statements are not evaluated. If it does find the invitation, it encodes the user ID and password and sends the xauth command to the server. It then waits on the response and evaluates it against the case statements just as the **send** command does.

Case Statements

Case statements are enclosed in switching statements. A case statement lists a set of conditions that the server response must satisfy for the case to be activated.

The list of conditions is followed by a colon ':'.

Case statements are processed in the order in which they are listed until the first match is found.

After a match is found for the conditions in a case statement, then the nested statements are executed.

A case condition may be a list of FTP codes and code ranges, a function expression, or one of the special cases, **any** and **timeout**.

If a case includes a list of FTP codes/ranges, the list must appear first, followed by any function expressions. The list is comma separated and enclosed in parentheses. Each item in the list must either be a single 3-digit code, or a range specified by two 3-digit codes separated by a double period '..'. The range is inclusive and it is recommended that the lower bound be specified first.

The special cases **any** and **timeout** must appear by themselves.

Examples of Case Statements

The following case condition will match if the returned FTP code is either 226 or 231.

The following case conditions will match if the returned FTP code is either 226 or 231, or between 250 and 299 inclusive. So 250 itself will match, as well as 251, 252 etc. up to 299

The following case conditions will match if the returned FTP code is in the 300s and the returned string contains the text “email address”.

The following case conditions will match if the returned FTP code is 500 or greater and the returned string contains the specified error message

If a case contains more than one condition they must be separated by **and**. The **and** operator specifies that all the listed conditions must be satisfied. So in the previous example, the FTP code must be between 500 and 599 AND the last reply must also contain the specified string. Both must be true. If either is false, the case will not match.

The **not** operator reverses the result of a function. We may for example want to make sure that the last response does not contain a certain string. For example:

There is no **or** operator. The same logic may be applied by using multiple case statements.

The following case condition will match if the send command timed out.

Case **any** is the catch all case, and if present should be the last case in the enclosing list. If it is followed by other case statements they will never be evaluated.

For example, the following case condition will always match.

If case statements overlap and two case statements would match the response, then the first one encountered will be executed.

Example:

If the case with the contains function appeared after the one without it, it would never get evaluated.

Continue

Unlike C and C++, execution inside a case statement does not fall through to the next case statement. Only the statements listed under the activated case are executed. Then execution continues at the next statement after the enclosing switching statement. The **continue** statement jumps to the statement following the enclosing switching statement. It does the same thing as a break, inside of a C/C++ switch statement, except it is not absolutely necessary.

Switching statements may not be nested. That is, neither a **send** statement nor an **xauth** statement may appear inside a **case** statement.

Jumps and Labels

A jump statement transfers execution to another part of the script. The jump destination must be defined by a label that also appears in the script. The Ipswitch example FireScripts use jumps to different code sequences from inside case statements, so the code that gets exedepends on which case was activated.

A label declaration consists of the word label, followed by the name of the label and a semi

A jump statement consists of the word jump, followed by the name of the jump destination, and a semicolon.

A label may not appear inside a case statement. You can't jump into a case statement.

Return

The return statement acts like a function in that it takes a single parameter, either true or false to indicate success or failure. It terminates script execution and returns to the caller. If it returns true, the connection is assumed to be logged in and authorized. If it returns false, the caller may either try again or abandon the connection.

Autodetect

The **autodetect** statement examines the last reply from the server to help determine the host type of the FTP server to which it is connected. Autodetect expects to examine the welcome banner so the statement should be placed immediately after the welcome banner is returned. Here are two example banners returned from two popular FTP servers. Autodetect would detect the first as being a Microsoft NT server, and the second as being an Ipswitch WS_FTP server.

If connection was made directly to the host FTP server and the welcome banner was already returned before the script begins execution, then **autodetect** should be the first statement in the script. If the connection was made to the firewall and the welcome banner from the host FTP server becomes available later in the script, the **autodetect** statement should be placed at that point. If the firewall swallows or replaces the welcome banner from the FTP host, or for some other reason, the FTP client never

sees the welcome banner, then leave out the **autodetect** statement. Ipswitch WS_FTP Professional will try to determine the host type after the script executes.

Autodetect does nothing if the host type in the site profile is set to anything other than 'Auto Detect.' The **autodetect** statement has no return value and does not change the flow of the script.

SSL Statements

The **tryssl** and **gossl** commands attempt to open a secure channel with the server via SSL. The difference between them is that if **gossl** fails, the script will terminate and return false, while if **tryssl** fails, the script will continue. The commands can appear more than once in the program. If a secure connection was not requested, or has already been established, the commands will do nothing. If the commands fail to go secure, they will display a message box to the user, asking the user if she wishes to continue in the clear, try again for SSL later in the sequence, or abandon the connection. If the user chooses to continue in the clear, future calls to **tryssl** or **gossl** will do nothing.

When the script completes, the script executive checks the SSL status of the connection to make sure that a request for a secure connection was honored. In the site profile, if the user selected **Use SSL**, then the script executive will issue a warning to the user if the connection is not secure. At this point, the user may abandon the connection. This warning is issued if a secure connection was attempted and the user chose to continue in the clear.

The placement of the attempts to open an SSL channel can be very important, depending on the type of firewall through which the script is connecting.

FireScript Key Words

Below is a complete list of all the keywords used and understood by the language. You may not use these words as label names.

gossl	tryssl	autodetect
send	xauth	case
continue	and	not
any	timeout	return
jump	label	true
false		

FireScript reserved words

The following words are reserved for future versions of the language and the parser. You should not use these words to name your labels.

switch	if	for
next	while	loop
break	function	int
bool	string	var
password	or	

FireScript statements

gossl	tryssl	autodetect
send	xauth	jump
return	continue	

FireScript intrinsic functions

contains	isempty

FireScript intrinsic variables

FwUserId	FwPassword	FwAccount
FwAddress	HostUserId	HostPassword
HostAccount	HostAddress	LastFtpCode
LastReply		

In this Chapter

Index

T

U