



Windows PowerShell Get-Help on Cmdlet 'ConvertTo-SecureString'

PS:\>Get-HELP ConvertTo-SecureString -Full

NAME

ConvertTo-SecureString

SYNOPSIS

Converts plain text or encrypted strings to secure strings.

SYNTAX

ConvertTo-SecureString [-String] <System.String> [[-AsPlainText]] [[-Force]] [<CommonParameters>]

ConvertTo-SecureString [-String] <System.String> [-Key <System.Byte[]>] [<CommonParameters>]

ConvertTo-SecureString [-String] <System.String> [[-SecureKey] <System.Security.SecureString>]
[<CommonParameters>]

DESCRIPTION

The `ConvertTo-SecureString` cmdlet converts encrypted standard strings into secure strings. It can also convert plain text to secure strings. It is used with

`ConvertFrom-SecureString` and `Read-Host`. The secure string created by the cmdlet can be used with `ConvertTo-SecureString` or

functions that require a parameter of type

SecureString . The secure string can be converted back to an encrypted, standard string using the ``ConvertFrom-SecureString`` cmdlet. This enables it to be stored in a file for later use.

If the standard string being converted was encrypted with ``ConvertFrom-SecureString`` using a specified key, that same key must be provided as the value of the Key or

SecureKey parameter of the ``ConvertTo-SecureString`` cmdlet.

PARAMETERS

`-AsPlainText <System.Management.Automation.SwitchParameter>`

Specifies a plain text string to convert to a secure string. The secure string cmdlets help protect confidential text. The text is encrypted for privacy and is

deleted from computer memory after it is used. If you use this parameter to provide plain text as input, the system cannot protect that input in this manner. To

use this parameter, you must also specify the Force parameter.

Required?	false
Position?	1
Default value	False
Accept pipeline input?	False
Accept wildcard characters?	false

`-Force <System.Management.Automation.SwitchParameter>`

Confirms that you understand the implications of using the AsPlainText parameter and still want to use it.

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

-Key <System.Byte[]>

Specifies the encryption key used to convert the original secure string into the encrypted standard string. Valid key lengths are 16, 24 and 32 bytes.

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

-SecureKey <System.Security.SecureString>

Specifies the encryption key used to convert the original secure string into the encrypted standard string. The key must be provided in the format of a secure string. The secure string will be converted to a byte array to be used as the key. Valid secure key lengths are 8, 12 and 16 code points.

Required?	false
Position?	1
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

-String <System.String>

Specifies the string to convert to a secure string.

Required?	true
Position?	0
Default value	None
Accept pipeline input?	True (ByValue)
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.String

You can pipe a standard encrypted string to this cmdlet.

OUTPUTS

System.Security.SecureString

This cmdlet returns the created SecureString object.

NOTES

Some characters, such as emoticons, correspond to several code points in the string that contains them. Avoid using these characters because they may cause

problems and misunderstandings when used in a password.

-- Example 1: Convert a secure string to an encrypted string --

```
PS C:\> $Secure = Read-Host -AsSecureString
```

```
PS C:\> $Secure
```

```
System.Security.SecureString
```

```
PS C:\> $Encrypted = ConvertFrom-SecureString -SecureString $Secure
```

```
PS C:\> $Encrypted
```

```
01000000d08c9ddf0115d1118c7a00c04fc297eb010000001a114d45b8dd3f4aa11ad7c0abdae980000000000
```

```
020000000000003660000a8000000100000005df63cea84bfb7d70bd6842e7efa79820000000004800000a000
```

```
000010000000f10cd0f4a99a8d5814d94e0687d7430b100000008bf11f1960158405b2779613e9352c6d1400
```

```
0000e6b7bf46a9d485ff211b9b2a2df3bd6eb67aae41
```

```
PS C:\> $Secure2 = ConvertTo-SecureString -String $Encrypted
```

```
PS C:\> $Secure2
```

```
System.Security.SecureString
```

The first command uses the `AsSecureString` parameter of the `'Read-Host'` cmdlet to create a secure string. After you enter the command, any characters that you type are

converted into a secure string and then saved in the `'$Secure'` variable.

The second command displays the contents of the `'$Secure'` variable. Because the `'$Secure'` variable contains a secure string, PowerShell displays only the

```
System.Security.SecureString type.
```

The third command uses the `'ConvertFrom-SecureString'` cmdlet to convert the secure string in the `'$Secure'` variable into an encrypted standard string. It saves the

result in the `'$Encrypted'` variable.

The fourth command displays the encrypted string in the value of the `'$Encrypted'` variable.

The fifth command uses the `'ConvertTo-SecureString'` cmdlet to convert the encrypted standard string in the `'$Encrypted'` variable back into a secure string. It saves

the result in the `'$Secure2'` variable. The sixth command displays the value of the `'$Secure2'` variable. The `SecureString` type indicates that the command was successful.

Example 2: Create a secure string from an encrypted string in a file

```
$Secure = Read-Host -AsSecureString
```

```
$Encrypted = ConvertFrom-SecureString -SecureString $Secure -Key (1..16)
```

```
$Encrypted | Set-Content Encrypted.txt
```

```
$Secure2 = Get-Content Encrypted.txt | ConvertTo-SecureString -Key (1..16)
```

The first command uses the `AsSecureString` parameter of the `'Read-Host'` cmdlet to create a secure string. After you enter the command, any characters that you type are

converted into a secure string and then saved in the `'$Secure'` variable.

The second command uses the `ConvertFrom-SecureString` cmdlet to convert the secure string in the `$Secure` variable into an encrypted standard string by using the specified key. The contents are saved in the `$Encrypted` variable.`

The third command uses a pipeline operator (``|``) to send the value of the ``$Encrypted` variable to the `Set-Content` cmdlet, which saves the value in the Encrypted.txt file.`

The fourth command uses the ``Get-Content` cmdlet to get the encrypted standard string in the Encrypted.txt file. The command uses a pipeline operator to send the encrypted string to the `ConvertTo-SecureString` cmdlet, which converts it to a secure string by using the specified key. The results are saved in the `$Secure2` variable.`

-- Example 3: Convert a plain text string to a secure string --

```
$Secure_String_Pwd = ConvertTo-SecureString "P@ssW0rD!" -AsPlainText -Force
```

> [CAUTION] > You should avoid using plain text strings in script or from the command line. The plain text can > show up in event logs and command history logs.

RELATED LINKS

Online

Version:

https://learn.microsoft.com/powershell/module/microsoft.powershell.security/convertto-securestring?view=powershell-5.1&WT.mc_id=ps-gethelp

ConvertFrom-SecureString

Read-Host