



## ***Windows PowerShell Get-Help on Cmdlet 'Set-AuthenticodeSignature'***

***PS:\>Get-HELP Set-AuthenticodeSignature -Full***

### **NAME**

Set-AuthenticodeSignature

### **SYNOPSIS**

Adds an Authenticode (/windows-hardware/drivers/install/authenticode)signature to a PowerShell script or other file.

### **SYNTAX**

Set-AuthenticodeSignature [-Certificate] <System.Security.Cryptography.X509Certificates.X509Certificate2> -Content  
<System.Byte[]> [-Force] [-HashAlgorithm

<System.String>] [-IncludeChain <System.String>] -SourcePathOrExtension <System.String[]> [-TimestampServer  
<System.String>] [-Confirm] [-WhatIf] [<CommonParameters>]

Set-AuthenticodeSignature [-Certificate] <System.Security.Cryptography.X509Certificates.X509Certificate2> [-FilePath]  
<System.String[]> [-Force] [-HashAlgorithm

<System.String>] [-IncludeChain <System.String>] [-TimestampServer <System.String>] [-Confirm] [-WhatIf]  
[<CommonParameters>]

Set-AuthenticodeSignature [-Certificate] <System.Security.Cryptography.X509Certificates.X509Certificate2> [-Force]  
[-HashAlgorithm <System.String>] [-IncludeChain

<System.String>] -LiteralPath <System.String[]> [-TimestampServer <System.String>] [-Confirm] [-WhatIf]  
[<CommonParameters>]

## DESCRIPTION

The ``Set-AuthenticodeSignature`` cmdlet adds an Authenticode signature to any file that supports Subject Interface Package (SIP).

In a PowerShell script file, the signature takes the form of a block of text that indicates the end of the instructions that are executed in the script. If there is a signature in the file when this cmdlet runs, that signature is removed.

## PARAMETERS

`-Certificate <System.Security.Cryptography.X509Certificates.X509Certificate2>`

Specifies the certificate that will be used to sign the script or file. Enter a variable that stores an object representing the certificate or an expression that gets the certificate.

To find a certificate, use ``Get-PfxCertificate`` or use the ``Get-ChildItem`` cmdlet in the Certificate ``Cert:`` drive. If the certificate is not valid or does not have ``code-signing`` authority, the command fails.

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

`-Content <System.Byte[]>`

This parameter appears in the syntax listing because it is defined in the base class that ``Set-AuthenticodeSignature`` is derived from. However, support for this parameter is not implemented in ``Set-AuthenticodeSignature``.

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

**-FilePath <System.String[]>**

Specifies the path to a file that is being signed.

Required? true  
Position? 1  
Default value None  
Accept pipeline input? True (ByPropertyName, ByValue)  
Accept wildcard characters? false

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

Allows the cmdlet to append a signature to a read-only file. Even using the Force parameter, the cmdlet cannot override security restrictions.

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

**-HashAlgorithm <System.String>**

Specifies the hashing algorithm that Windows uses to compute the digital signature for the file.

The default is SHA1. Files that are signed with a different hashing algorithm might not be recognized on other systems. Which algorithms are supported depends on the version of the operating system.

For a list of possible values, see `HashAlgorithmName` Struct ([/dotnet/api/system.security.cryptography.hashalgorithmname?view=netframework-4.7.2#properties](https://dotnet/api/system.security.cryptography.hashalgorithmname?view=netframework-4.7.2#properties)).

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

#### `-IncludeChain <System.String>`

Determines which certificates in the certificate trust chain are included in the digital signature. `NotRoot` is the default.

Valid values are:

- `Signer`: Includes only the signer's certificate.
- `NotRoot`: Includes all of the certificates in the certificate chain, except for the root authority.
- `All`: Includes all the certificates in the certificate chain.

Required?	false
Position?	named
Default value	<code>NotRoot</code>
Accept pipeline input?	False
Accept wildcard characters?	false

#### `-LiteralPath <System.String[]>`

Specifies the path to a file that is being signed. Unlike `FilePath`, the value of the `LiteralPath` parameter is used exactly as it is 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?	True (ByPropertyName)
Accept wildcard characters?	false

**-SourcePathOrExtension** <System.String[]>

This parameter appears in the syntax listing because it is defined in the base class that ``Set-AuthenticodeSignature`` is derived from. However, support for this parameter is not implemented in ``Set-AuthenticodeSignature``.

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

**-TimestampServer** <System.String>

Uses the specified time stamp server to add a time stamp to the signature. Type the URL of the time stamp server as a string.

The time stamp represents the exact time that the certificate was added to the file. A time stamp prevents the script from failing if the certificate expires because users and programs can verify that the certificate was valid at the time of signing.

Required?	false
Position?	named
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 is not 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.String

You can pipe a string that contains the file path to this cmdlet.

## OUTPUTS

System.Management.Automation.Signature

This cmdlet returns a Signature object representing the value it set.

### Example 1 - Sign a script using a certificate from the local certificate store

```
$cert=Get-ChildItem -Path Cert:\CurrentUser\My -CodeSigningCert  
$signingParameters = @{  
    FilePath      = 'PsTestInternet2.ps1'  
    Certificate    = $cert  
    HashAlgorithm = 'SHA256'  
}  
Set-AuthenticodeSignature @signingParameters
```

The first command uses the `Get-ChildItem` cmdlet and the PowerShell certificate provider to get the certificates in the `Cert:\CurrentUser\My` subdirectory of the certificate store. The `Cert:` drive is the drive exposed by the certificate provider. The `CodeSigningCert` parameter, which is supported only by the certificate provider, limits the certificates retrieved to those with code-signing authority. The command stores the result in the `$cert` variable.

The second command defines the `$signingParameters` variable as a HashTable with the parameters for the `Set-AuthenticodeSignature` cmdlet to sign the `PsTestInternet2.ps1` script. It uses the `FilePath` parameter to specify the name of the script, the `Certificate` parameter to specify that the certificate is stored in the `$cert` variable, and the `HashAlgorithm` parameter to set the hashing algorithm to SHA256.

The third command signs the script by splatting the parameters defined in `$signingParameters`.

> [!NOTE] > Using the `CodeSigningCert` parameter with `Get-ChildItem` only returns certificates that have > code-signing authority and contain a private key. If there is no private key, the certificates > cannot be used for signing.

### Example 2 - Sign a script using a certificate from a PFX file

```
$cert = Get-PfxCertificate -FilePath C:\Test\Mysign.pfx
```

```
$signingParameters = @{
```

```
    FilePath      = 'ServerProps.ps1'
```

```
    Certificate    = $cert
```

```
    HashAlgorithm = 'SHA256'
```

```
}
```

```
Set-AuthenticodeSignature @signingParameters
```

The first command uses the `Get-PfxCertificate` cmdlet to load the `C:\Test\MySign.pfx` certificate into the `$cert` variable.

The second command defines the `$signingParameters` variable as a HashTable with the parameters for the `Set-AuthenticodeSignature` cmdlet to sign the

`ServerProps.ps1` script. It uses the `FilePath` parameter to specify the name of the script, the `Certificate` parameter to specify that the certificate is stored in the

`$cert` variable, and the `HashAlgorithm` parameter to set the hashing algorithm to SHA256.

The third command signs the script by splatting the parameters defined in `$signingParameters`.

If the certificate file is password protected, PowerShell prompts you for the password.

- Example 3 - Add a signature that includes the root authority -

```
$signingParameters = @{
```

```
    FilePath      = 'C:\scripts\Remodel.ps1'
```

```
    Certificate    = $cert
```

```
    HashAlgorithm = 'SHA256'
```

```
    IncludeChain  = 'All'
```

```
    TimestampServer = 'http://timestamp.fabrikam.com/scripts/timestamp.dll'
```

```
}
```

```
Set-AuthenticodeSignature @signingParameters
```

The first command defines the `$signingParameters` variable as a HashTable with the parameters for the `Set-AuthenticodeSignature` cmdlet to sign the script. It uses



the `FilePath` parameter to specify the path to the script, the `Certificate` parameter to specify that the certificate is stored in the ``$cert`` variable, and the

`HashAlgorithm` parameter to set the hashing algorithm to SHA256. It uses the `IncludeChain` parameter to include all of the signatures in the trust chain, including the

root authority. It also uses the `TimeStampServer` parameter to add a timestamp to the signature. This prevents the script from failing when the certificate expires.

The second command signs the script by splatting the parameters defined in ``$signingParameters``.

## RELATED LINKS

Online

Version:

<https://learn.microsoft.com/powershell/module/microsoft.powershell.security/set-authenticodesignature?view=powershell-5.1>

[&WT.mc\\_id=ps-gethelp](#)

[Get-AuthenticodeSignature](#)

[Get-ExecutionPolicy](#)

[Get-PfxCertificate](#)

[Set-ExecutionPolicy](#)

[about\\_Execution\\_Policies](#)

[about\\_Signing](#)