**Full credit is given to all the above companies including the Operating System that this PDF file was generated!**

# *Windows PowerShell Get-Help on Cmdlet 'Export-Clixml'*

***PS:\>Get-HELP Export-Clixml -Full***

NAME

Export-Clixml

SYNOPSIS

Creates an XML-based representation of an object or objects and stores it in a file.

SYNTAX

Export-Clixml [-Depth <System.Int32>] [-Encoding {ASCII | BigEndianUnicode | Default | OEM | Unicode | UTF7 | UTF8 | UTF32}] [-Force] -InputObject <System.Management.Automation.PSObject> -LiteralPath <System.String> [-NoClobber] [-Confirm] [-WhatIf] [<CommonParameters>]

Export-Clixml [-Path] <System.String> [-Depth <System.Int32>] [-Encoding {ASCII | BigEndianUnicode | Default | OEM | Unicode | UTF7 | UTF8 | UTF32}] [-Force] -InputObject <System.Management.Automation.PSObject> [-NoClobber] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Export-Clixml` cmdlet serialized an object into a Common Language Infrastructure (CLI) XML-based representation stores it in a file. You can then use the

`Import-Clixml` cmdlet to recreate the saved object based on the contents of that file. For more information about CLI, see Language independence

(/dotnet/standard/language-independence).

This cmdlet is similar to `ConvertTo-Xml`, except that `Export-Clixml` stores the resulting XML in a file. `ConvertTo-XML` returns the XML, so you can continue to

process it in PowerShell.

A valuable use of `Export-Clixml` on Windows computers is to export credentials and secure strings securely as XML. For an example, see Example 3.

PARAMETERS

-Depth <System.Int32>

Specifies how many levels of contained objects are included in the XML representation. The default value is `2`.

The default value can be overridden for the object type in the `Types.ps1xml` files. For more information, see about_Types.ps1xml

(../Microsoft.PowerShell.Core/About/about_Types.ps1xml.md).

```
Required?                    false
Position?                    named
Default value                2
Accept pipeline input?       False
Accept wildcard characters?  false
```

-Encoding <System.String>

Specifies the type of encoding for the target file. The default value is Unicode .

The acceptable values for this parameter are as follows:

- `ASCII` Uses ASCII (7-bit) character set.

- `BigEndianUnicode` Uses UTF-16 with the big-endian byte order.

- `Default` Uses the encoding that corresponds to the system's active code page (usually ANSI).

- `OEM` Uses the encoding that corresponds to the system's current OEM code page.

- `Unicode` Uses UTF-16 with the little-endian byte order.

- `UTF7` Uses UTF-7.

- `UTF8` Uses UTF-8.

- `UTF32` Uses UTF-32 with the little-endian byte order.

```
Required?                    false
Position?                    named
Default value                Unicode
Accept pipeline input?       False
Accept wildcard characters?  false
```

-Force <System.Management.Automation.SwitchParameter>

Forces the command to run without asking for user confirmation.

Causes the cmdlet to clear the read-only attribute of the output file if necessary. The cmdlet will attempt to reset the read-only attribute when the command

completes.

```
Required?                    false
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-InputObject <System.Management.Automation.PSObject>

Specifies the object to be converted. Enter a variable that contains the objects, or type a command or expression that gets the objects. You can also pipe objects to `Export-Clixml`.

```
Required?                    true
Position?                    named
Default value                None
Accept pipeline input?       True (ByValue)
Accept wildcard characters?  false
```

-LiteralPath <System.String>

Specifies the path to the file where the XML representation of the object will be stored. Unlike Path , the value of the LiteralPath parameter is used exactly as it's typed. No characters are interpreted as wildcards. If the path includes escape characters, enclose it in single quotation marks. Single quotation marks tell PowerShell not to interpret any characters as escape sequences.

```
Required?                    true
Position?                    named
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-NoClobber <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet doesn't overwrite the contents of an existing file. By default, if a file exists in the specified path, `Export-Clixml` overwrites the file without warning.

```
Required?                    false
Position?                    named
Default value                False
```

```
    Accept pipeline input?       False
    Accept wildcard characters?  false

-Path <System.String>
    Specifies the path to the file where the XML representation of the object will be stored.

    Required?                    true
    Position?                    0
    Default value                None
    Accept pipeline input?       False
    Accept wildcard characters?  false

-Confirm <System.Management.Automation.SwitchParameter>
    Prompts you for confirmation before running the cmdlet.

    Required?                    false
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false

-WhatIf <System.Management.Automation.SwitchParameter>
    Shows what would happen if the cmdlet runs. The cmdlet isn't run.

    Required?                    false
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false

<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).

INPUTS

System.Management.Automation.PSObject

You can pipeline any object to this cmdlet.

OUTPUTS

System.IO.FileInfo

This cmdlet returns a FileInfo object representing the created file with the stored data.

NOTES

---------- Example 1: Export a string to an XML file ----------

```
"This is a test" | Export-Clixml -Path .\sample.xml
```

The string `This is a test` is sent down the pipeline. `Export-Clixml` uses the Path parameter to create an XML file named `sample.xml` in the current directory.

---------- Example 2: Export an object to an XML file ----------

```
Get-Acl C:\test.txt | Export-Clixml -Path .\FileACL.xml
$fileacl = Import-Clixml -Path .\FileACL.xml
```

The `Get-Acl` cmdlet gets the security descriptor of the `Test.txt` file. It sends the object down the pipeline to pass the security descriptor to `Export-Clixml`.

The XML-based representation of the object is stored in a file named `FileACL.xml`.

The `Import-Clixml` cmdlet creates an object from the XML in the `FileACL.xml` file. Then, it saves the object in the `$fileacl` variable.

------- Example 3: Encrypt an exported credential object -------

```
$Credxmlpath = Join-Path (Split-Path $Profile) TestScript.ps1.credential
$Credential | Export-Clixml $Credxmlpath
$Credxmlpath = Join-Path (Split-Path $Profile) TestScript.ps1.credential
$Credential = Import-Clixml $Credxmlpath
```

The `Export-Clixml` cmdlet encrypts credential objects by using the Windows Data Protection API (/previous-versions/windows/apps/hh464970(v=win.10)). The encryption

ensures that only your user account on only that computer can decrypt the contents of the credential object. The exported `CLIXML` file can't be used on a different

computer or by a different user.

In the example, the file in which the credential is stored is represented by `TestScript.ps1.credential`. Replace TestScript with the name of the script with which

you're loading the credential.

You send the credential object down the pipeline to `Export-Clixml`, and save it to the path, `$Credxmlpath`, that you specified in the first command.

To import the credential automatically into your script, run the final two commands. Run `Import-Clixml` to import the secured credential object into your script.

This import eliminates the risk of exposing plain-text passwords in your script.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/export-clixml?view=powershell-5.1&WT.mc_id=ps-gethelp

ConvertTo-Html

ConvertTo-Xml

Export-Csv

Import-Clixml

Join-Path

Securely Store Credentials on Disk https://powershellcookbook.com/recipe/PukO/securely-store-credentials-on-disk

Use PowerShell to Pass Credentials to Legacy Systems https://devblogs.microsoft.com/scripting/use-powershell-to-pass-credentials-to-legacy-systems/

Windows.Security.Cryptography.DataProtection