WinSCP Command-Line Simplified: Your Complete Reference

https://adamtheautomator.com/winscp-command-line/

WinSCP, a staple tool for secure file transfers, offers more than just a graphical user interface. The **WinSCP command-line** functionality allows IT professionals and sysadmins to manage and automate file transfers directly from the console, enhancing efficiency and flexibility in their workflow.

To learn the ins and outs of the WinSCP GUI, check out this post's complementing post, The WinSCP GUI: The Ultimate Guide.

Discover every command-line feature WinSCP offers in this Ultimate Guide. You'll learn, step-by-step, how to leverage the **WinSCP command line** to maximize your productivity and streamline your file transfer processes.

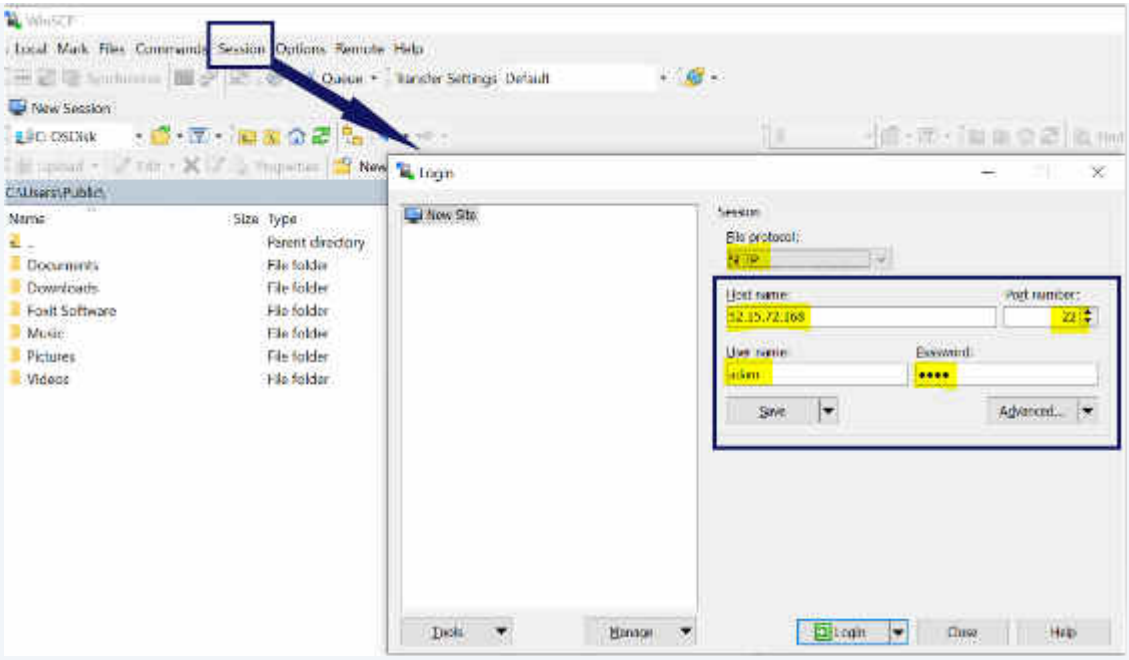
Let's dive into the world of command-line file management with WinSCP!

Prerequisites for Using the WinSCP Command Line

- A Windows XP+ PC – This tutorial will use Windows 10 Enterprise.
- A remote Linux host – This tutorial will use an Ubuntu 18.04.5 LTS machine.
- A user account on the remote SSH host with sudo permissions.

Generating a Session URL with the WinSCP Command Line

While the WinSCP GUI offers a convenient **Login** window for setting up connections, the **WinSCP command line** requires a different approach. Instead of interactive windows, you must specify connection attributes in a more direct manner.



Connection attributes

To instruct the **WinSCP command line** on where to connect, a session URL is used. This URL encapsulates the connection attributes defined in the GUI.

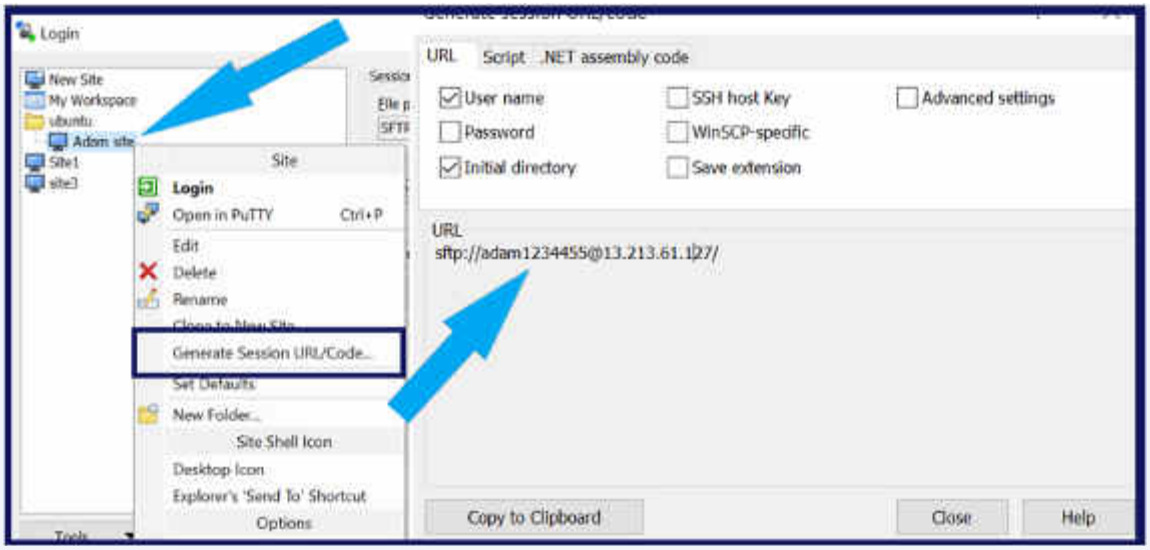
A basic session URL follows this structure:

```
<protocol>://<username>:<password>@<host name>/<folder path>
```

For example, to connect to the `1.1.1.1` host with username `adam` and password `pw` using SFTP in WinSCP, the session URL would be:

```
sftp://adam:pw@1.1.1.1
```

If you've set up a WinSCP site, you can easily find the session URL. Open the WinSCP GUI, click on **Session**, right-click your site, and select **Generate Session URL/Code**.



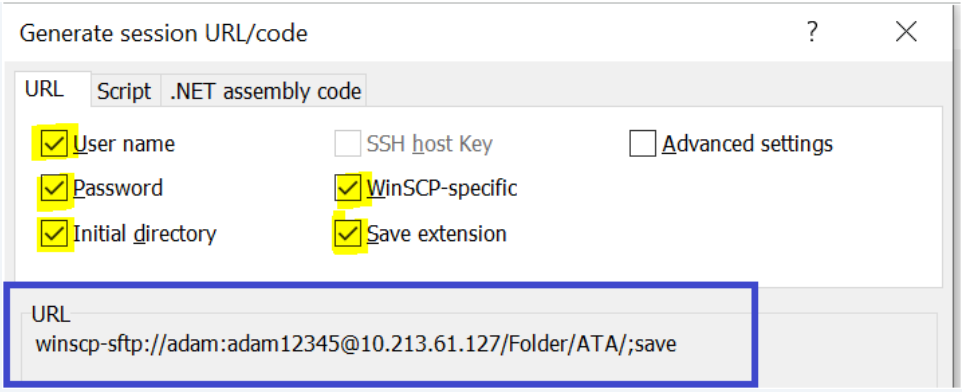
WinSCP site

The Session URL dialog box provides various customization options for your connection, each influencing how WinSCP interacts with the remote host.

- **Initial directory** – Specifies the remote directory that WinSCP opens upon connection.
- SSH host key – Uses an existing SSH key for remote host authentication.
- WinSCP-specific URL – Generates a unique session URL, like `WinSCP-sftp://`, specifically for WinSCP's understanding, avoiding conflicts with default web browsers.
- Save extension – Often paired with WinSCP-specific URLs to further minimize application conflicts.

When all options are enabled, the session URL format should resemble the following:

```
<protocol>://<username>:<password>@<host name>/<folder path><save extension>
```



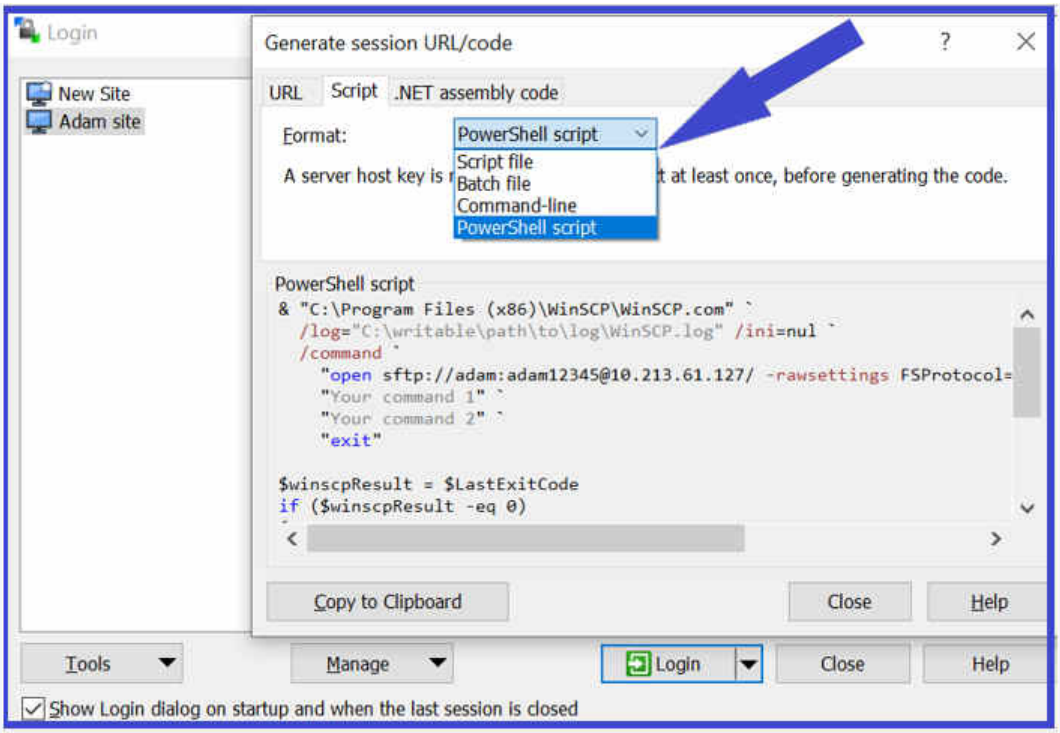
Generate Session URL/Code

Mastering the WinSCP Command-Line: Generating Session Connection Code

With your session URL ready, WinSCP goes a step further by offering code examples in the **Generate session URL/code** window's **Script** tab. Here, you can select from different script types:

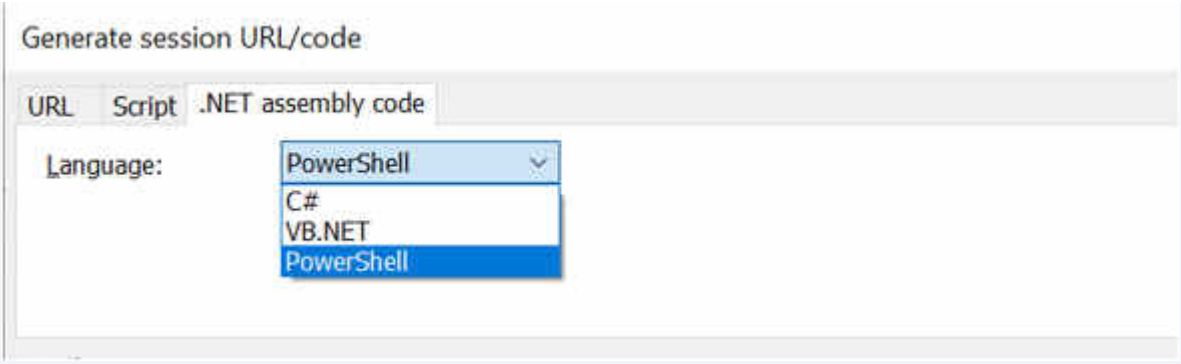
- Script file
- Batch file
- Command-line
- PowerShell script

Choose your preferred script type, and WinSCP will generate the necessary syntax for connecting to a remote host using the **WinSCP command line**.



Generating Session Connection Code

For coding in .NET, C#, or PowerShell, switch to the **.NET assembly code** tab, as depicted below.



.NET assembly code tab

Understanding WinSCP.exe vs. WinSCP.com in the WinSCP Command Line Context

Knowing how to generate a session URL is crucial for using **WinSCP command line** tools. Let's explore how to connect to a host using these tools, starting with understanding the differences between *winscp.exe* and *winscp.com*.

Winscp.exe doubles as the GUI launcher and a command-line utility for simple tasks. You can use it for basic commands if you specify the right parameters.

Think of *winscp.exe* as a bridge between the GUI features and command-line interface of WinSCP.

For more complex tasks, especially in scripting, *winscp.com* is your go-to. This console-based tool is ideal for automation scripts and supports a range of SSH functions. It provides a true, non-interactive **WinSCP command line** experience.

Regardless of your choice, begin by opening a command prompt (either *cmd.exe* or PowerShell) and navigate to the WinSCP installation directory.

```
cd 'C:\Program Files (x86)\WinSCP'
```

With the command prompt ready, let's proceed to using WinSCP for remote connections.

Utilizing WinSCP.exe: Connecting to Remote Hosts with Session URLs

One of the simplest methods to connect to a remote host with **WinSCP command line** is by running *winscp.exe* with the session URL as a parameter. For instance, to connect to the **54.179.19.216** host with username **automate** and password **automate** using **sftp**, and then navigate to the */tmp* directory, use the following command:

```
WinSCP.exe sftp://automate:automate@54.179.19.216/tmp/
```

For enhanced security, instead of using a password, you can connect to a remote host using a private key with the **/privatekey** parameter. While this method is more secure, the intricacies of setting up a private key fall beyond the scope of this tutorial on the **WinSCP command line**.

Related: How to Create SSH keys

Here's an example of connecting to the **54.179.19.216** host using the username **automate** and a private key **mykey.ppk** over scp.

```
winscp.exe scp://automate@54.179.19.216/tmp/ /privatekey=mykey.ppk
```

Efficiently Downloading Files with WinSCP.exe Without a Site

The **WinSCP command line** via *winscp.exe* allows you to swiftly transfer files using ad-hoc connections, even without a predefined WinSCP site. Let's explore downloading files from a remote host without a site configuration. For example, downloading all files in the */tmp* directory of the remote host 54.179.19.216 using SFTP.

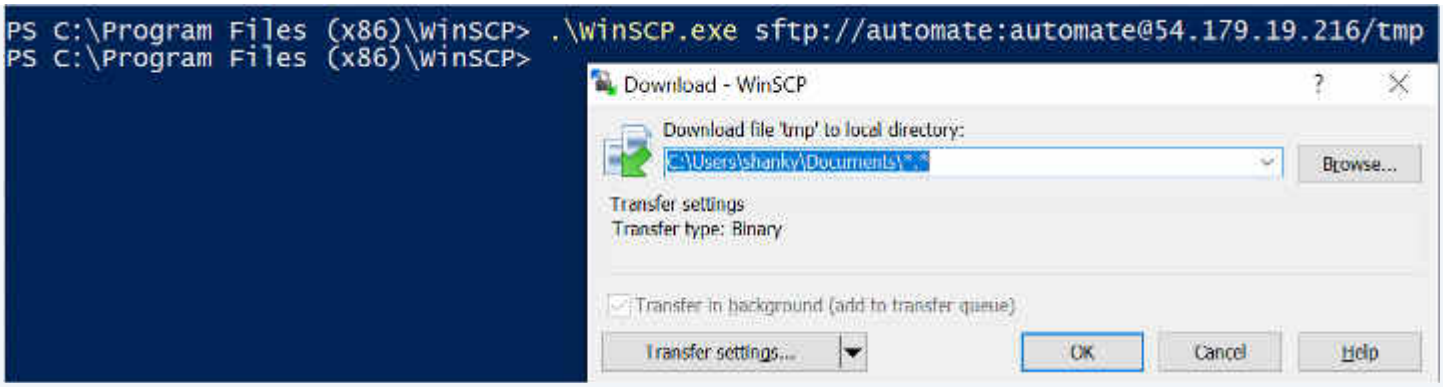
1. Start by generating a session URL. Below is an example connecting to the remote host with the credentials **automate** and landing in the */tmp* directory.

```
# Generated Session URL
sftp://automate:automate@54.179.19.216/tmp
```

2. Run *winscp.exe* with the session URL to access the WinSCP transfer settings dialog box. The local directory defaults to ~*Documents*, and WinSCP will target all files (***.***) in the remote directory.

```
# Command Syntax: winscp.exe [/path/[file]]
winscp.exe sftp://automate:automate@54.179.19.216/tmp
```

3. Click OK to initiate the transfer. WinSCP will download all files from the remote */tmp* directory to your chosen local directory over SFTP.



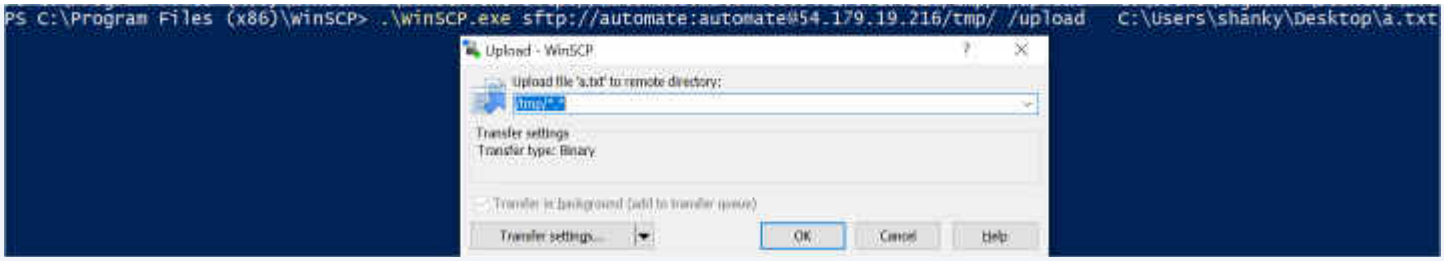
Downloading the Files without a Site

Uploading Files Using WinSCP.exe Without a Site

After mastering file downloads, let's switch to uploading files to a remote host using the **WinSCP command line** with *winscp.exe*. The process is similar to downloading, but this time you'll also need to use the `/upload` switch, specifying the file or folder you wish to upload.

```
# Uploading the file a.txt using winscp.exe to the remote server without a site.
.\WinSCP.exe sftp://automate:automate@54.179.19.216/tmp/ /upload C:\Users\shanky\Desktop\a.txt
```

Executing the command above brings up the WinSCP upload dialog, indicating the files to be uploaded (`*.*`) to the remote host's */tmp* directory.



Uploading the Files without a Site

After uploading, log into the remote host with an SSH client and verify the upload success with commands like **`pwd`** and **`ls -lh`**, which will display the directory contents, confirming the successful transfer.

```
$ pwd
/tmp
$ ls -lh
total 20K
-rw-rw-r-- 1 automate automate    0 Jun 25 10:18 a.txt
-rw-rw-r-- 1 automate automate    0 Jun 17 18:55 abc.txt
drwxrwxr-x 2 automate automate 4.0K Jun 21 20:42 folder1
```

File Uploaded Successfully on Remote Machine

Leveraging WinSCP.exe for Downloading Files Using a Site

While session URLs are practical for ad-hoc connections, having a pre-configured WinSCP site simplifies the process. If you've been utilizing WinSCP for some time, you're likely to have several sites already set up.

With *winscp.exe*, you can easily utilize these sites, created in the GUI, to connect to a remote host, bypassing the need to remember session URL details.

To locate saved sites in WinSCP, navigate to **Session** -> **Sites** -> **Site Manager**.

For instance, let's explore downloading files using a pre-existing site with the **WinSCP command line**.

1. Connect to the remote host using a configured WinSCP site. The example below demonstrates a connection using a site named **`Adamsite`**.

```
# Connect with a WinSCP site
winscp.exe Adamsite
```

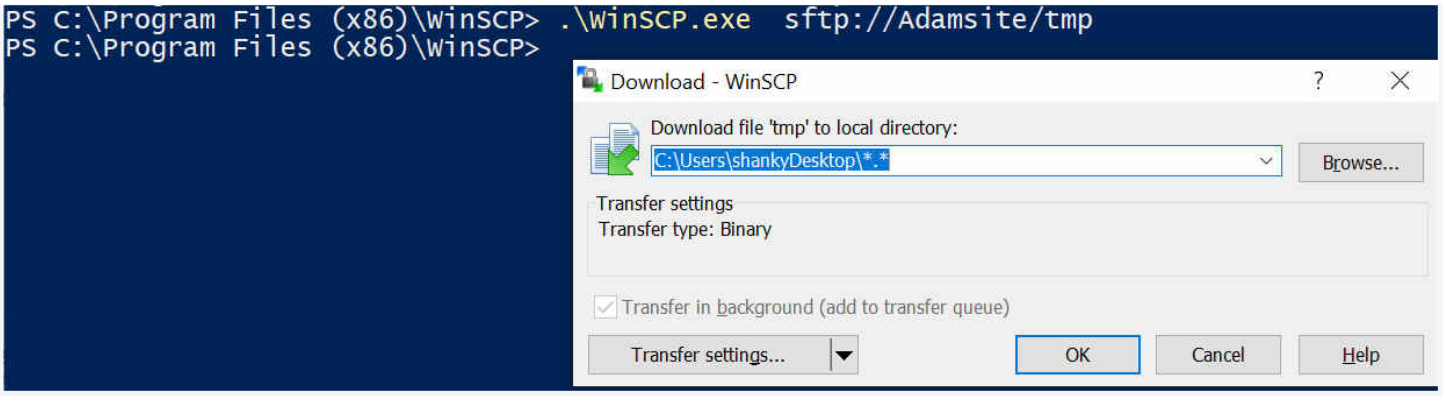
Once connected, WinSCP displays a status notification.

Connecting to Remote Machine Using a Site

2. Next, execute *winscp.exe* with the protocol (**`sftp`**), site name (**`Adamsite`**), and target directory (**`/tmp`**). This approach brings up the [WinSCP transfer settings dialog box], ready for file download actions.

```
# Initiate remote connection using a site
winscp.exe sftp://Adamsite/tmp
```

Click OK to start downloading all files from the */tmp* directory of the remote host to your local directory via SFTP.



Downloading Files Using a Site

For uploading files from your local machine to a remote host, the process is analogous. Use the **`/upload`** switch along with the file or directory path, like **`winscp.exe Site3 /upload .\license.txt`**. Full URL specification (**`sftp://Adamsite/tmp`**) is not necessary.

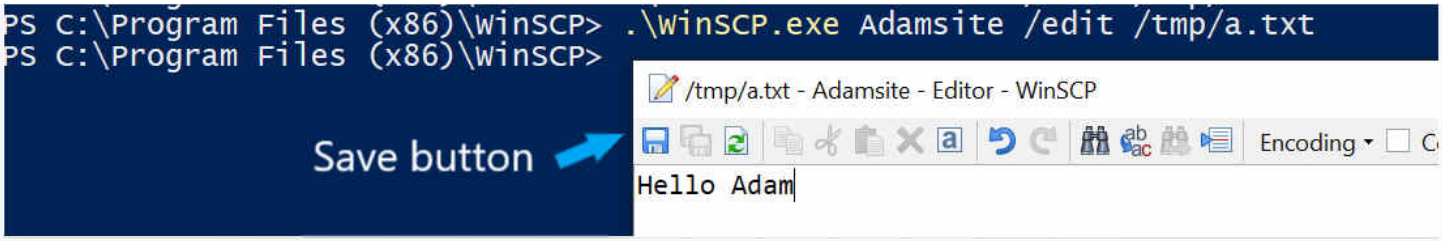
Streamlining Remote File Editing with WinSCP.exe

Need to edit a text file on a remote host? Skip the manual download-edit-upload cycle. The **WinSCP command line** offers a more efficient method using the **`/edit`** parameter. Simply specify the site name, **`/edit`**, and the remote file path.

Example:

```
# Editing a file on a remote host
winscp.exe /edit /Adamsite/path/to/file.txt
# Using the WinSCP command line to edit a remote file
.\WinSCP.exe Adamsite /edit /tmp/a.txt
```

Executing the command above launches your default text editor, allowing you to modify the remote file as needed. This seamless integration is a key feature of the **WinSCP command line** capabilities.



Editing Remote Files Using a Site

After editing, simply save the file. WinSCP takes care of updating the file on the remote host, streamlining the edit-upload process.

Optimizing Workflow with WinSCP.exe Session Logging

For a comprehensive record of your actions, the **WinSCP command line** offers session logging. This feature is invaluable for tracking the commands executed during a session, aiding in troubleshooting and record-keeping.

To enable session logging, include up to three parameters when connecting to a session:

```
# Parameters for session logging
/log="<log file path>" /loglevel=<level> /logsize=<size>
```

- **`/log`** – Specifies the path for the log file.
- **`/loglevel`** – Sets the verbosity of the logs, ranging from *Reduced* (**`1`**) to *Debug* (**`2`**).
- **`/logsize`** – Defines the size and rotation of the log file, in the format **`<total archived logs>*<max log file size>`**.

Below is an example of *winscp.exe* connecting to a host and logging activity to **`C:\winscp.log`** at a *Debug* level. The command maintains up to five 10MB log files (**`5*10MB`**).

```
# Example of WinSCP session logging
winscp.exe sftp://automate@54.179.19.216/tmp/ /log="C:\\winscp.log" /loglevel=2 /logsize=5*10M
```

Exploring Advanced Features with WinSCP.com Interactive Commands

While *winscp.exe* provides a great introduction to remote connections, *winscp.com* elevates your command line experience, offering deeper interaction and control.

Begin by opening *winscp.com* in a command line environment. You'll enter an interactive session reminiscent of SSH, indicated by the **`winscp>`** prompt.

```
.\WinSCP.com
```

Connect to a remote computer using the **open** command, followed by the desired site name.

```
open Adamsite
```

After connecting to the remote host via **WinSCP command line**, as illustrated with **Adamsite**, you'll find yourself in an environment akin to an SSH session. Here, you can execute commands and interact with the remote host efficiently.

```
PS C:\Program Files (x86)\WinSCP> .\WinSCP.com
winscp> open Adamsite
Searching for host...
Connecting to host...
Authenticating...
Using username "automate".
Authenticating with pre-entered password.
Authenticated.
Starting the session...
Session started.
Active session: [1] Adamsite
winscp> pwd
/tmp
winscp> ls
```

Interactively Running Commands using winscp.com

Enhancing Security: Connecting with a New Key Pair (Host Key) in WinSCP

Public-key authentication enhances security when connecting to sessions. To use it, first obtain the host key fingerprint for your session. Use the **ssh-keygen** command in the WinSCP folder following the syntax below.

This command generates an SSH key pair, essential for secure authentication. Once executed, you'll receive a fingerprint necessary for *winscp.com* session connections.

```
ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
```

```
root@ip-192-168-1-48:/etc/ssh# ssh-keygen -l -f /etc/ssh/ssh_ho
2048 SHA256:x4DeZzv4jcWEk2zeeJgr5JeJ+z0xA+lGa3LC0q/B+88 root@ip
```

Verifying SSH Key Fingerprint

To connect with the **hostkey** parameter, provide the generated fingerprint as shown in the example below.

Include the prefix **ssh-rsa 2048** when using the fingerprint from **ssh-keygen**.

```
winscp.com open sftp://automate:automate@54.179.19.216/  -hostkey="ssh-rsa 2048
x4DeZzv4jcWEk2zeeJgr5JeJ+z0xA+lGa3LC0q/B+88="
```

Maximizing Efficiency with WinSCP.com: Using the /command Parameter

While interactive sessions are great for certain tasks, non-interactive command execution is crucial for scripting and automation. *winscp.com* supports this through the **/command** parameter, enabling you to establish a session, execute a command, and disconnect in a single step.

For example, use the **/command** parameter to transfer a local file to a remote host. The example below demonstrates copying a file from *C:\abc\abc.txt* to the */tmp* directory on the remote host **13.213.61.127**.

The `/command` parameter takes two arguments as strings: one to establish the session and another for the actual command execution.

```
# Non-interactive file transfer using WinSCP
WinSCP.com /command "open sftp://adam:adam@13.213.61.127/tmp" "put C:\\abc\\abc.txt"
Session log for adam@13.213.61.127
C:\abc\abc.txt            |            0 B |     0.0 KB/s | binary |   0%
```

Automating Tasks with WinSCP.com: The /script Parameter

For more complex automation needs, the `/script` parameter in *winscp.com* allows you to execute a series of commands from a script. This feature is invaluable when managing repetitive tasks or complex operations on remote hosts.

To utilize the `/script` parameter in **WinSCP command line**, start by creating a script file named *upload_file_script.txt* on your desktop with your preferred text editor.

Prepare a blank text file named *a.txt* in the */tmp* directory of your remote computer.

Create a local directory at *C:\abc*.

Next, input the following contents into *upload_file_script.txt* and save it. This script downloads the *a.txt* file from the remote */tmp* directory and re-uploads it as *new_file.txt*.

```
# WinSCP Script for File Transfer
 open sftp://automate:automate@54.179.19.216/  -hostkey="ssh-rsa 2048
x4DeZzv4jcWEk2zeeJgr5JeJ+z0xA+lGa3LC0q/B+88="
 cd /tmp
 get a.txt C:\abc\
 close
 open sftp://automate:automate@54.179.19.216/ -hostkey="ssh-rsa 2048
x4DeZzv4jcWEk2zeeJgr5JeJ+z0xA+lGa3LC0q/B+88="
 cd /tmp
 put C:\abc\new_file.txt
 close
 exit
```

Execute this script using the `/script` parameter in WinSCP with the following command:

Use the **`/ini=nul`** switch to prevent WinSCP from saving any session configurations upon exit.

```
winscp.com /ini=nul /script=upload_file_script.txt
   # Output
   Authenticating...
   Session started.
   /tmp
   a.txt                     |           10 B |     0.0 KB/s | binary | 100%
   Session closed.
   # New session for upload
   Session started.
```

```
/tmp
C:\abc\new_file.txt        |                0 B |      0.0 KB/s | binary |    0%
Session closed.
```

Utilizing WinSCP.com for Key Conversion

WinSCP supports password and certificate-based or public-key authentication. To use public-key authentication, a compatible private key format is required. WinSCP assists in converting key formats for compatibility.

Related: Your Guide to X509 Certificates (For Mortals)

For instance, convert a PEM format private key (like those from AWS EC2) to a Putty-friendly format using **WinSCP command line**. The **/keygen** parameter in *winscp.com* facilitates this conversion.

- Apply the **/keygen** parameter, followed by the key's current path.
- Include the **-o** parameter to specify the output file path of the converted key.
- Optionally, use the **-c** parameter to add a comment to the converted key.

```
.\WinSCP.com /keygen C:\Users\shanky\Desktop\testing.pem -o C:\Users\shanky\Desktop\testing.ppk -
c "Converted from OpenSSH format"
```

Conclusion

With this comprehensive guide to the **WinSCP command line**, alongside the WinSCP GUI Guide, you're now equipped to fully leverage WinSCP's capabilities. Whether it's file transfers, command execution, script running, or key conversions, WinSCP is a versatile tool for any IT professional.

How do you plan to integrate WinSCP into your workflow? Share your thoughts and experiences!