MyWebUniversity

python PowerShell FPDF Library PDF generator

Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Start-BitsTransfer'

PS C:\Users\wahid> Get-Help Start-BitsTransfer

```
NAME

    Start-BitsTransfer


SYNOPSIS

    Creates a BITS transfer job.



SYNTAX

    Start-BitsTransfer [-Source] <String[]> [[-Destination] <String[]>] [-ACLFlags

    <ACLFlagValue>] [-Asynchronous] [-Authentication {Basic | Digest | Ntlm |

    Negotiate | Passport}] [-CertHash <Byte[]>] [-CertStoreLocation

    <CertStoreLocationValue>] [-CertStoreName <String>] [-Confirm] [-Credential

    <PSCredential>] [-CustomHeaders <String[]>] [-CustomHeadersWriteOnly]

    [-Description <String>] [-DisplayName <String>] [-Dynamic] [-HttpMethod

    <String>] [-MaxDownloadTime <Int32>] [-NotifyCmdLine <String[]>] [-NotifyFlags

    <NotifyFlagValue>] [-Priority {Foreground | High | Normal | Low}]

    [-ProxyAuthentication {Basic | Digest | Ntlm | Negotiate | Passport}]

    [-ProxyBypass <String[]>] [-ProxyCredential <PSCredential>] [-ProxyList

    <Uri[]>] [-ProxyUsage {SystemDefault | NoProxy | AutoDetect | Override}]

    [-RetryInterval <Int32>] [-RetryTimeout <Int32>] [-SecurityFlags

    <SecurityFlagValue>] [-Suspended] [-TransferPolicy {None | Unrestricted |
```

Capped | BelowCap | NearCap | OverCapCharged | OverCapThrottled | UsageBased |
Roaming | IgnoreCongestion | PolicyUnrestricted | Standard | NoSurcharge |
NotRoaming | Always}] [-TransferType {Download | Upload | UploadReply}]
[-UseStoredCredential {None | Server | Proxy}] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The Start-BitsTransfer cmdlet creates a Background Intelligent Transfer Service (BITS) transfer job to transfer one or more files between a client computer and a server. The TransferType parameter specifies the direction of the transfer. By default, after the cmdlet begins the transfer, the command prompt is not available until the transfer is complete or until the transfer enters an error state. If the state of the returned BitsJob object is Error, the error code and description are contained in the object and can be used for analysis.

The Start-BitsTransfer cmdlet supports the download of multiple files from a server to a client computer, but it does not generally support the upload of multiple files from a client computer to a server. If you need to upload more than one file, you can use the Import-Csv cmdlet to pipe the output to the Add-BitsFile cmdlet to upload multiple files. Or, if you need to upload more than one file, consider a cabinet file (.cab) or a compressed file (.zip).

PARAMETERS

-ACLFlags <ACLFlagValue>

Specifies the owner and access control list (ACL) information to maintain for the transfer job. Specify one or more of the following values:

- o: Copy owner information with file.

- g: Copy group information with file.

- d: Copy discretionary access control list (DACL) information with file.

- s: Copy system access control list (SACL) information with file.

-Asynchronous [<SwitchParameter>]

Indicates that the cmdlet creates and processes BITS transfer job in the background. The command prompt reappears immediately after the BITS transfer job is created. The returned BitsJob object can be used to monitor status and progress.

-Authentication <String>

Specifies the authentication mechanism to be used at the server. The acceptable values for this parameter are:

- Basic : Basic is a scheme in which the user name and password are sent in clear text to the server or proxy.

- Digest : Digest is a challenge-response scheme that uses a server-specified data string for the challenge.

- Ntlm : NT LAN Manager (NTLM) is a challenge-response scheme that uses the credentials of the user for authentication in a Windows-based network environment.

- Negotiate (the default): Negotiate is a challenge-response scheme that negotiates with the server or proxy to determine which scheme to use for authentication. For example, this parameter value allows negotiation to determine whether the Kerberos protocol or NTLM is used.

- Passport : Passport is a centralized authentication service provided by Microsoft that offers a single logon for member sites.

-CertHash <Byte[]>

Specifies a SHA1 hash that identifies the certificate.

-CertStoreLocation <CertStoreLocationValue>

Specifies the certificate store location to use for to look up the certificate. Valid values are:

- CURRENT_USER

- LOCAL_MACHINE

- CURRENT_SERVICE

- SERVICES

- USERS

- CURRENT_USER_GROUP_POLICY

- LOCAL_MACHINE_GROUP_POLICY

- LOCAL_MACHINE_ENTERPRISE

-CertStoreName <String>

Specifies the name of the certificate store. Valid values are:

- CA: Certification Authority certificates

- MY: Personal certificates

- ROOT: Root certificates

- SPC: Software Publisher Certificate

-Confirm [<SwitchParameter>]

Prompts you for confirmation before running the cmdlet.

-Credential <PSCredential>

Specifies the credentials to use to authenticate the user to the server that is specified in the value of the Source parameter. The default is the current user.

Type a user name, such as "User01", "Domain01\User01", or "User@Contoso.com". Or, use the Get-Credential cmdlet to create the value for this parameter. When you type a user name, you are prompted for a password.

-CustomHeaders <String[]>

Specifies one or more custom HTTP headers to include in the request to the server. Specify an array of strings.

-CustomHeadersWriteOnly [<SwitchParameter>]

Indicates that the HTTP custom headers for this job are write-only.

Use this parameter when your custom headers include security information. Other programs on the same computer can't read the header. The BITS process can read the headers and send them over the HTTP connection.

You cannot change this value for a job after you set headers to write-only.

-Description <String>

Describes the BITS transfer job. The description is limited to 1,024 characters.

-Destination <String[]>

Specifies an array that contains the destination location and the names of the files that you want to transfer. The destination names are paired with the corresponding source file names. For example, the first file name specified in the Source parameter corresponds to the first file name in the Destination parameter, and the second file name in the Source parameter corresponds to the second file name in the Destination parameter. The Source and Destination parameters must have the same number of elements; otherwise, the command produces an error.

-DisplayName <String>

Specifies a display name for the BITS transfer job. The display name provides a user-friendly way to differentiate BITS transfer jobs.

-Dynamic [<SwitchParameter>]

Indicates that the transfer uses the dynamic setting.

-HttpMethod <String>

Specifies a method for the transfer other than the default method GET. If you specify GET, the parameter has no effect.

If you specify a method, the job takes foreground priority, which can't be changed.

-MaxDownloadTime <Int32>

Specifies the maximum time, in seconds, for transferring the files in a job. The default is 7,776,000 seconds or 90 days.

-NotifyCmdLine <String[]>

Specifies a program to run after the job finishes or encounters an error. The program runs in the context of the user who runs this cmdlet.

Specify the program name and any parameters as an array of strings.

-NotifyFlags <NotifyFlagValue>

Specifies the type of event notification you want to receive, such as job transferred events. Valid values are:

- 1: Generates an event when all files in the job have been transferred.

- 2: Generates an event when an error occurs.

- 4: Disables notifications.

The default value is 1|2.

-Priority <String>

Sets the priority of the BITS transfer job, which affects bandwidth usage.
The acceptable values for this parameter are:

- Foreground (default): Transfers the job in the foreground. Foreground transfers compete for network bandwidth with other applications, which can impede the user's overall network experience. However, if the Start-BitsTransfer cmdlet is being used interactively, this is likely the best option. This is the highest priority level.

- High : Transfers the job in the background with a high priority. Background transfers use the idle network bandwidth of the client computer to transfer files.

- Normal : Transfers the job in the background with a normal priority. Background transfers use the idle network bandwidth of the client computer to transfer files.

- Low : Transfers the job in the background with a low priority. Background transfers use the idle network bandwidth of the client to transfer files. This is the lowest background priority level.

-ProxyAuthentication <String>

Specifies the authentication mechanism to use at the Web proxy. The acceptable values for this parameter are:

- Basic : Basic is a scheme in which the user name and password are sent in clear-text to the server or proxy.

- Digest : Digest is a challenge-response scheme that uses a server-specified data string for the challenge.

- Ntlm : NTLM is a challenge-response scheme that uses the credentials of the user for authentication in a Windows-based network environment.

- Negotiate (the default): Negotiate is a challenge-response scheme that negotiates with the server or proxy to determine which scheme to use for authentication. For instance, this parameter value allows negotiation to determine whether the Kerberos protocol or NTLM is used.

- Passport : Passport is a centralized authentication service provided by Microsoft that offers a single logon for member sites.

-ProxyBypass <String[]>

Specifies a list of host names to use for a direct connection. The hosts in the list are tried in order until a successful connection is achieved.
If you specify this parameter the cmdlet bypasses the proxy. If this parameter is used, the ProxyUsage parameter must be set to Override ; otherwise, an error occurs.

-ProxyCredential <PSCredential>

Specifies the credentials to use to authenticate the user at the proxy.
You can use the Get-Credential cmdlet to create a value for this parameter.

-ProxyList <Uri[]>

Specifies a list of proxies to use. The proxies in the list are tried in order until a successful connection is achieved. If this parameter is specified and ProxyUsage is set to a value other than Override , the cmdlet generates an error.

-ProxyUsage <String>

Specifies the proxy usage settings. The acceptable values for this parameter are:

- SystemDefault (the default): Use the system default proxy settings.

- NoProxy : Do not use a proxy to transfer files. Use this option when you transfer files within a local area network (LAN).

- AutoDetect : Automatically detect proxy settings. BITS detects proxy settings for each file in the job.

- Override : Specify the proxies or servers to use. If the ProxyList parameter is also specified, the proxies in that list are used. If the ProxyBypass parameter is also specified, the servers in that list are used. In both cases, the first member of the list is used. If the first member is unreachable, the subsequent members are tried until a member is contacted successfully.

-RetryInterval <Int32>

Specifies the minimum length of time, in seconds, that BITS waits before an attempt to transfer the file after BITS encounters a transient error. The minimum allowed value is 60 seconds. If this value exceeds the RetryTimeout value from the BitsJob object, BITS does not retry the transfer. Instead, BITS sets the state of the BITS transfer job to the Error state.

The default is 600 seconds (10 minutes).

-RetryTimeout <Int32>

Specifies the length of time, in seconds, that BITS attempts to transfer the file after the first transient error occurs. Setting the retry period to 0 prevents retries and forces the job into the `BG_JOB_STATE_ERROR` state when an error occurs. If the retry period value exceeds the JobInactivityTimeout Group Policy setting (90-day default), BITS cancels the job after the JobInactivityTimeout Group Policy setting is exceeded.

The default is 1,209,600 seconds (14 days).

-SecurityFlags <SecurityFlagValue>

Specifies security flags for the HTTP request.

The flags you can set, from the least significant bit, are the following bits:

- 1: Enable CRL Check.

- 2: Ignore incorrect common names in the server certificate.

- 3: Ignore incorrect dates in the server certificate.

- 4: Ignore incorrect certification authorities in the server certificate.

- 5: Ignore incorrect usage of the server certificate.

- 12: Allow redirection from HTTPS to HTTP.

Use bits 9 through 11 to implement your redirection policy:

- 0,0,0: Redirects are automatically allowed.

- 0,0,1: Remote name is updated if a redirect occurs.
-0,1,0: BITS fails the job if a redirect occurs.

-Source <String[]>

Specifies the source location and the names of the files that you want to transfer. The source file names are paired with the corresponding destination file names. For example, the first file name specified in the Source parameter corresponds to the first file name in the Destination parameter, and the second file name in the Source parameter corresponds to the second file name in the Destination parameter. The Source and Destination parameters must have the same number of elements; otherwise, the command produces an error. You can use standard wildcard characters such as the asterisk (*) and the question mark (?). Or, you can use a range operator such as "[a-r]".

-Suspended [<SwitchParameter>]

Indicates that the cmdlet suspends the BITS transfer job. If the Suspended parameter is not specified, the job automatically begins the transfer job.
If the Suspended parameter is specified, the command prompt returns immediately after the BITS transfer job is created. You can use the Resume-BitsTransfer cmdlet to start the transfer job.

-TransferPolicy <CostStates>

Specifies the network cost states in which the transfer is allowed to be scheduled. The current cost state of the network is a bitmask that indicates the kinds of charges that would be incurred if a transfer was scheduled at this time. This cost state represents a bitmask; if the bit corresponding to the current network cost state is set, the transfer can be scheduled. If the bit corresponding to the current network cost state is not set, the transfer is ignored for scheduling purposes. You can submit any of the named values listed here, or add them together to provide a custom value.

The acceptable values for this parameter are:

- Unrestricted (or unknown) : 0x00000001 : the cost state for this network is not known.

- Capped : 0x00000002 : the cost state for this network is a capped plan, or a plan that has a data usage limit.

- BelowCap : 0x00000004 : the cost state for this network is below the data plan cap.

- NearCap : 0x00000008 : the cost state for this network is near the data plan cap.

- OverCapCharged : 0x00000010 : the cost state for this network is above the data plan cap, and such usage is charged.

- OverCapThrottled : 0x00000020 : the cost state for this network is above the data plan cap, and such usage is throttled.

- UsageBased : 0x00000040 : the cost state for this network is charged based on usage.

- Roaming : 0x00000080 : the cost state for this network incurs roaming charges. The cost state also includes one option (IgnoreCongestion) and a set of standard policies (Uncosted, Standard, NoSurcharge, NotRoaming, and Always) which are combinations of the discrete bit values.

- IgnoreCongestion : 0x80000000 : the job can be scheduled even if the network provider reports that the network is congested.

- PolicyUnrestricted : 0x80000021 : the set of cost states that do not consume the quota of a capped plan, or incur extra charges.

- Standard : 0x80000067 : a set of cost states suitable for moderate-priority transfers.

- NoSurcharge : 0x8000006f : the set of cost states that incur no surcharge for use.

- NotRoaming : 0x8000007f : the set of cost states that exclude the roaming state.

- Always : 0x800000ff : the set of all cost states.

The cost state also includes one option (IgnoreCongestion) and a set of standard policies (Always, NotRoaming, NoSurcharge, Standard, and Uncosted) which are combinations of the discrete bit values.

-TransferType <String>

Specifies the BITS transfer job type. The acceptable values for this parameter are:

- Download (the default): Specifies that the transfer job downloads files to the client computer.

- Upload : Specifies that the transfer job uploads a file to the server.

- UploadReply : Specifies that the transfer job uploads a file to the server and receives a reply file from the server.

-UseStoredCredential <AuthenticationTargetValue>

Specifies that credentials stored in the Windows Credential Manager should be used for authentication when required for the specified target server type. If this parameter is not specified and a server requires authentication, then explicit credentials must be included by using the Credential or ProxyCredential parameters. This parameter is a flag parameter whose values can be added together to create the desired behavior.

The acceptable values for this parameter are:

- None : Use only credentials provided by the Credential or ProxyCredential parameters. This is the default behavior if the parameter is not specified.

- Proxy : Credentials stored in the Windows Credential Manager are used for authentication for any proxy server that requires authentication. If no credentials in the Windows Credential Manager match the proxy server needing authentication, then you must specify credentials by using the ProxyCredential parameter.

- Server : This value is not supported and generates an error if specified.

-WhatIf [<SwitchParameter>]

Shows what would happen if the cmdlet runs. The cmdlet is not run.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

- Example 1: Create a BITS transfer job that downloads a file -

```
PS C:\> Start-BitsTransfer -Source
```

"http://server01/servertestdir/testfile1.txt" -Destination
"c:\clienttestdir\testfile1.txt"

This command creates a BITS transfer job that downloads a file from a server. The local and remote names of the file are specified in the Source and Destination parameters. Because the default transfer type is Download, the `http://Server01/servertestdir/testfile1.txt` file is transferred to `C:\clienttestdir\testfile1.txt` on the client. The command prompt returns when the file transfer is complete or when it enters an error state.

When you upload files to an HTTP location, the TransferType parameter must be set to Upload .

Because the Start-BitsTransfer cmdlet assumes that the first parameter is the source and that the second parameter is the destination when no value is specified, this command could be simplified as follows:

`Start-BitsTransfer "http://server01/servertestdir/testfile1.txt"
"c:\clienttestdir\testfile1.txt"`
Example 2: Create BITS transfer jobs that download multiple files

```
PS C:\> Import-CSV filelist.txt | Start-BitsTransfer
```

This command creates BITS transfer jobs that download multiple files from a server.

The command imports the source and destination file locations and then pipes the locations to the Start-BitsTransfer command. The Start-BitsTransfer command creates a new BITS transfer job for each of the files in `filelist.txt` and then transfers them concurrently to the client.

The contents of the filelist.txt file resemble the following information:

Source, Destination http://server01/servertestdir/testfile1.txt,
c:\clienttestdir\testfile1.txt http://server01/servertestdir/testfile2.txt,
c:\clienttestdir\testfile2.txt http://server01/servertestdir/testfile3.txt,
c:\clienttestdir\testfile3.txt http://server01/servertestdir/testfile4.txt,
c:\clienttestdir\testfile4.txt Note: First line of the file must include
Source, Destination header as in the example.
-- Example 3: Create a BITS transfer job that uploads a file --

```
PS C:\> Start-BitsTransfer -Source "c:\clienttestdir\testfile1.txt"
-Destination "http://server01/servertestdir/testfile1.txt" -TransferType Upload
```

This command creates a BITS transfer job that uploads a file to a server. The local and remote names of the file are specified in the Source and Destination parameters. Because the default transfer type is Download, the TransferType parameter must be set to Upload . The `C:\clienttestdir\testfile1.txt` file on the client is transferred to `http://Server01/servertestdir/testfile1.txt`.
The command prompt returns when the file transfer is complete or enters an error state.

The Start-BitsTransfer cmdlet downloads multiple files from a server to a client computer, but it does not typically upload multiple files from a client computer to a server. It is possible to work around this limitation by using the Import-Csv cmdlet to pipe the output to the Start-BitsTransfer cmdlet. If you need to upload more than one file, you can also use a .cab or .zip file.

Because the Start-BitsTransfer cmdlet assumes that the first parameter is the source and that the second parameter is the destination when no value is specified, this command could be simplified as follows:

`Start-BitsTransfer "c:\clienttestdir\testfile1.txt"
"http://server01/servertestdir/testfile1.txt" -TransferType Upload`
Example 4: Create a BITS transfer job that downloads multiple files

```
PS C:\> Start-BitsTransfer -Source
"http://server01/servertestdir/testfile1.txt",
"http://server01/servertestdir/testfile2.txt" -Destination
"c:\clienttestdir\testfile1.txt", "c:\clienttestdir\testfile2.txt"
```

This command creates a BITS transfer job that downloads multiple files from a server.

The local and remote names of the files are specified in the Source and Destination parameters. Because the default of the TransferType parameter is Download, the `http://Server01/servertestdir/testfile1.txt` and `http://Server01/servertestdir/testfile2.txt` files are transferred to `C:\clienttestdir\testfile1.txt` and `C:\clienttestdir\testfile2.txt` on the client computer. The command prompt returns when the file transfer is complete or enters an error state.

Example 5: Create a BITS transfer job that downloads a file using a specific set of credentials

```
PS C:\> $Cred = Get-Credential
PS C:\> Start-BitsTransfer -DisplayName MyJob -Credential $Cred -Source
"http://server01/servertestdir/testfile1.txt" -Destination
"c:\clienttestdir\testfile1.txt"
```

This example creates a BITS transfer job that downloads a file from a server by using a specific set of credentials.

The first command retrieves a set of credentials from the user by calling the Get-Credential cmdlet. The returned PSCredential object is stored in the $Cred variable.

The second command uses the Credential parameter to pass the PSCredential object that is stored in the $Cred variable to the Start-BitsTransfer cmdlet.
A new BITS transfer job is created that downloads the

`http://server01/servertestdir/testfile1.txt` file to the client. The
specified credentials are used to authenticate the user at the server.
Additionally, the optional DisplayName parameter is used to give the BITS
transfer job a unique name.

Example 6: Create BITS transfer jobs that download multiple files

```
PS C:\> Import-CSV filelist.txt | Start-BitsTransfer -Asynchronous -Priority
Normal
```

This command creates BITS transfer jobs that download multiple files from a
server. The files are downloaded sequentially, but they are available
immediately when the transfer job is complete.

The command imports the source and destination file locations and then pipes
them to the Start-BitsTransfer command. The Start-BitsTransfer command creates
a new BITS transfer job for each of the files in filelist.txt and then
transfers them sequentially to the client.

The contents of the filelist.txt file resemble the following information:

Source, Destination http://server01/servertestdir/testfile1.txt,
c:\clienttestdir\testfile1.txt http://server01/servertestdir/testfile2.txt,
c:\clienttestdir\testfile2.txt http://server01/servertestdir/testfile3.txt,
c:\clienttestdir\testfile3.txt http://server01/servertestdir/testfile4.txt,
c:\clienttestdir\testfile4.txt Note: First line of the file must include
Source, Destination header as in the example.

Example 7: Create a BITS transfer job that downloads multiple files

```
PS C:\> Start-BitsTransfer -Source C:\clientsourcedir\*.txt -Destination
c:\clientdir\ -TransferType Download
```

In the preceding example, the Start-BitsTransfer command creates a new BITS
transfer job. All of the files are added to this job and transferred

sequentially to the client.

> [!NOTE] > The destination path cannot use wildcard characters. The
destination path supports relative directories, root paths, or implicit
directories (that is, the current directory). Destination files cannot be
renamed by using a wildcard character. Additionally, HTTP and HTTPS URLs do
not work with wildcards. Wildcards are only valid for UNC paths and local
directories.

Example 8: Create BITS transfer jobs that upload multiple files

```
PS C:\> Import-CSV filelist.txt | Start-BitsTransfer -TransferType Upload
```

This command creates BITS transfer jobs that upload multiple files to a server.

The command imports the source and destination file locations and then pipes
them to the Start-BitsTransfer command. The Start-BitsTransfer command creates
a new BITS transfer job for each of the files in filelist.txt and then
transfers them concurrently to the server.

The contents of the filelist.txt file resemble the following information:

Source, Destination c:\clienttestdir\testfile1.txt,
http://server01/servertestdir/testfile1.txt c:\clienttestdir\testfile2.txt,
http://server01/servertestdir/testfile2.txt c:\clienttestdir\testfile3.txt,
http://server01/servertestdir/testfile3.txt c:\clienttestdir\testfile4.txt,
http://server01/servertestdir/testfile4.txt Note: First line of the file must
include Source, Destination header as in the example.

Example 9: Download a file from a server on a network to a client on a
different network that are connected by a proxy server

```
PS C:\> Start-BitsTransfer -Source .\Patch0416.msu -Destination
$env:temp\Patch0416.msu -ProxyUsage Override -ProxyList BitsProxy:8080
-ProxyCredential Server01\Admin01
```

This command uses the Start-BitsTransfer cmdlet to copy a patch file from a server on one network to a client on a different network when the networks are connected only by a proxy server.

This scenario arises when an Internet-connected server downloads files and then distributes them to computers on disconnected or isolated networks that have no Internet access.

BITS can detect proxy server settings automatically. However, if the proxy servers are not configured for automatic detection, you can override the automatic detection mechanism and identify the proxy server explicitly, as shown in this example.

The command uses the Source parameter to specify the location of the patch on the server computer and the Destination parameter to specify the intended location of patch on the client computer. It uses the ProxyUsage parameter with a value of Override to override the automatic proxy server detection mechanism. To identify the proxy server, it uses the ProxyList parameter. The value of the ProxyList parameter is a URI with a `<Name:Port>` format. Finally, it uses the ProxyCredential parameter to specify the credentials of an administrator who has permission to connect to the proxy server.

REMARKS

To see the examples, type: "get-help Start-BitsTransfer -examples".
For more information, type: "get-help Start-BitsTransfer -detailed".
For technical information, type: "get-help Start-BitsTransfer -full".
For online help, type: "get-help Start-BitsTransfer -online"