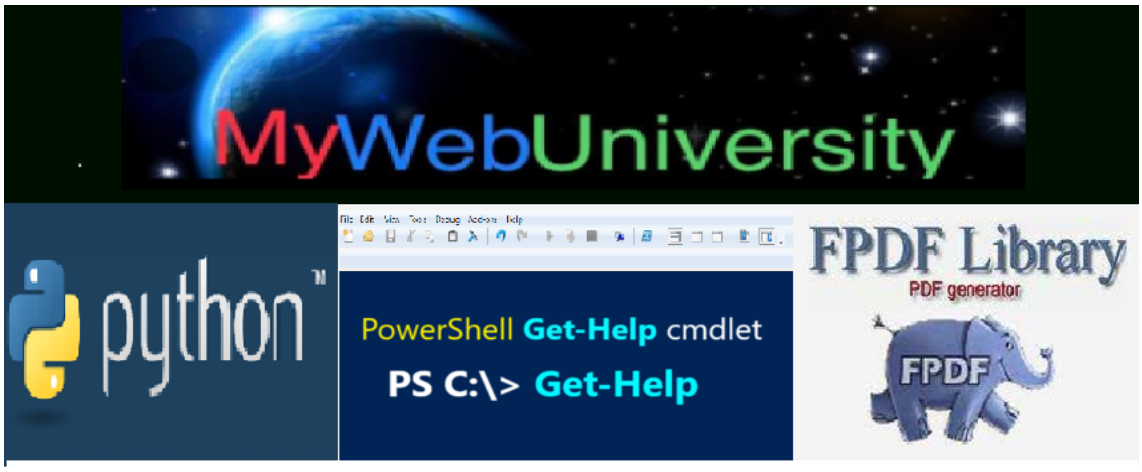

Full credit is given to all the above companies including the Operating System that this PDF file was generated!

Windows PowerShell Get-Help on Cmdlet 'Get-Credential'

PS:\>Get-HELP Get-Credential -Full

NAME

Get-Credential

SYNOPSIS

Gets a credential object based on a user name and password.

SYNTAX

Get-Credential [-Credential] <System.Management.Automation.PSCredential> [<CommonParameters>]

Get-Credential [[-UserName] <System.String>] -Message <System.String> [<CommonParameters>]

DESCRIPTION

The `Get-Credential` cmdlet creates a credential object for a specified user name and password. You can use the credential object in security operations.

Beginning in PowerShell 3.0, you can use the Message parameter to specify a customized message on the dialog box that prompts the user for their name and password.

The `Get-Credential` cmdlet prompts the user for a password or a user name and password. By default, an authentication dialog box appears to prompt the user. However,

in some host programs, such as the PowerShell console, you can prompt the user at the command line by changing a registry entry. For more information about this

registry entry, see the notes and examples.

PARAMETERS

-Credential <System.Management.Automation.PSCredential>

Specifies a user name for the credential, such as User01 or Domain01\User01 . The parameter name, `-Credential`, is optional.

When you submit the command and specify a user name, you're prompted for a password. If you omit this parameter, you're prompted for a user name and a password.

Starting in PowerShell 3.0, if you enter a user name without a domain, `Get-Credential` no longer inserts a backslash before the name.

Credentials are stored in a PSCredential (/dotnet/api/system.management.automation.pscredential)object and the password is stored as a SecureString

(/dotnet/api/system.security.securestring).

> [!NOTE] > For more information about SecureString data protection, see > How secure is SecureString?

(/dotnet/api/system.security.securestring#how-secure-is-securestring).

Required? true

Position? 1

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Message <System.String>

Specifies a message that appears in the authentication prompt. This parameter is designed for use in a function or

script. You can use the message to explain to

the user why you are requesting credentials and how they will be used.

This parameter was introduced in PowerShell 3.0.

Required? true

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-UserName <System.String>

Specifies a user name. The authentication prompt requests a password for the user name. By default, the user name is blank and the authentication prompt requests

both a user name and password.

When the authentication prompt appears in a dialog box, the user can edit the specified user name. However, the user cannot change the user name when the prompt

appears at the command line. When using this parameter in a shared function or script, consider all possible presentations.

This parameter was introduced in PowerShell 3.0.

Required? false

Position? 1

Default value None (blank)

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

None

You can't pipe objects to this cmdlet.

OUTPUTS

System.Management.Automation.PSCredential

This cmdlet returns a credential object.

NOTES

You can use the PSCredential object that `Get-Credential` creates in cmdlets that request user authentication, such as those with a Credential parameter.

By default, the authentication prompt appears in a dialog box. To display the authentication prompt at the command line, add the ConsolePrompting registry entry (`HKLM:\SOFTWARE\Microsoft\PowerShell\1\ShellIds\ConsolePrompting`) and set its value to True . If the ConsolePrompting registry entry does not exist or if its value is False, the authentication prompt appears in a dialog box. For instructions, see the examples.

The ConsolePrompting registry entry works in the PowerShell console, but it does not work in all host programs.

For example, it has no effect in the PowerShell Integrated Scripting Environment (ISE). For information about the effect of the ConsolePrompting registry entry, see the help topics for the host program.

The Credential parameter is not supported by all providers that are installed with PowerShell. Beginning in PowerShell 3.0, it is supported on select cmdlets, such as the `Get-Content` and `New-PSDrive` cmdlets.

---------- Example 1 ----------

$c = Get-Credential

This command gets a credential object and saves it in the `$c` variable.

When you enter the command, a dialog box appears requesting a user name and password. When you enter the requested information, the cmdlet creates a PSCredential

object representing the credentials of the user and saves it in the `$c` variable.

You can use the object as input to cmdlets that request user authentication, such as those with a Credential parameter. However, some providers that are installed

with PowerShell do not support the Credential parameter.

---------- Example 2 ----------

$c = Get-Credential -credential User01

$c.Username

User01

This example creates a credential that includes a user name without a domain name.

The first command gets a credential with the user name User01 and stores it in the `$c` variable. The second command displays the value of the Username property of

the resulting credential object.

---------- Example 3 ----------

$Credential = $host.ui.PromptForCredential("Need credentials", "Please enter your user name and password.", "", "NetBiosUserName")

This command uses the PromptForCredential method to prompt the user for their user name and password. The command saves the resulting credentials in the `$Credential`

variable.

The PromptForCredential method is an alternative to using the `Get-Credential` cmdlet. When you use PromptForCredential , you can specify the caption, messages, and

user name that appear in the message box.

For more information, see the PromptForCredential (/dotnet/api/system.management.automation.host.pshostuserinterface.promptforcredential)documentation in the SDK.

---------- Example 4 ----------

```
Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\PowerShell\1\ShellIds" -Name ConsolePrompting -Value $true
```

This example shows how to modify the registry so that the user is prompted at the command line, instead of by using a dialog box.

The command creates the ConsolePrompting registry entry and sets its value to True. To run this command, start PowerShell with the "Run as administrator" option.

To use a dialog box for prompting, set the value of the ConsolePrompting to false ($false) or use the `Remove-ItemProperty` cmdlet to delete it.

The ConsolePrompting registry entry works in some host programs, such as the PowerShell console. It might not work in all host programs.

---------- Example 5 ----------

```
$User = "Domain01\User01"
$PWord = ConvertTo-SecureString -String "P@sSwOrd" -AsPlainText -Force
$Credential = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $User, $PWord
```

The first command saves the user account name in the `$User` parameter. The value must have the "Domain\User" or "ComputerName\User" format.

The second command uses the `ConvertTo-SecureString` cmdlet to create a secure string from a plain text password. The command uses the AsPlainText parameter to

indicate that the string is plain text and the Force parameter to confirm that you understand the risks of using plain text.

The third command uses the `New-Object` cmdlet to create a PSCredential object from the values in the `$User` and `$PWord` variables.

---------- Example 6 ----------

Get-Credential -Message "Credential are required for access to the \\Server1\Scripts file share." -User Server01\PowerUser

PowerShell Credential Request

Credential are required for access to the \\Server1\Scripts file share.

Password for user Server01\PowerUser:

This command uses the Message and UserName parameters of the `Get-Credential` cmdlet. This command format is designed for shared scripts and functions. In this case,

the message tells the user why credentials are needed and gives them confidence that the request is legitimate.

---------- Example 7 ----------

Invoke-Command -ComputerName Server01 {Get-Credential Domain01\User02}

PowerShell Credential Request : PowerShell Credential Request

Warning: This credential is being requested by a script or application on the SERVER01 remote computer. Enter your credentials only if you

trust the remote computer and the application or script requesting it.

Enter your credentials.

Password for user Domain01\User02: **********

PSComputerName : Server01

RunspaceId : 422bdf52-9886-4ada-ab2f-130497c6777f

PSShowComputerName : True

UserName : Domain01\User01

Password : System.Security.SecureString

This command gets a credential from the Server01 remote computer. The command uses the `Invoke-Command` cmdlet to run a `Get-Credential` command on the remote

computer. The output shows the remote security message that `Get-Credential` includes in the authentication prompt.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.security/get-credential?view=powershell-5.1&WT.mc_id=ps-gethelp

PromptForCredential