MyWebUniversity
python
PowerShell Get-Help cmdlet
PS C:\> Get-Help
FPDF Library
PDF generator
FPDF
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 'ConvertFrom-SecureString'*

***PS:\>Get-HELP ConvertFrom-SecureString -Full***

NAME

ConvertFrom-SecureString

SYNOPSIS

Converts a secure string to an encrypted standard string.

SYNTAX

ConvertFrom-SecureString [-SecureString] <System.Security.SecureString> [-Key <System.Byte[]>] [<CommonParameters>]

ConvertFrom-SecureString [-SecureString] <System.Security.SecureString> [[-SecureKey] <System.Security.SecureString>] [<CommonParameters>]

DESCRIPTION

The `ConvertFrom-SecureString` cmdlet converts a secure string ( System.Security.SecureString ) into an encrypted standard string ( System.String ). Unlike a secure

string, an encrypted standard string can be saved in a file for later use. The encrypted standard string can be converted back to its secure string format by using the `ConvertTo-SecureString` cmdlet.

If an encryption key is specified by using the Key or SecureKey parameters, the Advanced Encryption Standard (AES) encryption algorithm is used. The specified key
must have a length of 128, 192, or 256 bits because those are the key lengths supported by the AES encryption algorithm. If no key is specified, the Windows Data
Protection API (DPAPI) is used to encrypt the standard string representation.

PARAMETERS

-Key <System.Byte[]>

Specifies the encryption key as a byte array.

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

-SecureKey <System.Security.SecureString>

Specifies the encryption key as a secure string. The secure string value is converted to a byte array before being used as the key.

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

-SecureString <System.Security.SecureString>

Specifies the secure string to convert to an encrypted standard 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.Security.SecureString

You can pipe a SecureString object to this cmdlet.

OUTPUTS

System.String

This cmdlet returns the created plain text string.

NOTES

- To create a secure string from characters that are typed at the command prompt, use the AsSecureString parameter of the `Read-Host` cmdlet. - When you use the

Key or SecureKey parameters to specify a key, the key length must be   correct. For example, a key of 128 bits can be specified as a byte array of 16 decimal

numerals.   Similarly, 192-bit and 256-bit keys correspond to byte arrays of 24 and 32 decimal numerals,   respectively. - 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: Create a secure string --------------

```
$SecureString = Read-Host -AsSecureString
```

This command creates a secure string from characters that you type at the command prompt. After entering the command, type the string you want to store as a secure
string. An asterisk (`*`) is displayed to represent each character that you type.

Example 2: Convert a secure string to an encrypted standard string

```
$StandardString = ConvertFrom-SecureString $SecureString
```

This command converts the secure string in the `$SecureString` variable to an encrypted standard string. The resulting encrypted standard string is stored in the
`$StandardString` variable.

Example 3: Convert a secure string to an encrypted standard string with a 192-bit key

```
$Key = (3,4,2,3,56,34,254,222,1,1,2,23,42,54,33,233,1,34,2,7,6,5,35,43)
$StandardString = ConvertFrom-SecureString $SecureString -Key $Key
```

These commands use the Advanced Encryption Standard (AES) algorithm to convert the secure string stored in the `$SecureString` variable to an encrypted standard
string with a 192-bit key. The resulting encrypted standard string is stored in the `$StandardString` variable.

The first command stores a key in the `$Key` variable. The key is an array of 24 decimal numerals, each of which must be less than 256 to fit within a single unsigned
byte.

Because each decimal numeral represents a single byte (8 bits), the key has 24 digits for a total of 192 bits (8 x 24). This is a valid key length for the AES
algorithm.

The second command uses the key in the `$Key` variable to convert the secure string to an encrypted standard string.

RELATED LINKS

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

ConvertTo-SecureString

Read-Host