



PowerShell HelpWriter - Help Manual

© 2020 by SAPIEN Technologies Inc., all rights reserved

1. Welcome to PowerShell HelpWriter	3
2. Introduction	4
About PowerShell HelpWriter	4
How to Buy PowerShell HelpWriter	10
3. Getting Started	12
Installing PowerShell HelpWriter	12
Staying Up-to-date	17
4. Basic Orientation	18
Customizing Your Workspace	18
Create a Help File for a Module	21
Review the Starter Help	23
Write the Help Content	25
Edit a Help File	30
Test in Get-Help	33
Ship your Help File	34
5. Using PowerShell HelpWriter	35
Create New Help Topic for a Module	35
Create an Empty Help Topic	37
Write Help for Parameters	43
Writing Help Content for Parameters	46
Write Examples in Help	47
Writing Help Content for Examples	50
Write Help for Inputs	53
Writing Help Content for an Input Type	56
Write Help for Outputs	58
Writing Help Content for an Output Type	60
Write Notes in Help	61
Writing Note Content	63
Add Related Links	64
Writing Help for a Related Link	66
Edit Help Topics as XML	67
Switching Between the Designer and Editor	67
Opening XML Files in the XML Editor	68
Formatting XML Files	69

Using Empty or Closing Tags 71

6. Getting Help 73

7. SAPIEN Updates 74

8. Appendices 78

 Appendix A: Manual Version 78

 Appendix B: Icon License Attribution 79

1 Welcome to PowerShell HelpWriter

Writing and editing Windows PowerShell help files has never been easier!



PowerShell HelpWriter is the premier editor for creating and editing help files for all command types, including cmdlets, functions, workflows, and CIM commands. Focus on your content and let PowerShell HelpWriter handle the XML!

For a complete list of current features, visit the [PowerShell HelpWriter product page](#).

About this documentation

This help is designed to show you how to use PowerShell HelpWriter—you can do a quick overview to get started, work through the topics in detail, and refer back to this guide for additional information when needed.

Getting started - new users

- [Download and install PowerShell HelpWriter](#).
- Quickly learn the basics of the program in [Getting Started](#) ¹².
- Visit the [support forum](#) to get help from SAPIEN staff and other experienced PowerShell HelpWriter users.

2 Introduction

This section provides an overview of the PowerShell HelpWriter features, lets you know how to get answers to your questions, and shows you how to purchase directly online or through a reseller.

2.1 About PowerShell HelpWriter

PowerShell HelpWriter is the first professionally designed environment for writing and editing Windows PowerShell help files. This section provides an overview of the features and benefits of PowerShell HelpWriter.

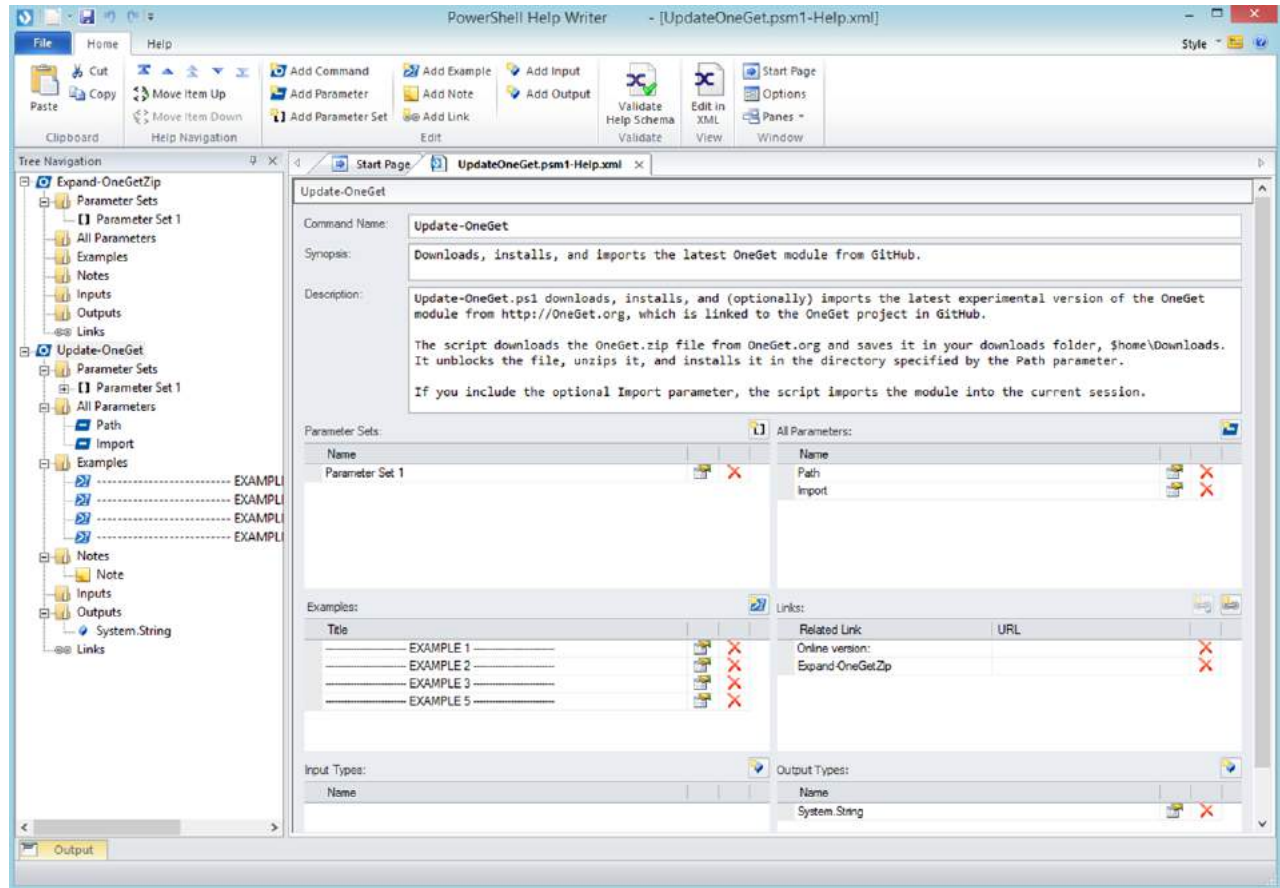
What is PowerShell HelpWriter?

PowerShell HelpWriter is the premier editor for Windows PowerShell XML help files.

- Create help files quickly and easily for a module or from scratch, or edit existing help files.
- The PowerShell HelpWriter Designer provides a studio environment for writing and editing help with the resource you need close at hand, including syntax displays, a built-in XML editor, and quick navigation.
- PowerShell HelpWriter can generate help for all commands in a modules. The help files match the code, including cmdlet attributes, parameters and parameter attributes, inputs and outputs, and even an optional online help link. And, your files can include any help you have already written, including comment-based help, so no effort is ever wasted.
- PowerShell HelpWriter works with the simplest and most complex modules, including modules with nested modules, and names them correctly so Get-Help can find them. It works on all supported command types—cmdlets, functions, workflows, and CIM commands.
- Your help files are ready to go. You can ship the help files in your module, use them in Updatable Help, and share them in an open source project with no post-processing.

Work in a Help Designer

Tired of typing XML in Notepad or settling for comment-based help? Welcome to the PowerShell HelpWriter Designer, a professional studio for writing and editing help topics. You don't need any expertise in XML and you don't need to learn the complex structure of the Windows PowerShell Help XML schema.

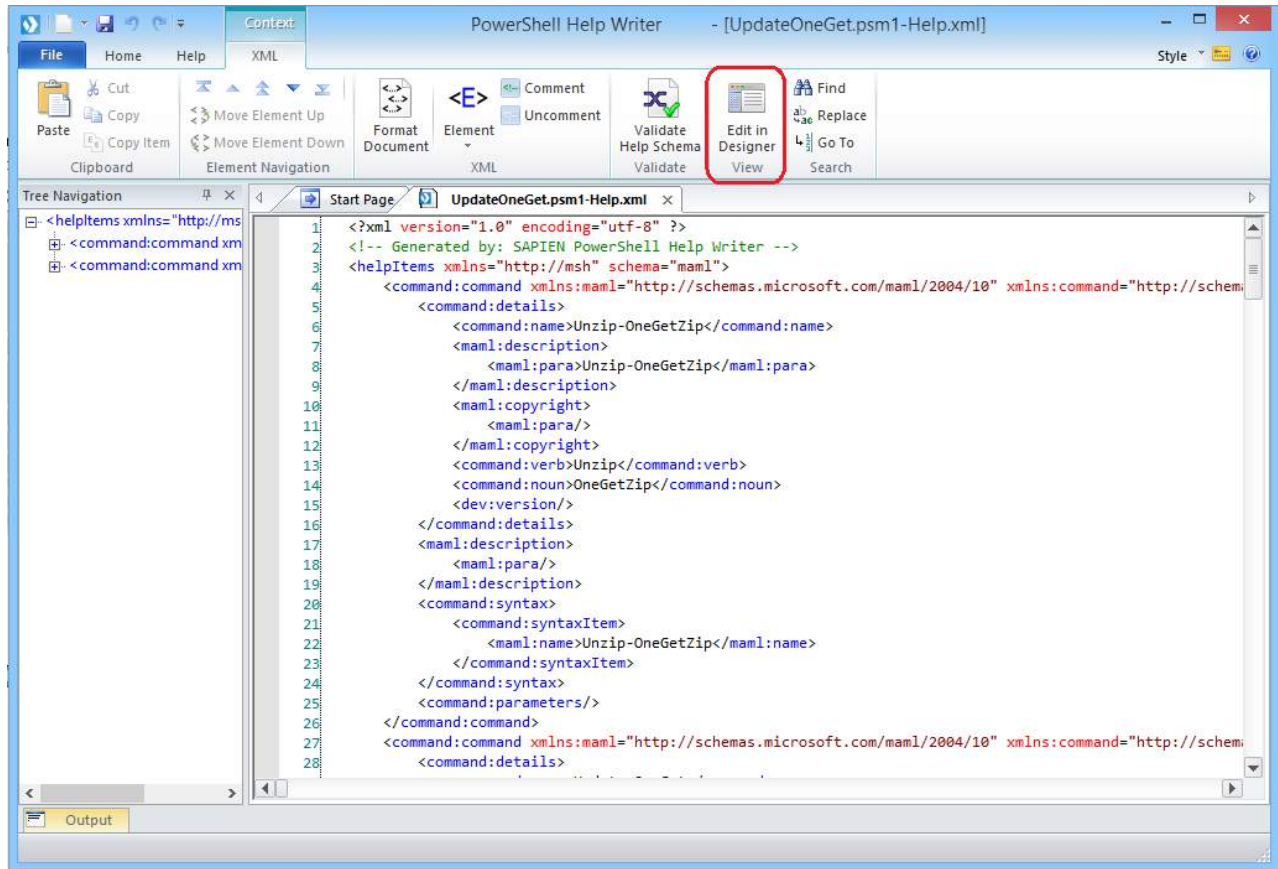


Edit in XML

Whenever you need it, you can view or edit the help file in the built-in XML editor. To view the underlying XML, click the **Edit in XML** button. The PowerShell HelpWriter XML editor is a complete XML editor—not just a read-only viewer.

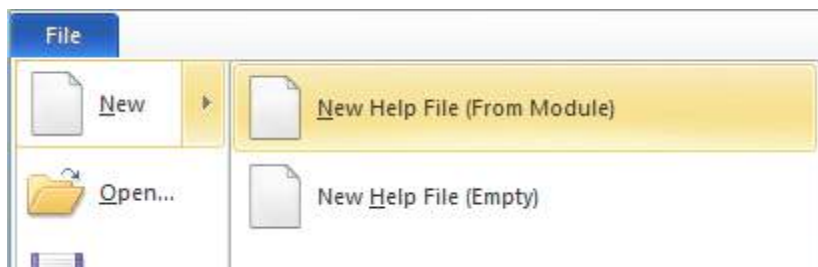
Want to switch back? Click **Edit in Designer**.

For more information, see [Edit Help Topics as XML](#) ⁶⁷

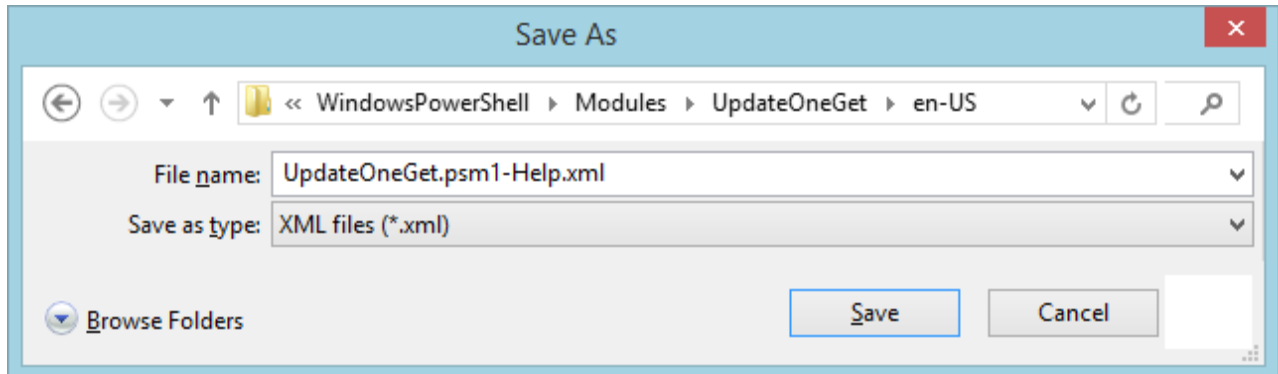


Generate starter files for any module

No need to start from scratch. PowerShell HelpWriter analyses your module and generates starter help files that match the cmdlet code.



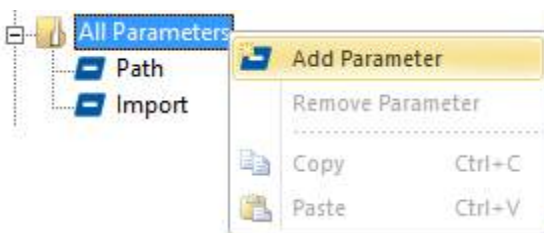
It implements the naming rules that allow the Get-Help cmdlet to find your help file.



The starter files include cmdlet attributes, parameters and parameter attributes, input and output types, and even an online help link for commands that support online help, so you can focus on writing descriptions.

A screenshot of a configuration window titled 'Update-OneGet > Path'. It contains several input fields and checkboxes. 'Parameter Name' is 'Path'. 'Description' is an empty text area. 'Parameter Type' is 'String'. 'Default Value' is an empty text field. 'Aliases' is 'PSPATH'. 'Position' is '0' with a dropdown arrow. 'Pipeline Input' is 'false' with a dropdown arrow. At the bottom, there are three checkboxes: 'Required' (checked), 'Accepts wildcard characters' (unchecked), and 'Dynamic' (unchecked).

👍 If the code changes, you can add and remove parameters from your help topic.



Focus on examples

The PowerShell HelpWriter designer features an environment especially designed for writing examples with an editable prompt and monospace font for formatting output.

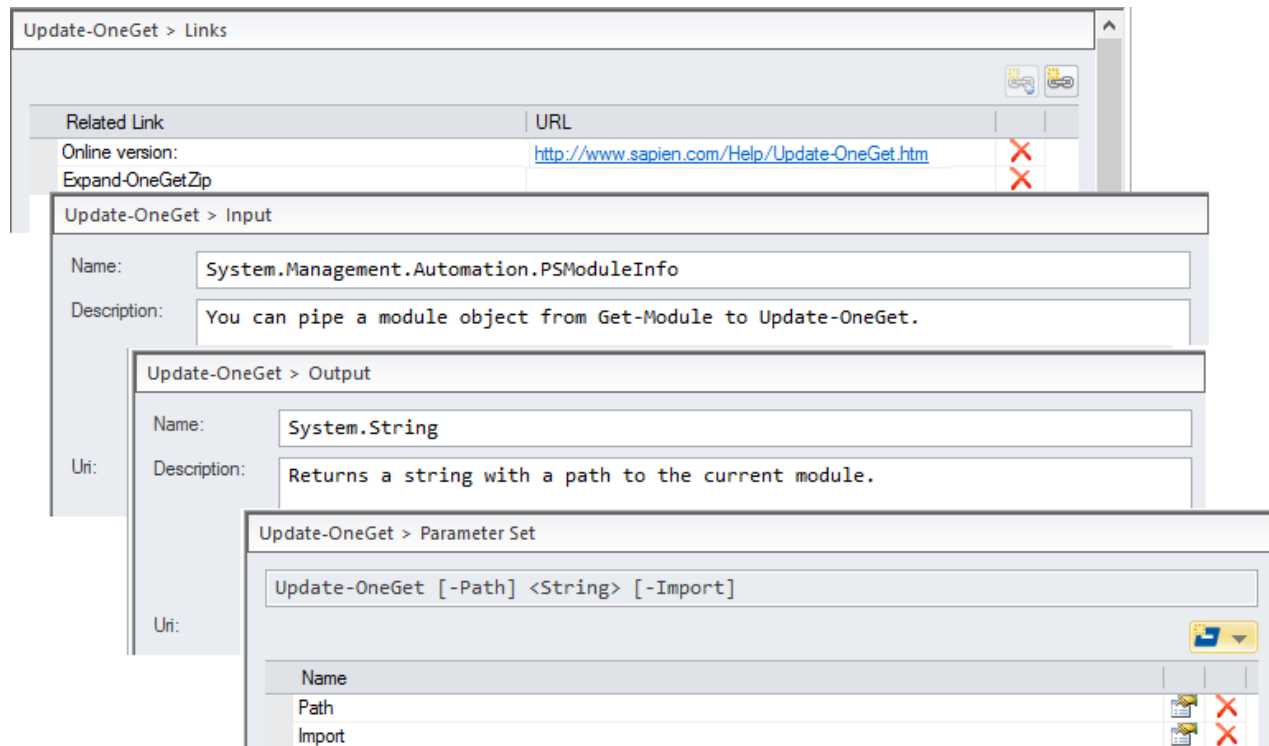
For more information, see [Write Examples in Help](#)^[47].

Update-OneGet > ----- EXAMPLE 1 -----	
Title:	----- EXAMPLE 1 -----
Introduction:	[ADMIN] C:\PS>
Command:	<pre>.\Update-OneGet.ps1 -Path \$home\Documents\Test\OneGet \$home\Documents\Test\OneGet \$home\Documents\Test\OneGet.ps1 \$home\Documents\Test\OneGet.psd1</pre>
Remarks:	<p>This command installs the newest OneGet module in the \$home\Documents\Test\OneGet directory. To import it: Import-Module \$home\Documents\Test\OneGet.psd1</p>

Get the details right

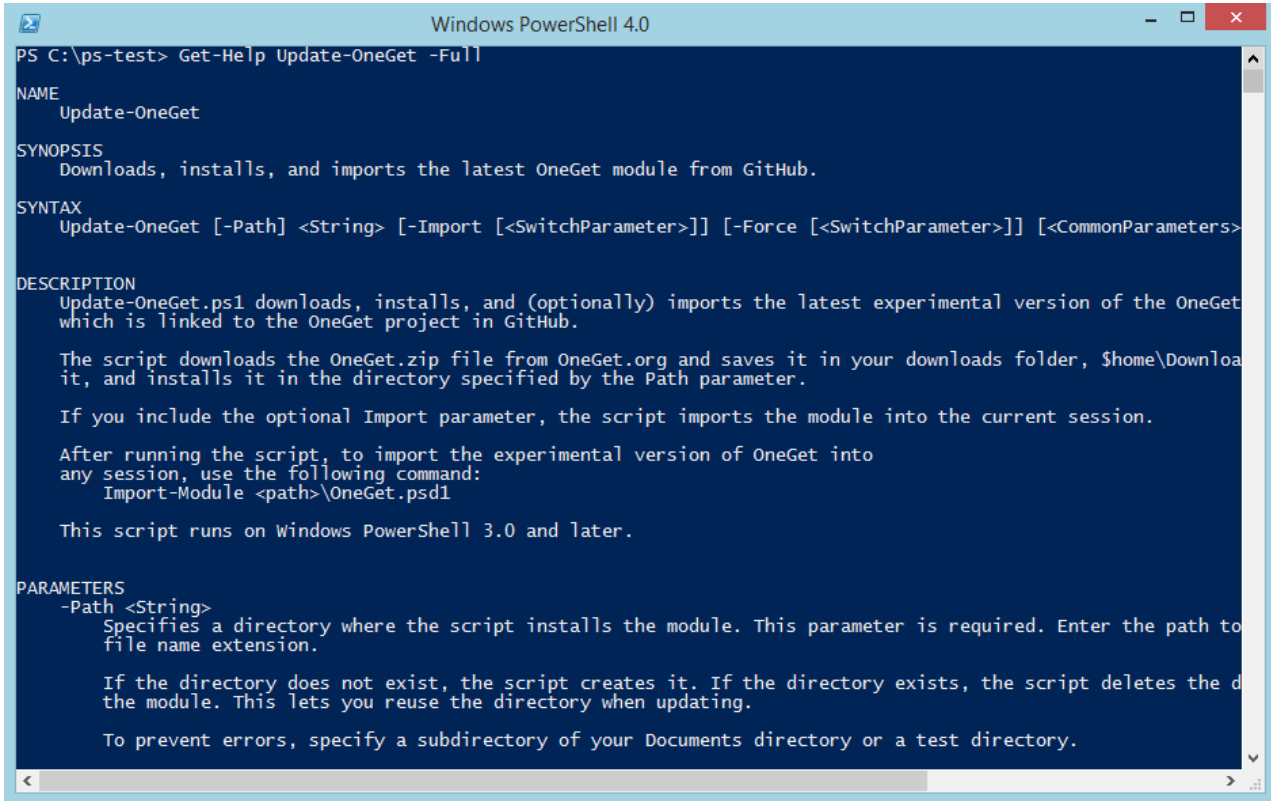
PowerShell HelpWriter makes it easy to include even detailed information for advanced readers.

For more information, see [Write Help for Inputs](#)^[53], [Write Help for Outputs](#)^[58], [Write Notes in Help](#)^[61], and [Add Related Links](#)^[64].



Test in Get-Help

You can test your help files as you work, validate them, and run them in Windows PowerShell.



```
Windows PowerShell 4.0
PS C:\ps-test> Get-Help Update-OneGet -Full

NAME
    Update-OneGet

SYNOPSIS
    Downloads, installs, and imports the latest OneGet module from GitHub.

SYNTAX
    Update-OneGet [-Path] <String> [-Import [<SwitchParameter>]] [-Force [<SwitchParameter>]] [<CommonParameters>]

DESCRIPTION
    Update-OneGet.ps1 downloads, installs, and (optionally) imports the latest experimental version of the OneGet
    which is linked to the OneGet project in GitHub.

    The script downloads the OneGet.zip file from OneGet.org and saves it in your downloads folder, $home\Downloa
    it, and installs it in the directory specified by the Path parameter.

    If you include the optional Import parameter, the script imports the module into the current session.

    After running the script, to import the experimental version of OneGet into
    any session, use the following command:
        Import-Module <path>\OneGet.psd1

    This script runs on Windows PowerShell 3.0 and later.

PARAMETERS
    -Path <String>
        Specifies a directory where the script installs the module. This parameter is required. Enter the path to
        file name extension.

        If the directory does not exist, the script creates it. If the directory exists, the script deletes the d
        the module. This lets you reuse the directory when updating.

        To prevent errors, specify a subdirectory of your Documents directory or a test directory.
```

2.2 How to Buy PowerShell HelpWriter

You can buy PowerShell HelpWriter online with all major credit cards. As soon as your transaction completes you can download and install the program.

For answers to your pre-order questions, check out the [SAPIEN Frequently Asked Questions](#) or post in the [Trial Software / Pre-sales Technical Questions](#) forum.

Order link and PowerShell HelpWriter product page

Online orders:

<https://www.sapien.com/store/powershell-helpwriter>

Worldwide authorized resellers:

<https://www.sapien.com/company/resellers>

PowerShell HelpWriter product page:

https://www.sapien.com/software/powershell_helpwriter

3 Getting Started

This section shows you how to install PowerShell HelpWriter and keep it up-to-date.


3.1 Installing PowerShell HelpWriter

To get started using PowerShell HelpWriter, follow the instructions below to download and install the program.

Downloading PowerShell HelpWriter

All SAPIEN Technologies software products are downloadable only. Download registered products from your [SAPIEN Account Registered Products page](#).

Select the 64-bit version of PowerShell HelpWriter to download. The installer software will save to your default download folder (e.g., *PHW20Setup_2.3.46_010320_x64*).

 Starting with the PowerShell HelpWriter 2020 product release, 32-bit versions are no longer available. Current owners of a license that includes a 32-bit product will have access to that from their [SAPIEN Account Registered Products page](#).

Want to try before you buy? You can [download a trial version here](#).

Installing PowerShell HelpWriter

Follow these instructions to install PowerShell HelpWriter.

How to install PowerShell HelpWriter

1. In your default download folder, double-click on the downloaded program (e.g., *PHW20Setup_2.3.46_010320_x64*).
2. Reply **Yes** to the *"Do you want to allow this app to make changes to your device?"* prompt.

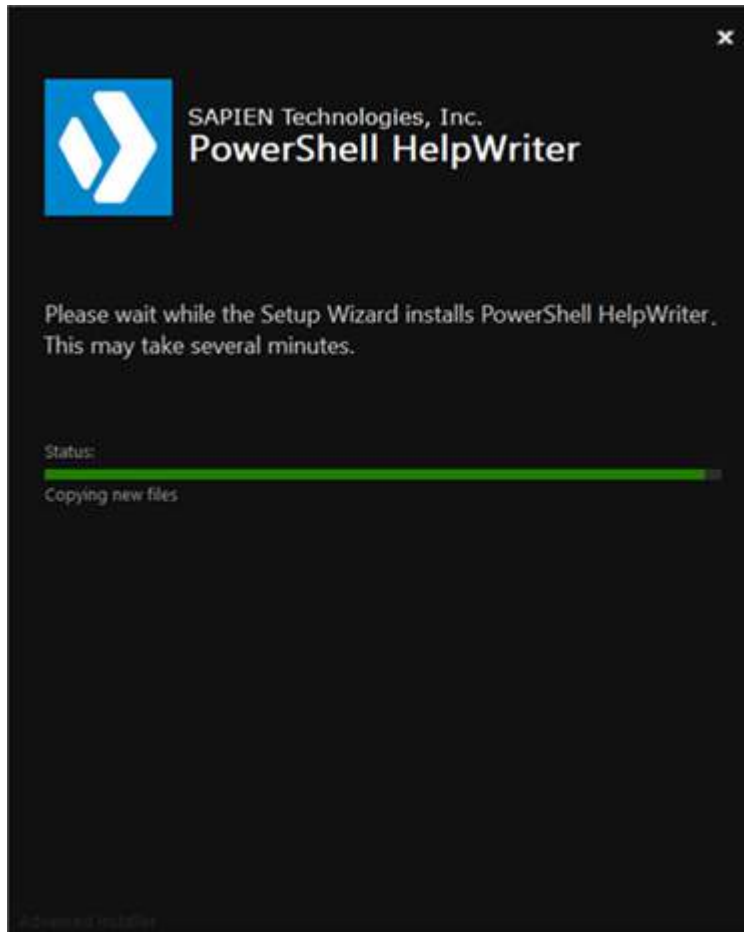
The installation wizard will first check several items, such as available disk space and the presence of previous builds. If the environment is adequate, the installer will display the legal agreement which you must accept to proceed:

- a. **Read** the terms of the license agreement.
- b. **Accept** the terms of the license agreement. You should never accept license terms unless you have read them, and you understand them.
- c. Once you have accepted the terms, click **Install**.

 The software will install in the default location as shown, unless you change the path.



3. The installation may take several minutes.



4. When PowerShell HelpWriter successfully completes the installation, click **Finish**.



Troubleshooting Installation

If you encounter problems installing PowerShell HelpWriter, please report them in the [Installation Issues support forum](#).

Activating and Deactivating PowerShell HelpWriter

Product Activation

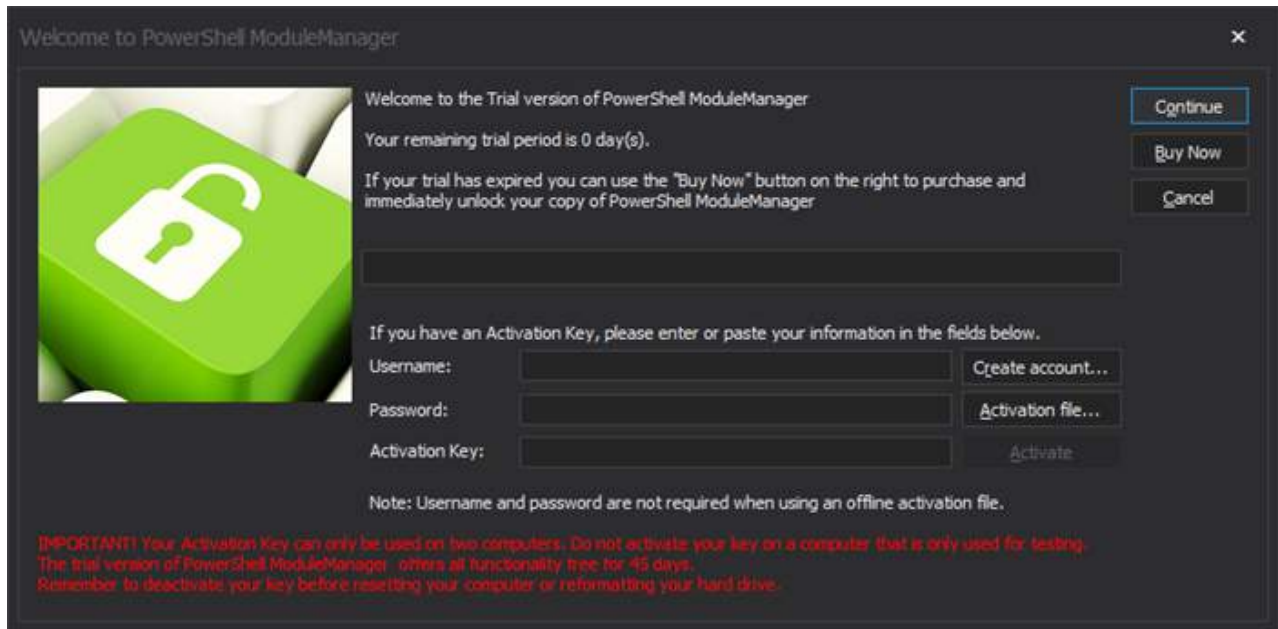
Registration is required to activate and operate the product, and also to obtain any customer service or technical support benefits. Registration only takes a few moments to complete and provides you with access to special offers including preferred pricing on renewals. *You will need an active internet connection to complete product registration.*

An active internet connection may not be required if you have a legitimate reason for needing [offline access](#). To request offline activation [please fill out this request](#). All requests are considered on a case-

by-case basis.

To activate PowerShell HelpWriter

The first time you launch a SAPIEN product, the Welcome screen is displayed.



The steps to activate the product vary depending on whether or not you already have a SAPIEN account.

👍 Follow the steps in the [Quick Guide to SAPIEN Software Activation](#) to activate the software.

If you are unable to activate the product, contact sales@sapien.com.

Product Deactivation

As outlined in our [End-User License Agreement](#), each single-user activation key is entitled to a maximum of two devices to be activated and operating at any given time. You may deactivate your devices to free up your activations at your own leisure, but there are also certain circumstances where proper deactivation is crucial in order to prevent the loss of your allotted two activations.

❗ Uninstalling the software from your device does **not** deactivate the activation key.

To deactivate your activation key

In the top-right of PowerShell HelpWriter above the ribbon, click the gold certificate button.



The Activation Information window will open.

👍 Follow the steps in the [SAPIEN Software Activation / Deactivation FAQ](#) to deactivate your activation key.

3.2 Staying Up-to-date

We are continually updating PowerShell HelpWriter, both to remove bugs and to add and improve product features. We recommend always staying current with the most recent version to ensure that you are taking advantage of the latest features, functionality, and product stability.

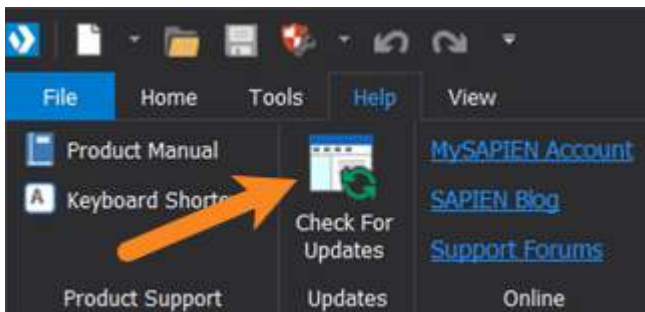
i The details for every PowerShell HelpWriter release are available in the [version history](#).

Check for Updates

By default, PowerShell HelpWriter will automatically check for software updates. You can also manually check for updates.

To check for updates

- On the **Help** ribbon (Updates section) > click **Check For Updates** to open the [SAPIEN Updates](#) ⁷⁴ tool and see if there is a new PowerShell HelpWriter build available:



4 Basic Orientation

This section shows you how to easily customize your workspace, and provides a basic orientation to PowerShell HelpWriter by walking through the steps to create a help file from a module. The topics are presented in the order that you would perform the tasks, but you can easily jump to any topic and work your way through..

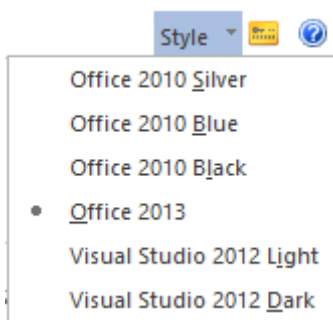
4.1 Customizing Your Workspace

You can adjust the appearance and behavior of PowerShell HelpWriter to meet your needs.

Change the Color Style

To change the color themed style

- In the upper right corner, click **Style** and then select a **style name**.



Your changes are saved automatically and persist after closing.

Show or Hide the Start Page

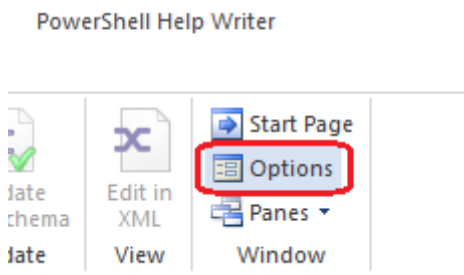
The PowerShell HelpWriter Start page opens when you start PowerShell HelpWriter. It lists recently used files, references for help, and topics of interest. If you close it, it remains closed until you start PowerShell HelpWriter again.

The **Show start page on startup** option (on the Start Page) determines whether the start page opens whenever you start PowerShell HelpWriter.

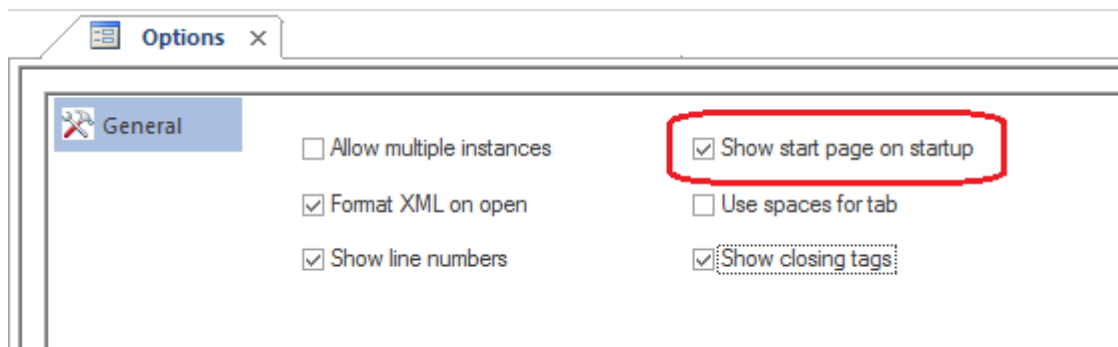
To Show / Hide the Start Page

To prevent the Start Page from opening when you start PowerShell HelpWriter:

1. On the Home tab > in the Windows section, click **Options**.



2. Click to clear the **Show start page on startup** option.



Your changes are saved automatically when you click outside of the Options window.

👍 To restore the Start page, select the **Show start page on startup** check box.

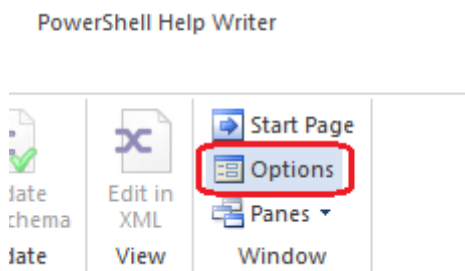
Change the Font

You can change the font, style, or size of the text in the Designer input fields and XML Editor. A single setting determines the fonts for both displays.

The default font is **Consolas, 10 pt., Regular**

To change the font

1. On the Home tab > in the Windows section, click **Options**.

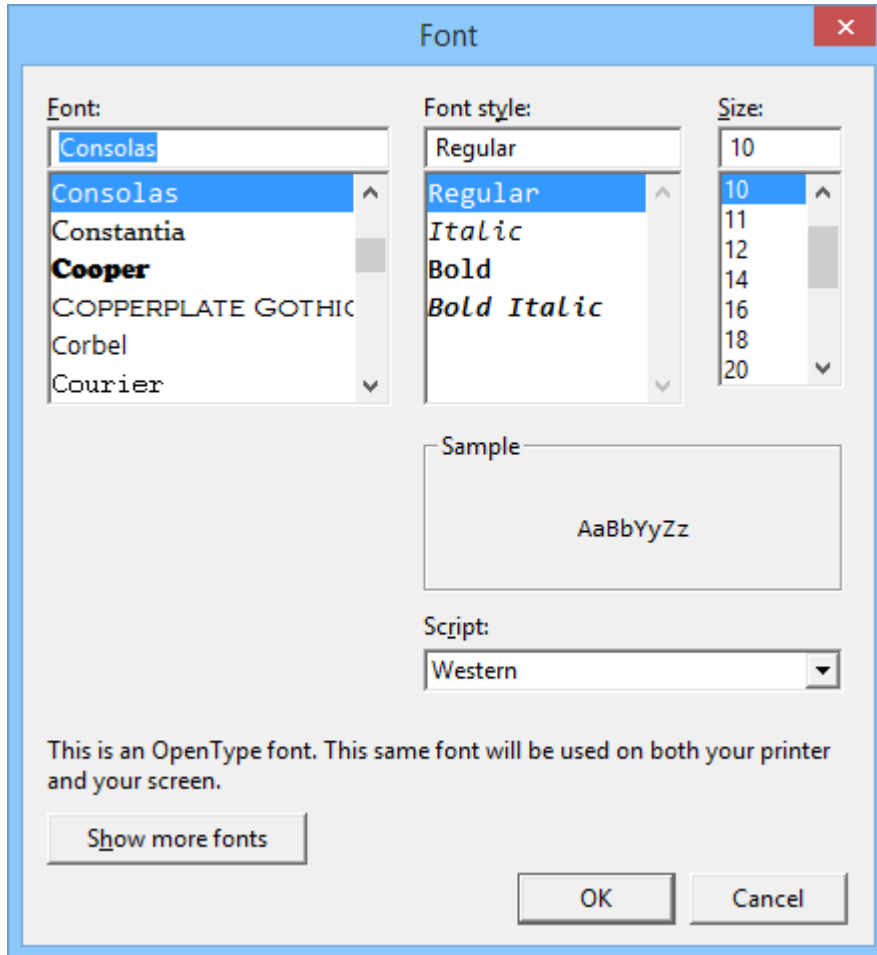


2. On the Options page > in the Default Font section, click the **ellipsis (...)**.

ℹ Editing the value in the text box doesn't change the font—it's just a display.



3. Adjust the settings in the Font dialog box.



Change the Default Directory

Typically, you save your help files where the **Get-Help** cmdlet looks for them, that is, in a language-specific subdirectory of the module directory, such as `$home\Documents\Modules\<moduleName>\en-US`. This location lets you use Get-Help to test your help file as you write it. However, you might decide to keep your draft help files in a separate directory.

When you save each help file for the first time, PowerShell HelpWriter suggests the default directory. Thereafter, it suggests the directory most recently used to save that help file.

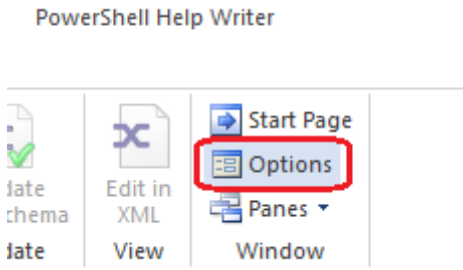
The default location is:

`$home\Documents\SAPIEN\Help Files`

`(%UserProfile%\Documents\SAPIEN\Help Files).`

To change the default directory

1. On the Home tab > in the Windows section, click **Options**.



2. In the Default Directory section > type a new path or click the ellipsis (...) and navigate to the new path.

Default Directory:

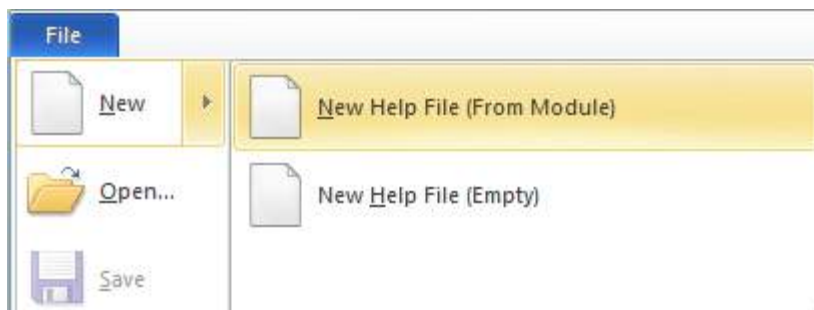
3. Restart PowerShell HelpWriter to make the change effective.

4.2 Create a Help File for a Module

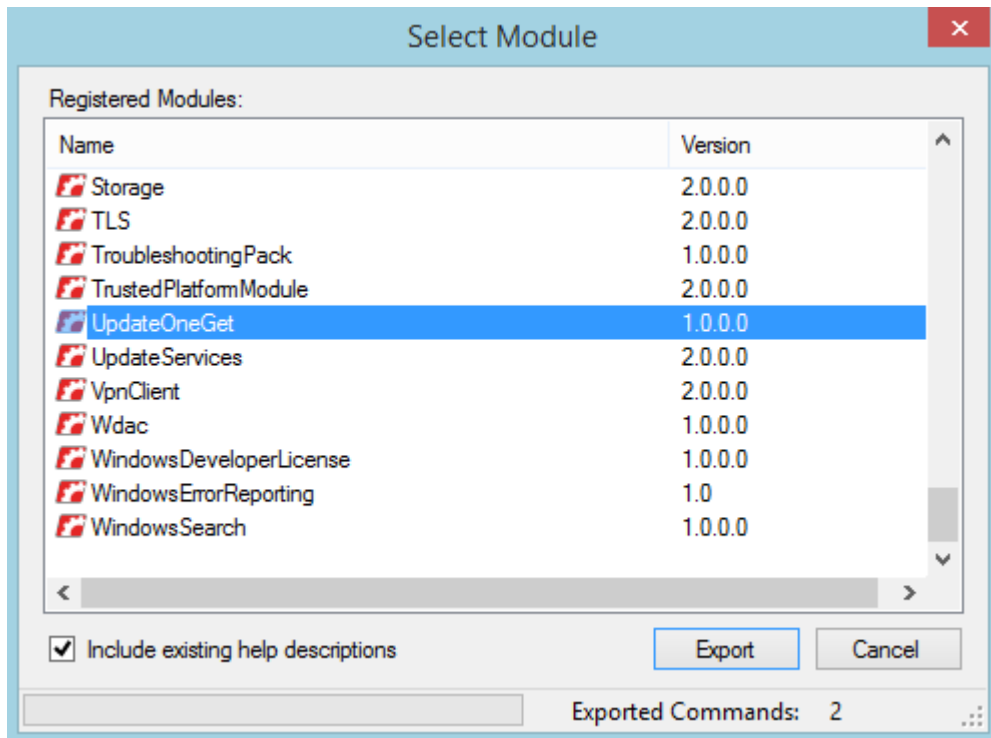
This is the first step in a set of instructions designed to help you quickly get started using PowerShell HelpWriter by demonstrating how to create a new help file from a module. The steps to review, write, edit, test, and ship the file are included.

To create a new help file from a module

1. Click **File > New > New Help File (From Module)** (*Ctrl+N*):

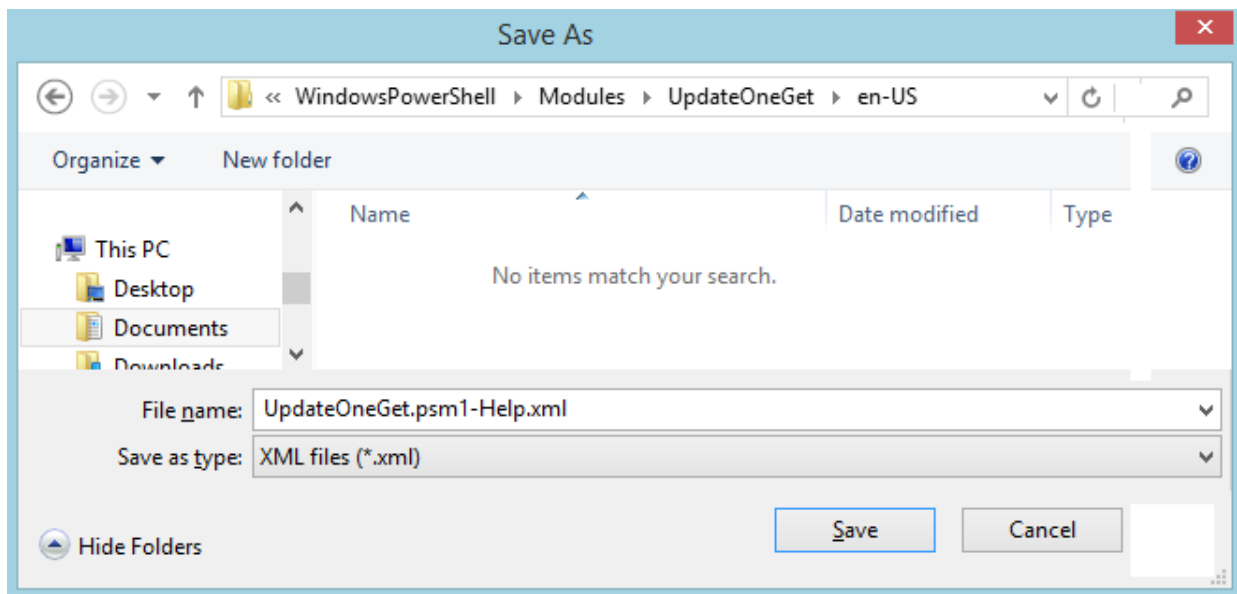


2. Select a module:



The Select Module window lists all modules in all paths in your PSModulePath environment variable. It shows how many commands the module exports and lets you include or exclude existing help sources, including comment-based help.

3. Save the help file for the module:



By default, PowerShell HelpWriter saves your files in the most recently used location.

👍 For easy testing with the Get-Help cmdlet, save the help file where Get-Help looks for it; that is, in a language-specific subdirectory of the module directory.

To change the default file location

1. On the **Home** ribbon > in the Window section > click **Options**.
2. In the Directories section, **click the ellipsis (...)**, then navigate to the desired location and click **Select Folder**.

If your module has nested modules, PowerShell HelpWriter creates a separate help file for the cmdlets that are defined in each nested module and saves them in the same location.

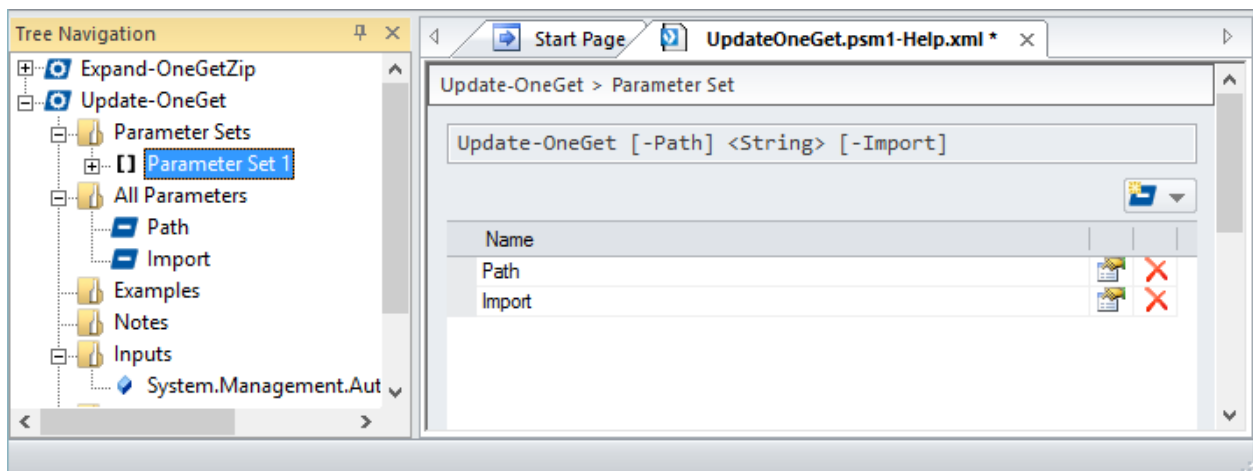
i To use an XML help file for a Windows PowerShell function, the function must have an ExternalHelp comment keyword. For more information, see [about Comment Based Help](#).

4.2.1 Review the Starter Help

After you [create a new help file from a module](#)^[21], you can open the new starter help file and review the fields in the designer.

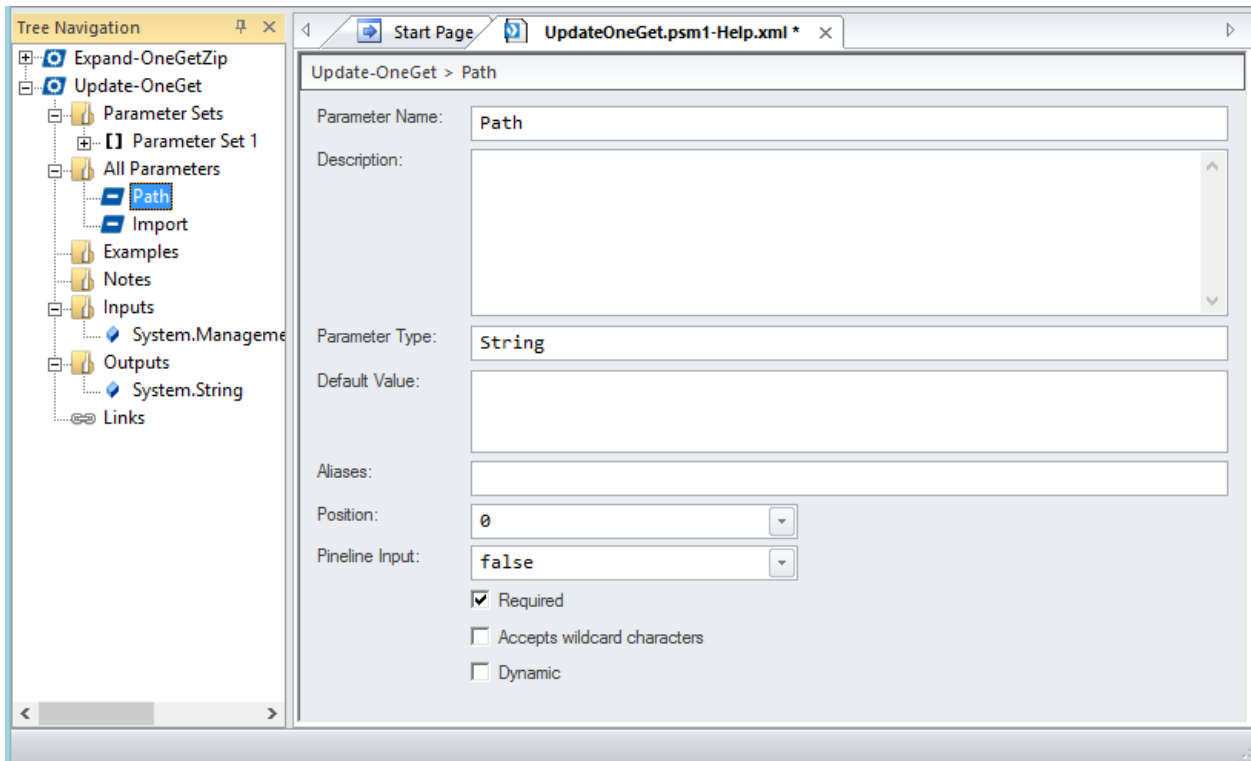
To see the command syntax

- In the tree navigation, click a parameter set. There will be one parameter set element for each parameter set in the cmdlet.
- i** The parameter nodes under each parameter set determines the contents of the syntax diagram. The attribute values are specific to each parameter set, but the descriptive fields are shared by all instances of the parameter in the help topic.



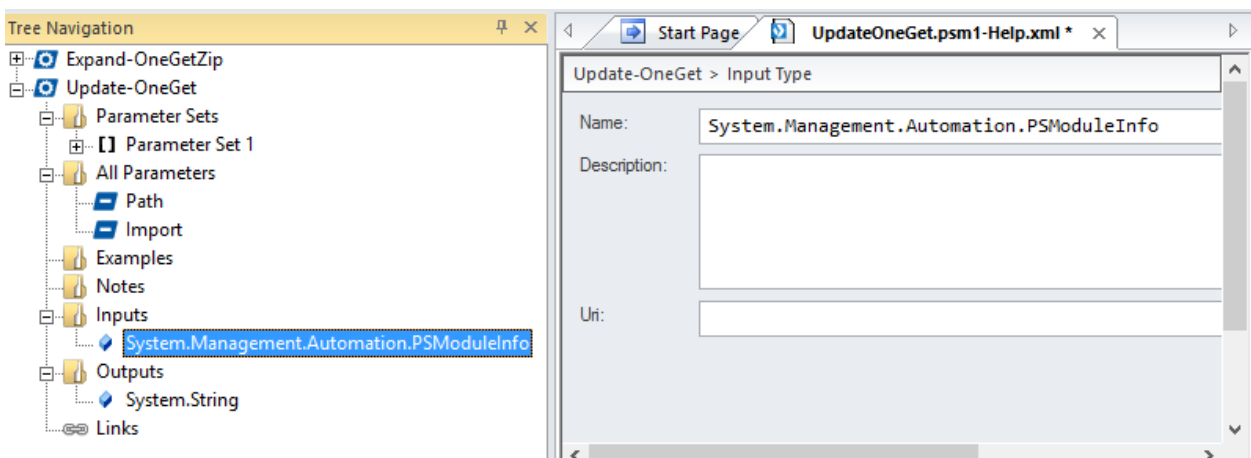
To see the parameter attributes

- In the All Parameters section of the tree navigation, click the parameter name.



To view the Inputs, Outputs, and Links

- Using the tree navigation, click the Inputs, Outputs, or Links node.



To view the XML for the file

- Right-click the tab for the file and click **Edit in XML**.

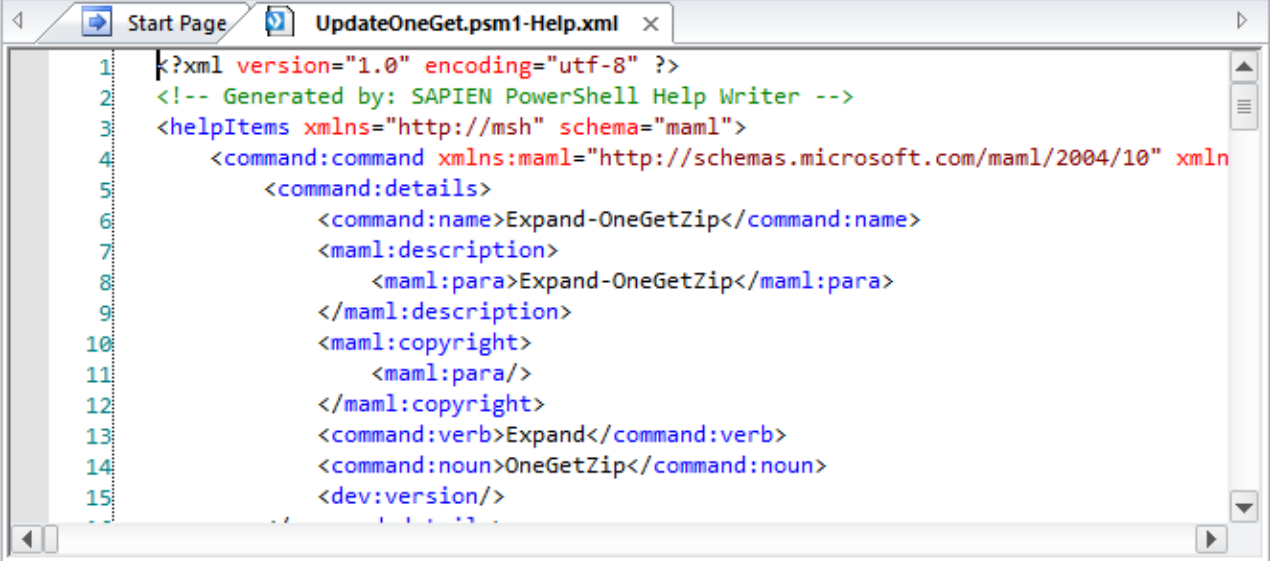
-OR-

- On the ribbon, click **Edit in XML**.

PowerShell HelpWriter saves any changes that you have made and opens the file in the XML editor.

👍 To return to Designer View, click **Edit in Designer**.

You can use the PowerShell HelpWriter XML Editor to view or edit any XML or text file. If you open a file that is not a help file, it opens in XML view automatically.



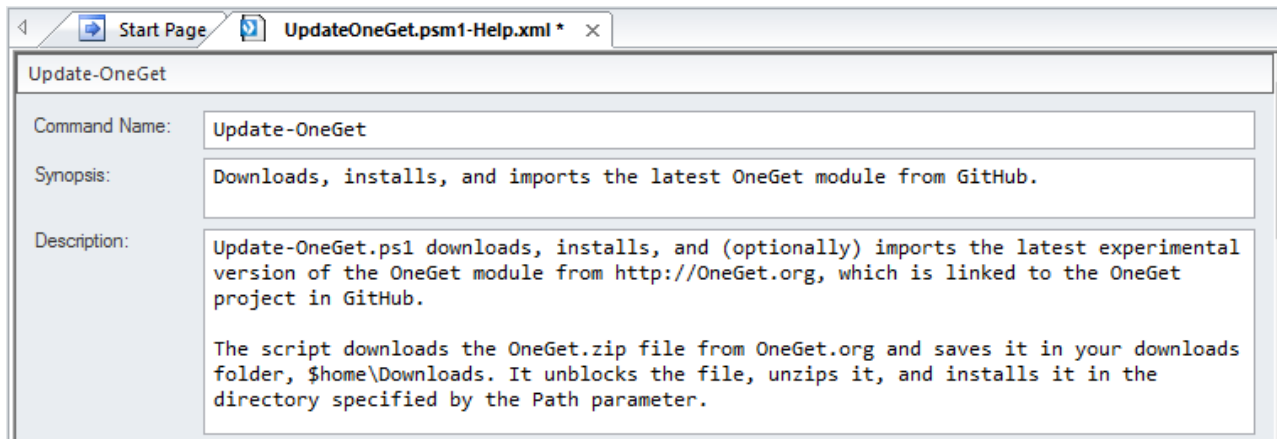
```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <!-- Generated by: SAPIEN PowerShell Help Writer -->
3 <helpItems xmlns="http://msh" schema="maml">
4   <command:command xmlns:maml="http://schemas.microsoft.com/maml/2004/10" xmlns
5     <command:details>
6       <command:name>Expand-OneGetZip</command:name>
7       <maml:description>
8         <maml:para>Expand-OneGetZip</maml:para>
9       </maml:description>
10      <maml:copyright>
11        <maml:para/>
12      </maml:copyright>
13      <command:verb>Expand</command:verb>
14      <command:noun>OneGetZip</command:noun>
15      <dev:version/>
```

4.2.2 Write the Help Content

Now it's time to write the help content. You can complete the fields in any order.

Draft the synopsis and description

Typically, you begin by drafting the synopsis and description. Type the content as you want it to appear in the Get-Help display. To create a paragraph break, just type it. Avoid tab characters. Instead, use spaces to indent paragraphs.



Draft the parameter descriptions

Next, draft the parameter descriptions.

To edit a parameter description

- In the tree navigation > in the All Parameters section, click the **parameter name**.

-OR-

- In the Designer > on the parameter row, click the **Edit icon**.

PowerShell HelpWriter pre-populates the parameter attributes from the code in the module, but you can edit any attribute value. For more information about documenting cmdlet parameters, including composing the descriptions, see [Write Help for Parameters](#)^[43].

Update-OneGet > Path

Parameter Name:

Description:
 Specifies a directory where the script installs the module. This parameter is required. Enter the path to a directory. Do not include a .zip file name extension.
 If the directory does not exist, the script creates it. If the directory exists, the script deletes the directory contents before installing the module. This lets

Parameter Type:

Default Value:

Aliases:

Position:

Pipeline Input:

☒ Required
☒ Accepts wildcard characters
☐ Dynamic

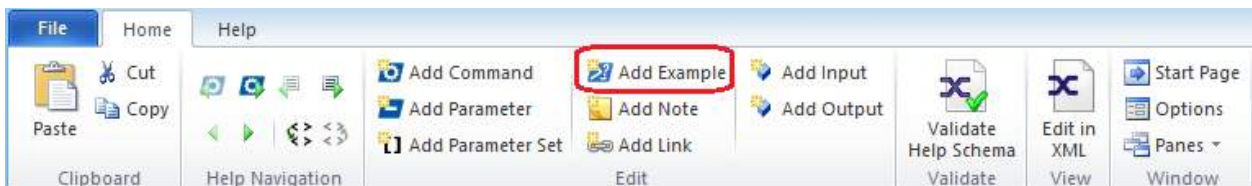
i The parameter nodes under each parameter sets determine the contents of the syntax diagram. The attribute values are specific to each parameter set, but the descriptive fields are shared by all instances of the parameter in the help topic.

Draft the examples

Now, work on examples, which are, arguably, the most important element in any help topic. Arrange the examples in order of complexity, beginning with the simplest commands. For more information about writing examples, see [Write Examples in Help](#)^[47].

To add an example

- On the Home tab, click **Add Example**.



You can use the Introduction, Command, and Remarks fields as designed, or type the prompt, command, and remarks content in the Command field. The Command field is typically used for all elements in a multiple-step command.

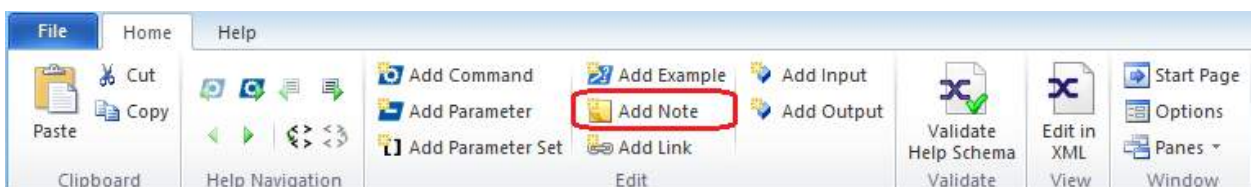
The screenshot shows a window titled 'Update-OneGet > ----- EXAMPLE 2 -----'. It contains four fields: 'Title' with the text '----- EXAMPLE 2 -----', 'Introduction' with 'C:\PS>', 'Command' with '.\Update-OneGet.ps1 -Path \$home\Documents\Test\OneGet -Import|', and 'Remarks' with 'This command installs the newest OneGet module in the \$home\Documents\Test\OneGet directory. The Import parameter imports the newly installed module into the current session.'

Draft the notes

At this point, you might want to add a sentence or two to the Notes section of the help topic. You can add anything to the Notes section, but it's particularly useful for contact information, troubleshooting advice, warnings about particular configurations and corner cases, and versioning.

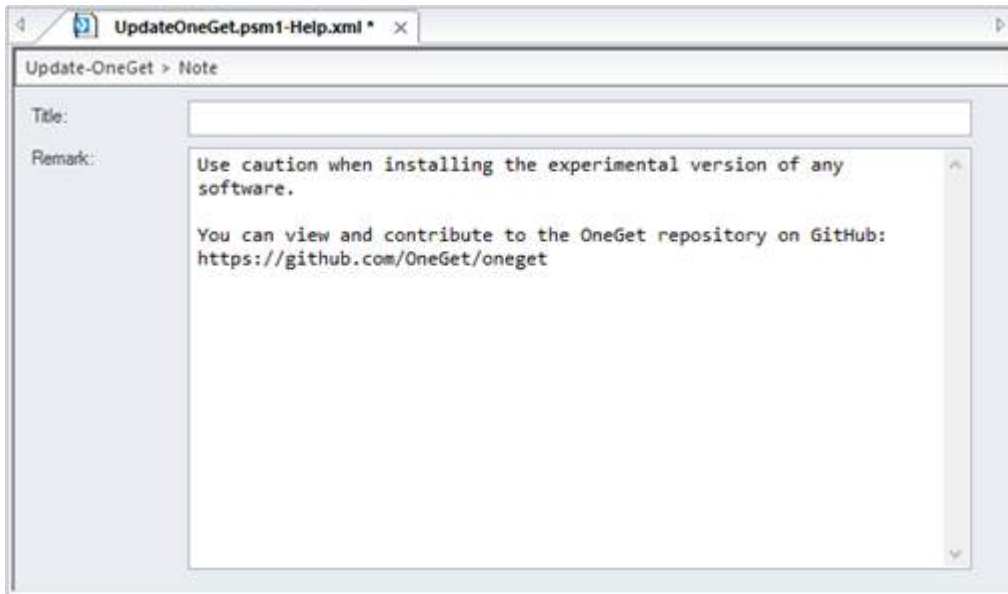
To add a note

- On the Home tab, click **Add Note**. For more information about Notes, see [Write Notes in Help](#).⁶¹



Type the notes as you want them to appear in the Get-Help display.

👍 You can create a separate Note item for each note or type all of the notes in a single Note item.



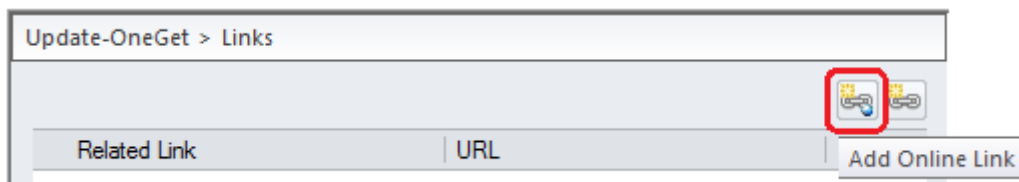
Add related links

The Related Links section of the file lists the names (and, optionally, the URLs) of related help topics. You can also use the URL of the first related link to enable support for online help, that is, the Online parameter of the Get-Help cmdlet.

If you use PowerShell HelpWriter to create a help topic from a module and the existing help for a command in the module includes an online help link, PowerShell HelpWriter adds the online link to the starter help topic. You can also add and delete online links and standard related links from help topics.

To add an online link

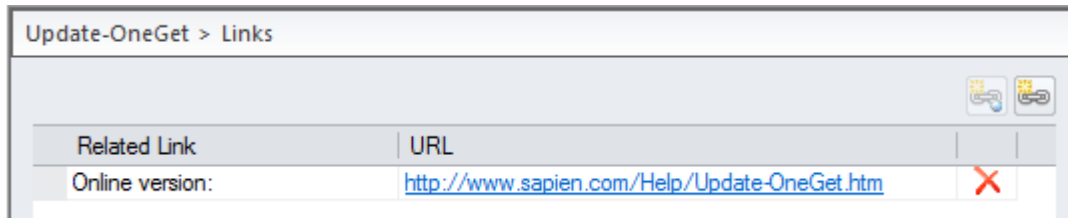
- Click the **Add Online Link** icon.



PowerShell HelpWriter adds the Online version: label.

To enable the link for your site

- Change the URL value to the location of the online version of this help topic.



To add a standard related link

- Click the **Add Link** icon and then type the name of the link.



For more details, see [Add Related Links](#)⁶⁴.

Save your work

When you change a PowerShell HelpWriter file (in any view), an asterisk appears on the tab beside the file name to indicate that there are unsaved changes in the file.

To save your changes

- Type **Ctrl+S**
- OR-
- Right-click the tab and click **Save** > click **File** > then click **Save** (or **Save As**)
- OR-
- Click the **Save** icon on the Quick Access Toolbar.



4.2.3 Edit a Help File

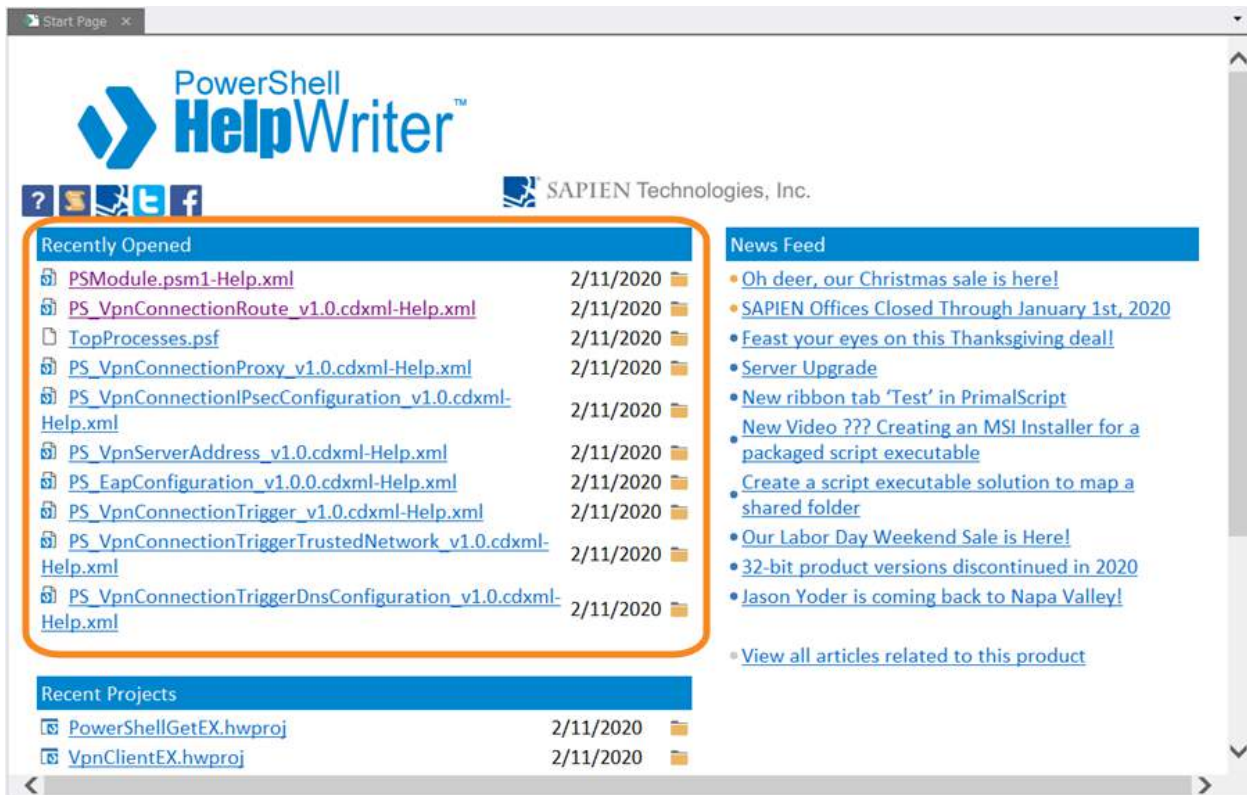
You can close and reopen your help file at any time.

To open a help file (regardless of how it was created)

- Click **File** > and then click **Open**.

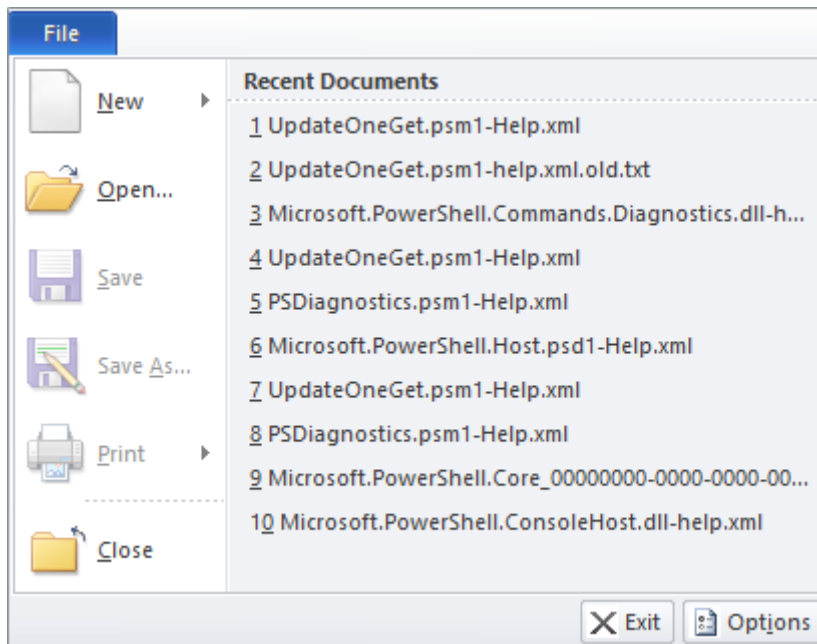
To open a recently used file

- Select it from the **Recently Opened** files list on the PowerShell HelpWriter Start Page.



-OR-

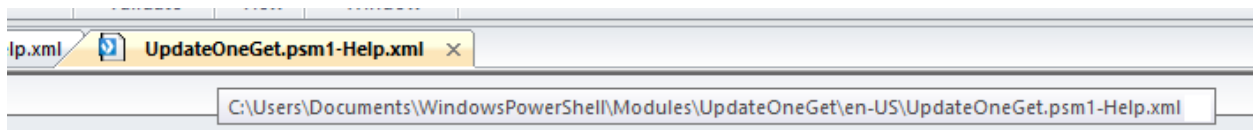
- Select it from the **File** tab > **Recent Documents**.



If you are working with multiple versions of a file, it's easy to forget which version you are editing.

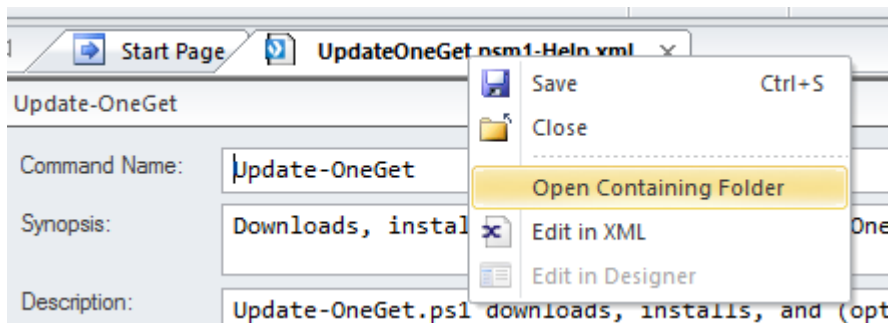
To see the fully qualified path to any open file

- Hover over the file tab.



To open the parent folder

- Right-click the tab and then click **Open containing folder**.



Make any changes you need and test again in the Get-Help display or in PowerShell HelpWriter XML View.

4.2.4 Test in Get-Help

When you have a first draft of your help content, use the **Get-Help** cmdlet to test it.

Save your help file in the location where **Get-Help** looks for it, that is, in a language-specific subdirectory of the module directory, such as:

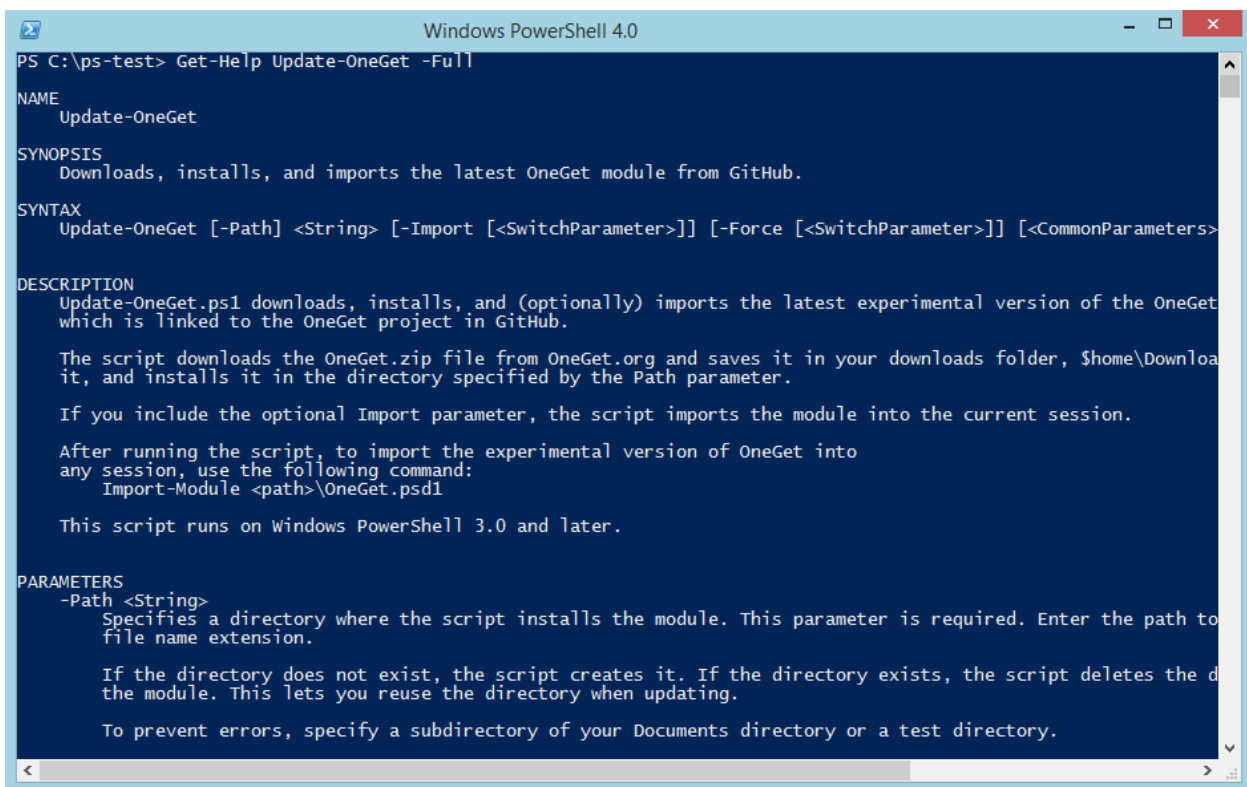
```
$home\Documents\WindowsPowerShell\Modules\<ModuleName>\en-US
```

If the help topic is for a function that has an **ExternalHelp** comment keyword, the help topic name must match the **ExternalHelp** keyword value. Otherwise, the help topic name must match the name of the file in which it is defined, and have a **-help.xml** file name extension. For example, if a cmdlet is defined in the Widgets.dll file, the help topic for the cmdlet must be in a Widgets.dll-help.xml file. For detailed information, see [Naming Help Files](#).

Use the Get-Help cmdlet to display the help topic. For example:

```
Get-Help -Full <CmdletName>
```

You can edit your file in PowerShell HelpWriter while the console window is open. However, to make your changes effective in Get-Help output, restart Windows PowerShell.



```
Windows PowerShell 4.0
PS C:\ps-test> Get-Help Update-OneGet -Full

NAME
    Update-OneGet

SYNOPSIS
    Downloads, installs, and imports the latest OneGet module from GitHub.

SYNTAX
    Update-OneGet [-Path] <String> [-Import [<SwitchParameter>]] [-Force [<SwitchParameter>]] [<CommonParameters>]

DESCRIPTION
    Update-OneGet.ps1 downloads, installs, and (optionally) imports the latest experimental version of the OneGet
    which is linked to the OneGet project in GitHub.

    The script downloads the OneGet.zip file from OneGet.org and saves it in your downloads folder, $home\Downloa
    it, and installs it in the directory specified by the Path parameter.

    If you include the optional Import parameter, the script imports the module into the current session.

    After running the script, to import the experimental version of OneGet into
    any session, use the following command:
        Import-Module <path>\OneGet.ps1

    This script runs on Windows PowerShell 3.0 and later.

PARAMETERS
    -Path <String>
        Specifies a directory where the script installs the module. This parameter is required. Enter the path to
        file name extension.

        If the directory does not exist, the script creates it. If the directory exists, the script deletes the d
        the module. This lets you reuse the directory when updating.

        To prevent errors, specify a subdirectory of your Documents directory or a test directory.
```

4.2.5 Ship your Help File

The help files that you create in PowerShell HelpWriter are ready to ship.

To ship your help file

- Verify that the help file name is the name that the Get-Help cmdlet expects. For details, see [Naming Help Files](#).

To include the help files in your module

- Place them in a language-specific subdirectory of the module directory, such as:

```
$home\Documents\WindowsPowerShell\Modules\<ModuleName>\en-US
```

You can also include the help file in an Updatable Help CAB file along with your About topics. For information about Updatable Help, see [Supporting Updatable Help](#).

5 Using PowerShell HelpWriter

This section shows you how to create new help topics from a module, and how to create an empty help topic. Information is also provided on writing the text for your help, adding related links, and editing help topics in XML.

5.1 Create New Help Topic for a Module

This topic covers the steps to create a new help topic from a module.

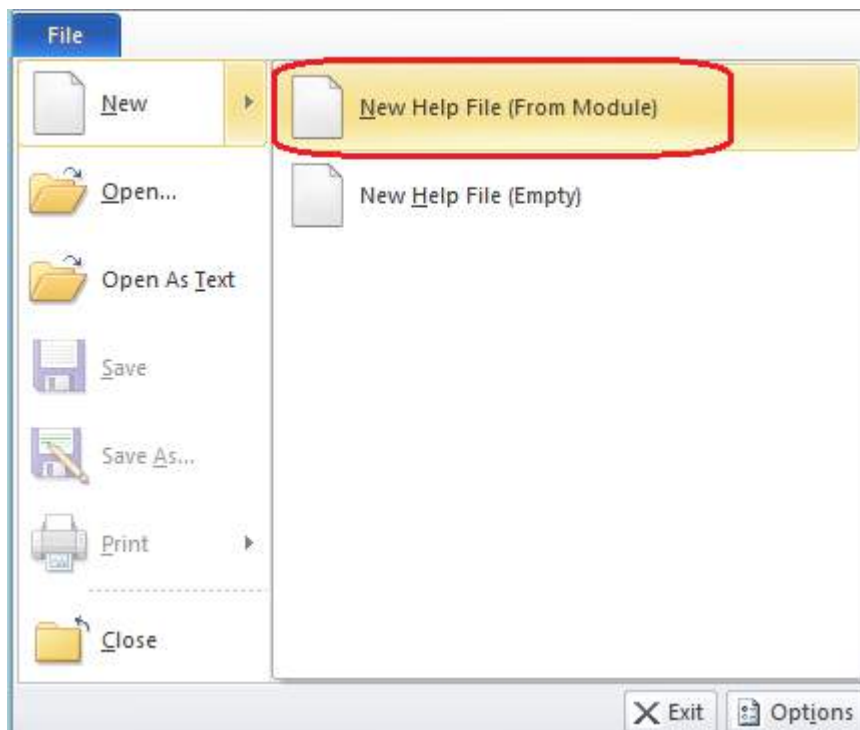
You can use PowerShell HelpWriter to create new XML-based help topics for all of the commands that a module exports. It works with all supported module types and command types, including commands that are defined in nested modules.

The result is a set of content-ready XML help topics that contain the command syntax, parameters, and parameter attributes in the commands. The help files are named for the module in which they are defined as the Get-Help cmdlet requires, so you can concentrate on the all-important descriptions and examples.

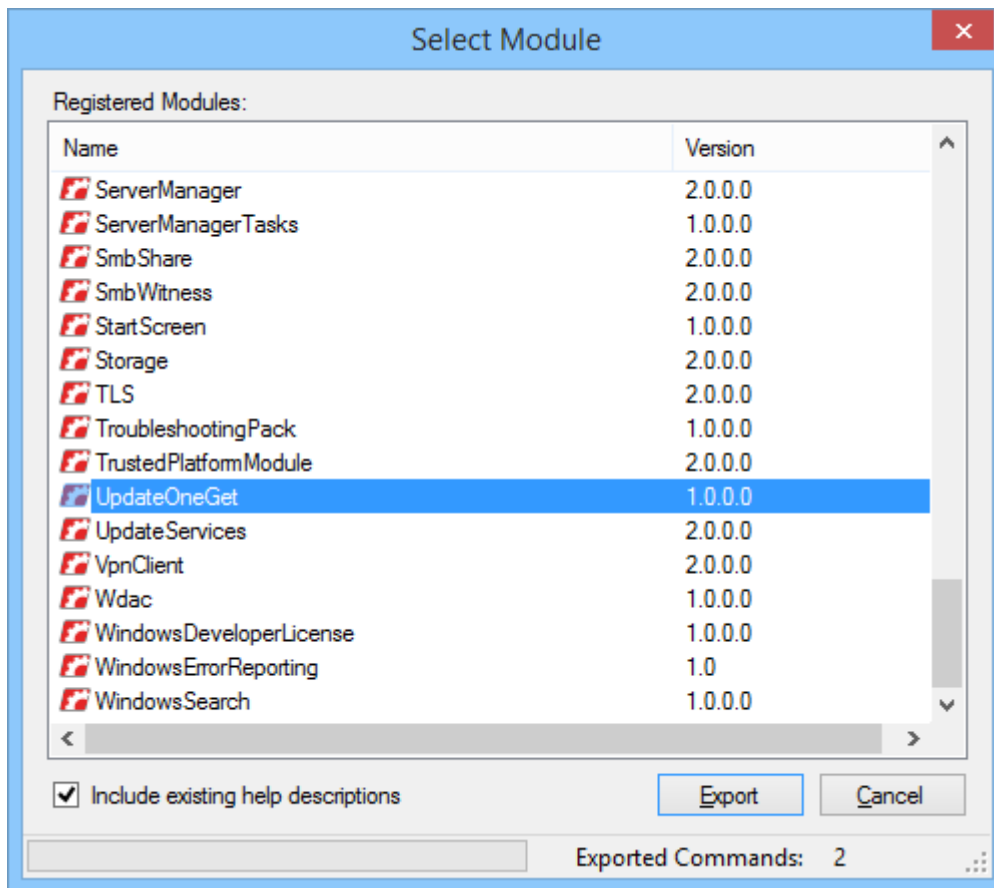
By default, PowerShell HelpWriter begins the new files with any existing help that it finds in the modules, including comment-based help for functions, but you can choose to exclude existing help.

To create a help file for a module

1. Click **File > New > New Help File (From Module) (Ctrl+N)**.



2. Select a module > then click **Export**.



i By default, PowerShell HelpWriter includes all help available for the command, including help in XML files and comment-based help. To omit it, clear the **Include existing help descriptions** checkbox.

3. Save the draft help files in any location. For help on naming and locating files, see [Naming Help Files](#).

i The Get-Help cmdlet looks for the help files in a language-specific subdirectory of the module directory. For example, if a module is saved in:

```
C:\Users\User01\Documents\WindowsPowerShell\Modules\<ModuleName>
```

Get-Help looks for English help files for the module in an **en-US** subdirectory:

```
C:\Users\User01\Documents\WindowsPowerShell\Modules\<ModuleName>\en-US\<Module>.psm1-help.xml
```

Get-Help looks for French help files for the module in an **fr-FR** subdirectory:

```
C:\Users\User01\Documents\WindowsPowerShell\Modules\<ModuleName>\fr-Fr\<Module>.psm1-help.xml
```

Now you are ready to write the text of your help file. You can start with [writing help for parameters](#).

5.2 Create an Empty Help Topic

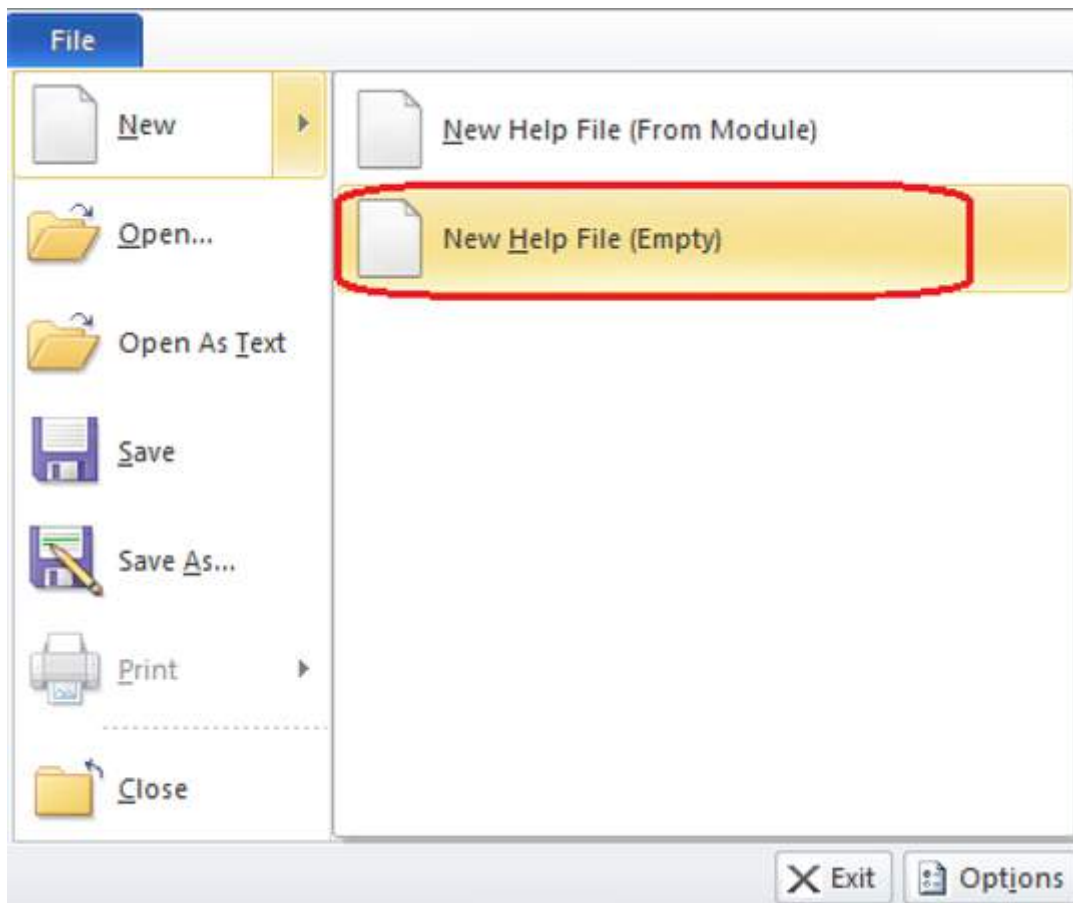
This topic covers the steps to create an empty help topic.

Usually, you write the code and then write the help. But, you might need to write help without the code, either because the code isn't available, or when you are using your help topic as a code specification.

Create a blank help file

To create a blank help file

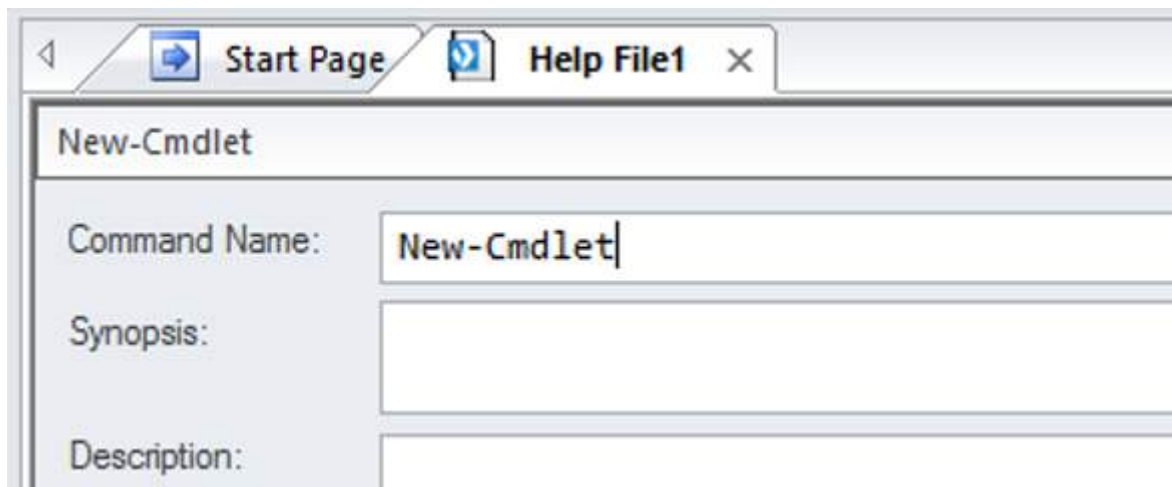
- Click File > New > New Help File (Empty).



Enter the command name

To specify the command name:

1. In the Tree Navigation, click the **New-Cmdlet** (top) node.
2. In the Designer, edit the **Command Name** value.



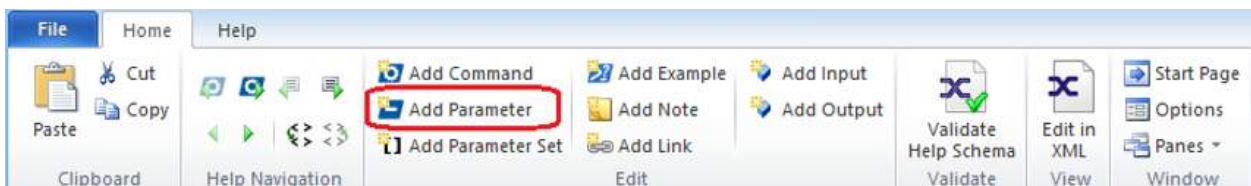
i When you change the default (New-Cmdlet) to the name of the command, the top node of the Tree Navigation changes to the new command name.

Add parameters

When you create a help file without code you need to specify the syntax of the command, including its parameters.

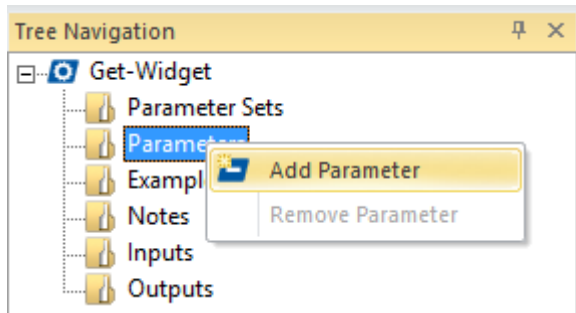
To add a parameter

- On the Home tab > click **Add Parameter**.



-OR-

- In the Tree Navigation > right click **Parameters** and then click **Add Parameter**.



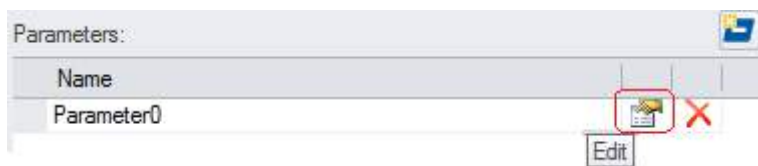
-OR-

- In the Designer > in the Parameters section, click the **Add Parameter** icon.



To add or edit parameter attributes

1. In the Parameters section > on the row of the specified parameter, click the **Edit** icon.



2. In the Designer > enter the **Parameter Name**, **Description**, and **Parameter Type**.

You can also enter the parameter attributes, including Aliases, Position, Pipeline Input, Required, and Accepts wildcard characters ("globbing").

The screenshot shows a configuration window for a PowerShell command parameter. The title bar reads 'Get-Widget > Name'. The window contains several fields and checkboxes:

- Parameter Name:** A text box containing 'Name'.
- Description:** A text area containing 'Specifies a name for the widget. The name must be unique in the domain.' with a vertical scrollbar on the right.
- Parameter Type:** A text box containing 'System.String'.
- Default Value:** An empty text box.
- Aliases:** An empty text box.
- Position:** A dropdown menu showing '0'.
- Pipeline Input:** A dropdown menu showing 'true (ByValue)'.
- Checkboxes:**
 - ☒ Required
 - ☒ Accepts wildcard characters
 - ☐ Dynamic

For more information about writing content and setting values, see [Write Help For Parameters](#)⁴³.

Add syntax and parameter sets

When you create a help file without code, you need to specify the parameter sets in the command. PowerShell HelpWriter creates a syntax statement for you from the parameter sets that you define.

Tips for adding parameter sets without code

- Before adding parameter sets, add the parameters in all parameter sets to the **Parameters** section of your help file.
- You do not need to add the [Common Parameters](#) to your help file. The **Get-Help** cmdlet adds them when it displays the help topic.
- If the command supports an attribute that adds parameters to the cmdlet, you must add the parameters that are associated with the attribute:
 - **SupportsPaging:** First, Skip, IncludeTotalCount
 - **SupportsShouldProcess:** Confirm, WhatIf
 - **SupportsTransactions:** UseTransaction

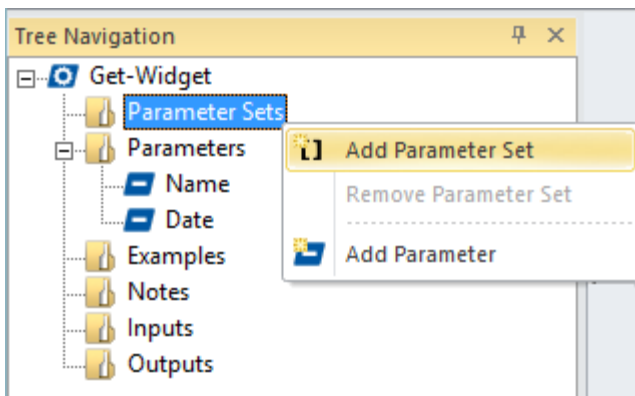
When you add one of these parameters, PowerShell HelpWriter adds the standard description for the parameter. It is a best practice to use the standard description, but you can delete or edit it.

To add a parameter set

- In the Tree Navigation > right-click **Parameter Sets** and click **Add Parameter Set**.

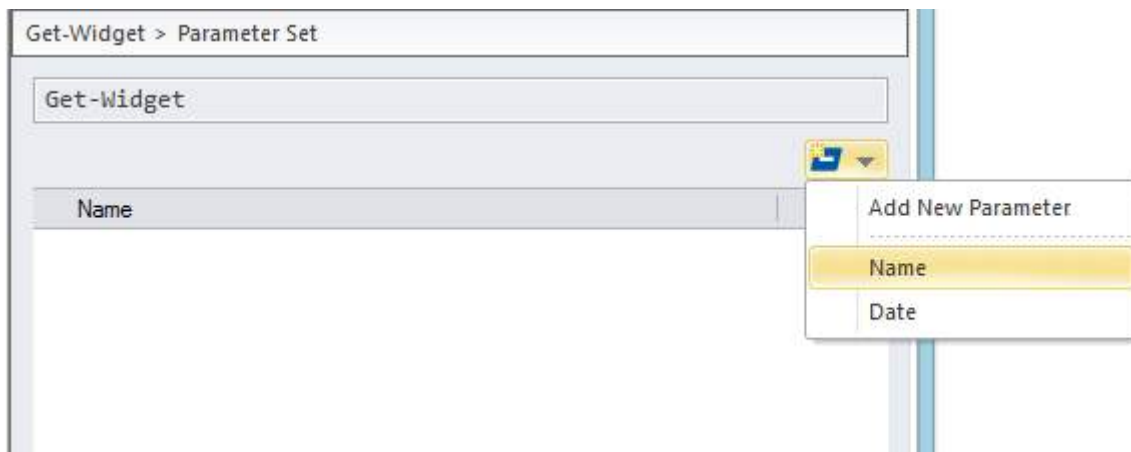
-OR-

- On the Home tab > click **Add Parameter Set**.

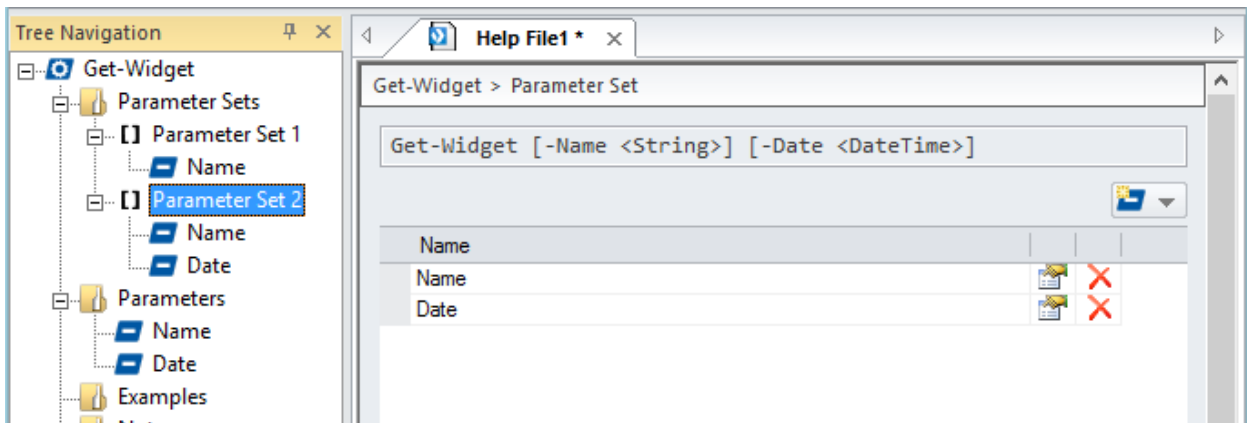


To add parameters to a parameter set

- In the parameter set editor > click the **Add Parameter** icon. Select a parameter or click **Add New Parameter**.

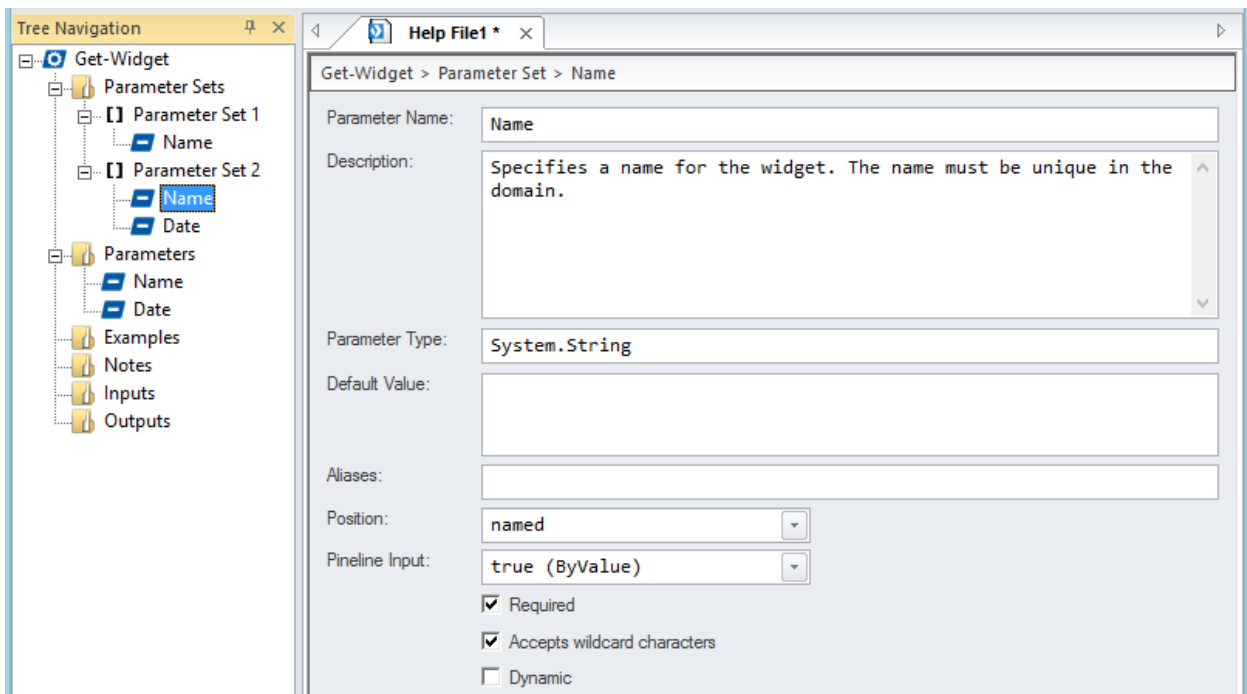


As you add parameters, PowerShell HelpWriter builds the syntax diagram.



To edit the attributes of a parameter in a particular parameter set

- In the parameter set tree > click the **parameter name**.



The values of parameter attributes (Position, Pipeline Input, Required, Accepts wildcard characters, Dynamic) are specific to the parameter in each parameter set. You can edit an attribute in one parameter set without affecting the attributes of the same parameter in other parameter sets. However, the other fields (Parameter Name, Description, Parameter Type, Default Value, and Aliases) are shared by all instances of the parameter.

👍 When the attributes of a parameter differ among parameter sets, the instance of the parameter in the **Parameters** section has the most important value. For example, if a parameter is **Required**

in any parameter set, set it to **Required** in the Parameters section.

Complete the help topic

To add the remainder of the help topic

- Click the item and type the values in the Designer fields.

For more information about writing content and setting values, see [Write Help for Inputs](#)^[53], [Write Help for Outputs](#)^[58], [Write Examples in Help](#)^[47], and [Add Related Links](#)^[64].

5.3 Write Help for Parameters

This topic explains how to write help for parameters.

About Help for Parameters

When you create a help file from the commands in a module, PowerShell HelpWriter adds the parameters in the command, including the parameter attributes, such as Required and Position.

For information about adding parameters and parameter sets from scratch, see [Create an Empty Help Topic](#)^[37].

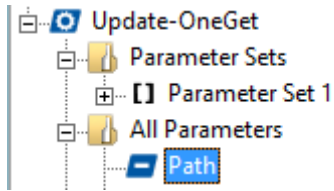
Key Points

- The parameter nodes under each parameter set are used to create the syntax diagram in help topic. When you create a help file from a module, PowerShell HelpWriter creates the syntax diagram from the cmdlet code. Unless the module changes, you do not need to edit the parameter set parameter nodes.
- You do not need to add a <CommonParameters> item to the Syntax or Parameters sections of your help file. The **Get-Help** cmdlet adds it automatically to all help files.

How to work with Help for Parameters

To write help for a parameter

1. In the tree navigation > in the **All Parameters** section, click the **parameter name**.



-OR-

In the Designer > in the **All Parameters** section, in the row for the parameter, click the **Edit** icon.



The editing dialog for the parameter opens.

A screenshot of the 'Update-OneGet > Path' parameter editing dialog. The dialog has the following fields and controls:

- Parameter Name:** A text box containing 'Path'.
- Description:** A large text area for entering a description.
- Parameter Type:** A dropdown menu showing 'String'.
- Default Value:** A text box for entering the default value.
- Aliases:** A text box for entering aliases.
- Position:** A dropdown menu showing '0'.
- Pipeline Input:** A dropdown menu showing 'false'.
- Required:** A checked checkbox.
- Accepts wildcard characters:** An unchecked checkbox.
- Dynamic:** An unchecked checkbox.

2. In the parameter editing dialog, enter the **Description**, **Default Value**, and **Accepts wildcard character** fields (PowerShell HelpWriter cannot infer these values from the command code). Then press **Ctrl+S** to save the file.

Update-OneGet > Path

Parameter Name:

Description:
 Specifies a directory where the script installs the module. This parameter is required. Enter the path to a directory. Do not include a .zip file name extension.
 If the directory does not exist, the script creates it. If the directory exists, the script deletes the directory contents

Parameter Type:

Default Value:

Aliases:

Position:

Pipeline Input:

☒ Required
☐ Accepts wildcard characters
☐ Dynamic

Here is the result in the Get-Help display.

```
PARAMETERS
-Path <String>
  Specifies a directory where the script installs the module. This parameter is required.
  Enter the path to a directory. Do not include a .zip file name extension.

  If the directory does not exist, the script creates it. If the directory exists, the
  script deletes the directory contents before installing the module. This lets you reuse
  the directory when updating.

  Required?                true
  Position?                0
  Default value
  Accept pipeline input?   false
  Accept wildcard characters? true
```

To return to Designer view

- Click on the **command name** in the **breadcrumb** trail at the top of the pane (e.g. Disable-PSTrace).
- OR-
- In the tree navigation > click the **command name** node.


5.3.1 Writing Help Content for Parameters

This topic provides guidelines for writing help content for parameters.

Parameter Fields

Use these guidelines to write help content for the parameters in your help file.

Parameter Name	The name of the parameter as it is spelled and capitalized in the command code. This is a best practice, even though Windows PowerShell is case-insensitive.
Description	<p>The parameter description describes the effect of using the parameter (as opposed to omitting it) and it explains requirements for the parameter's value, including format.</p> <p>For example:</p> <p>"The Path parameter directs the cmdlet to search for a widget in the specified path. By default, it searches for the widget in the current directory. Enter a full or relative path and the file name, including the file name extension."</p> <p>"When you use the Credential parameter, the cmdlet runs with the permissions of the credentialed user. The value is a <code>PSCredentialGet-Credential</code> object, such as the object that the cmdlet returns."</p>
Parameter Type	Specifies the .NET type of the parameter value. PowerShell HelpWriter gets the parameter type from the command code.
Default Value	The value that the cmdlet uses when the parameter is omitted or its value is null. Be sure to supply the default value when the parameter is not mandatory. PowerShell HelpWriter cannot infer this value from the command code.
Aliases	Alternate names for the parameter. You do not need to list partial names, such as the first three letters of the parameter name.
Position	Specifies the required position of the parameter value when the parameter name is omitted. For example, a value of 0 indicates that the parameter value must be the first (or only) unnamed parameter value in the command.

 When a help file has a parameter with a position of 0, Get-Help automatically increments the positions of parameters in the command so that they start at position 1.

Pipeline input	<p>Indicates whether you can pipe parameter values to the command.</p> <p>False: The cmdlet does not take parameter values from the pipeline.</p> <p>True (by value): You can pipe the parameter value to the command. The cmdlets uses the piped value, just so the value has the specified type or is a type that Windows PowerShell can convert to the specified type.</p> <p>True (by property name): You can pipe the parameter value to the command. However, the value is used only when its property name matches the name of the parameter. For example, you can pipe the Name property of an object to the Name parameter of the command.</p>
Required	<p>Indicates that the parameter is mandatory. If you omit the parameter, Windows PowerShell prompts you for a value. If you don't respond to the prompt, the command fails.</p>
Accepts wildcard characters	<p>The parameter value can include "*" or "?" characters. PowerShell HelpWriter cannot infer this value from the command code. Also known as "globbing".</p>
Dynamic	<p>PowerShell HelpWriter indicates when a parameter is dynamic. <i>Dynamic parameters</i> appear in a command only under certain conditions.</p> <p>For example, a parameter appears only when the command runs in a particular provider drive or only when another parameter is used with a particular value. You can elect to include or exclude dynamic parameters, or explain them only in a provider help topic or an About topic.</p>

5.4 Write Examples in Help

This topic explains how to write examples in help.

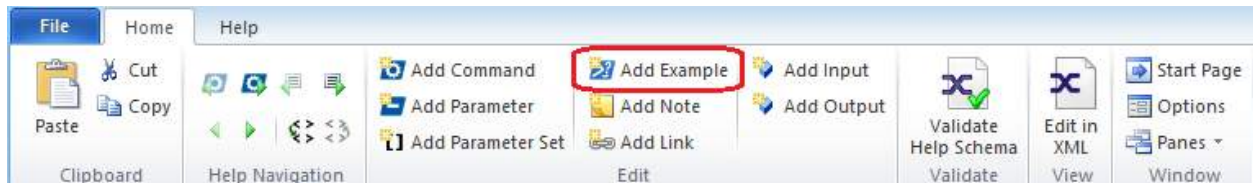
About Examples in Help

Examples might be the most important items in a help topic. Many readers learn best from seeing the cmdlet in action and trying the steps shown in the examples. Time spent crafting instructive examples in help is always well spent.

How to work with Examples in Help

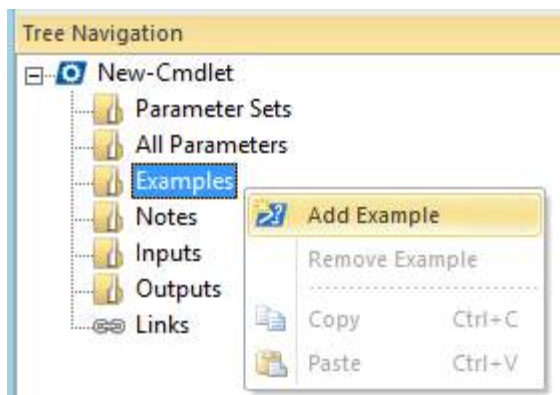
To add an example

- On the Home tab > click **Add Example**.



-OR-

- In the tree navigation > right-click **Examples** and then click **Add Example**.



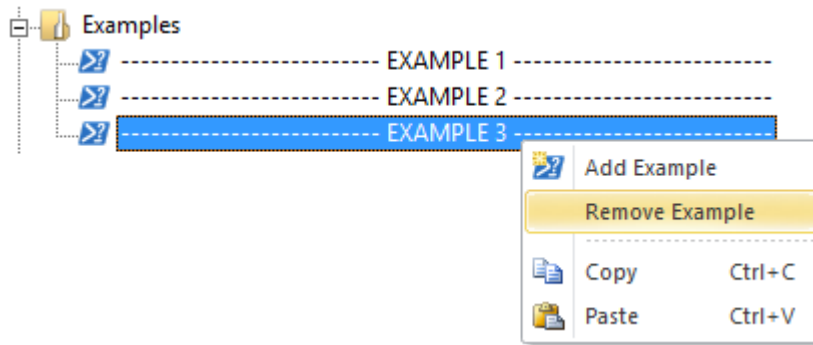
-OR-

- In the Designer, click the Add Example icon.



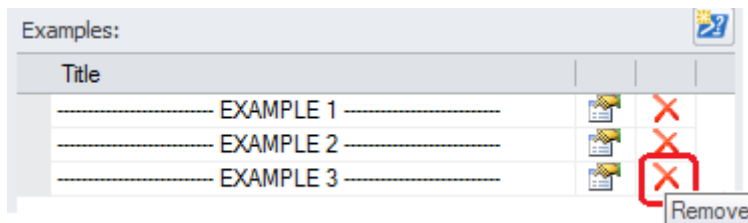
To delete an example

- In the tree navigation > right-click the example and then click **Remove Example**.



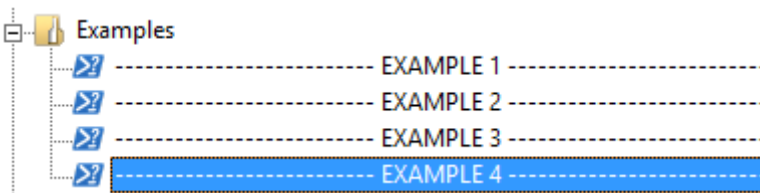
-OR-

- In the Designer > on the example row, click the **Remove** icon.



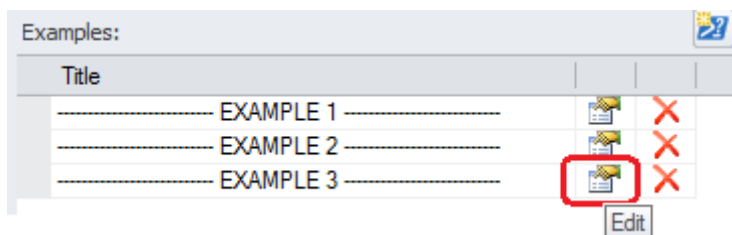
To write an example

- In the tree navigation > click the **example title**.



-OR-

- In the Designer > click the **Edit** icon.



To return to Designer view

- Click on the **command name** in the **breadcrumb** trail at the top of the pane (e.g. Disable-PSTrace).

-OR-

- In the tree navigation > click the **command name node**.

```
Update-OneGet > ----- EXAMPLE 1 -----
```

5.4.1 Writing Help Content for Examples

This topic explains how to write help content for examples.

About Help Content for Examples

Each example has **Title**, **Introduction**, **Command**, and **Remarks** sections. When you create a help file from a module, PowerShell HelpWriter adds a default title and introduction for you.

Arrange examples in order from the simplest to the most complex. Include examples that show how to use each of the cmdlet parameters, alone or in combination. The later examples should show how to use the cmdlet in a complex, real world scenario.

Help example fields

- **Title**

Enter a title that distinguishes the examples from each other. Titles allow users to refer to examples when discussing them. When you create a new example, PowerShell HelpWriter suggests a default title, but you can edit it.

- **Introduction**

Enter the command prompt. When you create a new example, PowerShell HelpWriter adds the default prompt, **PS C:\>**, but you can edit it to match your example. For example, you might want to show that Windows PowerShell was started with the "Run as administrator" option:


```
[ADMIN]: PS C:\>
```

Or, show that it is running in a Windows PowerShell provider drive:

```
PS Cert:\>
```

Or, that you ran an **Enter-PSSession** command to an alternate computer:

```
[Server01]: PS C:\Users\Admin\Documents>
```

 If the example includes a series of commands and responses, you might want to leave the Introduction field blank and place the introduction in the **Command** field immediately preceding the command.

Update-OneGet > ----- EXAMPLE 2 -----

Title: ----- EXAMPLE 2 -----

Introduction: C:\PS>

Command:

Remarks:

- **Command**

Enter the command code. To focus on the example, include full parameter names, even when they are optional, and avoid aliases.

You can also include the command prompt (PS C:\>) and the command output in the **Command** field. For example:

Update-OneGet > ----- EXAMPLE 2 -----	
Title:	----- EXAMPLE 2 -----
Introduction:	<div></div>
Command:	<div>C:\PS> .\Update-OneGet.ps1 -Path \$home\Documents\Test\OneGet -Import</div>
Remarks:	<div></div>

You can even include remarks in the **Command** field. This is especially useful for controlling spaces between elements of the example and in commands with multiple steps.

Update-OneGet > ----- EXAMPLE 2 -----	
Title:	----- EXAMPLE 2 -----
Introduction:	<div></div>
Command:	<div>C:\PS> .\Update-OneGet.ps1 -Path \$home\Documents\Test\OneGet -Import This command installs the newest OneGet module in the \$home\Documents Test\OneGet directory. The Import parameter imports the newly installed module into the current session.</div>
Remarks:	<div></div>

- **Remarks**

Explain what the command does and describe its results. For more complex commands, explain why the example uses the particular strategy or format. When writing remarks, be aware that the **Get-Help** cmdlet displays the content of the **Remarks** field after the contents of the **Command** field. Other help tools might display the contents in a different order.

Update-OneGet > ----- EXAMPLE 2 -----	
Title:	----- EXAMPLE 2 -----
Introduction:	C:\PS>
Command:	.\Update-OneGet.ps1 -Path \$home\Documents\Test\OneGet -Import
Remarks:	This command installs the newest OneGet module in the \$home\Documents\Test\OneGet directory. The Import parameter imports the newly installed module into the current session.

Here is the Get-Help display when each element of the example is typed in a separate field.

```
----- EXAMPLE 2 -----  
  
C:\PS>.\Update-OneGet.ps1 -Path $home\Documents\Test\OneGet -Import  
  
This command installs the newest OneGet module in the $home\Documents\Test\OneGet directory.  
The Import parameter imports the newly installed module into the current session.
```

5.5 Write Help for Inputs

This topic explains how to write help for Inputs.

About Help for Input Types

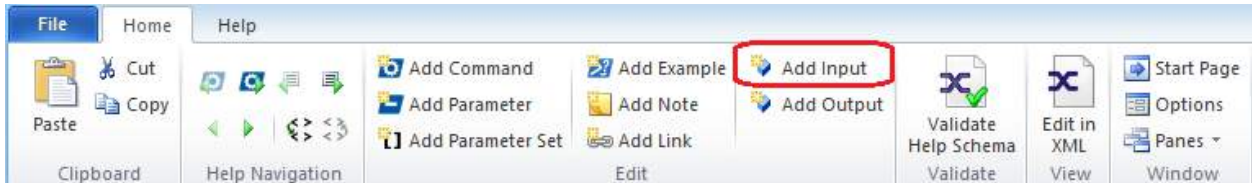
An input or input type is a .NET type that a parameter of the cmdlet takes from the pipeline by value (true by value). When you can pipe an object of the specified type to the cmdlet, it uses the object as the parameter value.

PowerShell HelpWriter gets the input types of a command from the command code. However, you can add and delete input types as needed, and add help for the input types.

How to work with Input Types

To add an input type

- On the Home tab > click **Add Input**.



-OR-

- In the tree navigation > right-click **Inputs** and then click **Add Input**.



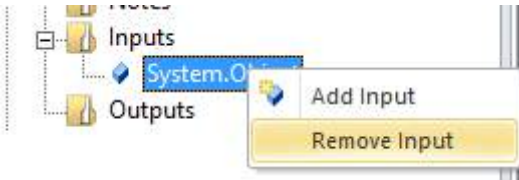
-OR-

- In the Designer > click the **Add Input** icon.



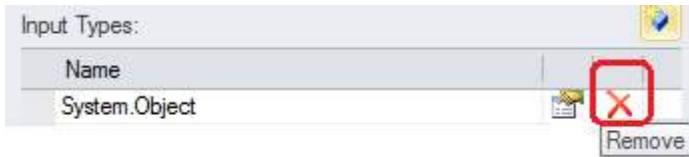
To delete an input type

- In the tree navigation > right-click an input type and click **Remove Input**.



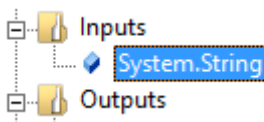
-OR-

- In the Designer > on the input type row, click the **Remove** icon.



To write help for an input type

1. In the tree navigation > click the **input type name**.



-OR-

- In the Designer > on the input type row, click the **Edit** icon.



2. Write the help content and save it (**Ctrl+S**).

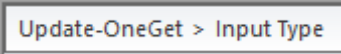
Update-OneGet > Input Type	
Name:	System.String
Description:	You can pipe the path as a string to Update-OneGet.
Uri:	https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx

To return to the command page of the Designer

- In the breadcrumb trail > click the **command name**.

-OR-

- In the tree navigation > click the **command name**.



5.5.1 Writing Help Content for an Input Type

This topic explains how to write help content for Input Types.

About Input Types

The **Inputs** section of a Windows PowerShell help topic lists the .NET types that you can pipe to the cmdlet. The types are used as the values of parameters that accept pipeline input by value, not just by property name. The input type name is the name of the parameter value type, not the name of the parameter. For example, if the **Name** parameter of the cmdlet takes a **System.String** value from the pipeline by value, the input type name is **System.String**.

For example, you can pipe a **System.String** object to the **Name** parameter of the **Get-Command** cmdlet. In this case, we pipe the "Get-ScheduledJob" string to **Get-Command**.

```
PS C:\> "Get-ScheduledJob" | Get-Command
```

CommandType	Name	ModuleName
-----	----	-----
Cmdlet	Get-ScheduledJob	PSScheduledJob

If you were writing help for the **Get-Command** cmdlet, in the Inputs section, you would list **System.String** and explain that you can pipe System.String values to the Name parameter of Get-Command.

How to find the Inputs of a cmdlet

To find the parameters of a command that take input by value, use the following command:

```
(Get-Command <commandName>).ParameterSets.Parameters |
where ValueFromPipeline |
Format-List -Property ParameterType, Name
```

For example, the **Set-LogProperties** cmdlet in the PSDiagnostics module has a **LogDetails** parameter that takes a **Microsoft.PowerShell.Diagnostics.LogDetails** object by value.

```
PS C:\> (Get-Command Set-LogProperties).ParameterSets.Parameters |
where ValueFromPipeline |
Format-List -Property ParameterType, Name
```

```
ParameterType : Microsoft.PowerShell.Diagnostics.LogDetails
Name          : LogDetails
```

In this case, the input type is **Microsoft.PowerShell.Diagnostics.LogDetails**. In the description, explain that you can pipe an object of this type to the **LogDetails** parameter.

Input Type fields

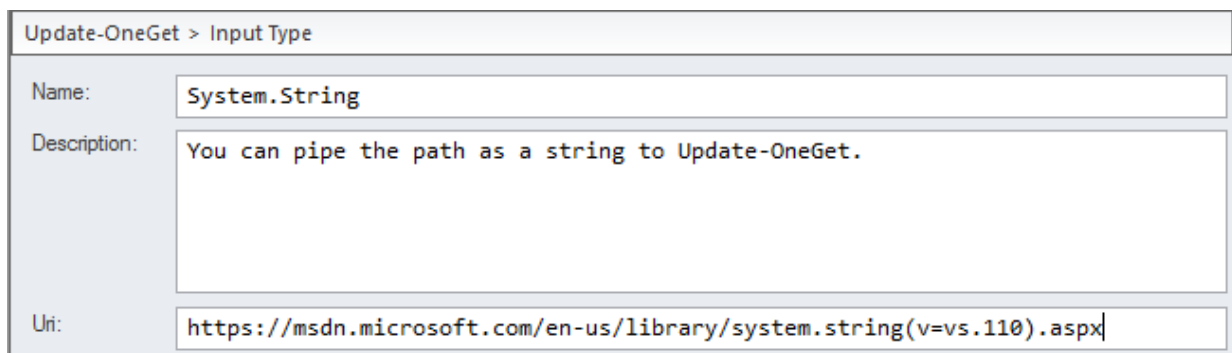
Each input type has a **Name**, **Description**, and **Uri** field.

Name	Enter the name of a type that a parameter takes by value.
Description	<p>When you pipe an object with the specified type to the cmdlet, the object, or one of its properties, is used as the value of one of the cmdlet parameters.</p> <ul style="list-style-type: none">• Tell which parameter is associated with the input type and how the value is used.• Explain how to get an object of that type. For example, the Get-Wid-get cmdlet returns this type.
Uri	Specifies the URL of the web page that describes the type, such as an MSDN reference page.

How to write help for an Input Type

To write help for an input type

1. Type the values in the Input Type editor.



Update-OneGet > Input Type

Name:

Description:

Uri:

2. View the result in the Get-Help display.

```
INPUTS
System.String
https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx
You can pipe the path as a string to Update-OneGet.
```

5.6 Write Help for Outputs

This topic explains how to write help for Outputs.

About Help for Output Types

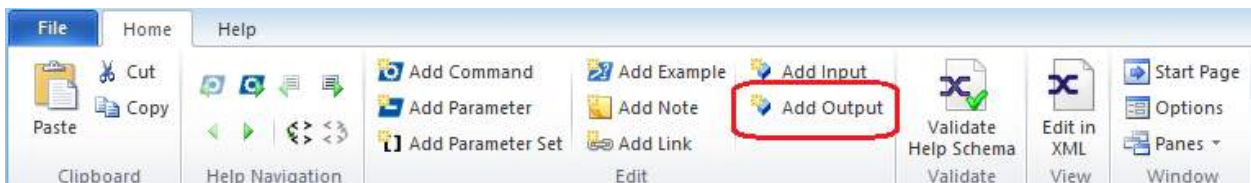
An output type is the .NET type of an object that the cmdlet returns as standard output.

PowerShell HelpWriter gets the output types of a command from the command code. However, you can add and delete output types as needed, and add help for the output types.

How to work with Output Types

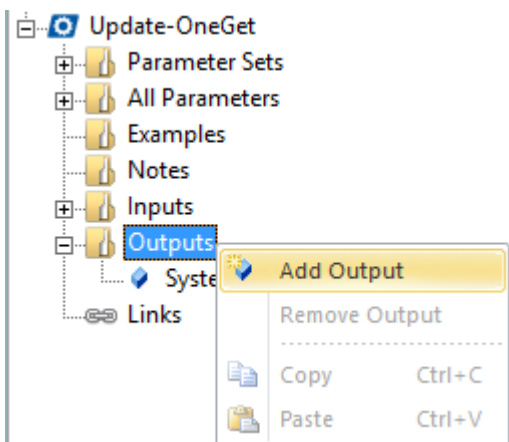
To add an output type

- On the Home tab > click **Add Output**.



-OR-

- In the tree navigation > right-click **Outputs** and then click **Add Output**.



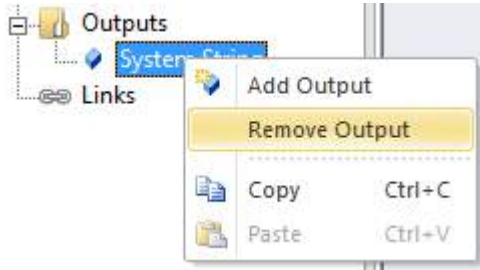
-OR-

- In the Designer > click the **Add Output** icon.



To delete an output type

- In the tree navigation > right-click the output type and then click **Remove Output**.



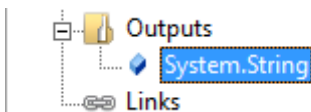
-OR-

- In the Designer > on the output type row, click the **Remove** icon.



To write help for an output type

1. In the tree navigation > click the **output type name**.



-OR-

- In the Designer > on the output type row, click the **Edit** icon.



2. Write the help content and save it (**Ctrl+S**).

Update-OneGet > Output Type	
Name:	<input type="text" value="System.String"/>
Description:	<input type="text" value="Update-OneGet returns the path to the installed module as a string."/>
Uri:	<input type="text" value="https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx"/>

To return to Designer view

- In the breadcrumb trail > click the **command name**.
- OR-
- In the tree navigation > click the **command name**.

Update-OneGet > Output Type

5.6.1 Writing Help Content for an Output Type

This topic explains how to write help content for Output Types.

About Output Types

Each output type has a **Name**, **Description**, and **Uri** field.

In some cases, the cmdlet returns different types when different parameters are used in the command. For example, some cmdlets return objects only when the PassThru parameter is used.

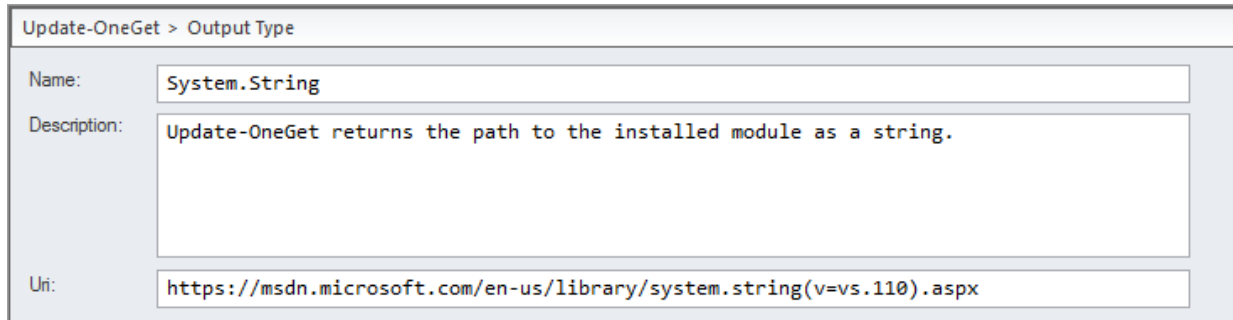
Output Type Fields

Name	Enter the .NET type of the output object.
Description	Explains the content of the output object and the conditions under which the cmdlet returns output of the specified type.
Uri	Specifies the URL of the web page that describes the type, such as an MSDN reference page.

How to write help for an Output Type

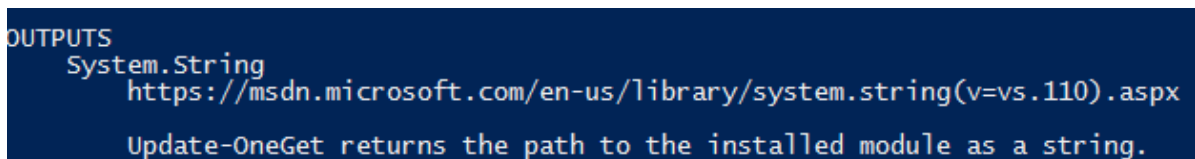
To write help for an output type

1. Type the values in the Output Type editor.



Update-OneGet > Output Type	
Name:	System.String
Description:	Update-OneGet returns the path to the installed module as a string.
Uri:	https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx

2. View the results in the Get-Help display.



```
OUTPUTS
System.String
https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx
Update-OneGet returns the path to the installed module as a string.
```

5.7 Write Notes in Help

This topic explains how to write notes in help.

About Notes in Help

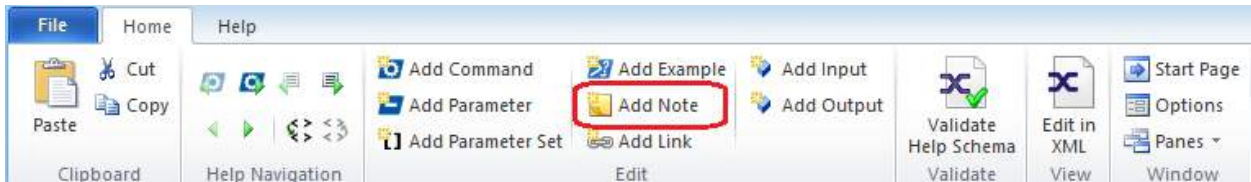
Each help topic has a **Notes** section. You can use the Notes section to tell give the reader extra information, such as how to contact the author, troubleshooting tips, guidance for atypical configurations, and system requirements.

 In the PSMAML XML schema, notes appear in the `<maml:AlertSet>` element.

How to work with Notes in Help

To add a note

- On the Home tab > click **Add Note**.



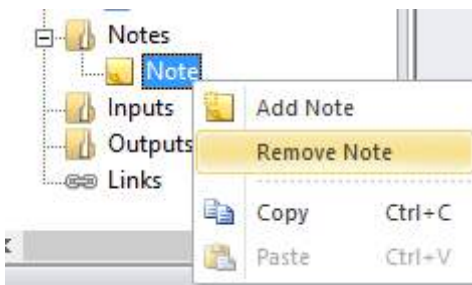
-OR-

- In the tree navigation > right-click **Notes** and then click **Add Note**.



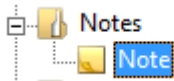
To delete a note

- In the tree navigation > right-click an note and click **Remove Note**.



To write a note

1. In the tree navigation > click the note.



2. Write the note and save it (**Ctrl+S**).



To return to the Designer view

- In the breadcrumb trail > click the **command name**.

-OR-

- In the tree navigation > click the **command name**.

Update-OneGet > Note

5.7.1 Writing Note Content

This topic explains how to write note content.

About Note Content

Each note has a **Title** and a **Remark** section. The title is optional and is typically omitted.

You can write all of your notes in a single Note item or create a separate note item (with optional title) for each note.

👍 Writing in a single Note item lets you control the spacing between notes.

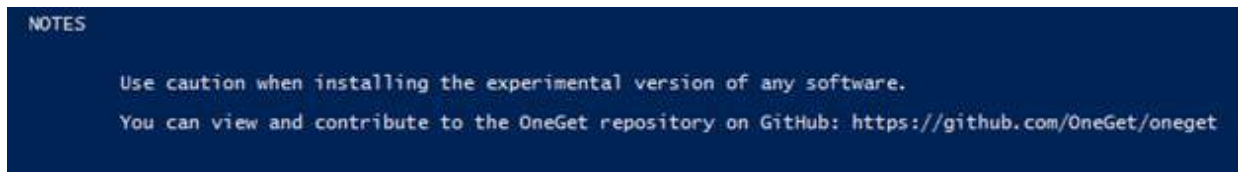
How to write Note content

To write the note content

1. Type the values in the Note editor.



2. View the results in the Get-Help display.



5.8 Add Related Links

This topic explains how to add related links to a help topic.

About Related Links

The Related Links section of a help topic consists of the names of related help topics with optional URLs.

Supporting Online Help

The Related Links section also implements support for online help in Windows PowerShell. The **Online** parameter of the **Get-Help** cmdlet (**Get-Help -Online**) uses the URL of the first item in the Related Links section (if the first Related Link has a URL).

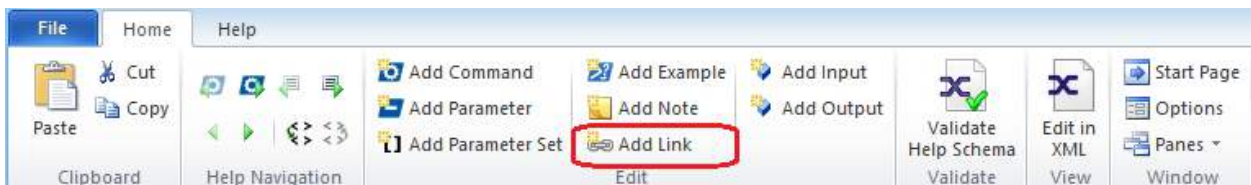
You can also support online help by adding the **HelpUri** attribute to any command type. If a command has both a **HelpUri** attribute and a URL in the first related link, the value of the URL in the first related link takes precedence over the value of the **HelpUri** attribute. For detailed instructions on supporting online help, including specific instructions for implementing online help in each command type, see [Supporting Online Help](#).

When you create a help topic from a module and the existing help includes an online link, PowerShell HelpWriter adds the online link to the starter help file. You can also add and delete online links from any help topic. The online link must be the first item in the Related Links section of the help topic and it must have a URL value that begins with **http** or **https**.

How to work with Related Links

To add a related link

- On the Home tab > click **Add Link**.



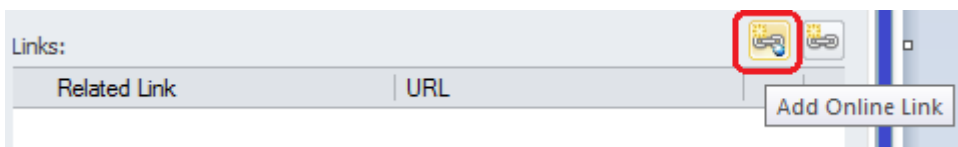
-OR-

- In the Designer > click the **Add Link** icon.



To add an online help link

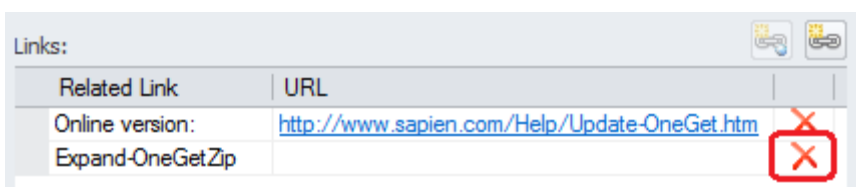
- In the Designer > click the **Add Online Link** icon.



i Because each help topic can have only one online link, the **Add Online Link** icon is disabled when the help topic has an online link.

To delete a related link

- In the Designer > on the Related Link row, click the **Remove** icon.



5.8.1 Writing Help for a Related Link

This topic explains how to write help for a related link.

About Related Link Help


The online link must be the first link in the related links section. The URL value must begin with http or https. You can have only one online link in each help topic.

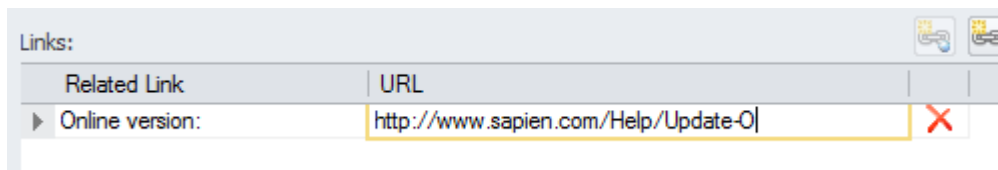
Online Help Related Link

When you add an online link, PowerShell HelpWriter adds the "Online version:" text for you.

To change the default URL to the URL of the online version of the help topic

- Click in the **URL field** > then **type the values** and save (**Ctrl+S**).

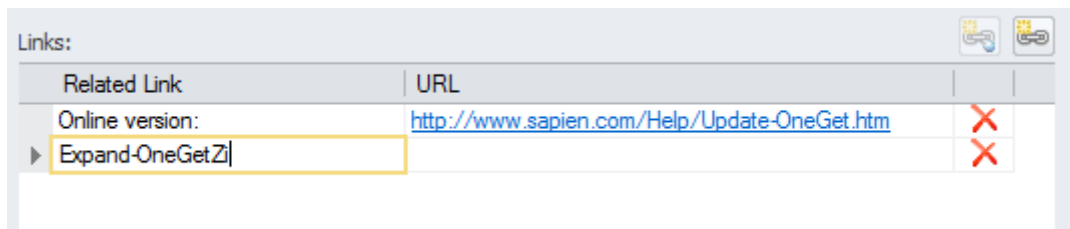
 After you add an online link the Add Online Link icon is disabled.




Standard Related Link

To edit a standard related link

- In the Links window > **edit the link name** and, optionally, the URL.



 If you are *not* supporting online help for the cmdlet, do **not** enter a URL in the first related link. Otherwise, **Get-Help** assumes that it is an online version link, regardless of the link name.

5.9 Edit Help Topics as XML

You can view and edit your help files in any text or XML editor, or use the PowerShell HelpWriter XML Editor. The PowerShell HelpWriter XML Editor can edit any text or XML file, including help files that you create in PowerShell HelpWriter and other tools.

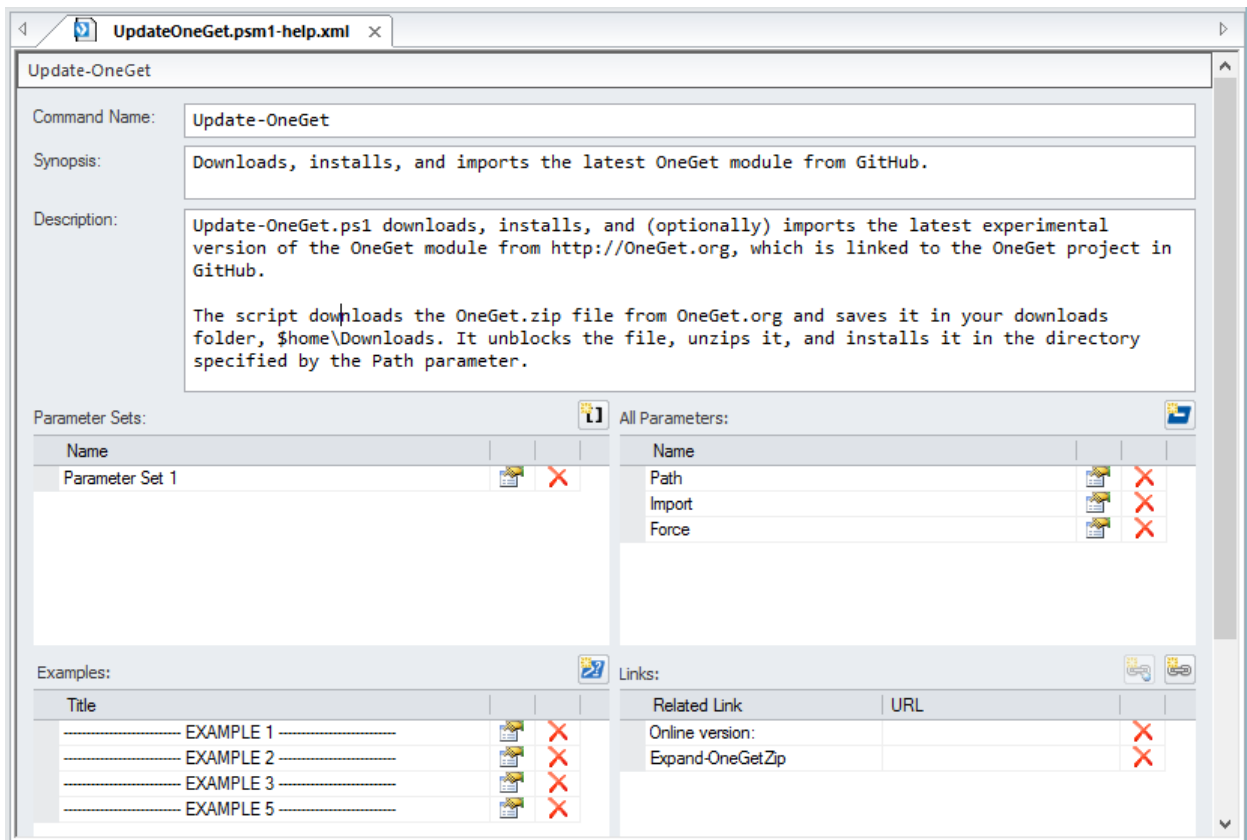
5.9.1 Switching Between the Designer and Editor

While writing or editing, you can switch back and forth from editing in the Designer to editing in the XML Editor.

- i When you switch from the Designer to the XML Editor (and back), the current file is saved, closed, and reopened to prevent overwriting changes made in the previous view.

To switch from the Designer and XML Editor

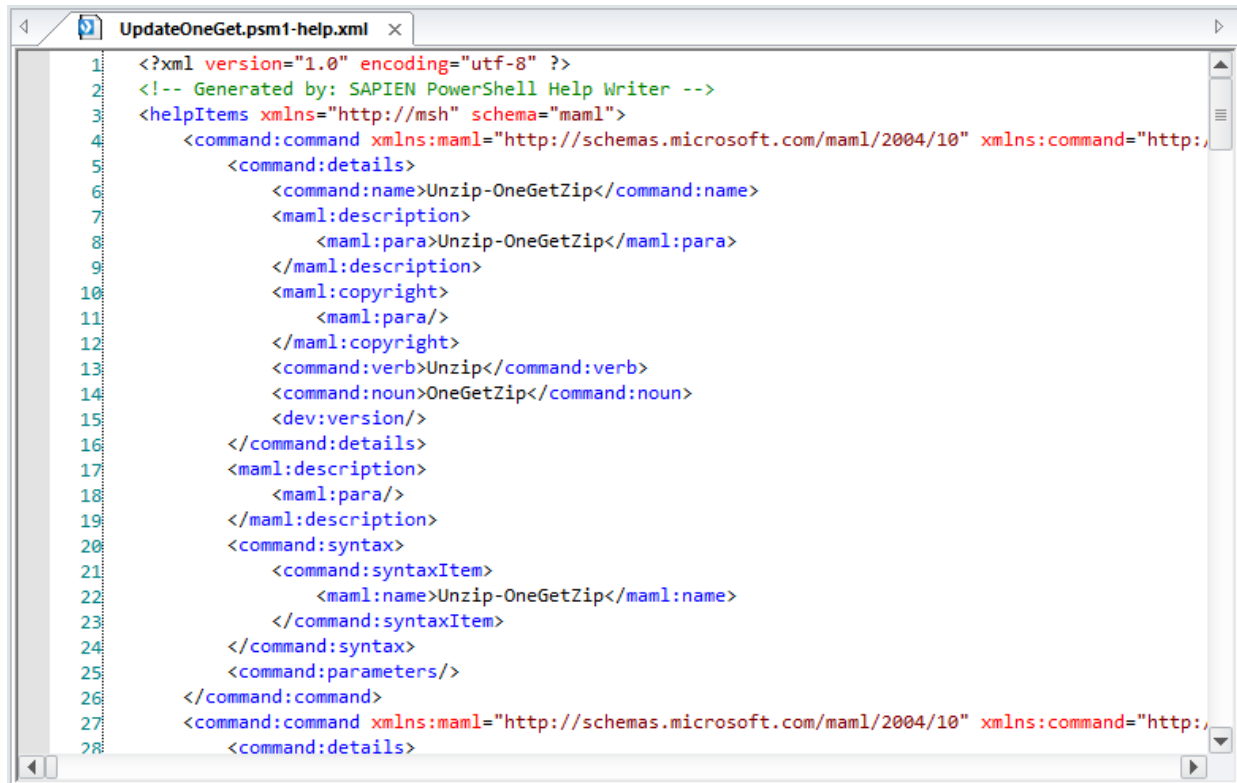
1. In PowerShell HelpWriter, create or open a help file.



2. On the ribbon, click **Edit in XML**.

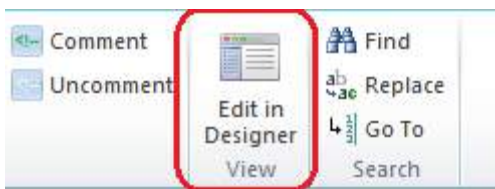


The file closes and reopens in the XML Editor. This isn't just a display. You can edit the XML, too.



To switch from the XML Editor to the Designer

- On the ribbon > click **Edit in Designer**.



5.9.2 Opening XML Files in the XML Editor

You can use the XML editor in PowerShell HelpWriter to edit any XML file.

- i** If you open an XML file that is not a Windows PowerShell help file, PowerShell HelpWriter automatically opens it in the XML Editor.

How to open XML files in the Editor

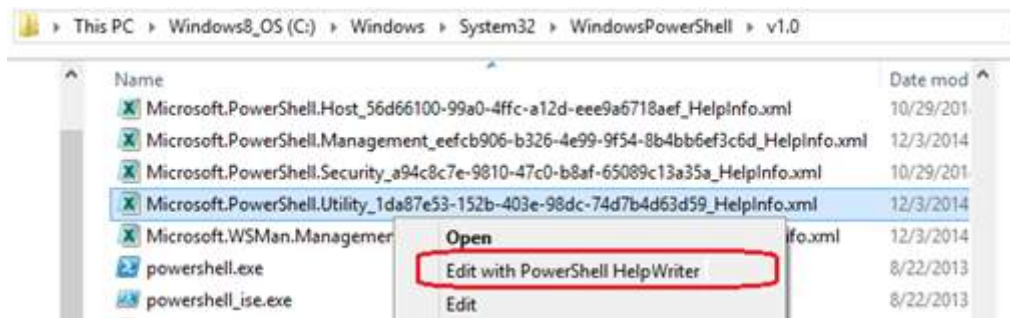
To open an XML file in the XML Editor

- In PowerShell HelpWriter > click **File** > **Open** (*Ctrl+O*) > navigate to a file, and then click **Open**.

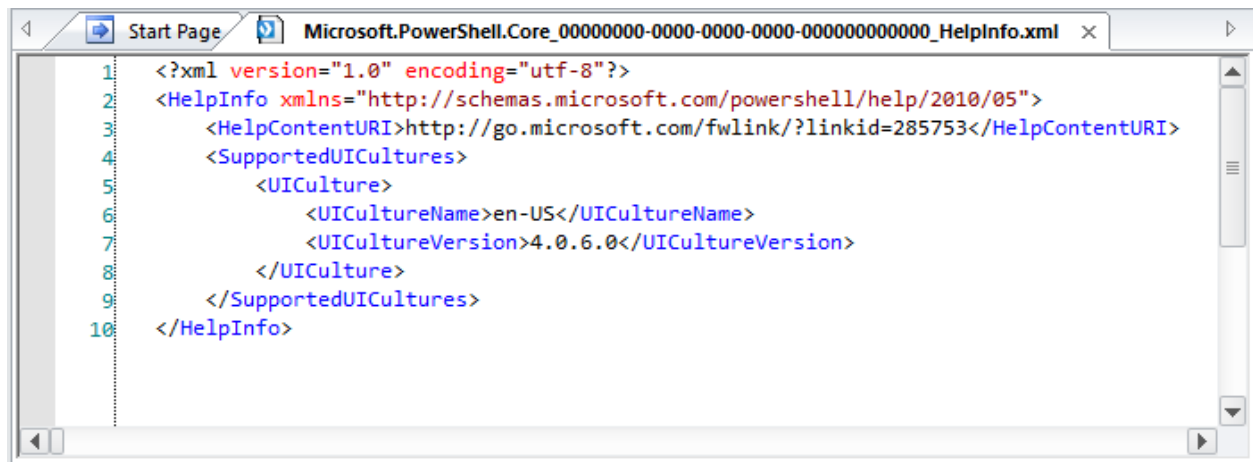
-OR-

- In File Explorer > right-click any XML file, and then click **Edit with PowerShell HelpWriter**.

For example, these HelpInfo.xml files for Updatable Help open directly in the XML Editor.



The file opens in XML Editor.



5.9.3 Formatting XML Files

PowerShell HelpWriter generates well-formatted ("pretty print") XML and automatically formats XML files that you open.

How to Format and Reformat XML Files

You can use the PowerShell HelpWriter editor to format or reformat any XML file.

To format an XML file

1. Open the file in PowerShell HelpWriter.

 Help files automatically open in the designer. Other XML files open in the XML editor.

To switch a help file to the XML editor, if necessary, click **Edit in XML**.

2. Click **Save**.

When you save, PowerShell HelpWriter writes the formatted XML to your file.

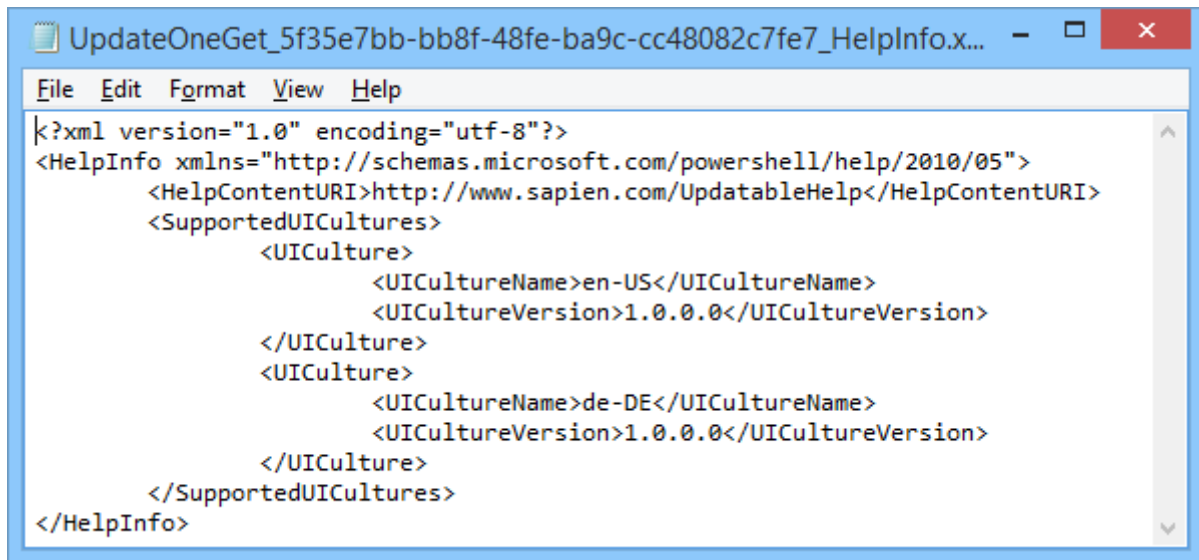
For example, this generated XML file, shown in Notepad with word wrap, has all elements on a single line.



When you open the file in PowerShell HelpWriter, it automatically formats the XML.



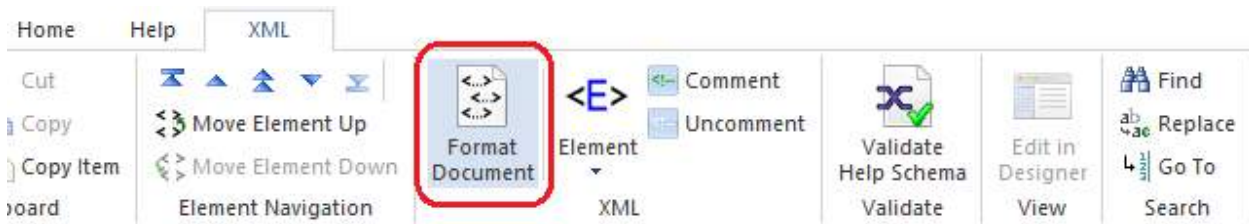
When you save, PowerShell HelpWriter writes the formatted XML to your file. Here's the result in Notepad.



To reformat an XML file

If you edit the XML and need to reformat it:

- On the XML tab, click **Format Document**.



5.9.4 Using Empty or Closing Tags

This topic explains how to use empty or closing tags.

About XML Element Tags

PowerShell HelpWriter creates separate opening and closing tags for all elements.

For example: `<maml:para> </maml:para>`

However, when an XML element is empty, you can replace the opening and closing tags with an empty element tag.

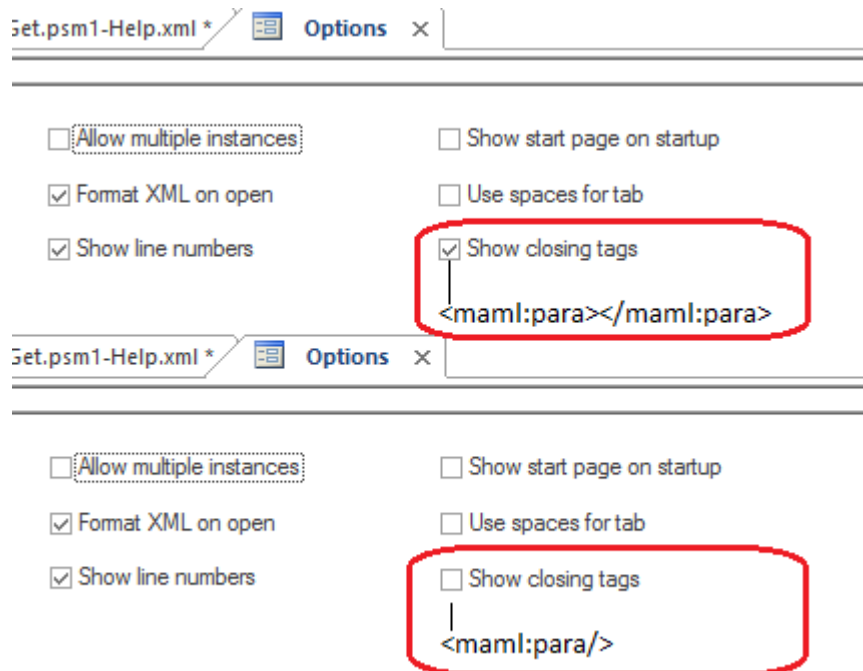
For example: `<maml:para/>`

👍 Both formats are considered to be well-formed XML.

Determining an Empty Tag Format


To determine the empty tag format

1. On the Home tab > click **Options**.
2. Check or clear the **Show closing tags** option.
3. To make the change effective > on the XML tab, click **Format Document**.



6 Getting Help

This help manual has been designed to provide all the information you will need for using PowerShell HelpWriter. In addition to the information in this guide, you can also ask questions in the [online support forums](#)^[73].

 View PowerShell HelpWriter product feature demonstrations and release details on [our blog](#).


Displaying the help manual

- On **Help** ribbon > in the **Product Support** section, click **Product Manual**.

User forums and support

Use the SAPIEN forums to get help with PowerShell HelpWriter, submit feature requests, and more.

- [PowerShell HelpWriter forum](#)
- [Scripting support forums](#)
- [All SAPIEN support options](#)

 To report a problem in the PowerShell HelpWriter forum, you will need to provide your SAPIEN product and OS [version information](#)^[73].

How to copy version information

To report a problem in the [PowerShell HelpWriter forum](#), you will need to include the product version and build, and also your OS version and build—and indicate 32 or 64-bit for each.

To copy the required version information

1. Click the **About** button in the top-right of the PowerShell HelpWriter workspace.
2. In the About PowerShell HelpWriter window, select **Copy Version Info**.
3. Paste the version information into your [PowerShell HelpWriter forum](#) post.

7 SAPIEN Updates

We are continually updating our software, both to remove bugs and to add and improve product features. We recommend always staying current with the most recent versions to ensure that you are taking advantage of the latest features, functionality, and product stability.

Every SAPIEN product has a built-in update tool—**SAPIEN Updates**—which will check for updates on all current activations and unexpired trial versions of our products. Available product updates are indicated in the SAPIEN Updates tool and also in the [Notifications dialog](#)^[74] (see below).

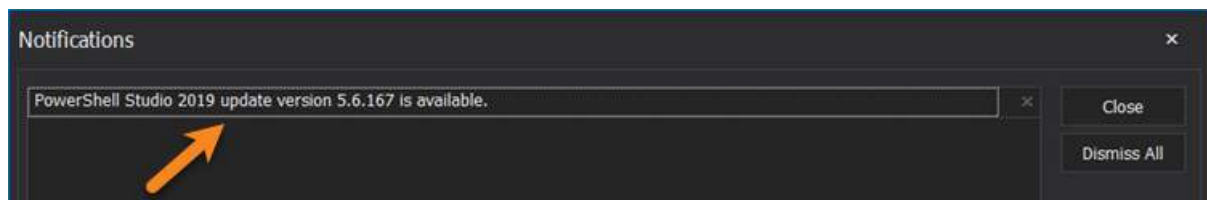
SAPIEN Notifications

SAPIEN products provide automatic notifications when there is a software update available, or when your maintenance is about to expire. Notifications are indicated by a 'flag' icon in the top-right of the program window:



How to view SAPIEN notifications

- Click the notification flag icon above the ribbon to open the Notifications dialog:



- If a product update is available, click the update notification to open the SAPIEN Updates tool.

👍 Click the X button to dismiss individual notifications or select **Dismiss All**. Dismissed notifications will not be shown again.

SAPIEN Updates - Tool Overview

The SAPIEN Updates tool indicates when an update is available for any SAPIEN program installed on your computer.

i To minimize the impact on your system, the tool does not run during Windows startup or continuously in the system tray.

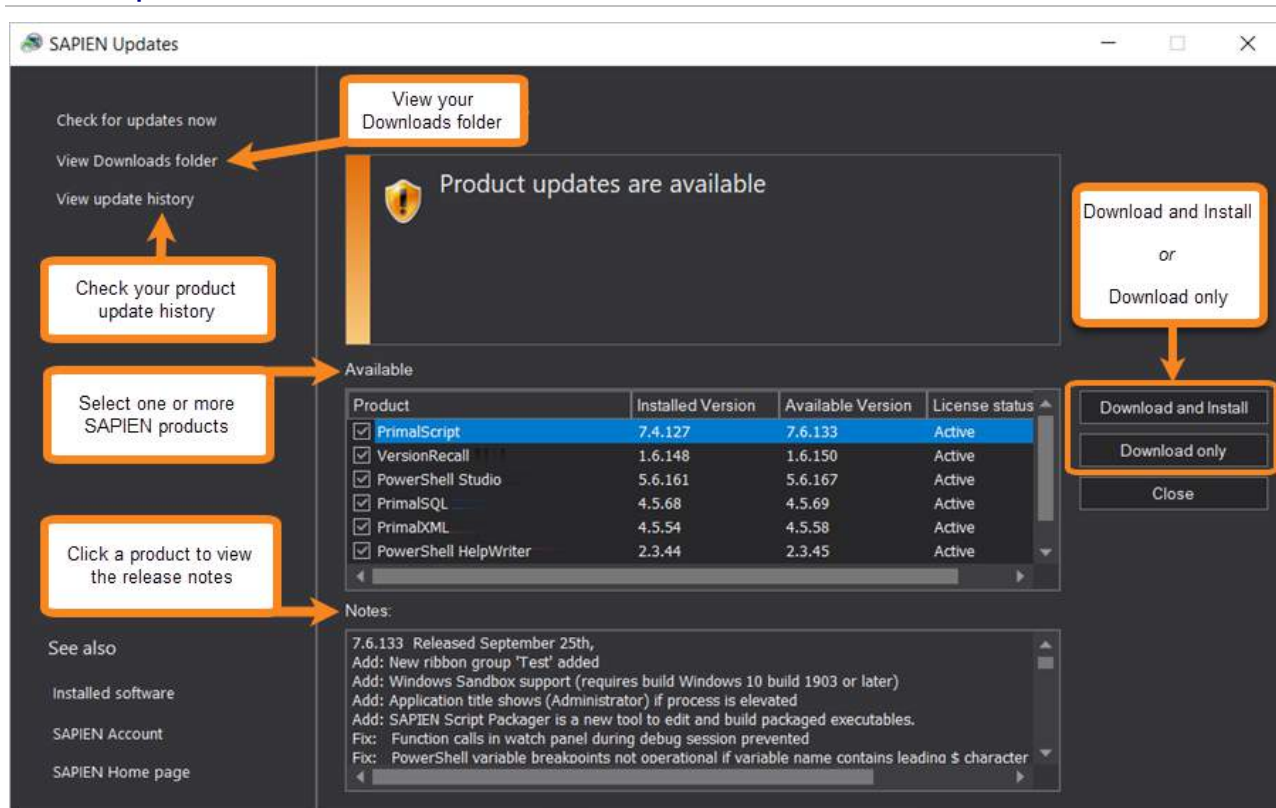
How to access the SAPIEN Updates tool

- On the **Help** or **Tools** ribbon > click **Check Now** or **Check For Updates** (Updates section).

-OR-



- Click the [notification icon](#) ⁷⁴ above the ribbon > then in the Notifications dialog, click the update notification.

SAPIEN Updates Tool



SAPIEN Updates Tool

SAPIEN Updates - Options

Check for updates now	Immediately checks to see if additional product updates are available.
View Downloads folder	Displays the Downloads folder in File Explorer.
View update history	Displays the history of all downloaded and installed product updates.
Available	<p>Displays a selectable list of available product updates.</p> <p> Select one or more products to Download or Download and Install.</p>
Download and Install	Downloads and installs the updates for the product(s) selected in the Available updates list.
Download only	Downloads the updates for the product(s) selected in the Available updates list.
Close	Closes the SAPIEN Updates tool.
Notes	<p>Displays a brief synopsis of what was changed, added, or fixed for the products selected in the Available window.</p> <p> The build history for all SAPIEN products is available here.</p>

Update On-Demand

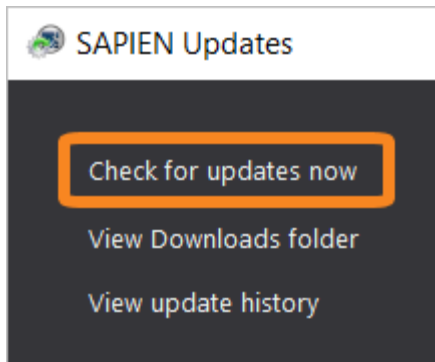
You don't need to wait to be notified when an update is available; you can check for updates at any time. This is particularly useful if you've heard about a new update and want to install it immediately, or if you are ready to start a new project and want to complete all updates before you begin.

How to check for updates on-demand

- On the **Help** or **Tools** ribbon > select **Check Now** or **Check For Updates** to open the SAPIEN Updates tool.

 These instructions may vary between SAPIEN products.

- In the SAPIEN Updates tool, select **Check for updates now**:



The latest product updates are displayed in the SAPIEN Updates **Available** window.

Security and Permissions

Installing updates to programs in a Program Files directory requires the permissions of a member of the Administrators group on the computer. When you click **Download and Install** in the [update tool](#)^[75], or install after downloading, you will be prompted for administrator credentials.

The update tool requires a functioning internet connection and unimpeded access through your internet firewall. For some installations, you might need to create a firewall rule to allow access or make some accommodations.

8 Appendices

Appendices for PowerShell HelpWriter Help Manual

[Appendix A: Manual and Product Version](#)  ⁷⁸

[Appendix B: Icon License Attribution](#)  ⁷⁹

8.1 Appendix A: Manual Version

Appendix A Manual Version

This help manual is in the process of being updated. Some features and images in this manual version may not reflect the current product functionality.

Blog articles

For the latest product tips and feature demonstrations, check out the PowerShell HelpWriter articles on the [SAPIEN blog](#).

Release details

To view a brief description of what was changed, added, or fixed in the most recent PowerShell HelpWriter builds, view the product [version history](#).

Need more help?

Please direct your product related questions to the [PowerShell HelpWriter support forum](#), and your scripting questions to the appropriate [Scripting Answers forum](#).

8.2 Appendix B: Icon License Attribution

Appendix B Icon License Attribution

Icons used in this manual are licensed under [Creative Commons Attribution 3.0 Unported \(CC BY 3.0\)](#).

Icons made by [Icomoon](#) from [www.flaticon.com](#) are licensed under [CC 3.0 BY](#):



[Thumb up](#)



[Information](#)



[Warning](#)
