SFXCL Automation Tips

1. Introduction

SecureFX® provides a command-line utility named SFXCL, a console application that can be used in automated file transfer scenarios. This document details "best practices" to employ in your efforts to implement an automated file transfer solution involving SFXCL.

The best practices to be employed when using SFXCL as a component of an automated file transfer project are as follows:

- Use correct command-line syntax
- Use a log file to facilitate troubleshooting
- Test manually before implementing automation

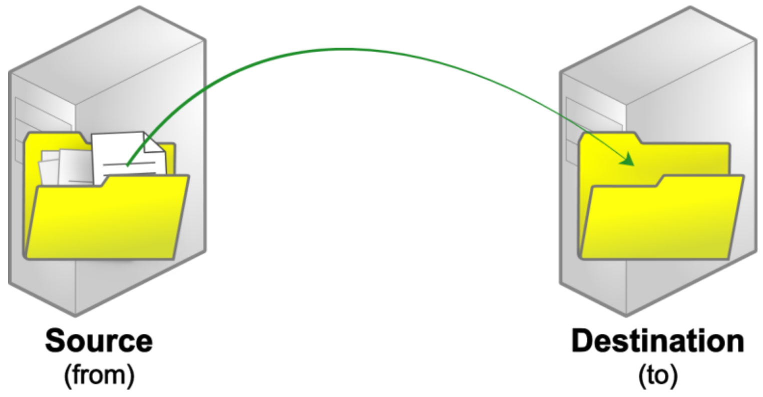
In addition to the best practices described above, this document provides information about SFXCL exit codes, using wildcards, ASCII vs. binary transfers, and how to generate unique log file names. Example batch files and VBScript code are provided as well.

In an effort to make patterns of usage more easily recognizable, components of an SFXCL command line that refer to a remote machine will be colored red throughout this document (associate "*Red* = *Remote*").

Note that while SFXCL can be used in an automated transfer scenario, SFXCL itself does not currently provide any scheduling functionality. Scheduling a task to run SFXCL is done with a third party scheduling service such as the "Scheduled Tasks" or "Task Scheduler" component that comes standard with Windows operating systems.

2. Use Correct Command-Line Syntax

File transfers involve copying one or more files *from* one location known as the "source" *to* another location known as the "destination".



The general syntax that needs to be used for any SFXCL operation is defined in the "**Getting Started / Command-Line Utility**" topic in the SecureFX help documentation:

```
SFXCL [options] source destination
```

Any options are placed prior to the source specification on the SFXCL command line. Source specifications are always placed prior to the destination specification on the SFXCL command line. Violating this positional structure will typically result in an error, "Copying local files not supported", or unexpected behavior that can be difficult to troubleshoot.

Sources and destinations can be specified using any of the following formats:

- **URL** specifications
- **Session** specifications
- **Local Path** specifications

2.1. URL Specifications

A URL specification makes use of an "ad hoc" connection in which all the information needed for SFXCL to connect to a remote host, authenticate, and know which files/folders to act upon is specified in the form of a URL. The general syntax for a URL specification is as follows:

```
protocol://user:password@host/file_or_folder_path
```

Explanation of URL components:

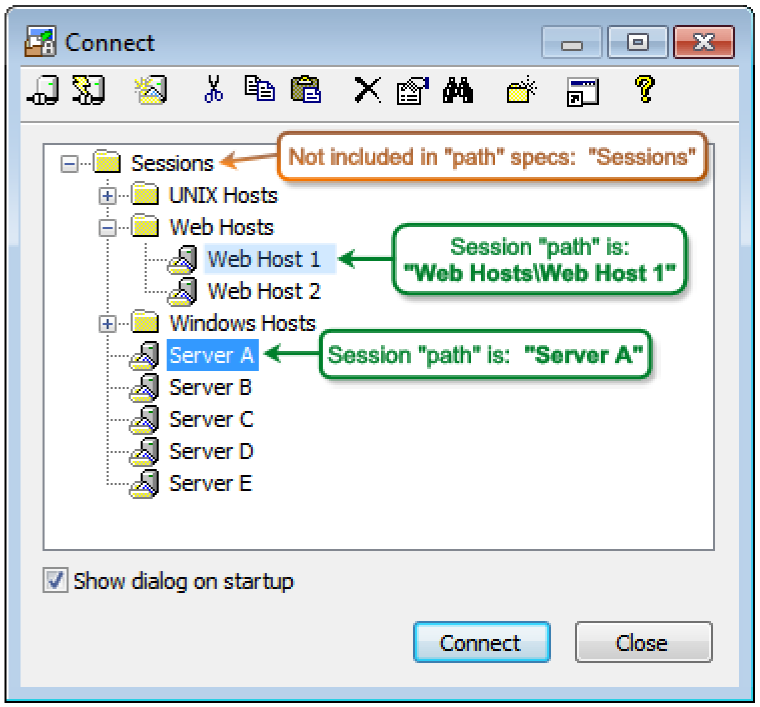
Protocol	Represents the communication protocol used to connect to the remote machine (file transfer server). This can be one of the following choices: `scp` SecureFX's "SCP" protocol `sftp` The "**S**SH **F**ile **T**ransfer **P**rotocol" `ftps` **FTP** over **S**SL `ftp` Plain-text FTP
User	Represents the account name used to authenticate to the remote machine (file transfer server)
Password	Represents the password used to authenticate to the remote machine (file transfer server)
Host	The IP address or host name of the remote machine (file transfer server)
file_or_folder_path	Represents a file or folder on the remote machine

2.2. Session Specifications

The general syntax for a session specification is as follows:

```
/S  "session_path"  "file_or_folder_path"
```

The `session_path` represents the "path" to a saved session as it appears in the **Connect** dialog in the SecureFX graphical user interface, including all of the visible components of the path to a session *except* the root "Sessions" folder. The graphic below portrays two example session path specifications. The saved session named "Web Host 1" resides in a subfolder named "Web Hosts", so its full session path specification is: "Web Hosts\Web Host 1". The saved session named "Server A" is located within the root "Sessions" folder, so its full session path specification is simply "Server A". Quotes are required in all cases where spaces are part of the session path or name.



2.3. Local Path Specifications

A "local path" specification is an absolute or relative path to a file or folder residing on the file system "local" to the SFXCL client machine or LAN. A local path specification could be a reference to a file or folder that resides on a local drive on the same machine as where the SFXCL process is running, or it could be a reference to a file or folder that resides on a network share local to the LAN on which the SFXCL machine resides. A best practice is to use absolute paths for local specifications. Both of the following examples are legitimate local path specifications:

```
C:\Temp\MyFile.txt
\\file_server\share\subfolder\AnotherFile.txt
```

2.4. File Transfer "Direction" (Upload vs. Download)

SFXCL is a client application that initiates a connection to a file server. The left-to-right order of the source (a.k.a "from") and destination (a.k.a "to") specifications on the SFXCL command line determines whether or not an upload or a download operation is to take place.

An "upload" is a transfer which copies a file from the machine on which SFXCL is running to a remote server.

A "download" is a transfer which copies a file from a remote server to the local machine on which SFXCL is running.

Wherever there is a specification pertaining to a remote server, we've colored the remote-specific text dark red (red = remote). When you see an example SFXCL command line in this document, you can immediately tell which part of the command line provides information about the remote server side of the operation.

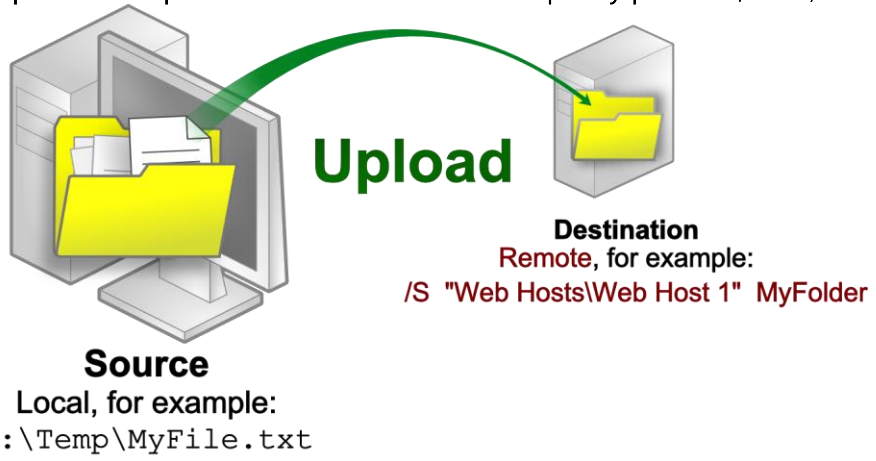
2.4.1. Upload Syntax

For an upload to take place, the source must be a local path and the destination must be a remote specification (either a URL or a session specification):

```
SFXCL  [Options]  local_source_path  remote_destination_spec
```

2.4.1.1. Upload Example #1: Saved Session

This upload example uses a saved session to specify protocol, host, username, and password used to connect.



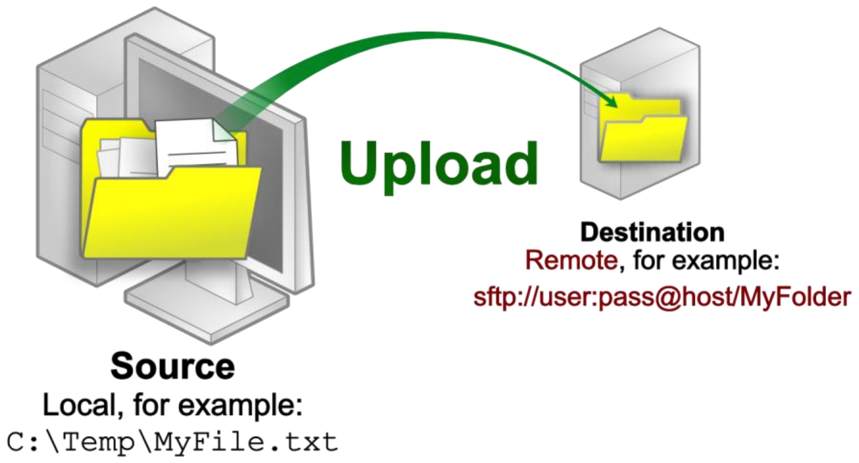
Local source file:	`C:\Temp\MyFile.txt`
Remote Host:	*Defined in saved session named "Web Hosts\Web Host 1".*
Connection Protocol:	*Defined in the saved session.*
Account Information:	*Defined in the saved session.*
Destination Folder:	`MyFolder`

Resulting SFXCL command line for Upload Example #1:

```
sfxcl  C:\Temp\MyFile.txt  /S "Web Hosts\Web Host 1" MyFolder
```

2.4.1.2. Upload Example #2: Ad Hoc URL Specification

This upload example uses an "ad hoc" connection with a URL format to specify protocol, host, username, and password used to connect.



Local source file:	`C:\Temp\MyFile.txt`
Remote Host:	hostname is "`host`"
Connection Protocol:	SFTP
Account Information:	Username = "`user`", Password = "`pass`"
Destination Folder:	`MyFolder`

Resulting SFXCL command line for Upload Example #2:

```
sfxcl  C:\Temp\MyFile.txt  sftp://user:pass@host/MyFolder
```

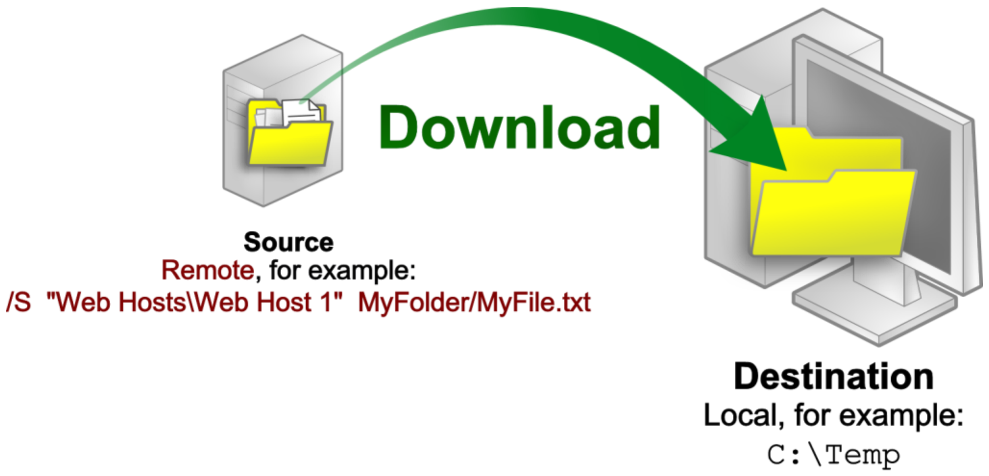
2.4.2. Download Syntax

For a download to take place, the source must be a remote specification (either a URL or a session specification) and the destination must be a local path:

```
SFXCL  [Options]  remote_source_spec  local_destination_path
```

2.4.2.1. Download Example #1: Saved Session

This download example uses a saved session to specify protocol, host, username, and password used to connect.



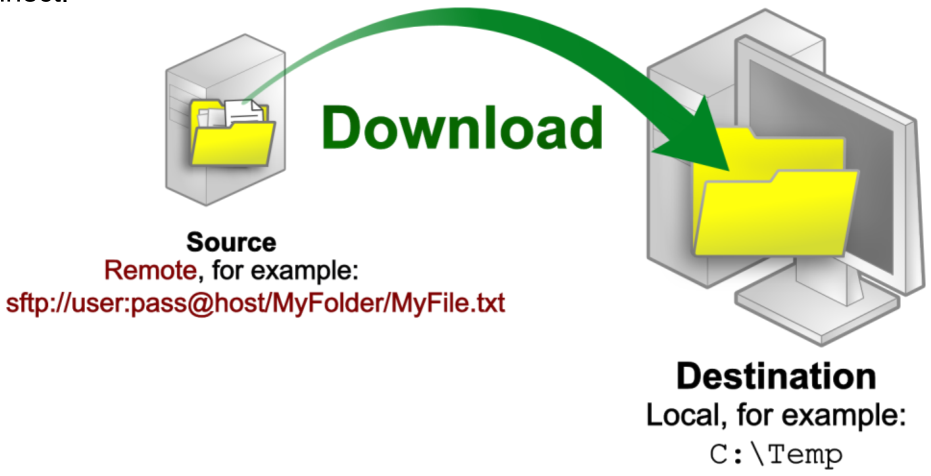
Remote Host:	*Defined in saved session named "Web Hosts\Web Host 1".*
Connection Protocol:	*Defined in the saved session.*
Account Information:	*Defined in the saved session.*
Remote File to Get:	`MyFolder/MyFile.txt`
Local Destination:	`C:\Temp`

Resulting SFXCL command line for Download Example #1:

```
sfxcl  /S "Web Hosts\Web Host 1" MyFolder/MyFile.txt  C:\Temp
```

2.4.2.2. Download Example #2: Ad Hoc URL Specification

This download example uses an "ad hoc" connection with a URL format to specify protocol, host, username, and password used to connect.



Remote Host:	hostname is "`host`"
Connection Protocol:	SFTP
Account Information:	Username = "`user`", Password = "`pass`"
Remote File to Get:	`MyFolder/MyFile.txt`
Local Destination:	`C:\Temp`

Resulting SFXCL command line for Download Example #2:

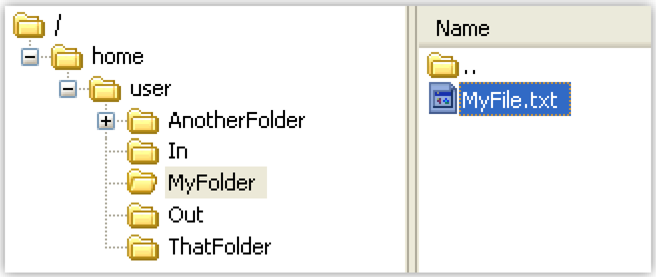
```
sfxcl  sftp://user:pass@host/MyFolder/MyFile.txt  C:\Temp
```

2.5. Absolute vs. Relative Paths

The remote file paths in the examples thus far have all been "relative". In other words, the path specified is relative to the current directory (also known as the "working directory"). Here's a brief refresher on the differences between the two types of path specifications:

- A "relative" path begins relative to the current directory on the file system.
- An "absolute" path always begins with the "/" character (meaning the "root" of a file system hierarchy), and includes all directories starting from the root of the file system down to the actual name of the file or folder being targeted.

The graphic and table below illustrates the difference between the two types of path specifications. Although there are various ways to specify relative paths to the same file depending on the value of the current directory, there is only one absolute path to the file "MyFile.txt" depicted in the graphic below.



If *Current Directory* is…	*Relative* Path to "MyFile.txt" would be...
`MyFolder`	`MyFile.txt` *or* `./MyFile.txt`
`ThatFolder`	`../MyFolder/MyFile.txt`
`AnotherFolder`	`../MyFolder/MyFile.txt`
`user`	`MyFolder/MyFile.txt` *or* `./MyFolder/MyFile.txt`
`home`	`user/MyFolder/MyFile.txt`
`/`	`home/user/MyFolder/MyFile.txt`

The *Absolute* path to "MyFile.txt" as in above graphic would <u>always</u> be:

```
/home/user/MyFolder/MyFile.txt
```

If you needed to download `MyFile.txt` (as depicted above), and you were connecting to a remote server and you weren't sure where the remote server was going to place you initially (your current directory), you would need to specify the path as an *absolute* path in order to avoid any problems retrieving the file. As examples, consider the following ways to download `MyFile.txt` (as depicted above) using *absolute* paths (the absolute paths' beginning slash characters are bolded for emphasis) in the remote specifications.

> **Note:** In the URL examples of absolute paths given below, pay particular attention to the presence of two consecutive slash characters "//" following the host component of the URL. The first slash separates the *host* specification from the *path* component while the second slash is the beginning of the absolute path specification.

Download using URL syntax and an absolute path to `MyFile.txt`:

```
sfxcl  sftp://user:pass@host//home/user/MyFolder/MyFile.txt  C:\Temp
```

Download using a saved session "Server A" and an absolute path to `MyFile.txt`:

```
sfxcl  /S "Server A" /home/user/MyFolder/MyFile.txt  C:\Temp
```

As additional examples of using absolute paths for the remote specification, consider the following upload command-line sequences:

Upload a file using URL syntax and an absolute path to the destination folder `MyFolder`:

```
sfxcl  C:\Temp\MyFile.txt  sftp://user:pass@host//home/user/MyFolder
```

Upload a file using a saved session "Server A" and an absolute path to the destination folder `MyFolder`:

```
sfxcl  C:\Temp\MyFile.txt  /S "Server A" /home/user/MyFolder
```

3. Use a Log File to Facilitate Troubleshooting

SFXCL will write debug information to the shell console from which the process was launched. However, the shell console will likely not be available for viewing if you plan to run SFXCL in an automated (unattended) manner. In order to facilitate troubleshooting, you should employ the best practice of instructing SFXCL to write debug information to a log file by using the **/LOG** command-line option and an accompanying log file path. For example:

```
sfxcl /LOG C:\SFXCL.log C:\file.txt /S "Server A" uploads
```

If you are unable to determine the cause of a problem upon reviewing the information contained in an SFXCL log file, you may desire to contact VanDyke Software technical support for assistance via e-mail or web. When contacting technical support for assistance in troubleshooting SFXCL problems, you should attach or upload the SFXCL log file with any sensitive information like usernames, passwords, etc. properly redacted.

If multiple simultaneous SFXCL instances are being launched in parallel, you should engineer the launching of each SFXCL instance so that a unique log file is specified each time. Otherwise, multiple instances of SFXCL will potentially be competing for access to the same log file, resulting in a log file that contains jumbled information that will not help you in your troubleshooting efforts. Examples that demonstrate unique log file name generation within CMD Batch files and VBScript code are provided in the "Generating Unique Log File Names" section.

4. Test Manually Before Implementing Automation

Prior to adding an SFXCL command to a batch file or a script, you should employ the best practice of manually testing it on the command line in order to work out any syntax errors or other mistakes that are difficult to track down otherwise. It is much easier to confirm that a command line is ready for automation with immediate direct feedback via manual testing rather than checking log files to see why the SFXCL command is failing in an automated batch file or script.

4.1. Use the Same Account for Manual Testing and Automation

SFXCL requires access to and knowledge of a valid configuration folder location. SFXCL loads settings and host data (such as session names, location for the known host key database for SFTP and SCP connections, etc.) from the profile associated with the user account context under which the command process is launched.

When manually testing your SFXCL command-line task, be sure to run SFXCL under the same user account context as the one you will ultimately be using for the automated task you are preparing.

For example, if your automated task is to be launched with the user account context of user "AUTOMATION_USER", make sure you're manually testing as that user. If you do all the manual testing of your SFXCL command as "Bob", and then set up an automated task to run the same command as "AUTOMATION_USER", you'll likely encounter an error since all the configuration settings you've been using during manual testing come from Bob's profile.

If you're using SFTP or SCP as the connection protocol, the manual testing phase is the perfect time to validate the host key fingerprint to ensure that you are connecting to the actual SSH server you intend to instead of a man-in-the-middle spoof.

4.2. Using a Configuration Folder from another Account

If your situation is one in which you are unable to manually test SFXCL as the user account context under which SFXCL will ultimately be running once automated, there are a few suggestions for making the impossible become less so.

1) Test manually using a user account that you do have access to – one which can log on to the machine directly and run SecureFX, setting up session configuration and global options as needed.

2) Make use of the **/F** command-line option to "point" SFXCL to the configuration folder location that is being used by your test user account. For example, consider that account "Bob" may have the configuration folder, "`C:\Users\Bob\AppData\Roaming\VanDyke\Config`". When Bob sets up SecureFX configurations, sessions, saves host keys, etc., he'll be working with a SecureFX configuration that is profile-specific to Bob. When setting up the automated task that will run under a different account context (say, AUTOMATION_USER), Bob should use the **/F** command-line option to point SFXCL to use Bob's configuration folder, as in the following example:

```
sfxcl  /F "C:\Users\Bob\AppData\Roaming\VanDyke\Config"  sftp://user:pass@host/file.txt  C:\downloads
```

3) Ensure that the AUTOMATION_USER account has permissions to read and access the SecureFX configuration folder indicated by the value of the **/F** command-line option.

4.3. Avoid Hanging SFXCL Processes Needing User Input

By default, SFXCL will prompt for user input should it be needed. For example, if credentials are not properly specified on the command line as part of a URL specification, or not properly saved in a session configuration, SFXCL will prompt for username/password/etc., waiting for user input. Another example of an "input required" prompt occurs when SFXCL is instructed to connect to an SFTP server for which a corresponding host key does not currently exist in the SecureFX configuration's Known Hosts database. If at any time user input is required by SFXCL after being launched as part of an automated task, there will be no user available to provide the input required for SFXCL to continue with the operation – leaving an SFXCL.exe process running forever until forcibly closed via the Windows Task Manager or a system restart. In order to avoid such situations where user input is required, SFXCL provides a **/NOPROMPT** command-line option which will cause the SFXCL process to terminate with a non-success exit code if any user interaction is ever required. For example:

```
sfxcl  /NOPROMPT /F "C:\Users\Bob\AppData\Roaming\VanDyke\Config"  sftp://user:pass@host/file.txt  C:\downloads
```

5. SFXCL Exit Codes

The exit code for SFXCL will always be **zero (0) for success** and **non-zero** (-infinity to +infinity) **for any failure**.

Non-success SFXCL exit codes are not currently documented and are subject to change in future releases. You should avoid trying to associate a specific SFXCL exit code with the cause of a failure. Instead, use the information contained within the SFXCL log file to determine the cause of a failure.

6. Using Wildcards

SFXCL provides the ability to use wildcards to transfer a number of files matching a pattern. The two wildcards supported by SFXCL are displayed in the following table.

Wildcard	Meaning
*	Used to represent zero or more characters
?	Used to represent any single character

The following example uploads all `.txt` files within the local machine's `C:\SalesReports` folder to a folder named “ReportsArchive” in the initial working directory on the remote machine named "host":

```
sfxcl  C:\SalesReports\*.txt  sftp://user:pass@host/ReportsArchive
```

The following example downloads all `.dat` files found in the initial working directory on a remote machine named "host" to the `D:\Incoming` folder on the local machine where SFXCL is running:

```
sfxcl  sftp://user:pass@host/*.dat  D:\Incoming
```

The following example downloads all files that begin with the word "`file`", followed by exactly three characters (doesn't matter what the characters are – there just needs to be three), followed by the extension "`.csv`":

```
sfxcl  sftp://user:pass@host/file???.csv  D:\CSV_Files
```

7. ASCII vs. Binary Transfers

When transferring text files between disparate systems (UNIX or Mac OS X vs. Windows, etc.) you may need to be aware of the need to transfer files such that lines are separated by an end of line (EOL) marker matching the EOL convention used by the recipient system.

- Windows systems store text files with each line separated by two ASCII characters: Carriage Return (CR, ASCII 13 decimal) and Line Feed (LF, ASCII 10 decimal).
- UNIX and Mac OS X systems store text files with each line separated by a single LF character.

Files transferred as ASCII result in EOL characters being converted (or "translated") from the EOL convention on the sender's side to the EOL convention of the receiver's convention. For example, a text file on a Windows machine transferred as ASCII to a UNIX machine would have all the CRLF character sequences translated/converted to a single LF character when stored on the UNIX machine.

In SecureFX 6.7 and newer versions, the SFXCL command-line utility provides a **/TRANSFERTYPE** option that can be used to force all files to be transferred as either ASCII or Binary. As examples:

Perform ASCII Conversion:

```
sfxcl  /TRANSFERTYPE ascii  F:\TextFiles  sftp://user:pass@host/uploads
```

Perform No Conversion (Binary):

```
sfxcl  /TRANSFERTYPE binary  F:\Graphics  /S "Server A" uploads
```

8. Generating Unique Log File Names

When the log file name specified with SFXCL's **/LOG** argument already exists, SFXCL will append information to the existing log file. As mentioned earlier in the section titled "Use **/LOG** to Facilitate Troubleshooting", if multiple simultaneous SFXCL instances are being launched in parallel, you should engineer the launching of each SFXCL instance such that a unique log file is used. Otherwise, multiple instances of SFXCL will potentially be competing for access to the same log file, resulting in a log file that contains jumbled information that will not help you in your troubleshooting efforts.

This section provides example code demonstrating unique log file name generation within batch files and VBScript code. We encourage you to employ similar techniques for generating unique log file names if your automation scenario involves the possibility of multiple simultaneous instances of SFXCL being launched in parallel.

8.1. Generating a Unique Log File Name within a Batch File

```
:: The command 'date /T' returns a value matching the pattern:
::    Tue 03/24/2010   (That is, DAY_OF_WEEK MM/DD/YYYY)
:: MyDate will have YYYYMMDD format
for /f "tokens=2-4 delims=/ " %%a in ('date /T') do set MyDate=%%c%%a%%b

:: The variable %TIME% produces a value matching the pattern:
::    14:30:14.85     (That is, HH:MM:SS.MS)
:: MyTime will have HHMMSS.MS format
for /f "tokens=1-4 delims=:. " %%a in ("%TIME%") do set MyTime=%%a%%b%%c.%%d

:: logname will end up having the following pattern:
::    20100324143014.85.log  (That is, YYYY-MM-DD--HH-MM-SS.MS.log)
set LOGFILE=C:\Temp\SFXCL_LOG__%MyDate%%MyTime%.txt

echo Unique Log File: %LOGFILE%
```

8.2. Generating a Unique Log File Name within VBScript Code

```
' Use WMI to get at the current time values
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * from Win32_OperatingSystem")
For Each objItem In colItems
    strLocalDateTime = objItem.LocalDateTime
Next
' strLocalDateTime has the following pattern:
' 20111013093717.418000-360   [ That is,  YYYYMMDDHHMMSS.MILLIS(zone) ]
' Take the left-most 18 digits...
strLogFile = Left(strLocalDateTime, 18)

' Now that we have a unique name, let's build a path with it
strLogFile = "C:\Temp\SFXCL_LOG__" & strLogFile & ".txt"

' ... Display for all the world to see:
MsgBox "Unique filename is: " & strLogFile
```

9. Example Batch Files

9.1. Upload .CSV Files

```
@echo off

:: GENERATE A UNIQUE LOG FILE NAME:
for /f "tokens=2-4 delims=/ " %%a in ('date /T') do set MyDate=%%c%%a%%b
for /f "tokens=1-4 delims=:. " %%a in ("%TIME%") do set MyTime=%%a%%b%%c.%%d
set LOGFILE=C:\Temp\SFXCL_LOG__%MyDate%%MyTime%.txt

:: Run SFXCL - upload .CSV files from local "Out" to remote "In" folder:
SFXCL  /NOPROMPT  /RETRYCOUNT 0  /LOG "%LOGFILE%"  C:\Out\*.CSV  sftp://user:pass@192.168.232.149/In
```

```
:: Check for failure
if not %errorlevel%==0 goto FAILURE
:: Otherwise... Success!
goto SUCCESS

:FAILURE
echo.    Transfer Failure.  See %LOGFILE% for details.
goto END

:SUCCESS
echo.    Transfer Success.
goto END

:END
echo.  Batch file exiting.
```

9.2. Download .DAT Files

```
@echo off

:: GENERATE A UNIQUE LOG FILE NAME:
for /f "tokens=2-4 delims=/ " %%a in ('date /T') do set MyDate=%%c%%a%%b
for /f "tokens=1-4 delims=:. " %%a in ("%TIME%") do set MyTime=%%a%%b%%c.%%d
set LOGFILE=C:\Temp\SFXCL_LOG__%MyDate%%MyTime%.txt

:: Run SFXCL - download .DAT files from remote "Out" to local "In" folder:
SFXCL  /NOPROMPT  /RETRYCOUNT 0  /LOG "%LOGFILE%"  sftp://user:pass@192.168.232.149/Out/*.DAT  C:\In

:: Check for failure
if not %errorlevel%==0 goto FAILURE
:: Otherwise... Success!
goto SUCCESS

:FAILURE
echo.    Transfer Failure.  See %LOGFILE% for details.
goto END

:SUCCESS
echo.    Transfer Success.
goto END

:END
echo.  Batch file exiting.
```

10. Example VBScript Code

10.1. Upload .CSV Files

```
' MinimalSFXCLUploadExample.vbs
' Generate a unique log file name
For Each item In GetObject("winmgmts:\\.\root\cimv2").ExecQuery(_
   "Select * from Win32_OperatingSystem")
    strLocalDateTime = item.LocalDateTime
Next
strLogFile = "C:\Temp\SFXCL-" & Left(strLocalDateTime, 18) & "-Log.txt"

' Launch SFXCL - Upload all .CSV files from local "Out" folder
' to remote "In" folder:
nResult = CreateObject("WScript.Shell").Run(_
    "SFXCL /NOPROMPT /RETRYCOUNT 0 /LOG """ & strLogFile & _
    """ C:\Out\*.CSV  sftp://user:pass@192.168.232.149/In", _
    0, _
    True)

' Check for failure
If nResult <> 0 Then
    WScript.Echo "Transfer Failure.  See " & strLogFile & " for details."
Else
    WScript.Echo "Transfer Success."
End If
```

10.2. Download .DAT Files

```
' MinimalSFXCLDownloadExample.vbs

' Generate a unique log file name
For Each item In GetObject("winmgmts:\\.\root\cimv2").ExecQuery(_
   "Select * from Win32_OperatingSystem")
    strLocalDateTime = item.LocalDateTime
Next
strLogFile = "C:\Temp\SFXCL-" & Left(strLocalDateTime, 18) & "-Log.txt"

' Launch SFXCL - Download all .DAT files from remote "Out" folder
' to local "In" folder:
nResult = CreateObject("WScript.Shell").Run(_
    "SFXCL /NOPROMPT /RETRYCOUNT 0 /LOG """ & strLogFile & _
    """  sftp://user:pass@192.168.232.149/Out/*.DAT  C:\In", _
    0, _
    True)

' Check for failure
If nResult <> 0 Then
    WScript.Echo "Transfer Failure.  See " & strLogFile & " for details."
Else
    WScript.Echo "Transfer Success."
End If
```