PowerShell Get-Help on command 'Import-Clixml'

PS C:\Users\wahid> Get-Help Import-Clixml

NAME

Import-Clixml

SYNOPSIS

Imports a CLIXML file and creates corresponding objects in PowerShell.

SYNTAX

Import-Clixml [-First <System.UInt64>] [-IncludeTotalCount] -LiteralPath <System.String[]> [-Skip <System.UInt64>] [<CommonParameters>]

Import-Clixml [-Path] <System.String[]> [-First <System.UInt64>] [-IncludeTotalCount] [-Skip <System.UInt64>] [<CommonParameters>]

DESCRIPTION

The `Import-Clixml` cmdlet imports objects that have been serialized into a Common Language Infrastructure (CLI) XML file. A valuable use of `Import-Clixml` on Windows computers is to import credentials and secure strings that were exported as secure XML using `Export-Clixml`. Example #2 (#example-2-import-a-secure-credential-object)shows how to use `Import-Clixml` to import a secure credential object.

The CLIXML data is deserialized back into PowerShell objects. However, the deserialized objects aren't a live objects. They are a snapshot of the objects at the time of serialization. The deserialized objects include properties but no methods.

The TypeNames property contains the original type name prefixed with `Deserialized`. Example #3 (#example-3-inspect-the-typenames-property-of-a-deserialized-object)show the TypeNames property of a deserialized object.

`Import-Clixml` uses the byte-order-mark (BOM) to detect the encoding format of the file. If the file has no BOM, it assumes the encoding is UTF8.

For more information about CLI, see Language independence (/dotnet/standard/language-independence).

PARAMETERS

-First <System.UInt64>

Gets only the specified number of objects. Enter the number of objects to get.

-IncludeTotalCount <System.Management.Automation.SwitchParameter>

Reports the total number of objects in the data set followed by the selected objects. If the cmdlet can't determine the total count, it displays Unknown total count . The integer has an Accuracy property that indicates the reliability of the total count value. The value of Accuracy ranges from `0.0` to `1.0` where `0.0` means that the cmdlet couldn't count the objects, `1.0` means that the count is exact, and a value between `0.0` and `1.0` indicates an increasingly reliable estimate.

-LiteralPath <System.String[]>

Specifies the path to the XML files. 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.

-Path <System.String[]>

Specifies the path to the XML files.

-Skip <System.UInt64>

Ignores the specified number of objects and then gets the remaining objects. Enter the number of objects to skip.

<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: Import a serialized file and recreate an object --

```
Get-Process | Export-Clixml -Path .\pi.xml
$Processes = Import-Clixml -Path .\pi.xml
```

--------- Example 2: Import a secure 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 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.

Example 3: Inspect the TypeNames property of a deserialized object

```
$original = [pscustomobject] @{
    Timestamp = Get-Date
    Label     = 'Meeting event'
}
$original | Add-Member -MemberType ScriptMethod -Name GetDisplay -Value {
    '{0:yyyy-MM-dd HH:mm} {1}' -f $this.Timestamp, $this.Label
}
$original | Get-Member -MemberType ScriptMethod

TypeName: System.Management.Automation.PSCustomObject

Name        MemberType   Definition
----        ----------   ----------
Equals      Method       bool Equals(System.Object obj)
```

```
    GetHashCode Method       int GetHashCode()
    GetType     Method       type GetType()
    ToString    Method       string ToString()
    Label       NoteProperty string Label=Meeting event
    Timestamp   NoteProperty System.DateTime Timestamp=1/31/2024 2:27:59 PM
    GetDisplay  ScriptMethod System.Object GetDisplay();

    $original | Export-Clixml -Path event.clixml
    $deserialized = Import-CliXml -Path event.clixml
    $deserialized | Get-Member

    TypeName: Deserialized.System.Management.Automation.PSCustomObject

    Name        MemberType   Definition
    ----        ----------   ----------
    Equals      Method       bool Equals(System.Object obj)
    GetHashCode Method       int GetHashCode()
    GetType     Method       type GetType()
    ToString    Method       string ToString()
    Label       NoteProperty string Label=Meeting event
    Timestamp   NoteProperty System.DateTime Timestamp=1/31/2024 2:27:59 PM
```

Note that the type of the object in `$original` is
System.Management.Automation.PSCustomObject , but the type of the object in
`$deserialized` is Deserialized.System.Management.Automation.PSCustomObject .
Also, the `GetDisplay()` method is missing from the deserialized object.

REMARKS

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