PrimalScript - Help Manual

1 Welcome to PrimalScript

Script and code editing has never been easier!

Welcome to PrimalScript, the industry's most mature, feature-filled script and code editor. Focused firmly on the needs of systems administrators, Web developers, and others working with script and other advanced languages, PrimalScript provides everything you need to become more efficient and more effective in your scripting and coding efforts.

About this documentation

This help is designed to show you how to use PrimalScript—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 PrimalScript.
- Get a quick overview of the user interface 19 and see how to customize your workspace 21.
- Learn how to use the powerful PrimalScript Editor 43.
- Visit the support forum to get help from SAPIEN staff and other experienced PrimalScript users.

2 Introduction

This section provides an overview of the PrimalScript features, shows you how to purchase directly on-line or through a reseller, and lets you know how to get answers to your questions.

2.1 About PrimalScript

PrimalScript is the leading Universal Scripting IDE for all your administrative and web-development tasks.

Key Features

- Supports over **50 languages and file types**.
- Supports **32-bit and 64-bit** platform development.
- Next generation **PowerShell** local and remote debugger.
- Supports **PowerShell V2, V3, V4 and V5** at the same time.
- Script against a remote machine's **Installed Module Set** (IMS).
- **Remote** VBScript, JScript and PowerShell debugger.
- For a complete list of current features, visit the PrimalScript product page.

What's New

We are always updating and improving PrimalScript. You can learn about the latest product updates on our blog and in the release build log.

- Check out the latest PrimalScript tips and product feature demonstrations on the SAPIEN blog.
- View a brief synopsis of what was changed, added, or fixed in the most recent PrimalScript build in the product version history.
- Submit feedback and suggestions.

2.2 How to Buy PrimalScript

You can buy PrimalScript online with all major credit cards. As soon as your transaction completes, you will be able to download and install 8 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 PrimalScript product page

Online orders:

https://www.sapien.com/store/primalscript

Worldwide authorized resellers:

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

PrimalScript product page:

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

3 Getting Started

This section shows you how to download and install PrimalScript, how to keep the application updated with the latest builds, and how to find additional help.

3.1 Installing PrimalScript

To get started using PrimalScript, follow the instructions below to download and install the program. It is also a good idea to review the security and firewall considerations.

Downloading PrimalScript

All SAPIEN Technologies software products are downloadable only. Download registered products from your SAPIEN Account Registered Products page.

Select the 64-bit version of PrimalScript to download. The installer software will save to your default download folder (e.g., *PSR23Setup_8.1.184_062123_x64.exe*).

Starting with the PrimalScript 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 PrimalScript

Follow these instructions to install PrimalScript.

How to install PrimalScript

1. In your default download folder, double-click on the downloaded program (e.g., *PSR23Setup_8.1.184_062123_x64.exe*).
2. Reply **Yes** to the "*Do you want to allow this app to make changes to your device?*" prompt.

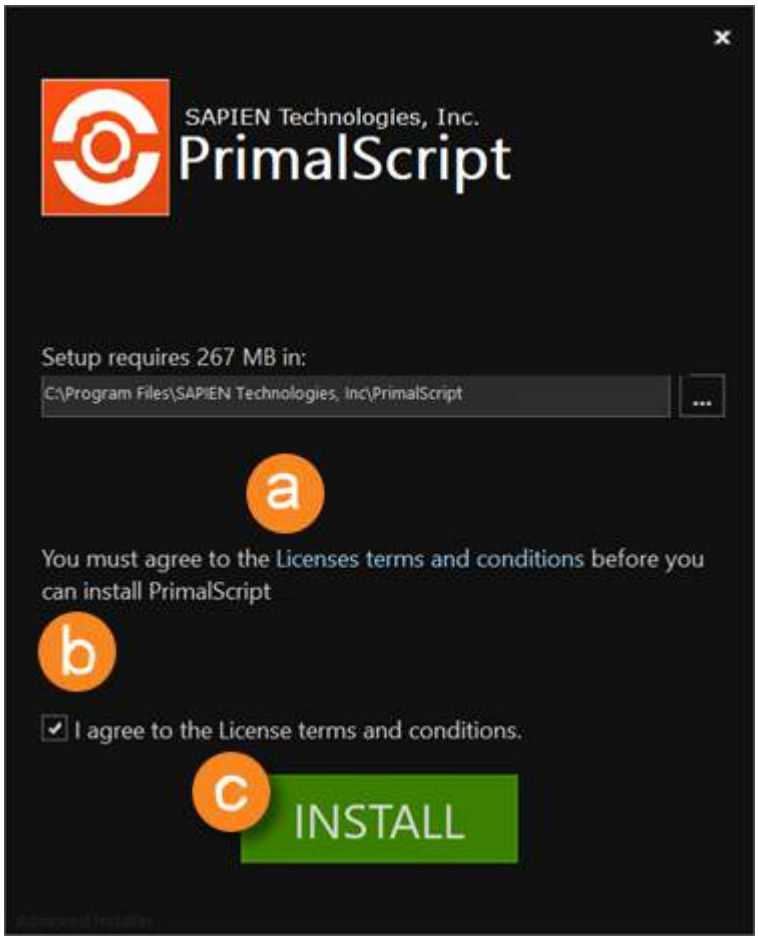
   The installation wizard will first check several items, such as available disk space and the presence of previous builds. If the environment is adequate, the installer will display the legal agreement which you must accept to proceed:

   a. **Read** the terms of the license agreement.

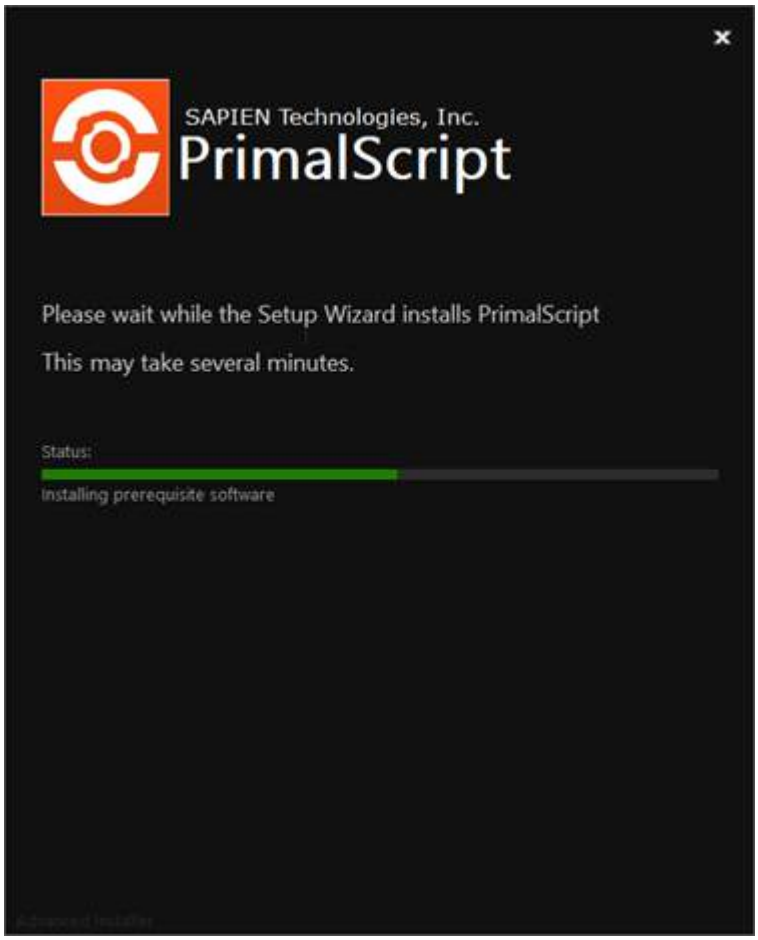
   b. **Accept** the terms of the license agreement. You should never accept license terms unless you have read them, and you understand them.

   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 PrimalScript successfully completes the installation, click **Finish**.

To install PrimalScript on Windows 7, Windows Server 2008, and earlier versions of Windows, install PrimalScript by right-clicking the installer package and selecting **Run as Administrator**. This ensures that PrimalScript installs with the privileges it requires to properly configure advanced components, such as the Remote Script Execution Engine 195.

If a previous version of PrimalScript is open, you are prompted to exit. If you don't exit, PrimalScript restarts after the installation is complete.

Silent Installation

Use this command if you need to install silently:

```
PSRxxSetup_x.x.xxx_xxxxxx_x64.exe /exenoui /qn
```

Example:

```
PRS21Setup_8.0.152_072621_x64.exe /exenoui /qn
```

Troubleshooting Installation

If you encounter problems installing PrimalScript, please report them in the Installation Issues support forum.

Use these Installer Log parameters to output to a log file: `Installer.exe /exenoui /qn /L*v .\PSR_Install.log`

Security

You might need Administrator privileges to install PrimalScript if you plan to add local firewall exceptions for the Remote Script Execution Engine 195 or the ASP Debugger.

Other PrimalScript features require the permissions of a member of the Administrators group on the computer, including:

- Enumerating WMI namespaces and classes (for the WMI Explorer and WMI Wizard)
- Debugging scripts
- Changing the Windows PowerShell execution policy (Set-ExecutionPolicy)
- Changing the Windows Script Host trust policy
- Scanning for new command-line tools
- Accessing links in the Info Browser

This is only a partial list; other features might also require administrative access.

When running PrimalScript on Windows 7 and earlier versions of Windows, User Account Control (UAC) can prevent some features from installing or running correctly. On these systems, we recommend that you configure PrimalScript to run as administrator.

To run PrimalScript as an administrator (one time)

- Each time you open PrimalScript, **right-click the PrimalScript icon** or executable and select **Run as Administrator**.

To run PrimalScript as an administrator (every time)

1. **Right-click the PrimalScript icon** or executable, and then select **Properties**.
2. Select the **Shortcut** tab.
3. Click **Advanced**.
4. Select the **Run as administrator** checkbox.
5. Click **OK**.

When the configuration is complete, Windows displays a UAC warning or an authentication dialog box each time you start PrimalScript.

Firewall Considerations

The following PrimalScript actions might trigger a firewall warning:

- PrimalScript installs a service that supports the Remote Script Execution Engine. When the RSEE service attempts to open the port on which it listens, a firewall warning is triggered. For more information, consult the chapter on the Remote Script Execution Engine 195.
- To ensure that your copy of PrimalScript is current, PrimalScript checks a text file on the SAPIEN.com web site that contains the current PrimalScript version number. Your firewall might warn you when PrimalScript attempts to read this file for the first time. PrimalScript does not transmit any personally-identifiable information when checking this file.
- PrimalScript also accesses the Web to display its product registration page (after the initial installation) and to display Web pages when you click on links in the Info Browser.
- If you configure PrimalScript to use ASP Debugging (see the Scriptable COM Components chapter), your firewall software might alert you when PrimalScript initially opens the port required for debugging communications.

Command-Line Interface

You can start PrimalScript from the command-line in Cmd.exe or Windows PowerShell.

Switch	Description	Example
PrimalScript *filename*	Opens the specified file.	`C:\> <path>\PrimalScript.exe C:\Scripts\Test.asp` `PS C:\> & <path>\PrimalScript.exe C:\Scripts\Test.asp`
PrimalScript *filename* /l *line*	Opens a file to the specified line.	`C:\> <path>\PrimalScript.exe Test.asp /l6` `PS C:\> & <path>\PrimalScript.exe .\Test.asp /l6`

Scanning for Tools

When PrimalScript runs for the first time, it scans your system for scripting-related tools and adds them to the Tools Browser.

To add additional tools to the Tools Browser after the one-time scan

- Right-click anywhere in the **Tools Browser** and click **Scan for Tools**.

To customize the Tools Browser, including reorganizing the groups and adding and removing tools

- Right-click anywhere in the **Tools Browser** and click **Customize**.

Activating and Deactivating PrimalScript

Software activations are outlined in our End-User License Agreement. The number of activations allowed will differ depending on your type of license. For Perpetual Licenses, each licensed user is allowed to have a maximum of two devices activated and operating at any given time for personal use. For Subscriptions*, each licensed user is allowed to have the software activated on a total of 20 devices with a maximum of two devices operating simultaneously at any given time for personal use.

Product Activation

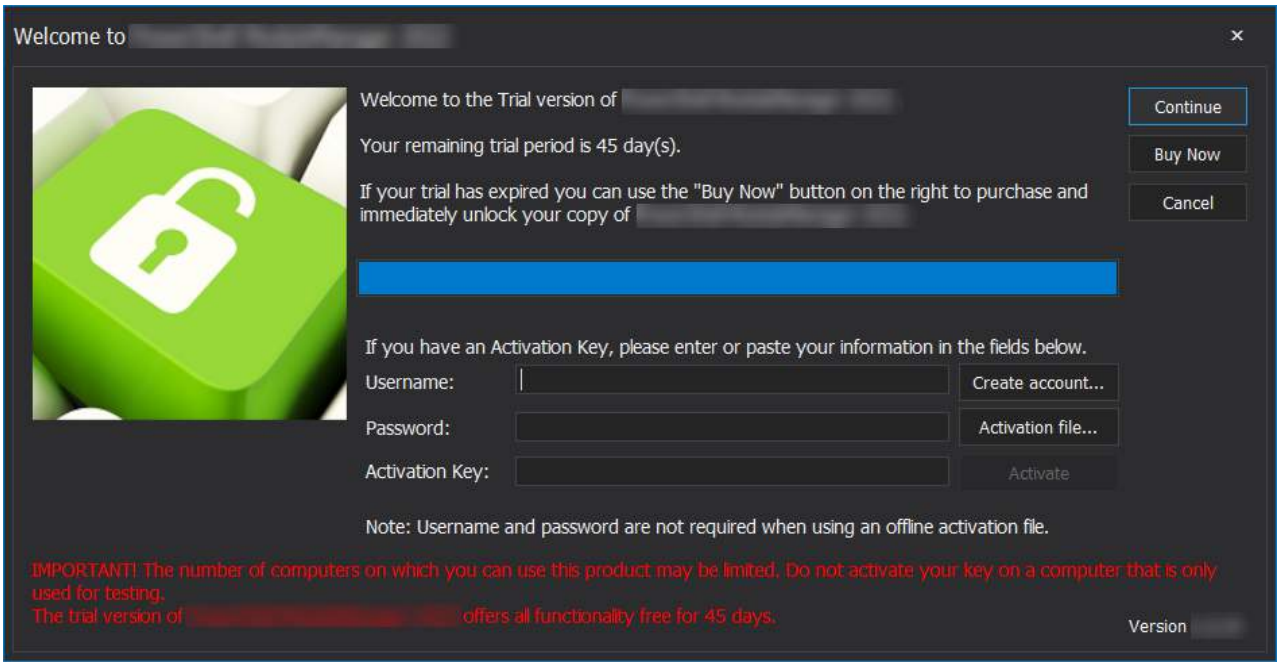
Registration is required to activate and operate the product, and also to obtain any customer service or technical support benefits. Registration only takes a few moments to complete and provides you with access to special offers including preferred pricing on renewals. *You will need an active internet 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. Please note: Activation keys belonging to Subscriptions* are not eligible for Offline Activation.

* *Information about software activations for Subscriptions only applies to SAPIEN Technologies products with a Subscription purchase offer.*

To activate PrimalScript

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

Removing a software activation, also known as "deactivating", allows you to free up an activation for use on another device.

You may deactivate your devices to free up your activations at your leisure, but there are certain circumstances where proper deactivation is crucial to prevent the loss of your allotted activations 14.

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

To deactivate your activation key

In the top-right of PrimalScript above the ribbon, click the Activation Information 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 PrimalScript, 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.

The details for every PrimalScript release are available in the version history.

Check for Updates

By default, PrimalScript 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 Now** to open the SAPIEN Updates 226 tool and see if there is a new PrimalScript build available:

3.3 Getting Help

This help manual has been designed to provide all the information you will need for using PrimalScript. In addition to the information in this guide, you can also ask questions in the online support forums 17.

View PrimalScript product feature demonstrations and release details on our blog.

Accessing the help manual

To view the help manual online

- In PrimalScript, on the **Help** ribbon > in the **Product Help** section, click **PrimalScript Manual**.

The SAPIEN Information Center provides direct access to all of the SAPIEN product manuals.

User forums and support

SAPIEN Technologies provides a variety of ways to get help with PrimalScript, including community support forums for your scripting questions.

Support Options

Every registered PrimalScript perpetual or subscription license with active maintenance includes basic support in our PrimalScript product support forums.

If your PrimalScript maintenance has expired, you must renew in order to obtain support.

Premium Support

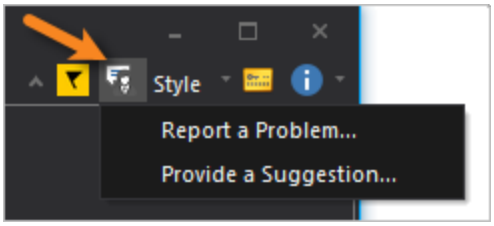
SAPIEN also offers Premium Support, an elevated support option, at an additional cost. Premium Support gives you access to our direct technical ticketing system and guarantees a response within 24 hours, as well as personalized attention until the issue is resolved.

Support Forums

SAPIEN provides support forums where our development team answers user questions. Our support technicians monitor the forums daily, but response times are not guaranteed.

PrimalScript Forums

The **Send Feedback** menu on the top-right of the ribbon header provides direct links to support options:



- **Report a Problem...**
  Opens the PrimalScript forum where you can report a problem with the software or ask a product-specific question.
  You will need to provide your PrimalScript and OS version information 18 to obtain support.
- **Provide a Suggestion...**
  Opens the Feature Request page on the SAPIEN site where you can make a feature request or suggestion.

Scripting Forums

Our scripting forums offer community support for answers to your scripting questions.

- **Windows PowerShell**

See the Windows PowerShell board in the Scripting Answers forum.

- **Scripting - including UI scripting**
  See the Scripting Answers forums.

How to copy version information

To report a problem in the PrimalScript 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 PrimalScript workspace:

2. In the About PrimalScript window, click **Copy Version Info.**
3. Paste the version information into your PrimalScript forum post.

4 Basic Orientation

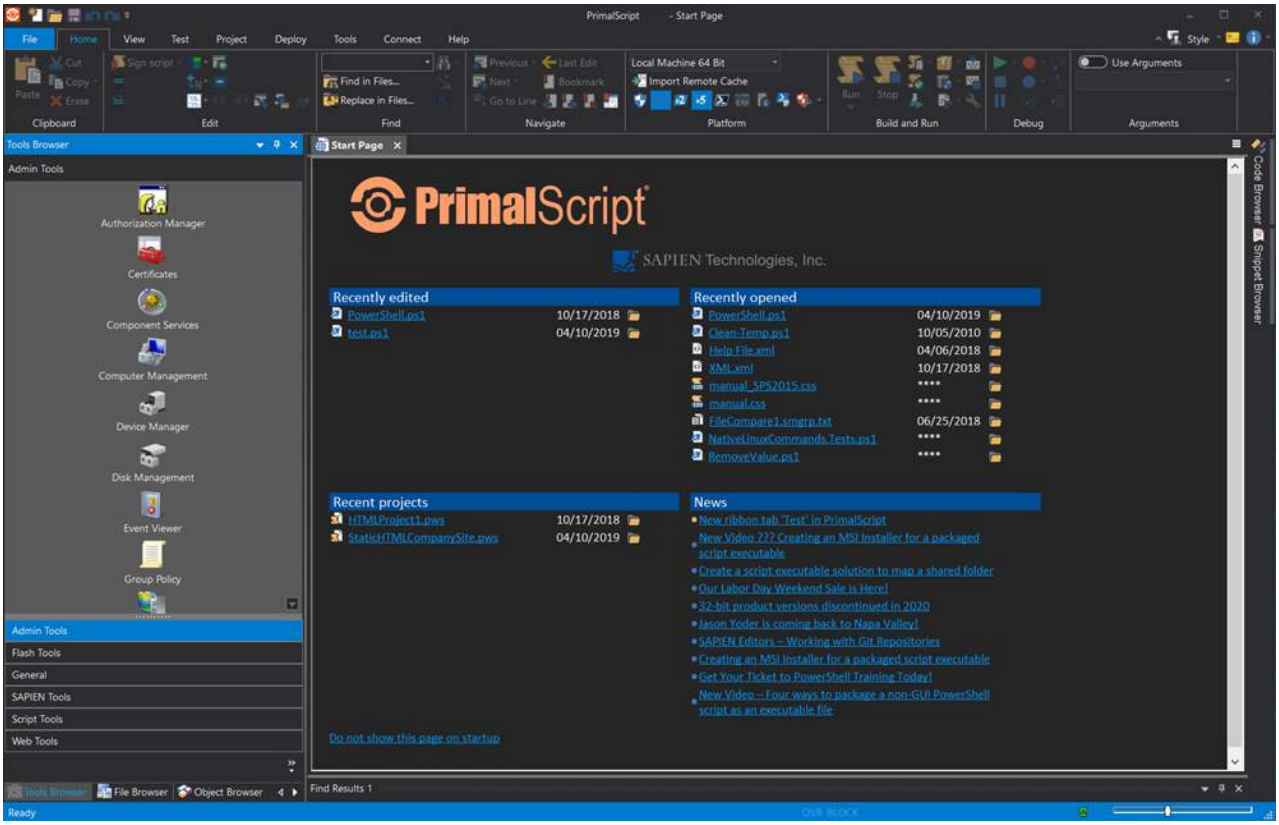
This section provides a brief overview of the main user interface elements, shows you how to customize your workspace, and covers PrimalScript templates and template variables.

4.1 User Interface

Before getting started with PrimalScript we recommend that you take a moment to review some of the main user interface elements.

Start Page

When you start PrimalScript, the Tools Browser appears on the left and the Start Page opens as a document. This is the initial default view, which can be easily customized.



PrimalScript Program Window - Start Page

The Start Page lists recently edited files, recently opened files and projects, and the latest news feeds from the SAPIEN blog.

To prevent the Start Page from opening when PrimalScript starts

- At the bottom of the Start Page, click **Do not show this page on startup**.

To restore the Start Page on startup

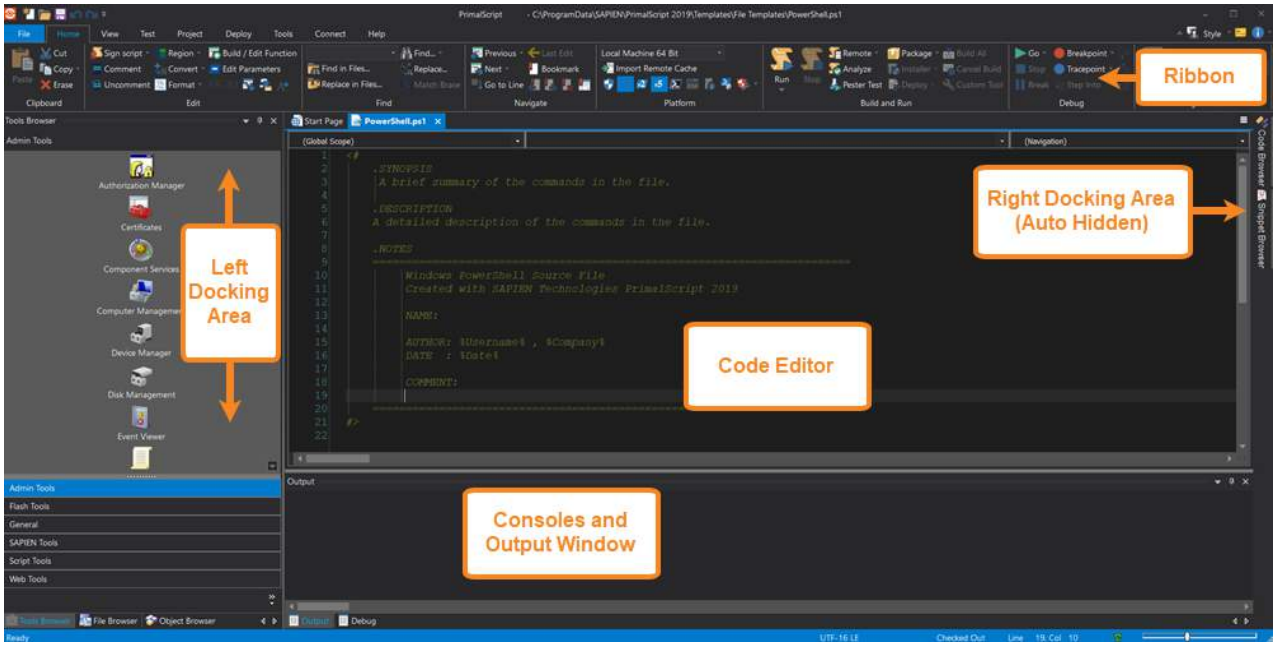
- Click **File** > **Options** > **Application** > **General** > and check (or clear) **Show Launch page on startup**.

To display the Start Page (without changing startup settings)

- Click **View** and, in the Other section, click **Start Page**.

Editing Environment

This figure shows the main user interface elements:



PrimalScript User Interface - Main Elements

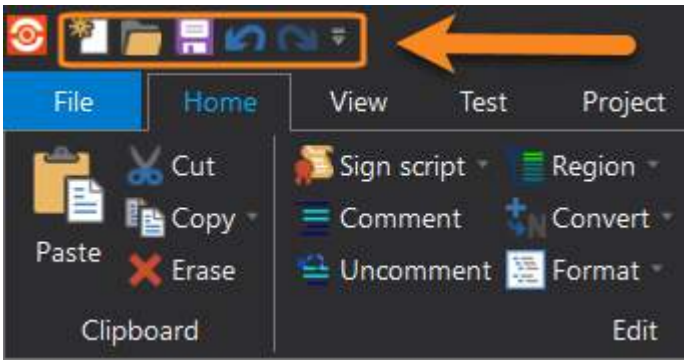
Because PrimalScript displays certain user interface elements according to Windows visual themes, PrimalScript might look slightly different on your system.

The components of the PrimalScript editing environment can be resized or hidden 34.

To maximize the entire PrimalScript program window in your display, double-click the top title bar. Double-click the title bar again to restore the window down to the previous size.

Quick Access Toolbar

The Quick Access Toolbar on the top-left of the program window provides direct access to frequently used functions:



Quick Access Toolbar

You can customize the toolbar 23 by adding and removing controls, and you can also choose to show the toolbar below the ribbon.

4.2 Customizing Your Workspace

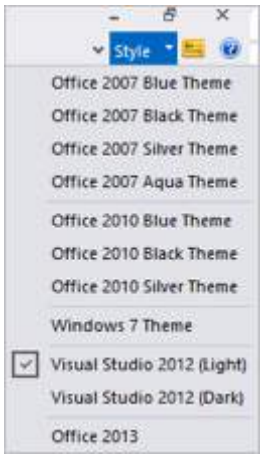
The PrimalScript workspace can be easily customized to suit your personal preference.

4.2.1 Selecting a Style

A style is a visual layout, skin, or theme. You can save your style settings and share them with others.

To change the style for PrimalScript

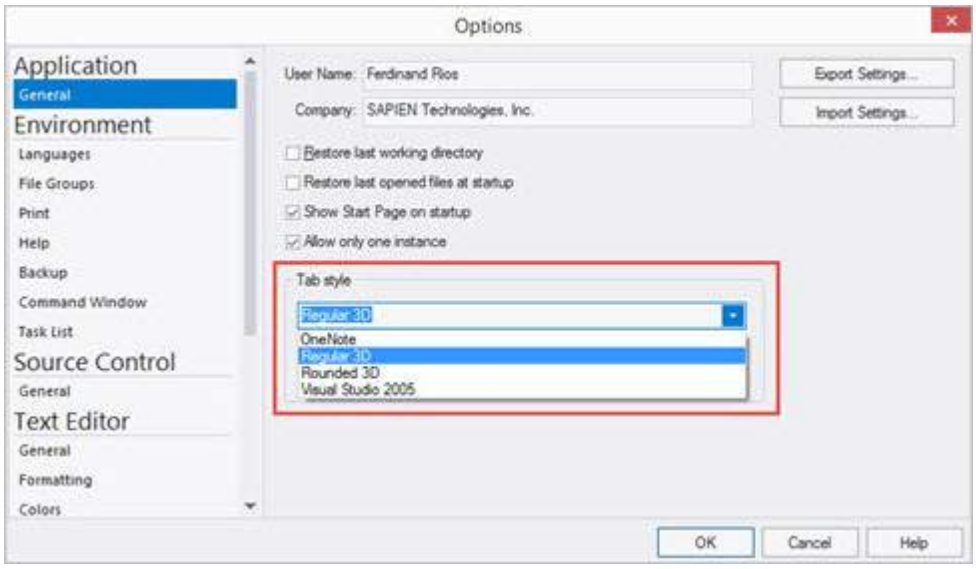
- In the upper right corner, click Style and then click a style:



The default style is Visual Studio 2012 (Light).

To change the tab style

1. Click **File** > **Options** > **Application** > **General**.
2. From the Tab Style section, select a style:

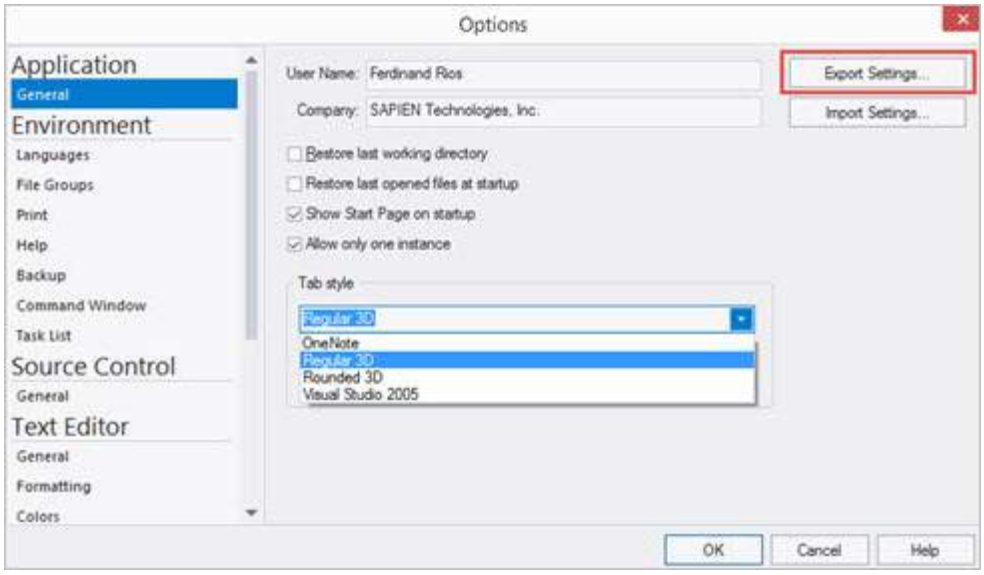


The default tab style is Regular 3D.

To save settings

You can save your settings, including style settings, in an XML file. This allows you to restore your settings, if needed, and share them with others.

1. Click **File** > **New** > **Application** > **General** and then click **Export Settings**:



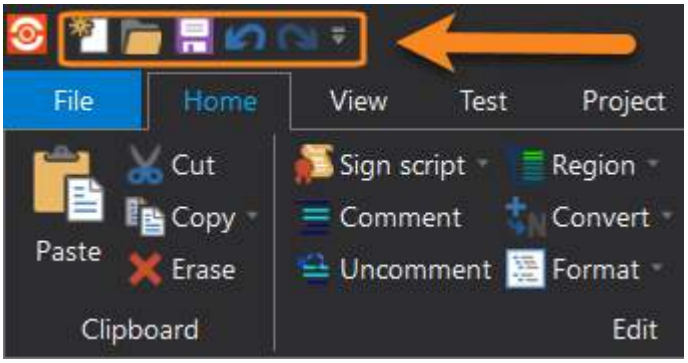
2. Save the settings XML file.

To import or restore settings

1. Click **File** > **New** > **Application** > **General** and then click **Import Settings**.
2. Navigate to the settings XML file and then click **OK**.

4.2.2 Customizing the Quick Access Toolbar

The **Quick Access Toolbar** at the top-left of the program window provides access to your most frequently used tools. This topic explains how to add or remove controls, how to show the Quick Access Toolbar above or below the ribbon, and how to reset the toolbar.



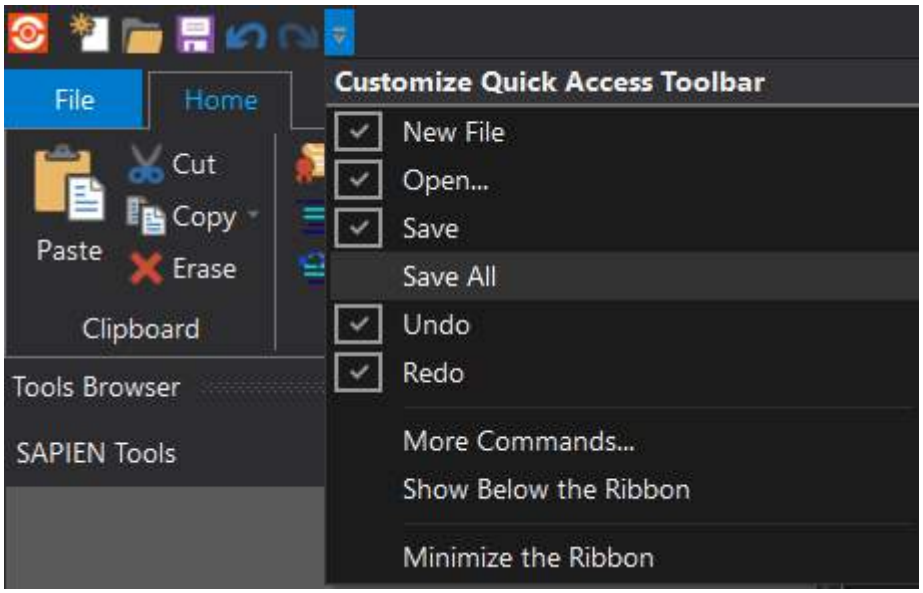
Quick Access Toolbar

Adjusting the Quick Access Toolbar default controls

The Quick Access Toolbar contains a set of default button controls, which you can choose to show or hide.

To show or hide default controls on the toolbar

- Click the **drop-down arrow** on the far right of the toolbar > then **check or uncheck a control**:

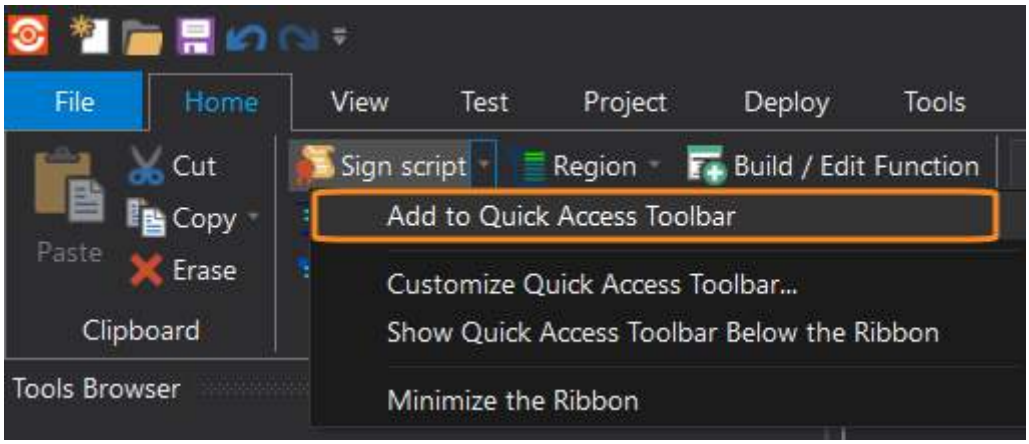


Adding additional commands to the Quick Access Toolbar

You can add additional commands to the Quick Access Toolbar.

To add a new button to the toolbar

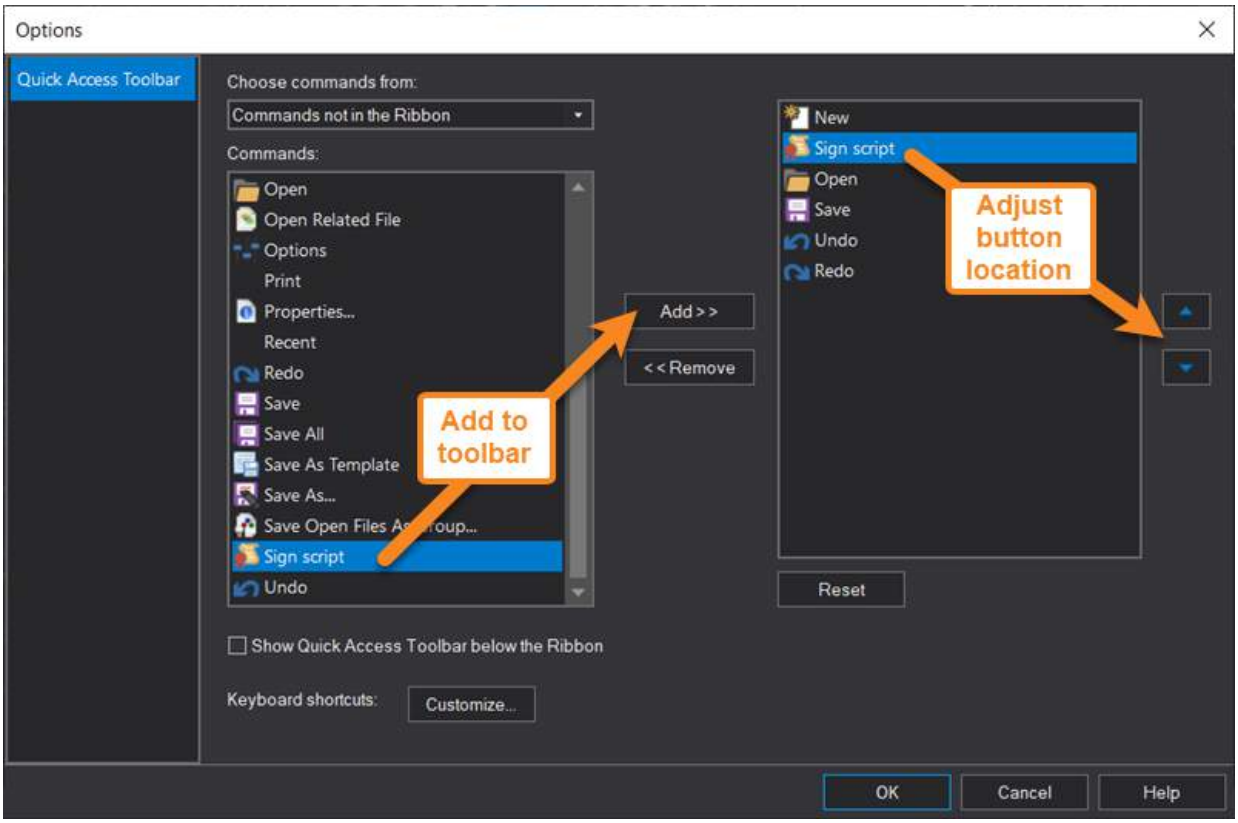
- **Right-click on a ribbon control** > select **Add to Quick Access Toolbar**:



-OR-

1. Click the **drop-down arrow** on the far right of the toolbar > then select **More Commands...**
2. In the Options dialog, locate and **select the command** you want to add, click **Add** > >, then click **OK**.

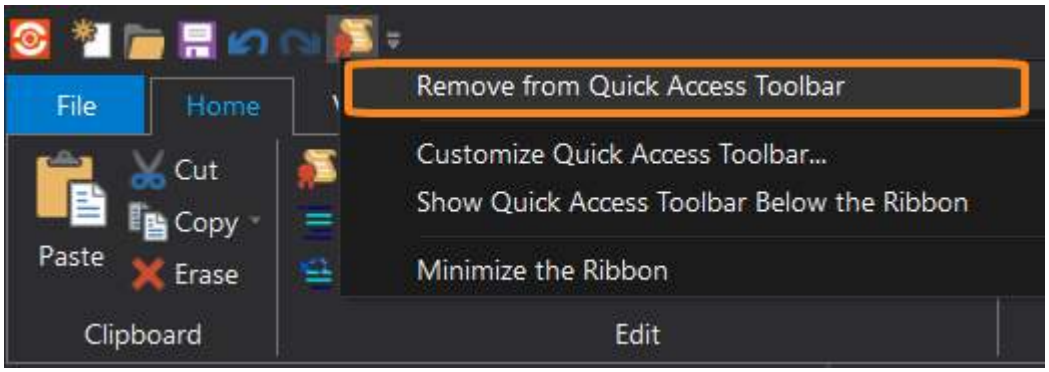
   Click the blue Up or Down arrows to adjust the button location on the toolbar.



The control now appears on the Quick Access Toolbar:

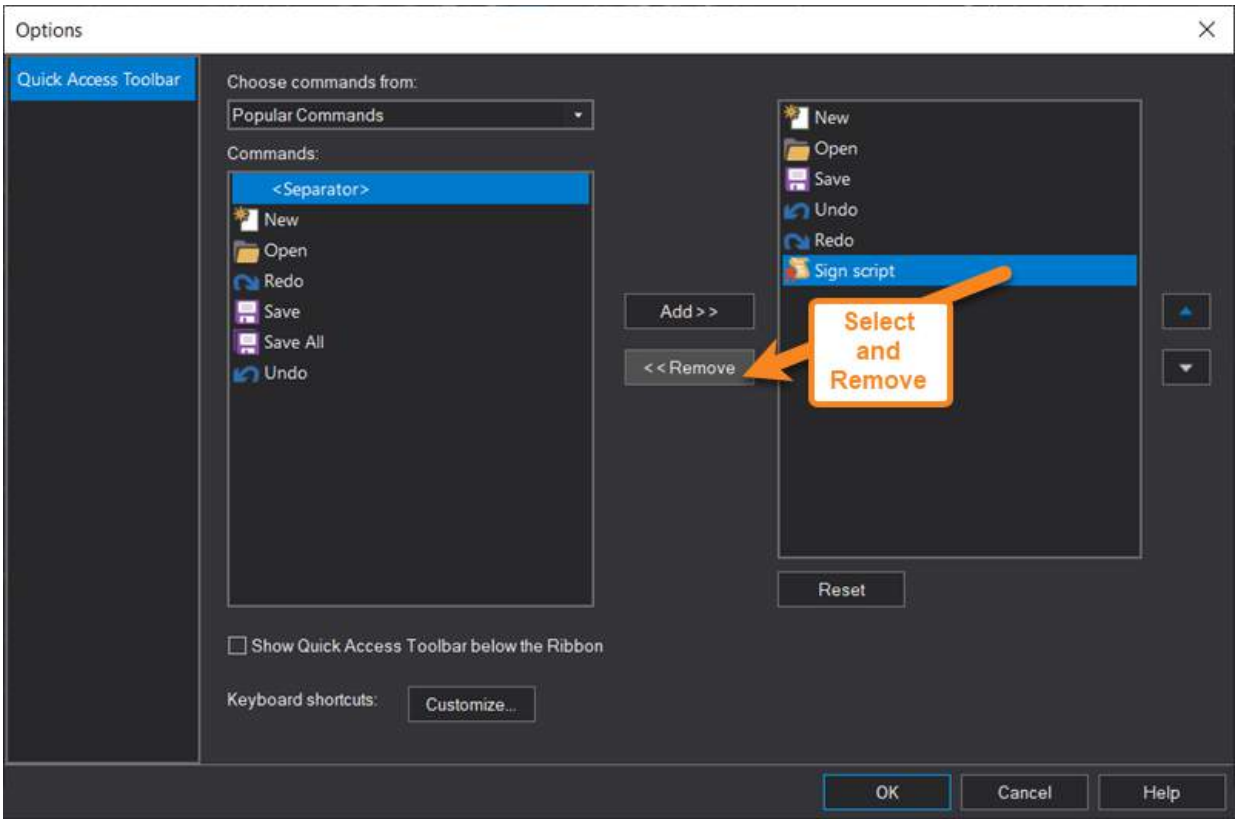
To remove a command from the toolbar

- **Right-click the command on the toolbar** > select **Remove from Quick Access Toolbar**:



-OR-

1. Click the **drop-down arrow** on the far right of the toolbar > then select **More Commands...**
2. In the Options dialog, **select the command** you want to remove, click **Remove** > >, then click **OK**.



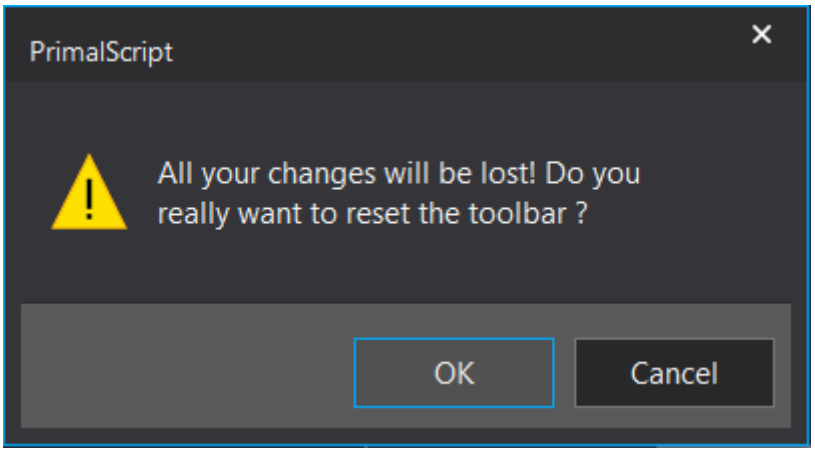
Restoring the Quick Access Toolbar default options

You can reset the buttons on the Quick Access Toolbar to the default options.

To reset the Quick Access Toolbar

1. Click the **drop-down arrow** on the far right of the toolbar > then select **More Commands...**

2. In the Options dialog, click **Reset** > then click **OK** to confirm.



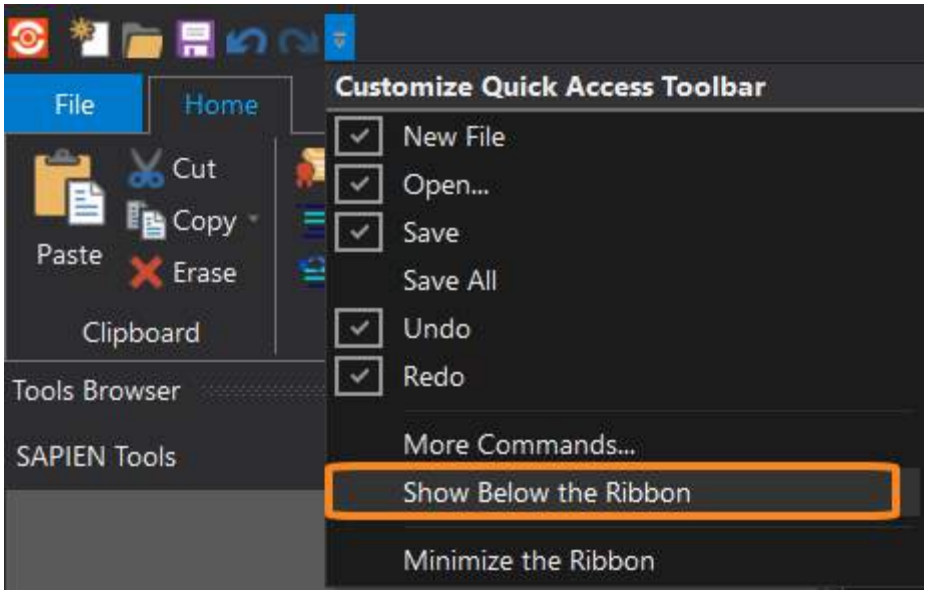
The Quick Access Toolbar is restored to the default commands.

Adjusting the Quick Access Toolbar location

You can adjust the Quick Access Toolbar to appear above or below the ribbon.

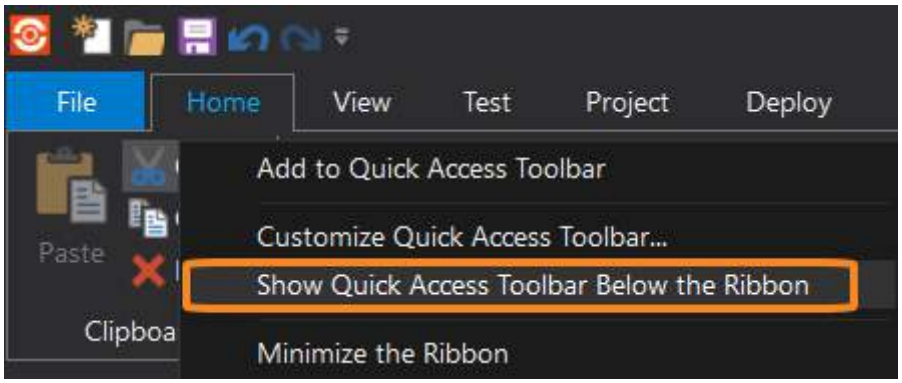
To show the toolbar below the ribbon

- Click the **drop-down arrow** on the far right of the Quick Access Toolbar > then select **Show Below the Ribbon**:



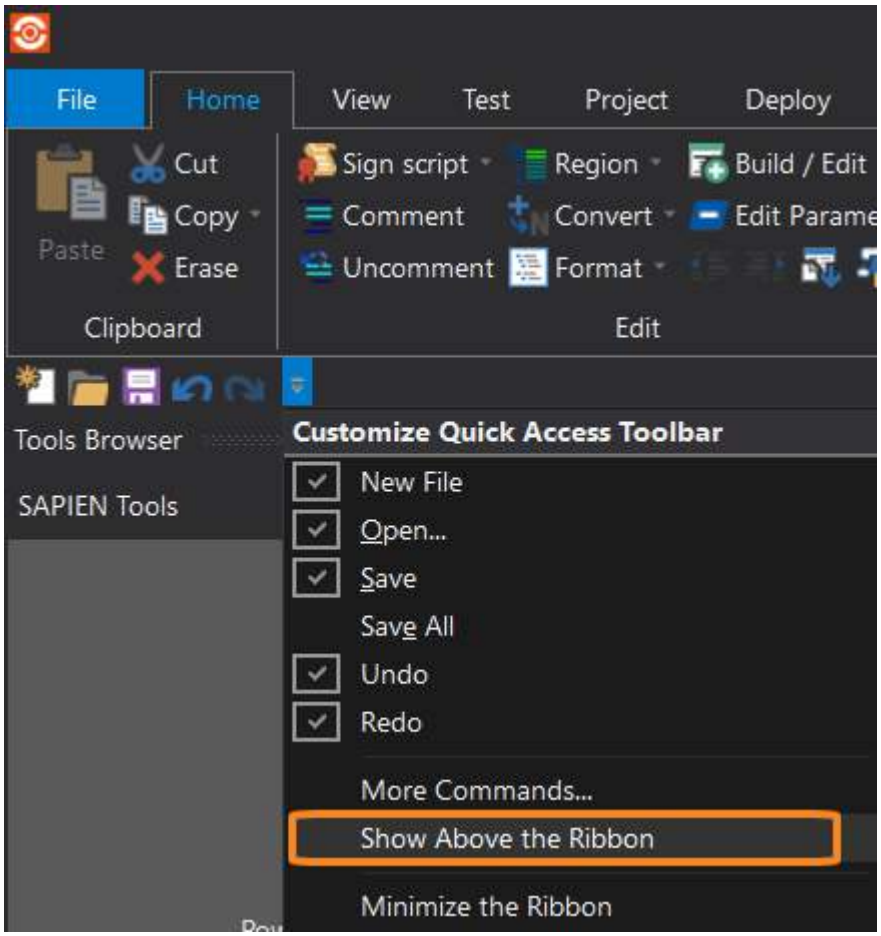
*-**OR**-*

- **Right-click any ribbon control** > then select **Show Quick Access Toolbar Below the Ribbon**:



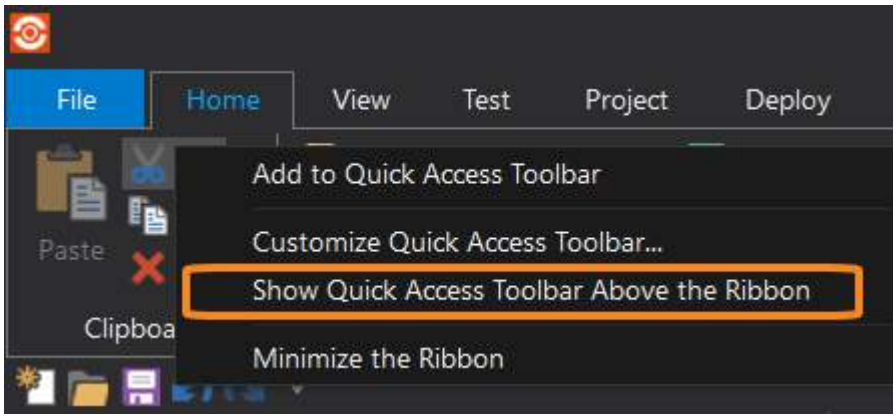
To show the toolbar above the ribbon

- Click the **drop-down arrow** on the far right of the Quick Access Toolbar below the ribbon > then select **Show Above the Ribbon**:



-OR-

- **Right-click any ribbon control** > then select **Show Quick Access Toolbar Above the Ribbon**:



Quick Access Toolbar - Default controls

Default button details (from left to right):

New File (*Ctrl+N*) — Create a new, empty document.

Open (*Ctrl+O*) — Open an existing document.

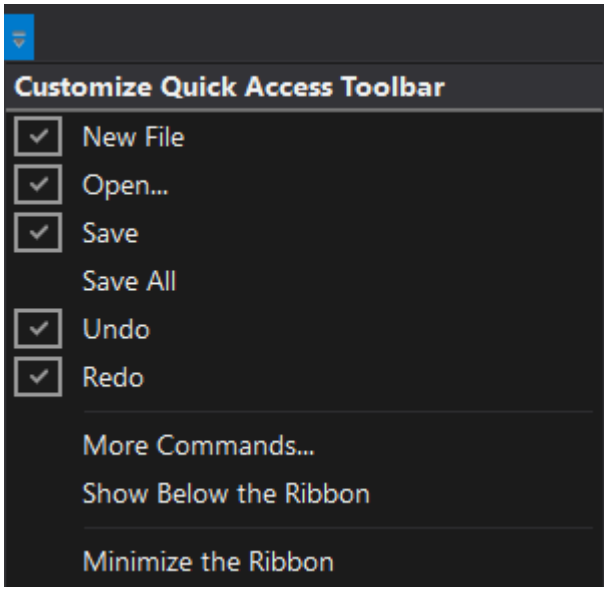
Save (*Ctrl+S*) — Save the active document.

Undo (*Alt+Backspace*) — Undo the last action.

Undo and Redo invoke the PrimalScript infinite undo feature. PrimalScript maintains an extensive history of all file editing actions along with the file. This allows you to undo actions which were made days or even months in the past, even if you have saved the file many times. Undo and Redo work only on NTFS file systems.

Redo (*Alt+Insert*) — Redo the previously undone action.

Customize Quick Access Toolbar — Click the drop-down icon on the right of the toolbar to display the options:



New File
Select or deselect to show the **New File** button on the toolbar.

Open...
Select or deselect to show the **Open...** button on the toolbar.

Save
Select or deselect to show the **Save** button on the toolbar.

Save All
Select or deselect to show the **Save All** button on the toolbar.

Undo
Select or deselect to show the **Undo** button on the toolbar.

Redo
Select or deselect to show the **Redo** button on the toolbar.

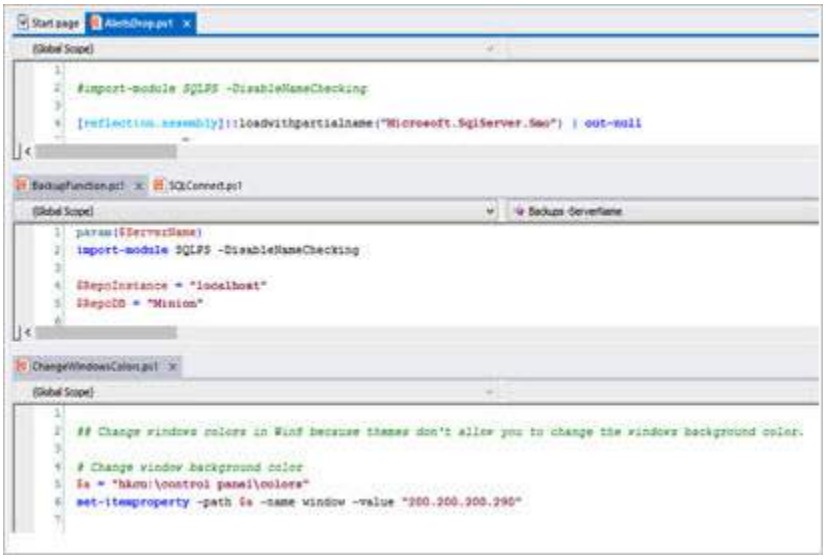
More Commands

4.2.3 Working with Tabs and Panes

You can customize the tabs and panes in the PrimalScript window, and also create file groups.

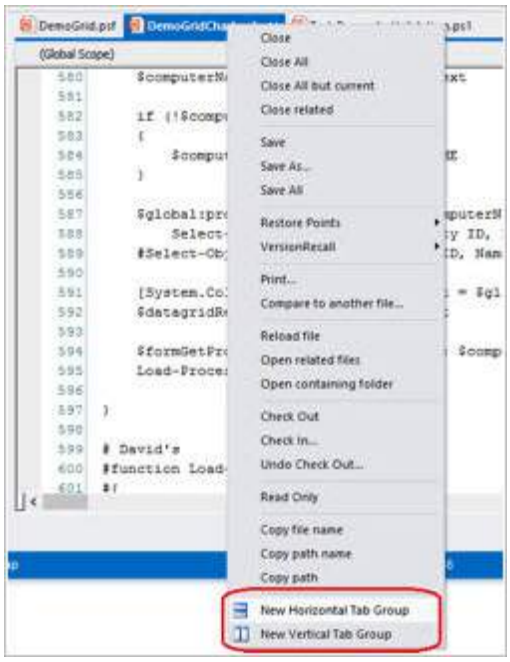
Arrange Files in Tab Groups

You can arrange files in horizontal or vertical tab groups in the PrimalScript window, which makes it easier to view and edit related files.



To create a tab group of files

1. Right-click a file tab and then click **New Horizontal Tab Group** or **New Vertical Tab Group**:



2. To move the files between file groups, click and drag the file tab.

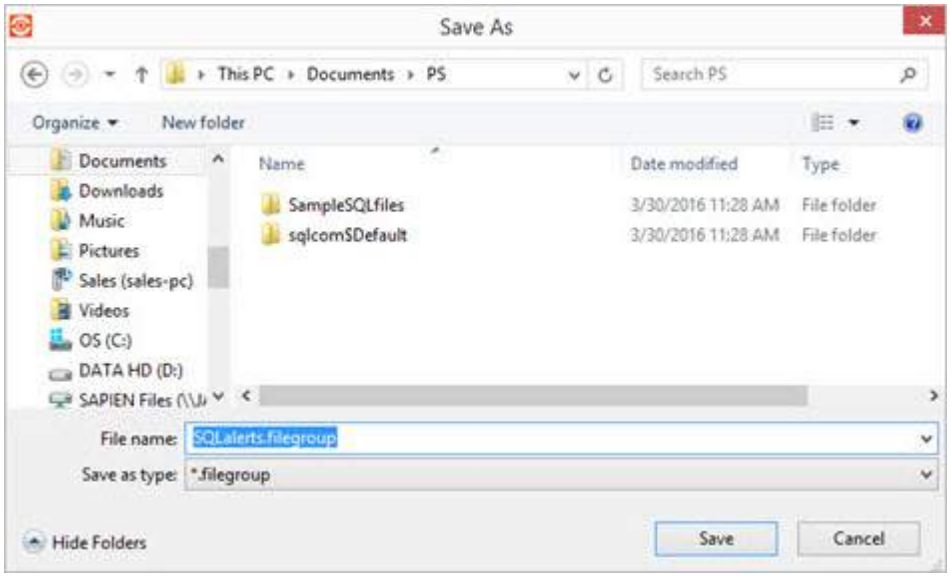
Create File Groups

You can save files in file groups. File groups let you open related files easily in PrimalScript without changing the file type or file structure. This is a great way to manage files that you often view together, such as the files in a Windows PowerShell module.

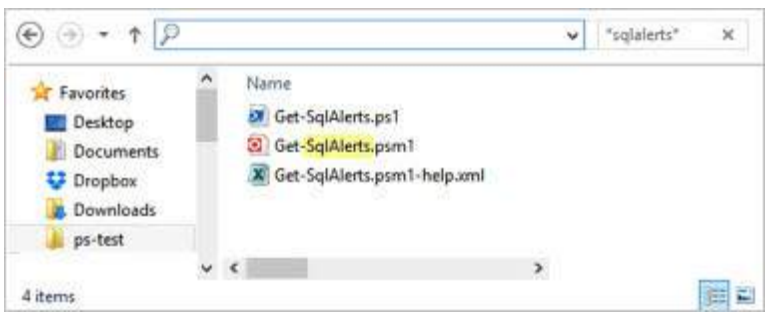
Unlike projects, PrimalScript opens all files in a file group. When you open a project, the related files are available, but not open.

To create a file group

1. In PrimalScript, open the files that you want to group. Close all other files.
2. From the **File** menu, click **Save open files as group**.
3. Name and save the .filegroup file. The .filegroup file does not need to be in the same location as the files in the group:

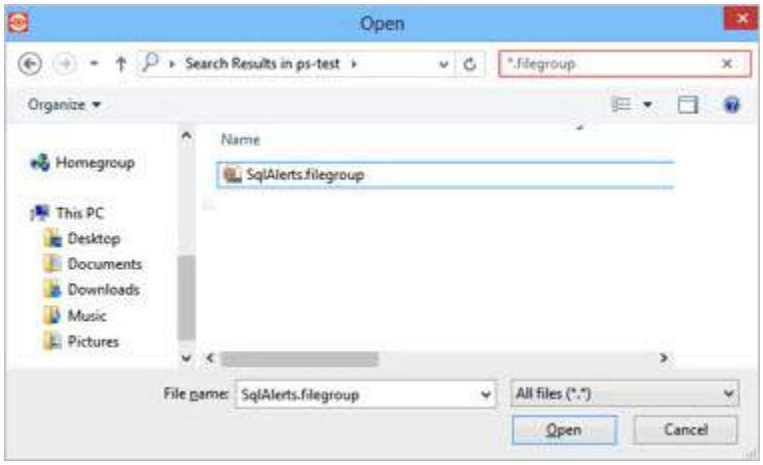


The files in the file group are unchanged:



To open the files in a file group

1. In PrimalScript, click **File** and then click **Open**.
2. Navigate to the **.filegroup** file and then click **Open**:



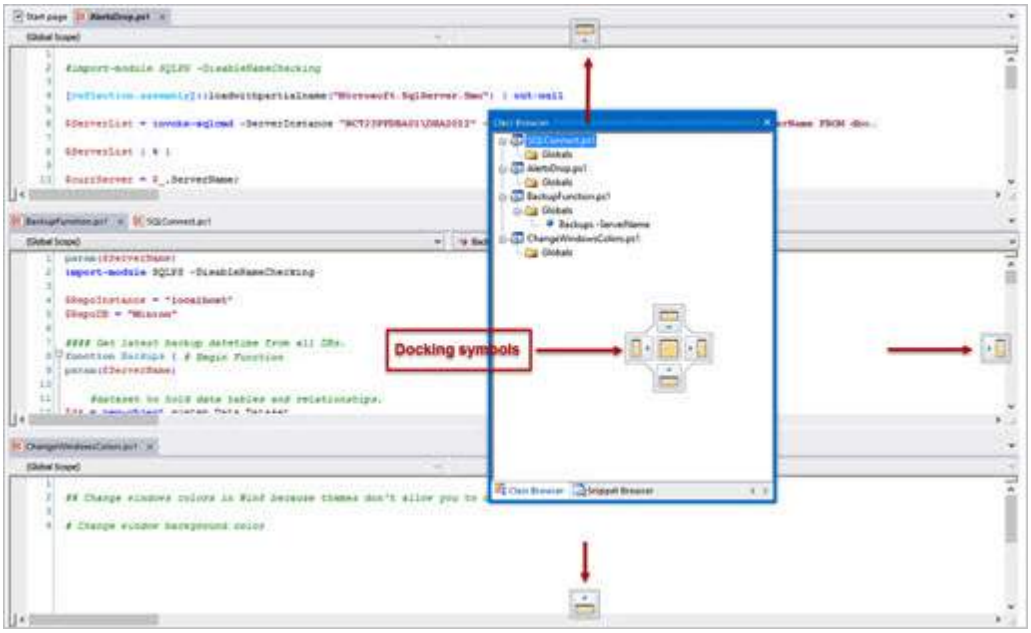
Docking and Undocking Panes

You can un-dock, move, and re-dock panes—allowing them to float, even across multiple monitors—and convert them to tabs.

In PrimalScript, tools appear in panes or tabs; files appear only in tabs.

To un-dock a pane

- Click the title bar and drag the pane to its new location:



-OR-

- Right-click the title bar and then click **Floating**.

To re-dock a pane

- Drag it to one of the docking symbols that appears while you are dragging the pane.

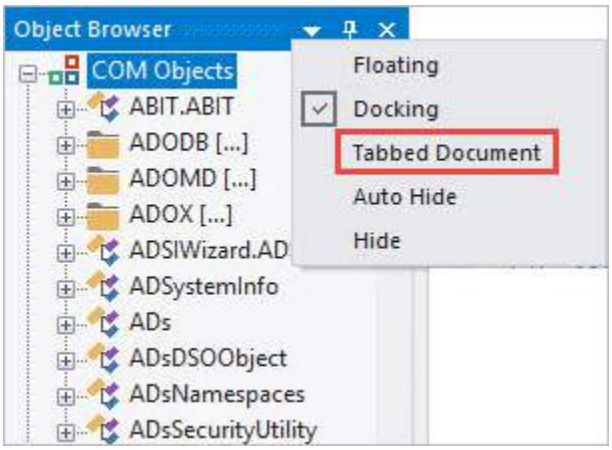
To convert a pane to a tab

- Right-click the title bar and then click Tabbed Document.

To restore a tabbed pane to a standard pane

- Right-click and click Tabbed Document again.

  You cannot convert a file tab to a pane:



4.3 Templates

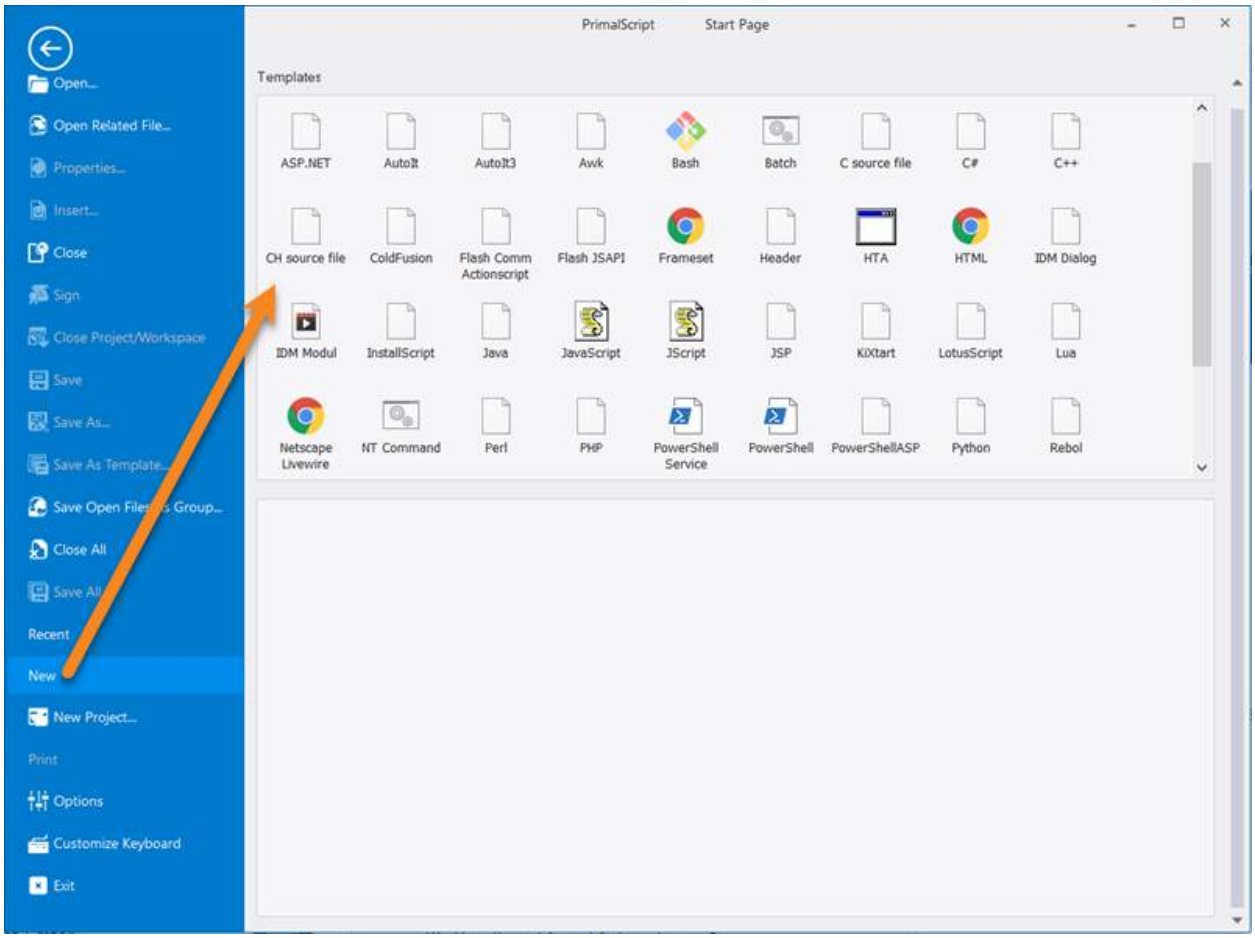
PrimalScript includes pre-defined templates, and allows you to create your own. This section shows you how to work with templates and template variables.

4.3.1 Predefined Templates

PrimalScript includes predefined coding templates so you don't have to start every file from scratch.

To see the pre-defined templates

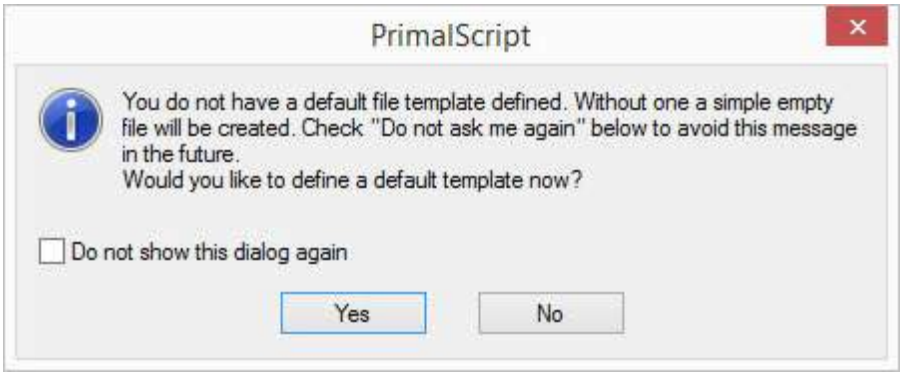
- Click **File** > **New**:



4.3.2 Select a Default Template

When you open a new file by clicking the **New File** icon or by pressing **Ctrl**+**N**, PrimalScript uses your default template unless you specify a different one.

If you have not yet selected a template as your default, when you create a new file, PrimalScript displays this dialog box:



To select a default template

1. Press **Ctrl**+**N**.
2. In this dialog box prompt, click **Yes**.
3. In the template gallery, **double-click** a template.

-OR-

1. Click **File**, **Options**, **Environment**, **Directories** and, in the **Templates** box, select your **Templates** path.
2. In the <***Templates path***>**File Templates** subdirectory, copy a template file.
3. Paste the template file in the <***Templates path***> directory (not in the **File Templates** subdirectory).
4. Change the base name of the file to **Default**. Do not change the file name extension.

For example, to make the JavaScript.js file your default template, copy it from <***Templates path***>**File Templates\JavaScript.js,** and paste it to <***Templates path***>\ **Default.js**

If you have more than one file named Default.* in the <***Templates path***> directory, PrimalScript uses the one that appears first in alphanumeric order.

4.3.3 Change the Templates Directory

You can create custom templates, save them in a shared template directory, and then change the default templates directory to your shared directory.

Default Templates Directory

By default, the templates directory is: **%ProgramData%\SAPIEN\PrimalScript yyyy\Templates\File Templates**.

To change the default directory for PrimalScript templates

- Click **File**, **Options**, **Environment**, **Directories** and enter the path in the **Templates** box.

4.3.4 Create Custom User Templates

You can create a custom template for any supported file type. To simplify the process, edit an existing PrimalScript template. To create different templates for the same language, use different base names and the same file name extension.

For example, you might create a custom template to:

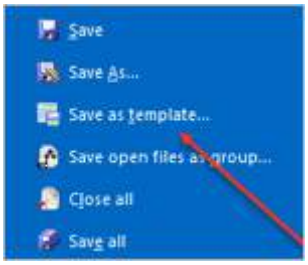
- Add custom header information. If you're a contractor who writes scripts for many different companies, create a template for each one.
- Include common debugging information

- Create sections and standard comments

To create a user template

1. Open any script or template:

2. To save the file as a template, click **File** > **Save as template**:



3. Choose a name and filename extension for the file. Save the file in the **%UserProfile%\AppData\Roaming\SAPIEN\User Templates** directory.

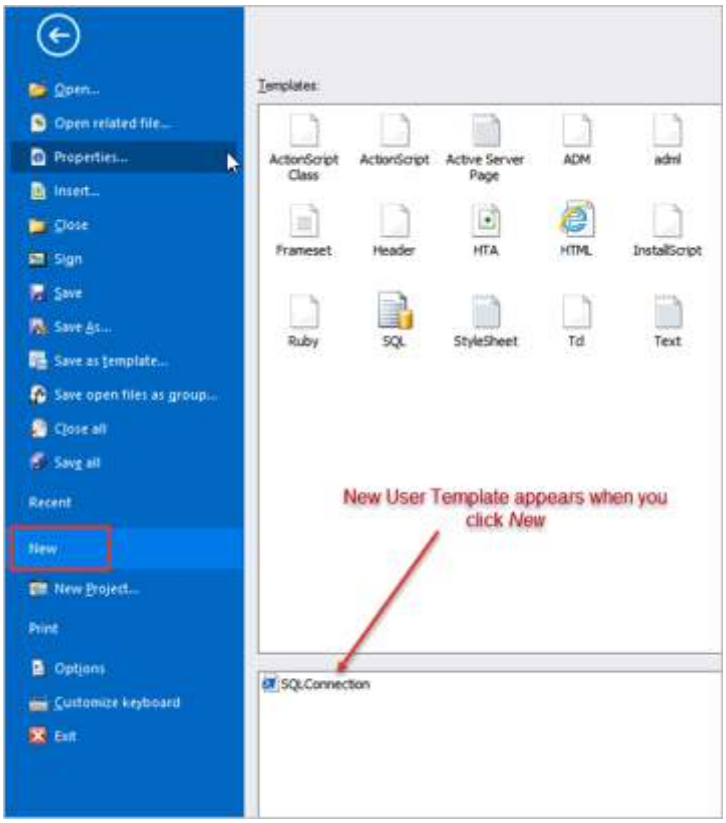
Pay special attention to the file name extension. PrimalScript uses the extension to associate the template with a scripting language:

To select a user template

1. Click **File** > **New**.

   User templates appear in the bottom half of the Templates page.

2. Select the user template:



4.3.5 Template Variables

In addition to text that you type, you can use predefined variables in a template. These variables are expanded and replaced whenever you create a new file from the template.

Predefined Template Variables

Variable	Description
$CARET	Places the cursor at the specified point in the file.
$COMPANY	Value in File\Options\Application\General\Company
$DATE	Current system date
$NAME	Value in File\Options\Application\General\User Name
$TIMEDATE	Current system date and time
$TIME	Current system time

You cannot define additional variables or include Windows environmental variables in your template.

Snippet-specific Variables for Templates

Use the following variables in template snippets.

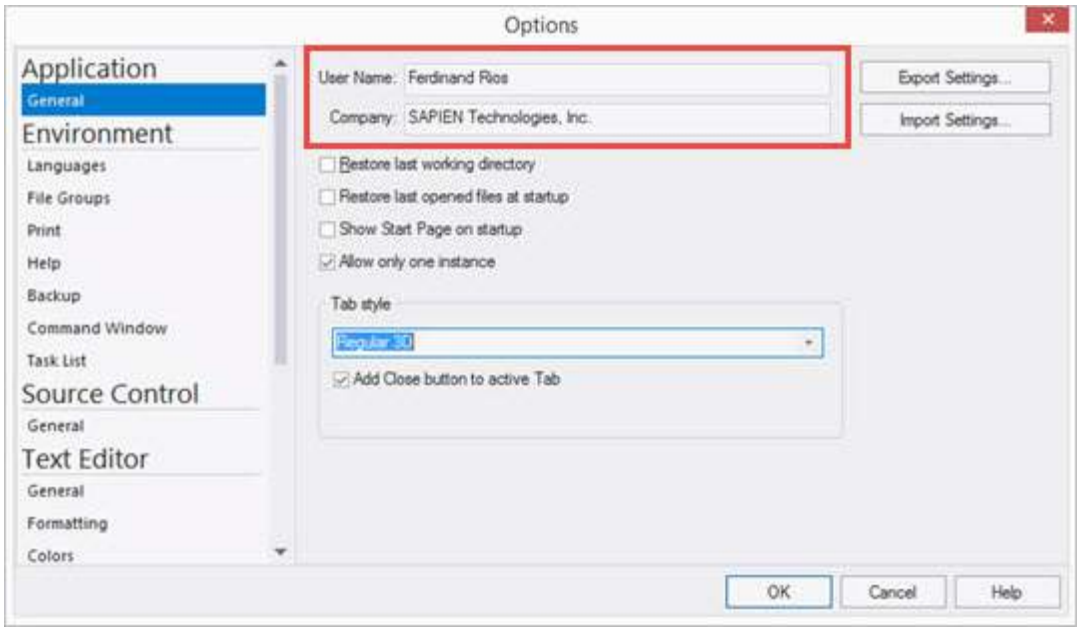
Variable	Description
$SELECTION	Replaced by the currently selected item.
$STARTSEL	Marks the beginning of the selection AFTER the snippet insert.
$ENDSEL	Marks the end of the selection.

Specify Default Name and Company Variables

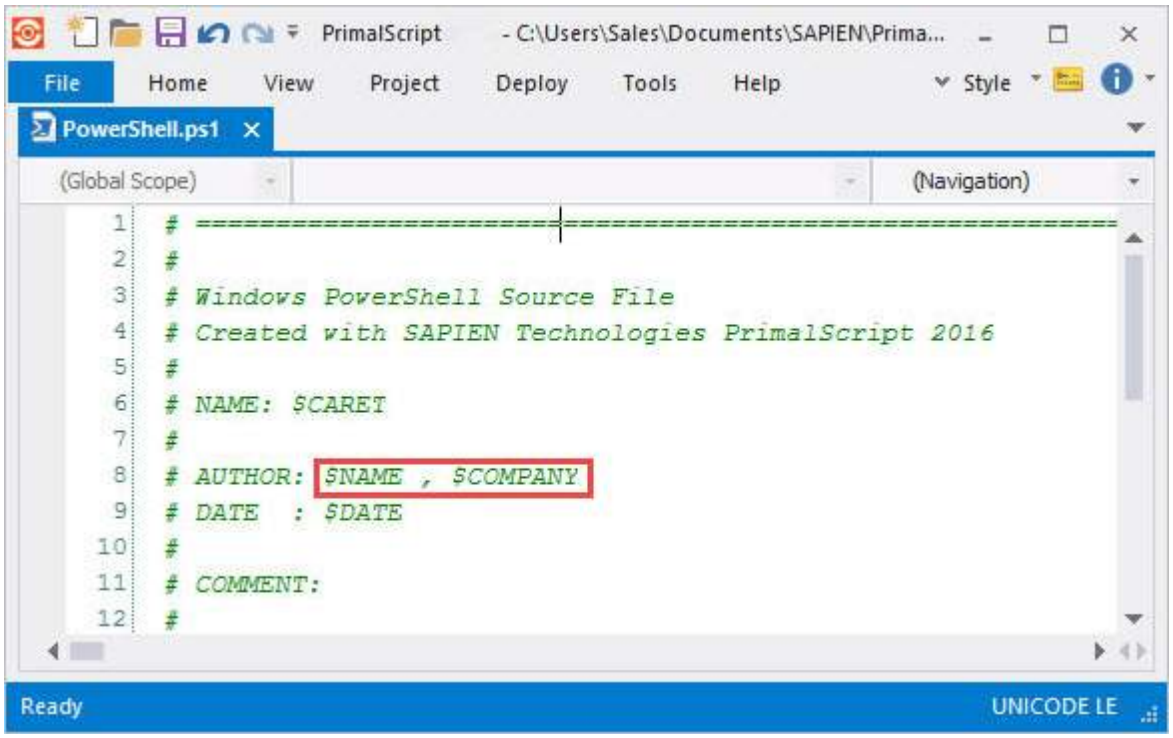
You can specify default values for the name and company template variables. The values are specific to each copy of PrimalScript.

To specify the name and company

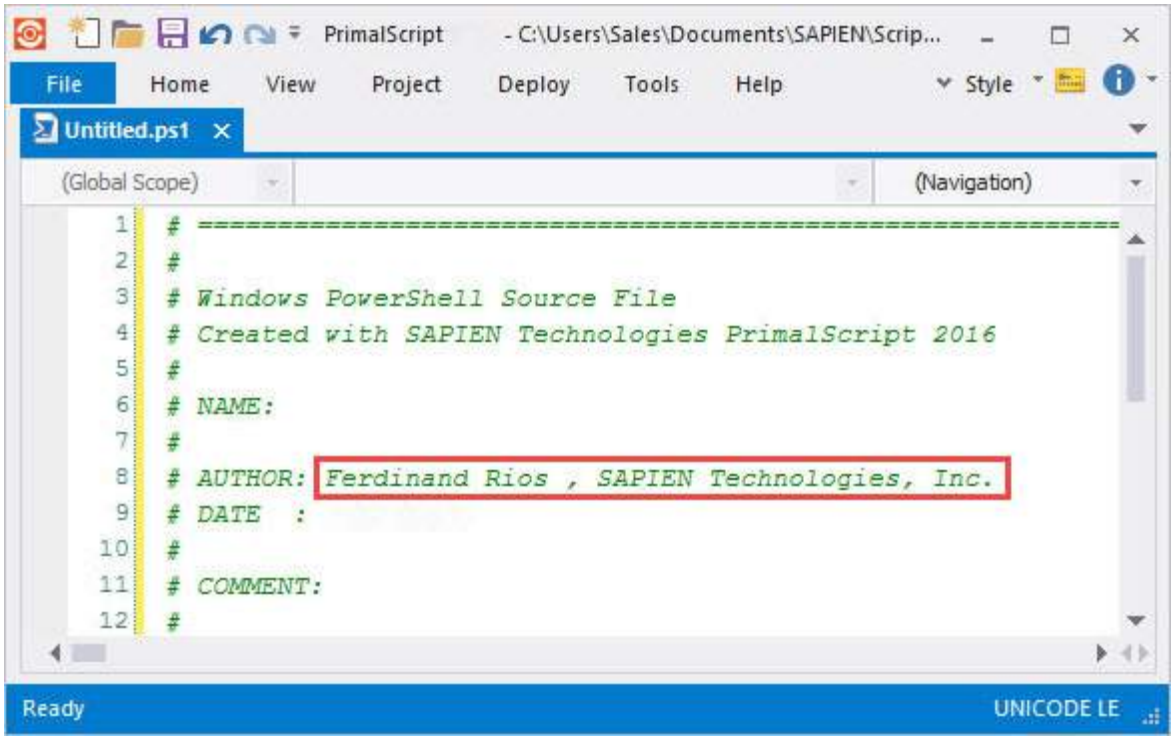
1. Click **File** > **Options** > **Application** > **General**.
2. Type the values in the **User Name** and **Company** fields.



Then, when you use a template that has $NAME and $COMPANY variables:



The specified values appear in the template-based file:



5 PrimalScript Editor

PrimalScript is more than just a script editor; it's actually a complete environment, including dozens of built-in tools and functions to make scripting and software development more efficient. However, at the heart of PrimalScript is the industry's most powerful and flexible code editor. While it's easy to start using the PrimalScript editor without any training, a number of its most efficient and effective features can be easily overlooked.

In this section you will learn about all of the core editor features within PrimalScript, as well as a number of tips for using PrimalScript more efficiently.

5.1 Editing Aids

PrimalScript includes many aids that make editing scripts and code easier.

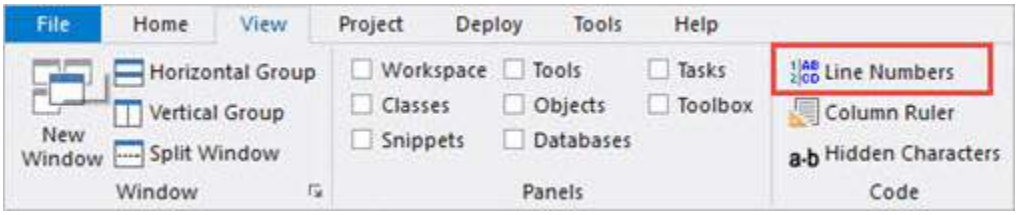
5.1.1 Line and Column Numbering

PrimalScript provides optional line numbering and a column ruler, making it easier to navigate to the desired line and column. The current line and column are also displayed in PrimalScript's status bar.

- Line numbers are displayed by default. To hide them, click **View** > **Line numbers**.
- The column ruler is hidden by default. To display it, click **View** > **Column ruler**.

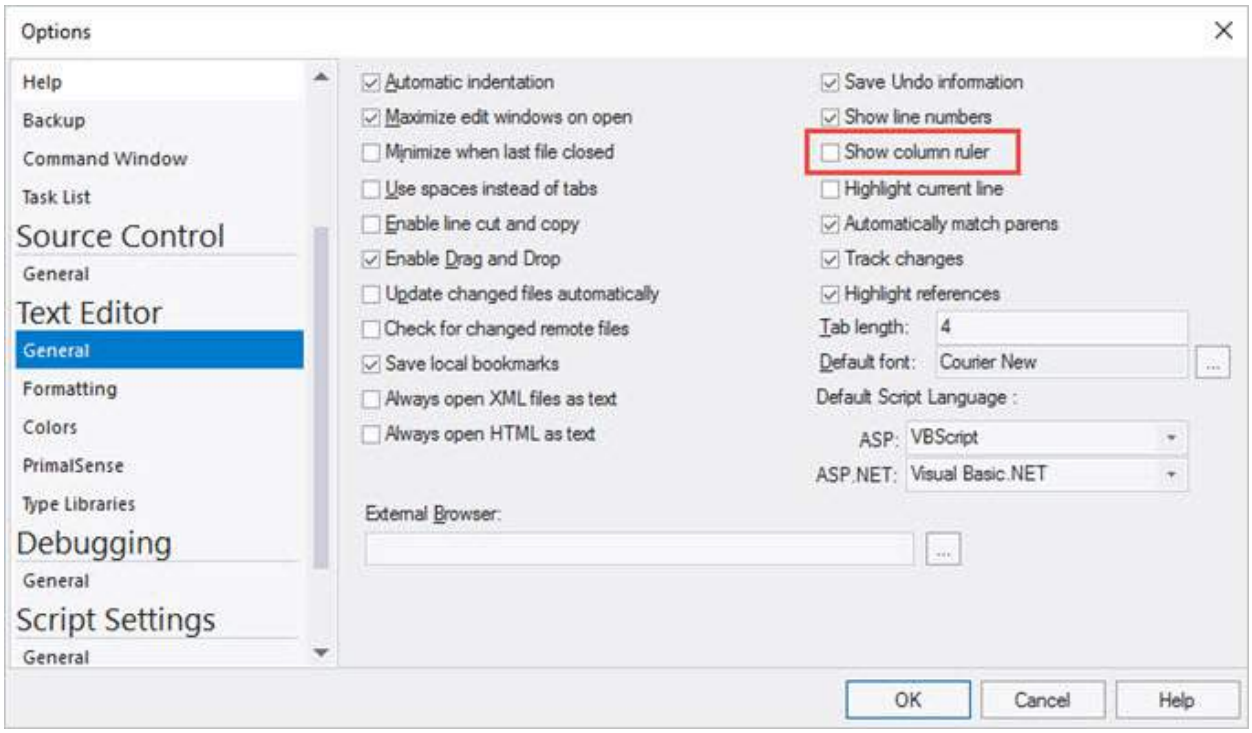
To show and hide line numbers and the column ruler

- On the ribbon, click **View** and then click the **Line numbers** or **Column ruler** toggle button:

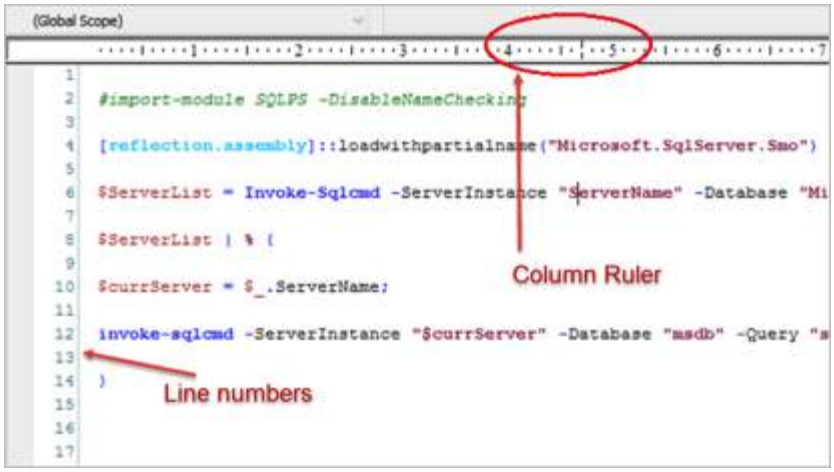


To display the column ruler by default

- Click **File** > **Options** > **Text Editor** > **General** > **Show column ruler**:

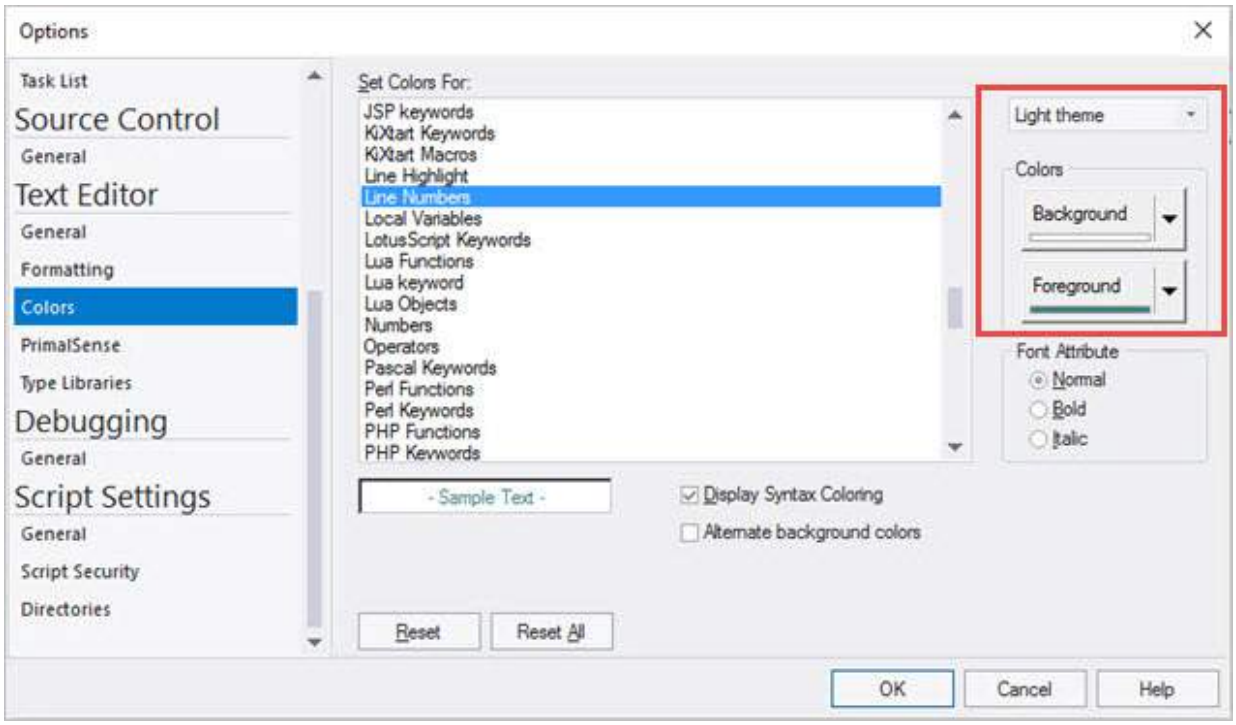


The numbers on the column ruler represent the **tens**, and the ticks between them represent the **ones**. Therefore, the cursor is at position 47 in the following image:



To change the colors of the line numbers

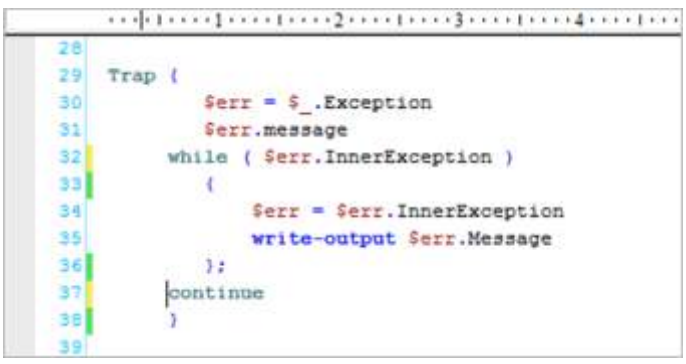
1. Click **File** > **Options** > **Text Editor** > **Colors** and in the **Set Colors For** box, click **Line Numbers**.
2. In the Colors section, use the Background and Foreground controls to set the line number colors:



The column ruler is a good way to ensure proper positioning of elements in your code. You will stay more organized and be more productive if certain elements are lined up.

A good example of this can be found in the following screenshot. Notice how the curly brackets aren't lined up so it's harder to tell where code blocks begin and end:

Here is the same code block, with the brackets lined up with the column ruler. It's much cleaner and easier to follow, or see where to insert or delete code:



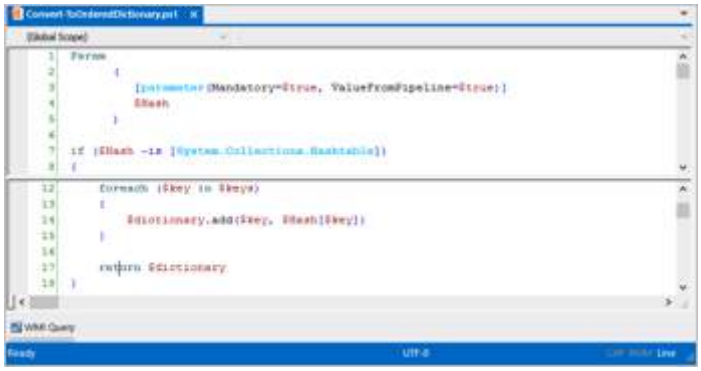
Notice that any line that has been changed—but not saved—has a yellow bar in its left margin. Saved changes have a green bar.

5.1.2 Split Screen

Split screen lets you scroll sections of the code editor independently of each other. This feature is especially helpful when you are editing different parts of the script at once, or viewing one part of a script while editing another.

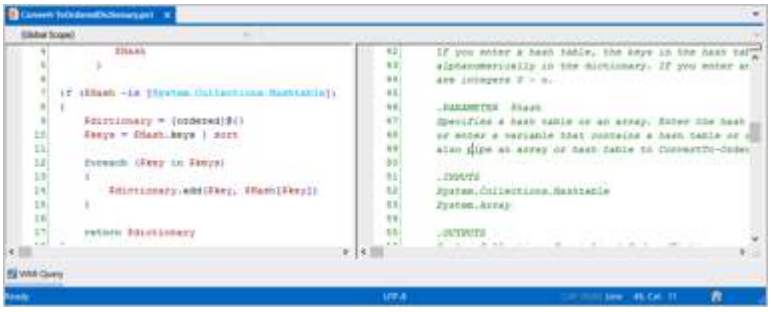
Split screen options

- **Horizontal splitter** divides the screen into two horizontal panes:



- **Cross splitter** divides the screen into four panes:

For two vertical panes, cross-split the screen and then close the horizontal split:



Split screen features

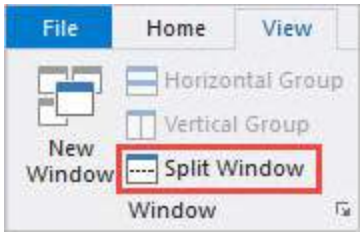
- Split panes scroll independently.
- Each split pane has its own column ruler.
- If you edit a line in a split pane of a file, the edit is effective in all split panes of the file. (It is two views of the same file; not two files.)

To split a screen horizontally

- Click the split screen divider, drag it down the page, and release.

To cross-split a screen

- Click **View** > **Split Window**:



To resize the split panes

- Click a split line and drag it to a new location.

To close the split panes

- Click a split line and drag it to the edge of the screen (top or bottom; left or right).

-OR-

- Click the center of a cross-split screen and drag it to any corner of the screen.

5.1.3 Display Hidden Characters

By default, PrimalScript does not display whitespace characters, such as spaces, tabs, and newlines (<Enter> key).

To display whitespace characters

- Click **View** and, in the Code section, click **Hidden characters**.

  Displaying hidden characters improves formatting, and can reveal extra-space errors that might interfere with the proper execution of your script:

5.1.4 Display as a Binary File

PrimalScript allows you to open a file in binary mode.

To display a document in hexadecimal mode

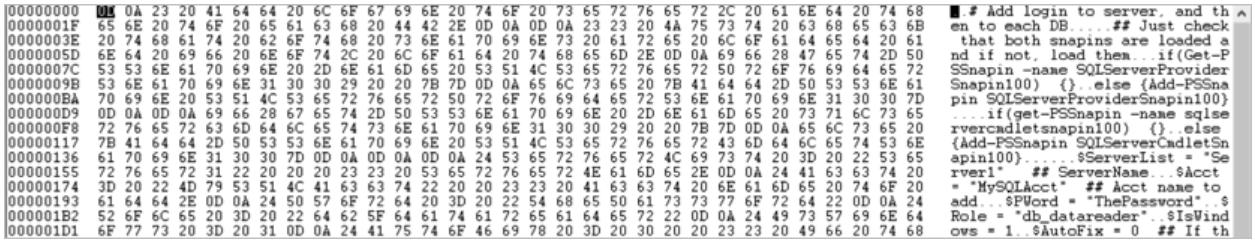
- In the code editor, right-click and then click **Open file as binary file**.

  PrimalScript closes the standard (ASCII) display and opens the file in binary mode. Because the file is closed and reopened as a binary file, it is not a toggle control.

To return to standard display

- Open the file again (**File** > **Recent Documents** ...)

  This display can make it easier to work with binary documents, because you can work directly with raw data that data isn't visible in a text editor:

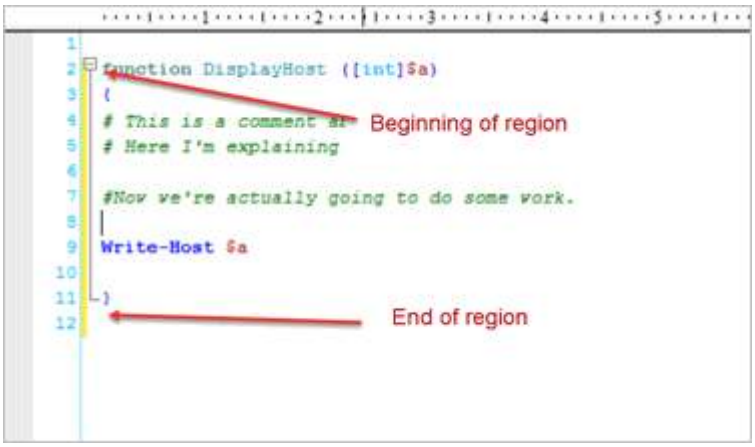


5.1.5 Code Folding

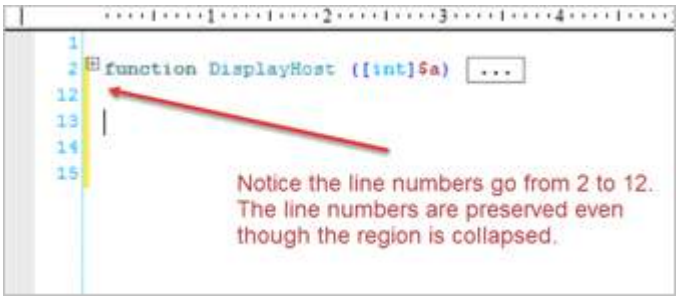
Code folding collapses and expands sections of your script, making it easier to focus on the sections that you need. PrimalScript creates foldable regions automatically from functions and subroutines, and it creates temporary foldable regions from selected text—but you can create custom persistent foldable regions for any lines in a file.

Automatic Foldable Regions

PrimalScript automatically creates foldable regions from declared functions and subroutines:



When collapsed, PrimalScript displays line numbers that show lines contained within the collapsed (or folded) region:



This ensures that line-number-based error messages remain accurate.

Temporary Foldable Regions

Whenever you select text, PrimalScript makes the selected text a temporary foldable region. You can expand and collapse these temporary regions, but they are not persisted when the file is closed and reopened.

Persistent Foldable Regions

You can create a persistent foldable region—one that is maintained even when PrimalScript and the file are opened and closed.

To create a foldable region

1. Select one or more lines of a file.
2. Click **Home** > in the Edit section, click **Region** > **Create Region**.
3. [Option] Edit the name of the region. The default value is **Persistent fold region**.

-OR-

- In your script, type **#region** and **#endregion** on commented lines.

  PrimalScript uses the appropriate comment characters for each scripting language:

  Two types of files—Powershell and SQL; chose the right comments for each one.

  Named regions are especially convenient because the name remains visible even when the region is folded to remind you of what the region contains:

Managing Foldable Regions

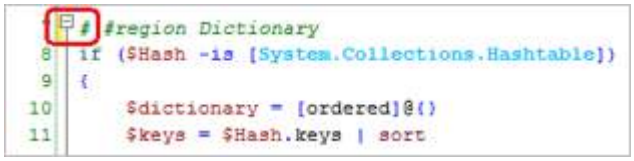
You can collapse (fold) and expand (unfold) foldable regions individually or as a unit.

To manage regions

- Click **Home** > in the Edit section, click **Region**:

To collapse a region

- Click the **minus sign** (-) at the beginning of the region:



To expand a folded region

- Click the **plus sign** (+) at the fold:

To collapse all foldable regions

- Click **Home** > in the Edit section, click **Region** > **Hide all**.

To expand all foldable regions

- Click **Home** > in the Edit section, click **Region** > **Expand all**.

To find foldable regions

- Search for **#region**

5.1.6 Vertical Groups

The Vertical Groups feature splits the screen into muliple vertical panes with one or more files in each pane. Unlike the Split Screen 46 feature, which displays the same file in multiple panes, the Vertical Groups feature displays different files in each pane.

To create a vertical group

1. Click a file tab.
2. Click **View** > in the Window section, click **Vertical group**.

   -OR-

   - Right-click a file tab > click **New Vertical Tab Group**.

     PrimalScript splits the screen and moves the selected tab to the new group.

To move a file to another vertical group

- Click a file tab and drag the file to the other group.

-OR-

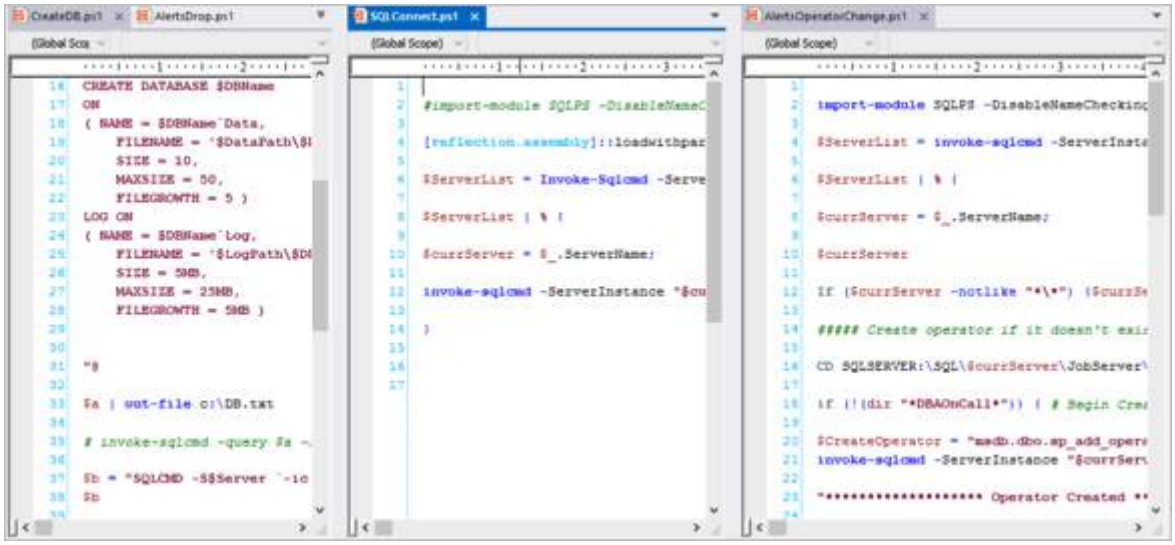
- Right-click a file tab > click **Move to previous tab group** or **Move to next tab group**.

To delete a vertical group

- Move all files to other tab groups or close them.

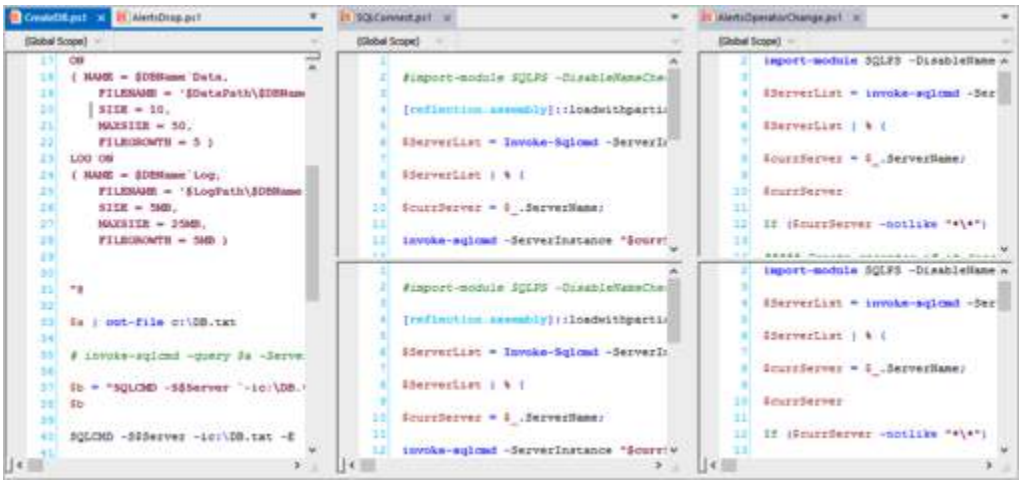
  A group is deleted when it contains no files.

  For example, this screen has three vertical groups. The leftmost group contains two files. Each of the other groups contains one file:



You can combine the Vertical Group feature (different files in each pane) with the Split Screen 46 feature (the same file in multiple panes):

You can also combine the Vertical Group feature with the cross-split ("split window") feature, which divides the window into four panes—each displaying the same file:



5.1.7 File Groups

PrimalScript lets you save files as a group without changing the file type, location, or any other property of the member files. When you open or close a file group, PrimalScript opens (or closes) all of the files in the group. This is a very handy way of managing related files, such as the files in a module or gem.

To create a file group

1. Open the files you want to include in the group.
2. Click **File** and then click **Save open files as group**.
3. Enter a name and location for the .filegroup file.

To open a file group

- Open the .filegroup file

  PrimalScript opens all files in the file group.

5.1.8 Open Related Files

You can open related files as a unit, even if the files are not part of a file group.

PrimalScript considers files to be related when the files are in the same directory and their file names begin with the same base name. For example, if you open Perfmon.bat and then click **Open related file**, PrimalScript displays files in the directory that begin with Perfmon, such as Perfmon.sql, Perfmon-collector.ps1, and PerfmonList.html.

To open files that begin with the same name

1. Open a file, or, if editing multiple files, click a file tab.

2. Click **File** > **Open related file**.

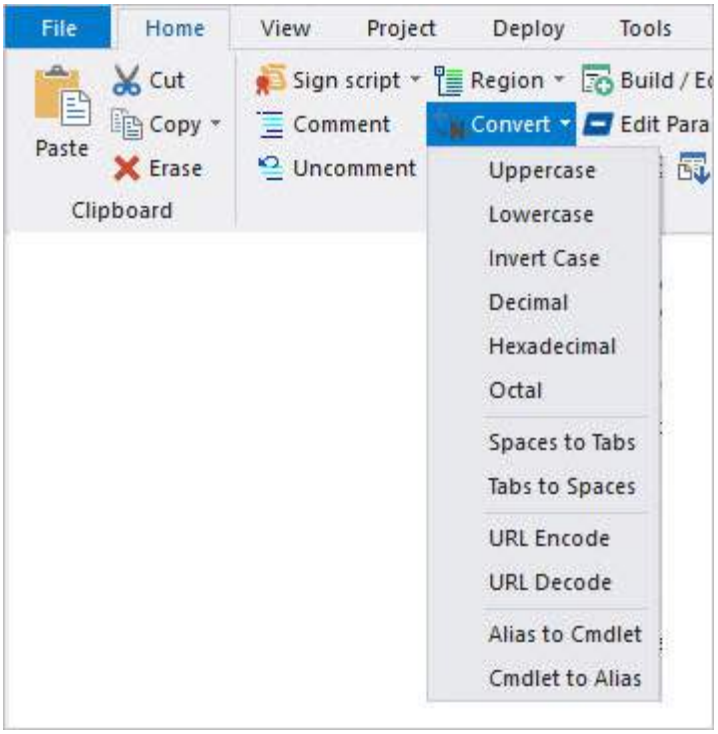
   The Related Files feature uses the entire name of the current file. If the current file is New-Task.ps1, it looks for files with names that begin with New-Task (New-Task*.*).

5.1.9 Conversion Features

You can use PrimalScript features to convert text and data.

To access the conversion features

- Click **Home** > in the Edit section, click **Convert**:



To convert text case

1. Click the text.
2. Click **Home** > in the Edit section, click **Convert;** then click **Uppercase**, **Lowercase**, or **Invert case**.

To convert numeric base

1. Click a number.
2. Click **Home** > in the Edit section, click **Convert;** then click **Decimal**, **Hexadecimal**, or **Octal**.

To encode or decode URL

1. Click a URL that includes spaces or special characters.

2. Click **Home** > in the Edit section, click **Convert;** then click **URL Encode** or **URL Decode**.

To convert Windows PowerShell cmdlet aliases to full cmdlet names (and back)

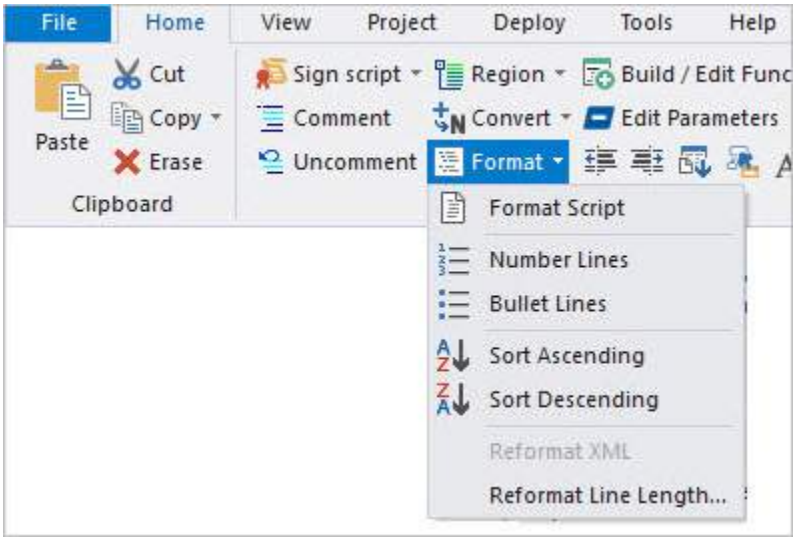
1. Click a cmdlet name or alias, or select all (*Ctrl+A*).
2. Click **Home** > in the Edit section, click **Convert;** then click **Alias to Cmdlet** or **Cmdlet to Alias**.

5.1.10 Formatting Features

PrimalScript makes it easy to format and reformat a script file.

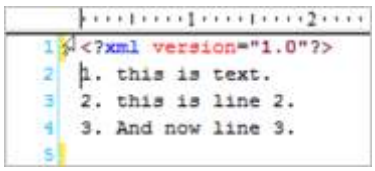
To open the format menu

- Click **Home** > in the Edit section, click **Format**:



To insert line numbers at the start of each line

- Click **Home** > in the Edit section, click **Format** > **Number Lines**:

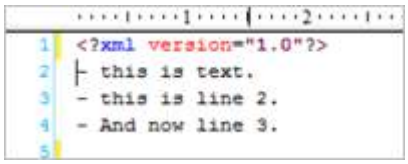


To sort line numbers

- Click **Home** > in the Edit section, click **Format** > **Sort Descending** or **Sort Ascending**.

To insert bullets at the start of each line

- Click **Home** > in the Edit section, click **Format** >**Bullet Lines**:



To reformat XML

- Click **Home** > in the Edit section, click **Format** > **Reformat XML**:

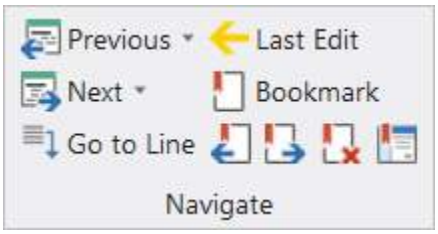
Formatted version:

5.2 Navigation Options

PrimalScript provides options to help you move quickly between different locations in your scripts.

Navigate - Menu Options

The **Home** ribbon > **Navigate** section contains navigation options:



- **Go to Line**
  Navigates to the specified line.
  1. Click **Home** > Navigate section > **Go to line** (*Ctrl*+*G*).
  2. Enter the line number, and click **OK**.

- **Last Edit**
  Navigates to the position of the last edit operation.
  o Click **Home** > Navigate section > **Last edit** (*Ctrl*+*E*).

Bookmarks

Bookmarks make it easier for you to return to specific sections of your script. Bookmarks appear as colored blocks in the left margin of the editing window. They are particularly useful when you are working in several parts of a script at once.

Every bookmark is a toggle; click to create it, click again to delete it.

To find a bookmarked section

- Scroll until you see the bookmark, or use the keyboard (*F2*) to jump directly to the next bookmark:

To toggle a bookmark on or off

1. Place the cursor on the line where you want the bookmark.
2. Click **Home** > in the Navigate section, click **Bookmark**.

-OR-

- Right-click in the margin (if you left-click, you create a breakpoint (red circle)).

-OR-

- Press *Ctrl+F2*.

To bookmark a search term

- Clicking **Mark All** will place a search-style (temporary) bookmark at each location where your search string occurs; you can then use *F3* and *Shift+F3* to move back and forth between bookmarks.

To jump to the next or previous bookmark

- Click **Home** > in the Navigate section, click **Next Bookmark** or **Previous Bookmark**.

-OR-

- Press *F2* (next) or *Shift+F2* (previous).

To clear (turn off) all bookmarks

- Click **Home** > in the Navigate section, click the **Clear all bookmarks** icon:

Global Bookmarks

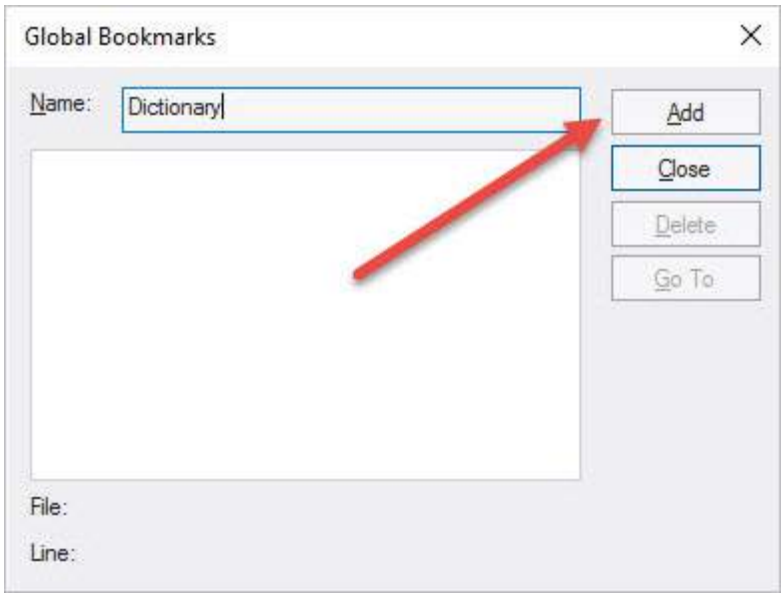
Global bookmarks, also known as *named bookmarks*, are designed to help you navigate through multiple files, but can also be useful in a single file.

Unlike standard bookmarks, you can't see global bookmarks in the code editor. They are only available in the Global Bookmarks dialog box.

To create a global bookmark

1. Position the cursor on the line where you want to create the global bookmark.
2. Click **Home** > in the Navigate section, click the **Global Bookmark** icon (*Alt+F2*):

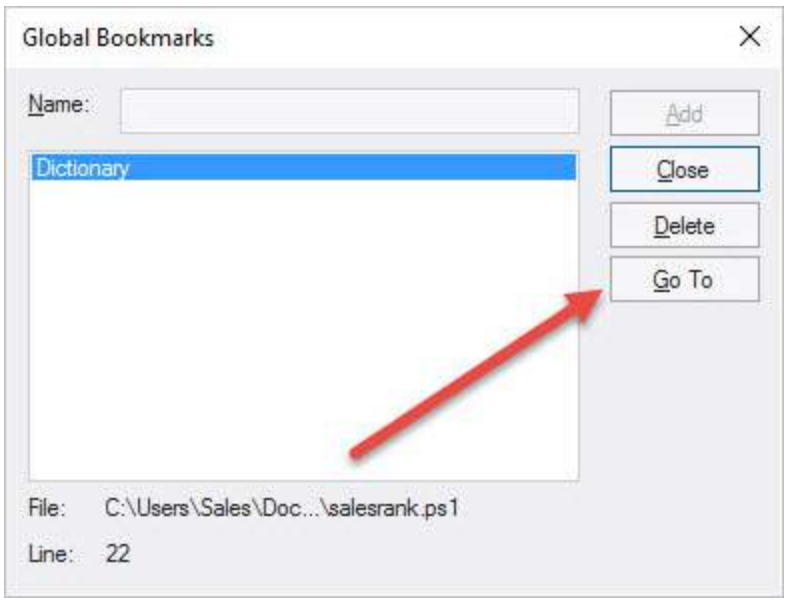
3. Enter a name for the bookmark and click **Add**:



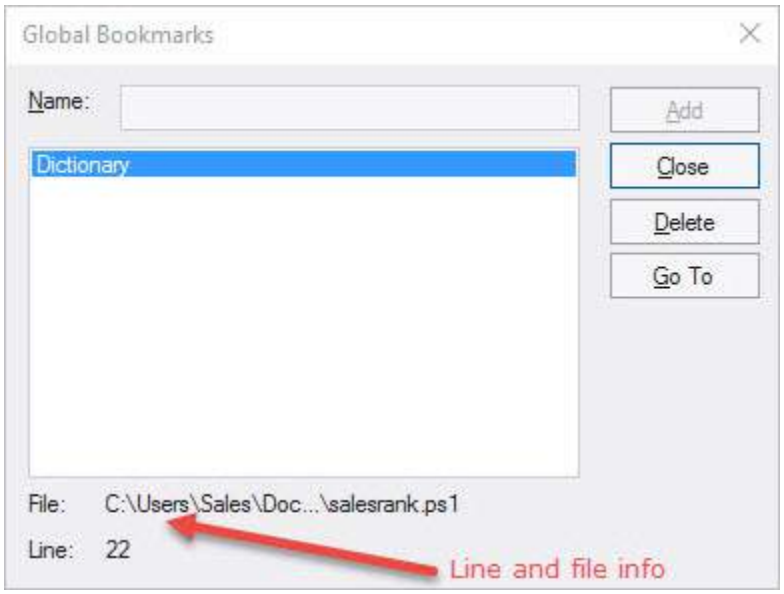
To jump to a global bookmark

1. Click **Home** > in the Navigate section, click the **Global Bookmark** icon (*Alt+F2*):

2. Click the bookmark name and then click **Go To**:



Global bookmarks are even powerful when used with multiple files. When you define bookmarks in more than one file, PrimalScript saves the path and file name along with the line number. When you click **Go To**, PrimalScript opens the file with the cursor on the bookmarked line:



5.3 Clipboard Integration

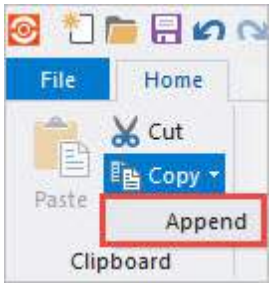
PrimalScript provides enhanced clipboard functionality.

Append to Clipboard

By default, when you copy, the copied data replaces data on the clipboard—but you can also append to data on the clipboard. When you paste, it pastes all data copied and appended. This is an excellent way to collect and assemble data from different parts of one or multiple scripts.

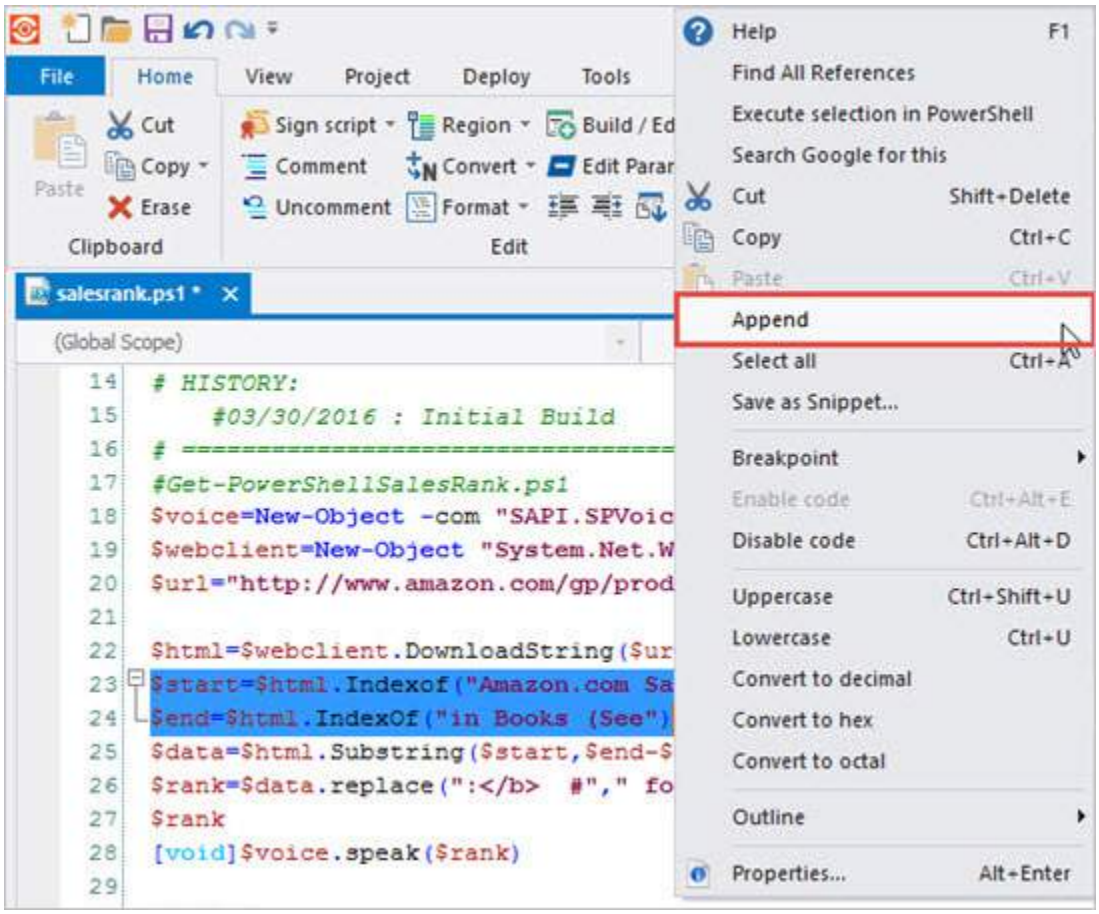
To append to the clipboard

- Click **Home** > in the Clipboard section, click the **Copy menu** (down-arrow) > click **Append**:



-OR-

- Right-click selected lines and click **Append**:



Paste from Copy Stack

PrimalScript saves the last several cut and copy operations in a last-in-first-out stack. When pasting, you can cycle through the copied / cut text, and then paste.

To paste from the copy stack

- Press *Ctrl+Shift+V* repeatedly.

5.4 Find and Replace Options

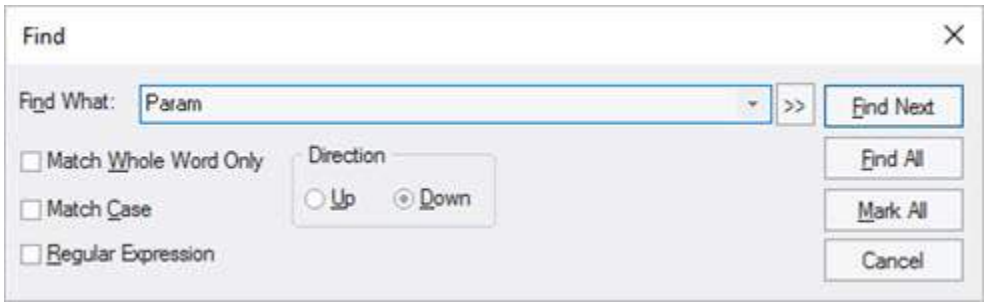
PrimalScript includes extensive, powerful search and replace features, including the ability to search multiple files 65, and to manage multi-file search results 66.

Search a File

PrimalScript performs a full-text search of files and allows you to change the contents of a single file or multiple files quickly and easily.

To find a word or word pattern

1. Click **Home** > in the Find section, click **Find** (*Ctrl+F*).
2. Enter the search term and press <**Enter**> or click **Find Next**:

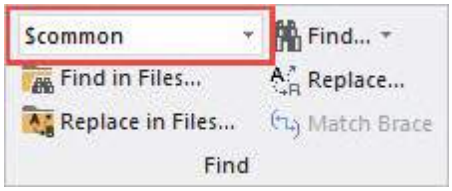


-OR-

- **Select a term** from the search history, and press <**Enter**>:

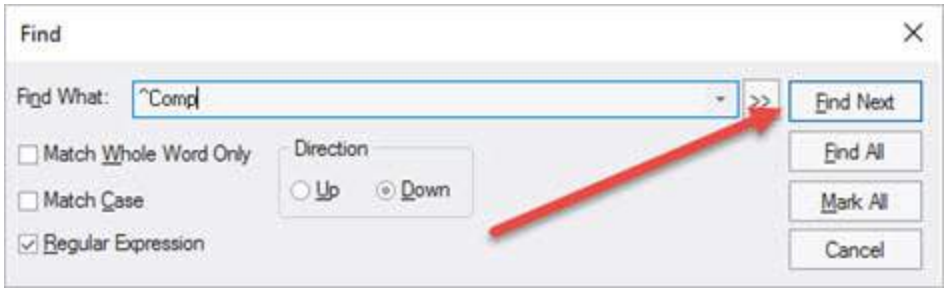
-OR-

- In the **Find** section, enter the search term, and press <**Enter**>:



To find the next instance of the term, click **Find Next again**.

You can use common search criteria such as whole word matching, regular expression matching, case matching, and searching forward or backward:



To find all instances of a word or word pattern

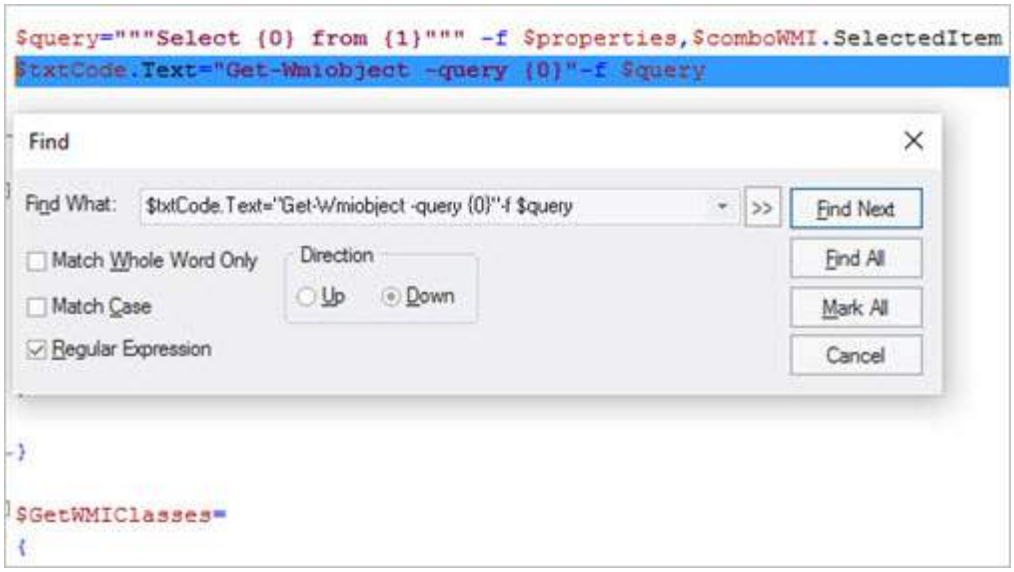
1. Click **Home** > in the Find section, click **Find** (*Ctrl+F*).
2. Enter the search term and press <**Enter**> or click **Find All**.

-OR-

- Select and right-click a term and then click **Find All References**.

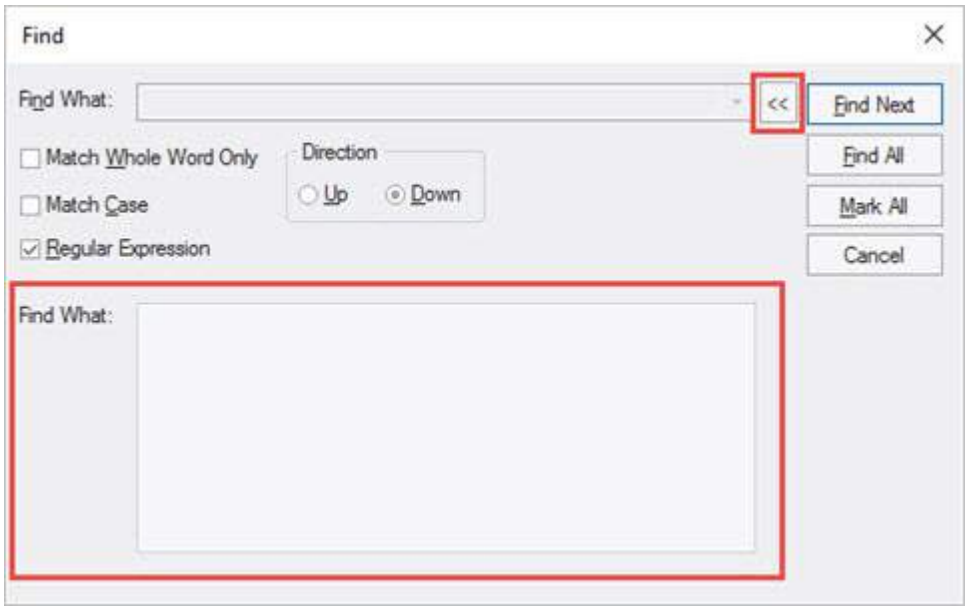
To find text from a script

1. Copy the text from the editor window (*Ctrl+C*).
   The copied text is automatically inserted in the Find box.
2. Click **Home** > in the Find section, click **Find** (*Ctrl+F*):



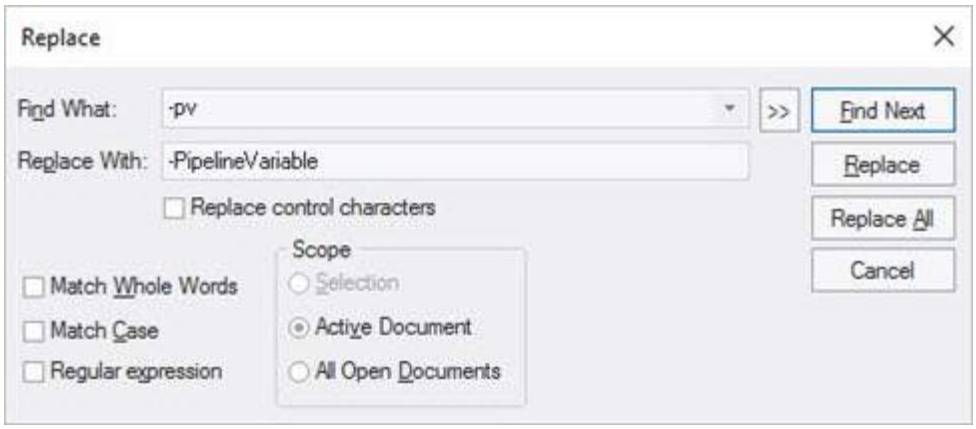
To find multi-line text

1. Click **Home** > in the Find section, click **Find** (*Ctrl+F*).
2. Click the **Expand** / **Contract** button.
3. Enter extended text and then click **Find** / **Find All** / **Mark All**:



To replace a search term

1. Click **Home** > in the Find section, click **Replace** (*Ctrl+H*).
2. Enter the search and replace terms and then press <**Enter**> or click **Find Next**:



To replace hidden characters

1. Click **Home** > in the Find section, click **Find** (*Ctrl+F*).
2. Enter a search term, check **Replace control characters**, and then click **Find** (or **Replace**).

To bookmark a search term

PrimalScript places a temporary search bookmark on each instance of the search term.

- Press *Ctrl+F*, enter a search term, and then click **Mark All**.

To go to search bookmarks

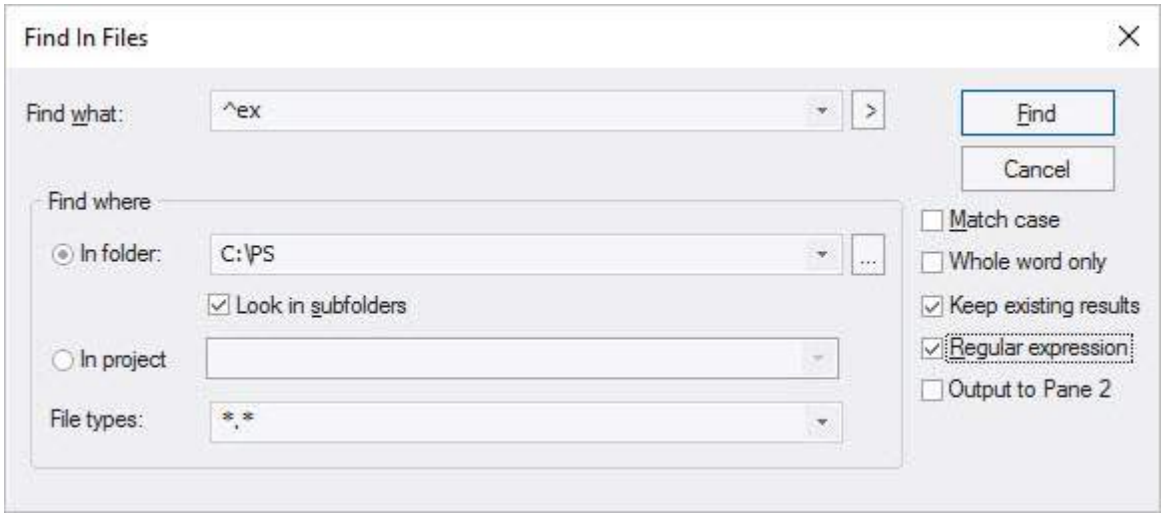
- Press *F3* and *Shift+F3* to move back and forth between bookmarks.

Search Multiple Files (full-text search)

PrimalScript performs a full-text search and replace in multiple files, even when the files are not open in PrimalScript.

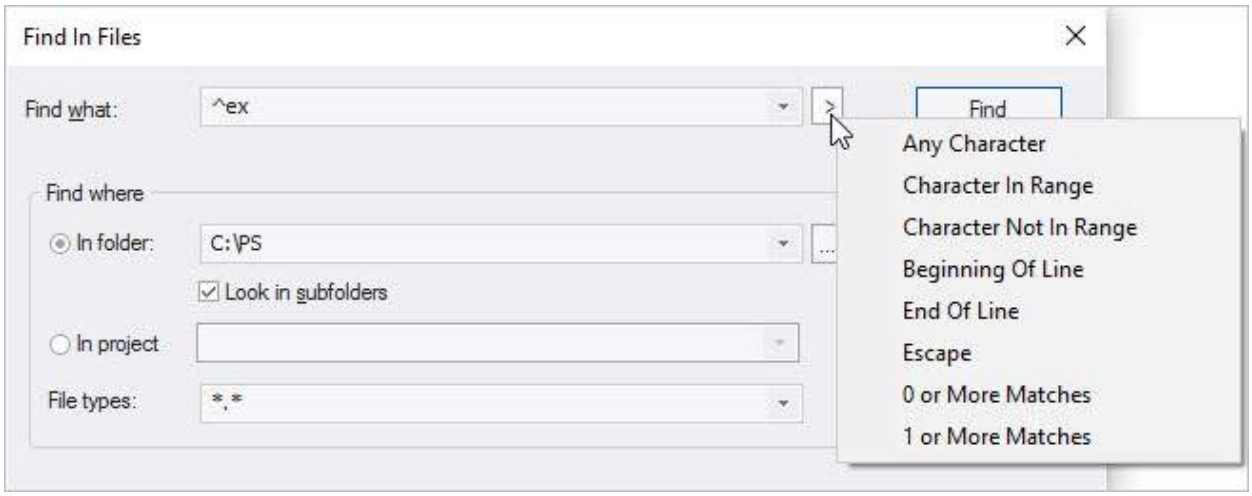
To find a term in multiple files

1. Click **Home** > in the Find section, click **Find in Files** (*Ctrl+Shift+F*).
2. In **Find what**, enter a search term. You can select Match case, Whole word, and Regular expression options.
3. In **Find where**, enter/navigate to a directory or project. To make the search recursive, click **Look in subfolders**.
4. In **File types**, type or select a file name pattern. The default value is all file types:



To use regular expression in a multi-file search

- In the Find in Files window, next to Find what, click the arrow:



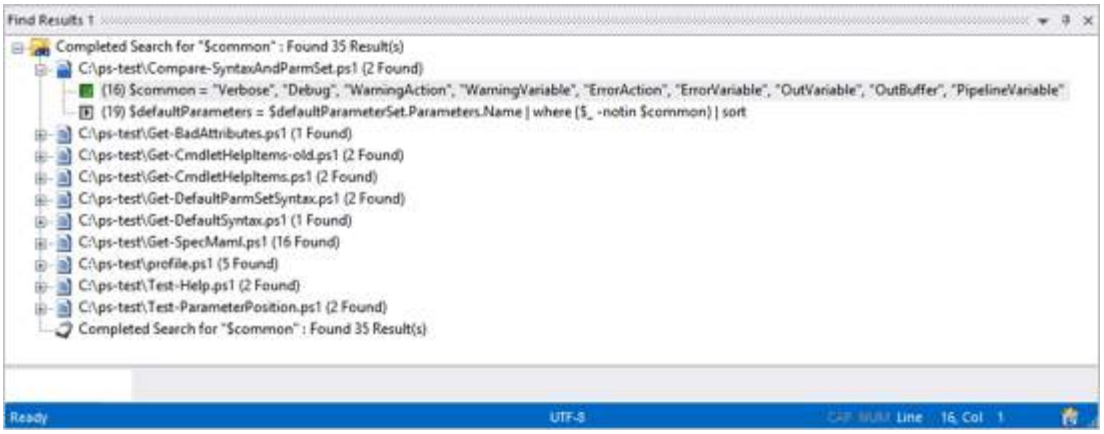
Managing Multi-File Search Results

The results of a multi-file search [65] appear in the Find Results pane.

The Find Results pane creates a to-do list that helps you manage your inspection of the search results.

Step 1: Find in Files

When you run a multi-file search ("Find in Files", *Ctrl+Shift+F*), a Find Results pane displays the results of the most recent search operations:

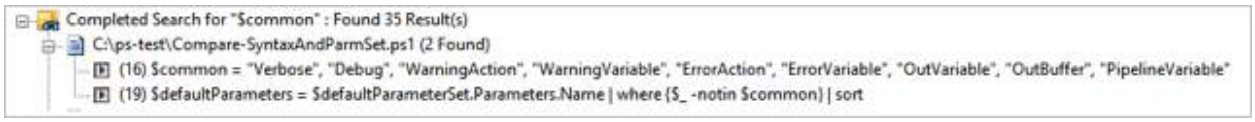


Step 2: Examine the results

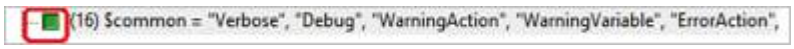
Find Results displays the line number and matches for each file in a to-do display that helps you track your investigation of the results.

To jump to the matching line in the file

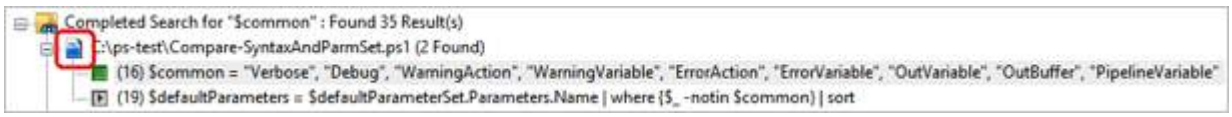
- Double-click the line in Find Results:



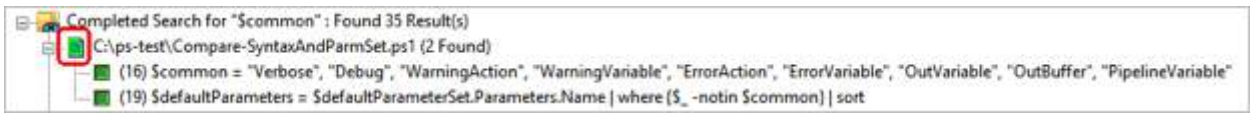
When you click a matched line in Find Results, the box to the left of the line turns green to indicate that you have viewed it:



Also, the file icon is partially shaded to show that you have viewed some, but not all, of the results for that file:



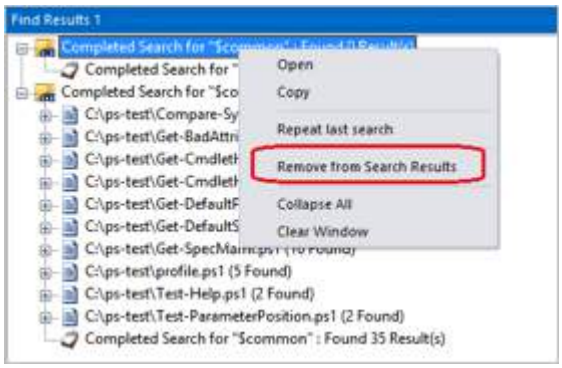
The file icon turns green when you have examined all matches for a file:



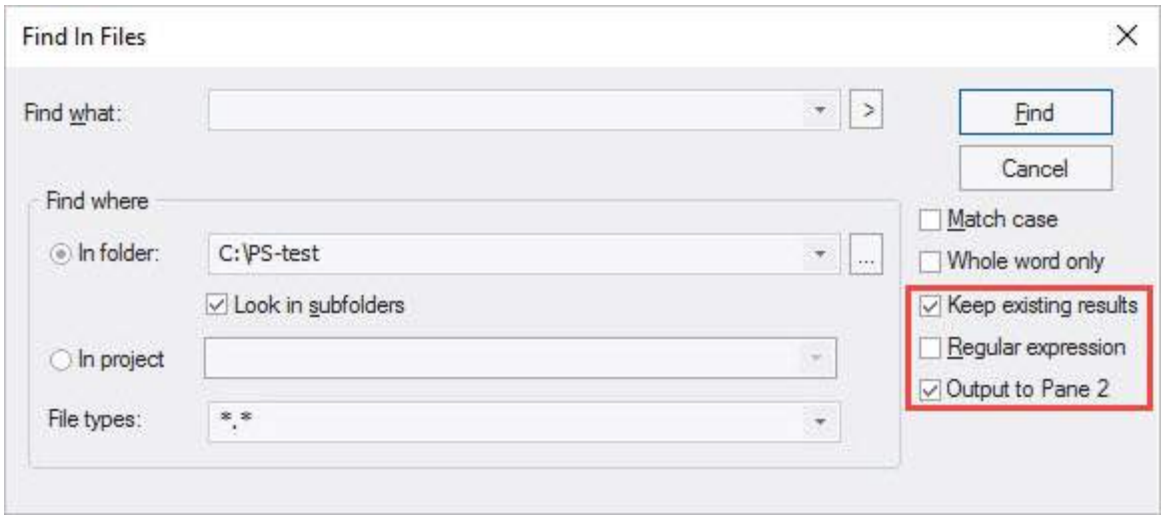
Step 3: Manage the results

To delete results from a Find Results pane

- Right-click the result and then click **Remove from Search Results** or **Clear Window**:



PrimalScript includes two panes, **Find Results 1** and **Find Results 2**. If you are using Find Results 1, you can direct the results of a new search to Find Results 2.



To display Find Results 1 and Find Results 2 manually

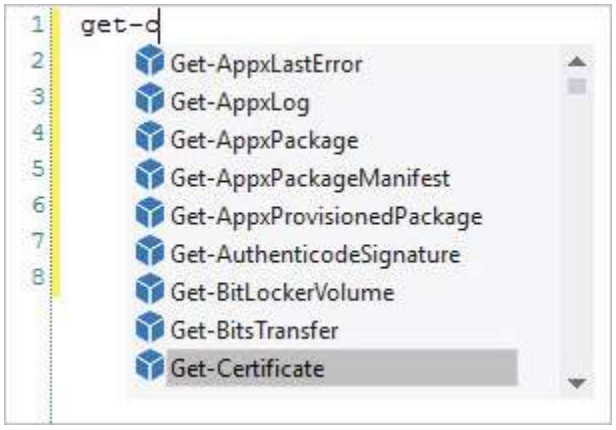
- Click **View** > in the Output section, click **Find Results 1** and/or **Find Results 2**.

5.5 PrimalSense™

PrimalSense™ is a powerful, flexible, code-hinting and code-completion feature.

About PrimalSense™

Because PrimalSense features Optimized Parsing Technology™ (OPT™), you'll never have to wait for it to display the help you need—it works instantly. By default, PrimalSense and OPT are active as soon as you have typed a few characters from a recognized keyword, object variable name, or other element.



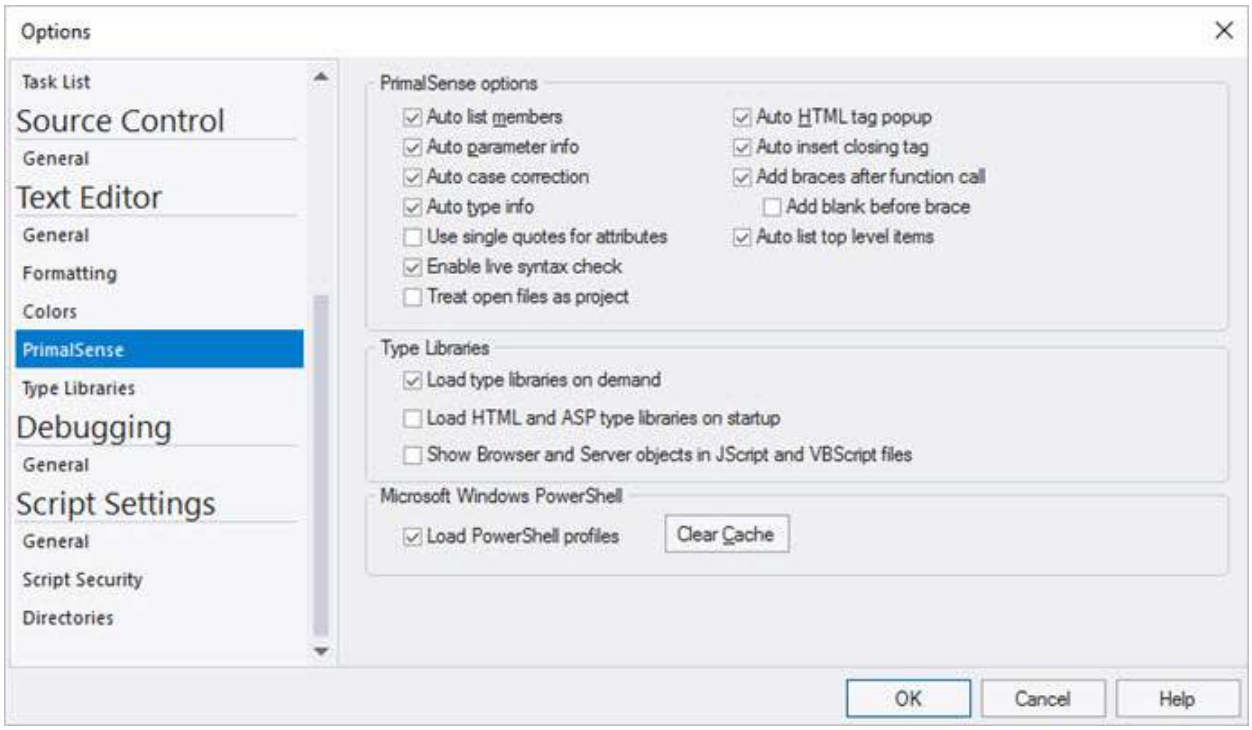
PrimalSense is not available for all languages. To request PrimalSense for a specific language, post a request in the SAPIEN PrimalScript Forum with a link to a language reference resource.

To activate PrimalSense immediately

- Press *Ctrl+Space*.

To customize PrimalSense

- Click **File** > **Options** > **Text Editor** > **PrimalSense**:



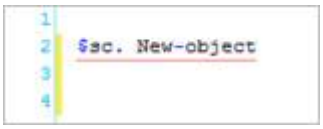
PrimalSense™ Features

PrimalSense works automatically in most cases and provides the following features:

Automatic Syntax Checking

PrimalSense underlines script errors as you type. In the background, PrimalScript submits your statements to the script engine for your scripting language. Because the errors come directly from the script engine, this feature helps you to avoid errors that occur at runtime.

This feature depends upon and reflects differences in script engines. For example, the VBScript engine reports only the first error it finds. The Windows PowerShell engine does not report all errors from a static syntax check.

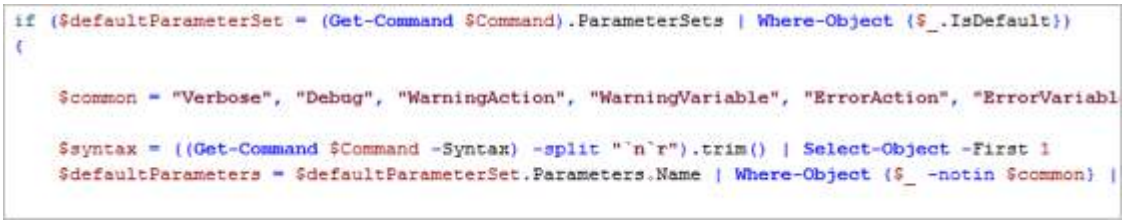


Syntax Coloring

PrimalSense automatically colors your code syntax to help make literals, statements, comments, and other elements stand out more clearly.

Case Correction

PrimalSense automatically corrects the case of intrinsic statements, variable names, and other elements, helping your code to appear more professional.

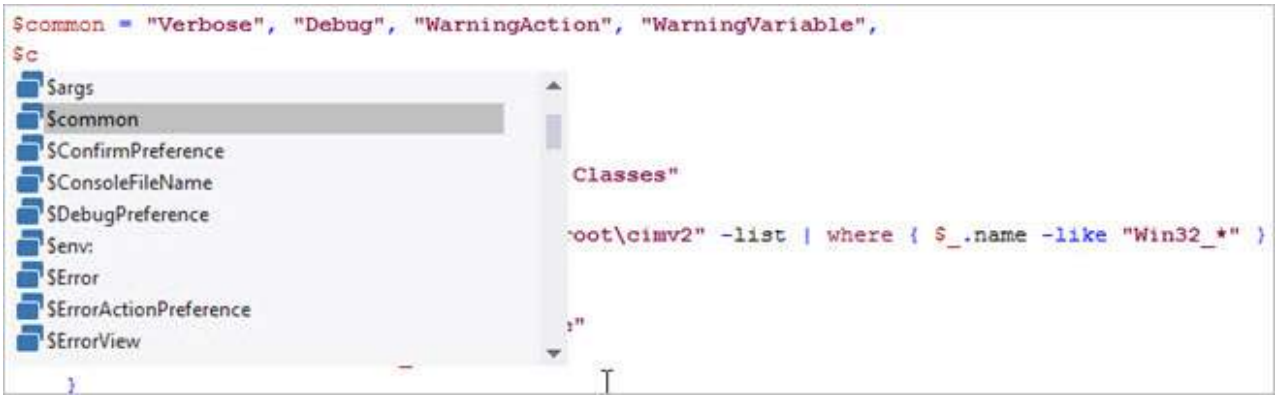


Member Lists

When working with classes and objects, PrimalSense displays pop-up lists of properties and methods. In many cases, PrimalSense can provide "deep" assistance, helping you work with sub-objects and their members.

Variables

PrimalSense completes the names of variables, functions, and other elements of your script. To accept the suggestion, press <Tab>.



Syntax Hints

PrimalSense provides pop-up "tool tips" to help remind you of proper syntax for objects, intrinsic keywords, and defined subroutines, functions, and classes.

5.6 Snippets

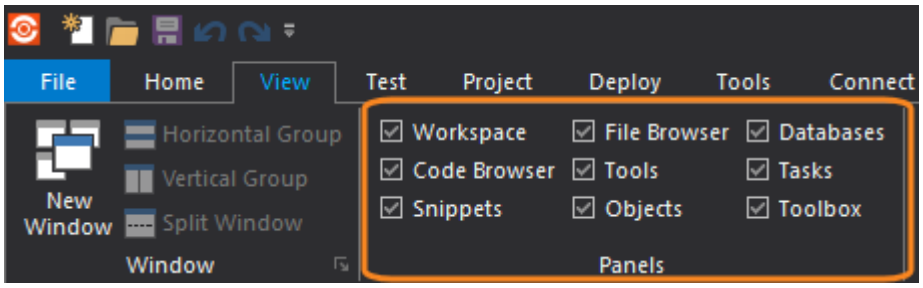
Snippets are a great way to speed up your coding. A snippet is a block of code that you define in PrimalScript that you can insert anywhere you like. There are many excellent uses for snippets. You can create a snippet that has code for creating a connection to a database, performing a simple loop, or just header comments or common functions. The possibilities are endless, and PrimalScript makes it easy to add and manage your snippets.

Accessing Snippets

Snippets are managed in the Snippet Browser 106.

To view the Snippet Browser

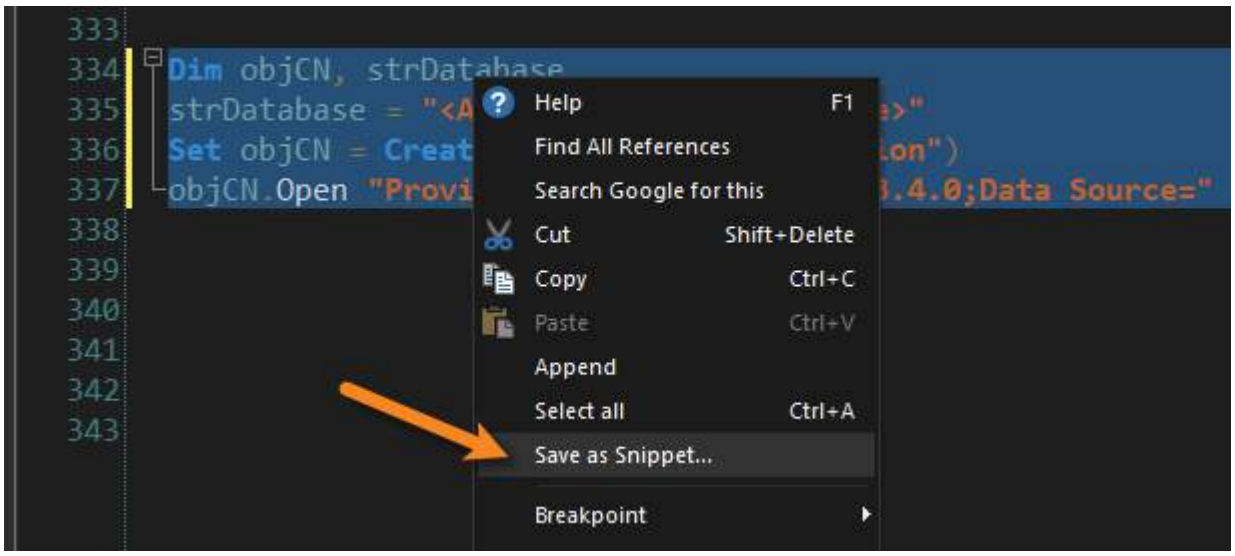
- On the ribbon, click **View** > in the Panels section, check **Snippets**:



Working with Snippets

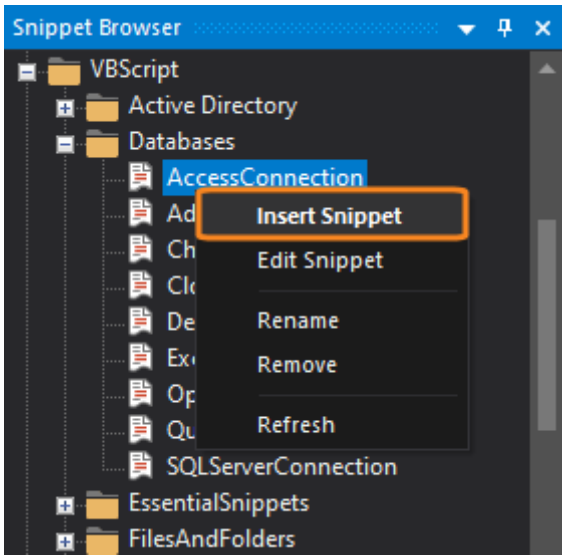
To create a snippet

- Highlight the code you want, right-click and choose **Save as Snippet...**:



To insert a snippet

- In your code, position the cursor where you want to insert the snippet. In the Snippet Browser 106, locate the desired snippet and either double-click or right-click and select **Insert Snippet**:



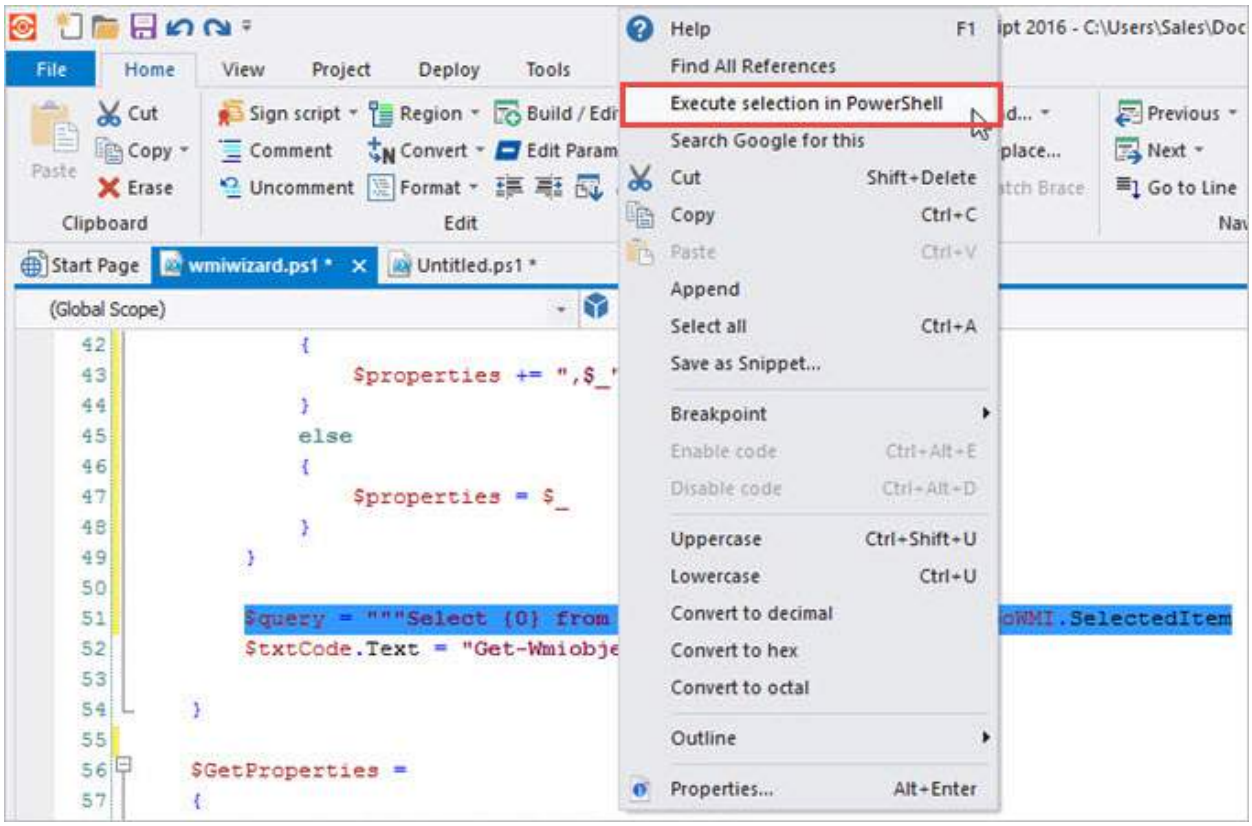
5.7 Run Selected Statements

In some scripting languages, such as Windows Powershell and SQL script, you can run selected statements without running the entire script.

You can use this feature to test part of your code, or to isolate a logic error.

To run selected statements

- Select the statements > right-click and then click **Execute selection in Powershell**:



6 Browser Panels

PrimalScript tools appear in browser panels. You can hide/show, move, resize, dock or float the panels, and even convert the panels to tabs. This section shows you how to work with browser panels, and provides information about each browser.

6.1 Working with Browser Panels

PrimalScript is installed with a default environment that you can customize by moving panels around.

To show/hide a browser panel

PrimalScript's browsers, panels, and consoles can be toggled to show or hide from the View tab.

- Click **View** >then **select or clear the appropriate checkbox** to show or hide the desired panel:

To move a browser panel

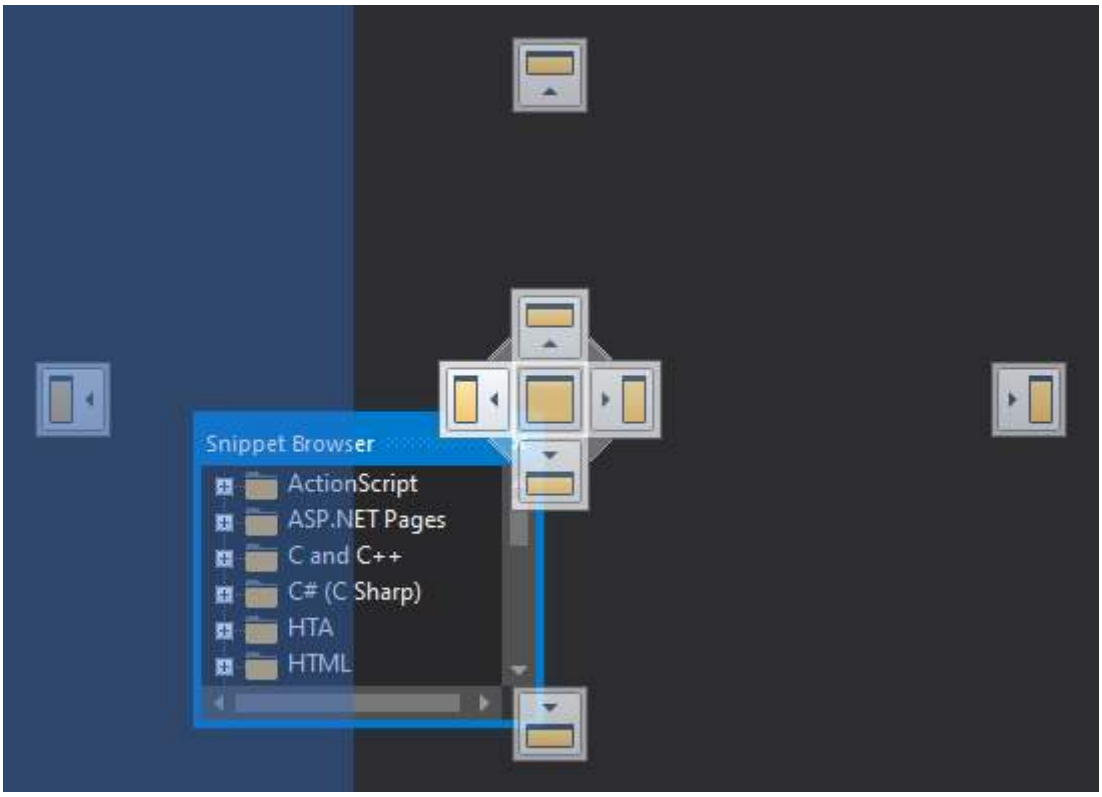
- Click the title bar and drag the panel to a new location.

To float a browser panel

- Right-click the title bar and select **Floating**.

To dock a browser panel

1. Click the title bar and drag the panel in any direction.
2. Use the docking guides to dock to the windows. For example, to dock a panel on the left edge, drag the panel to the left edge-docking guide or left center-docking guide. (The center guide represents all positions. It can be used interchangeably with the respective edge docking guides.)

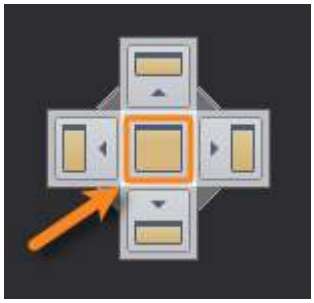


To convert a browser panel to a tab

- Right-click the title bar and then click **Tabbed Document** (it's a toggle).

-OR-

- Click the title bar and drag the panel to the center element in the center docking guide:



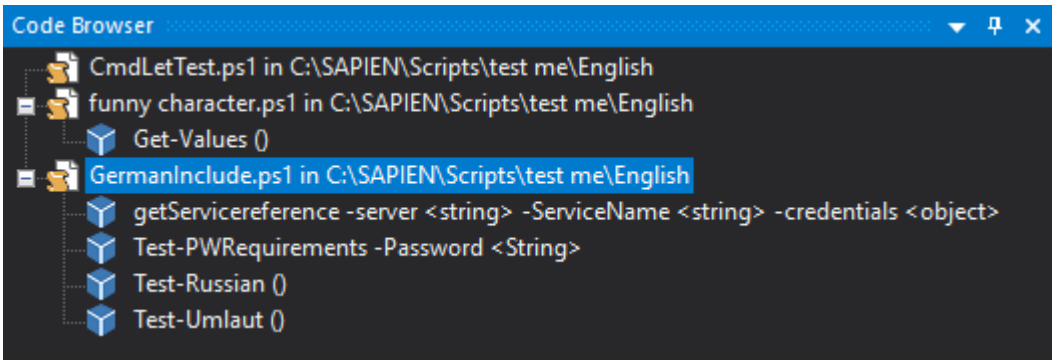
To convert a browser tab to a panel

- Right-click the title bar and then click **Tabbed Document** (it's a toggle).

  You cannot convert a file tab to a panel.

6.2 Code Browser

The Code Browser allows you to view and navigate classes and their methods in your current project or open files.

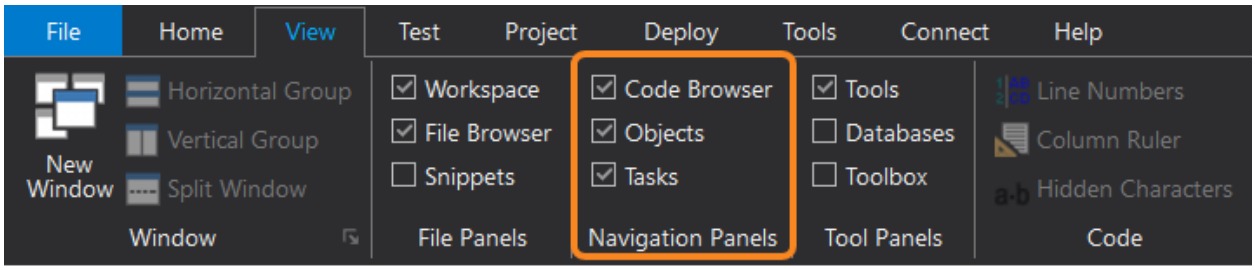


The list is read-only, and updates automatically as you add or remove classes to and from your project or file.

Accessing the Code Browser

To show / hide the Code Browser

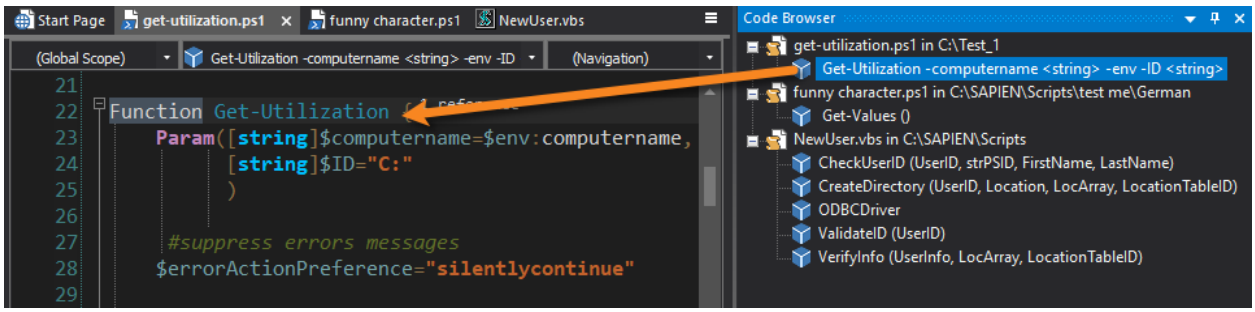
- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



Using the Code Browser

To use the Code Browser

- Double-click an item in the Code Browser to go to that location in the file:

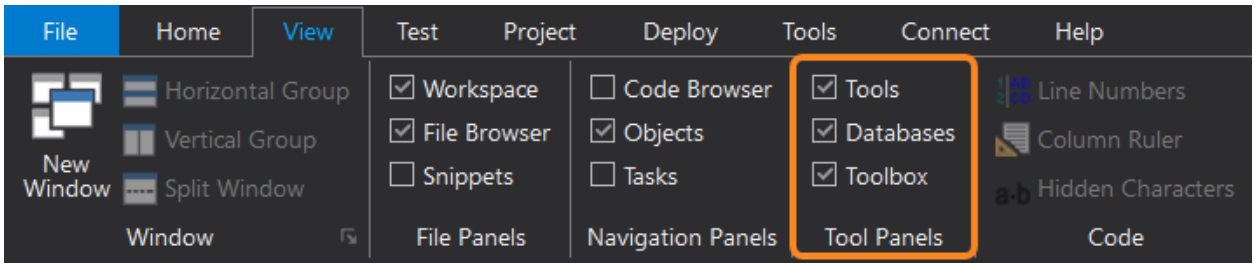


6.3 Database Browser

You can create and edit SQL (.sql) files in PrimalScript. To help with this task, PrimalScript provides a **Database Browser** that allows you to view database objects. Instead of typing connection information in your editor pane, you can drag it from the Database Browser pane.

To show / hide the Database Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):

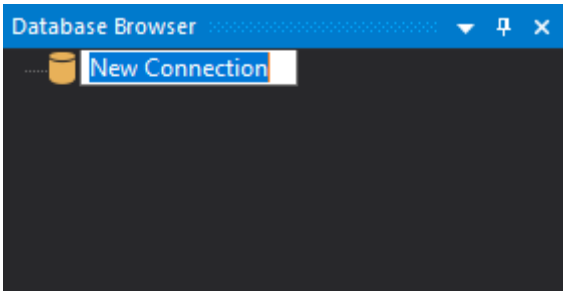


Database Connections

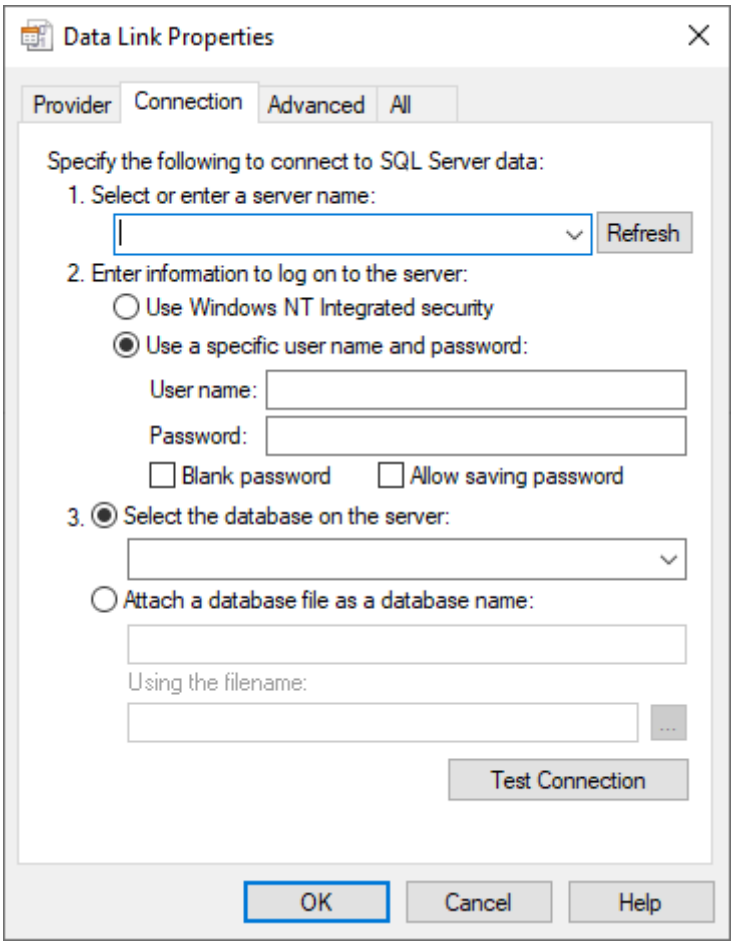
To work with database objects you must create a connection.

To create a connection

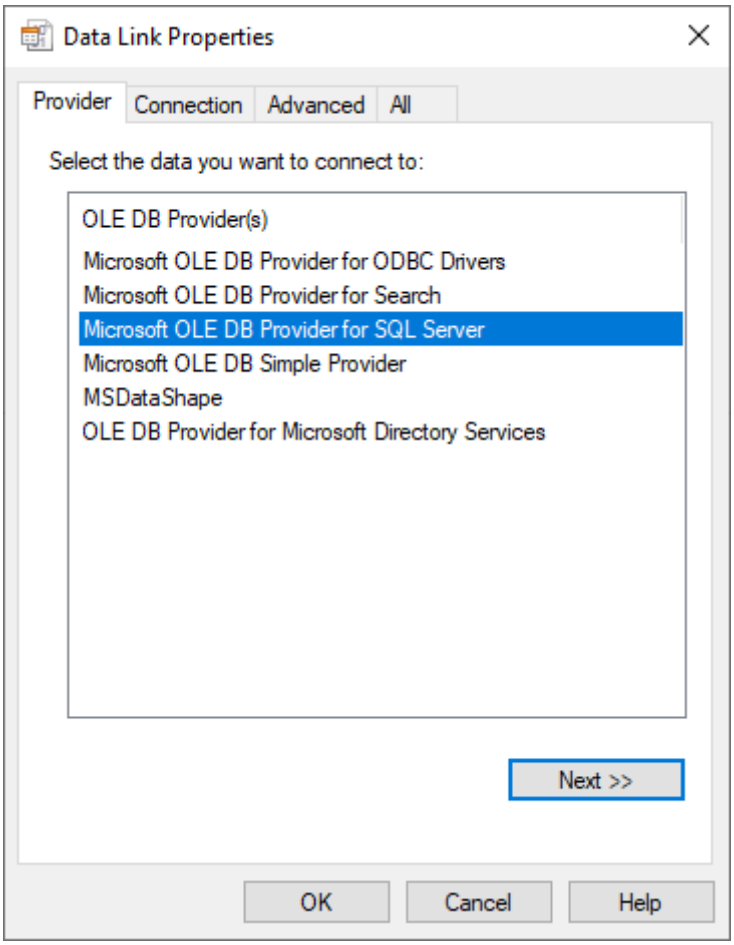
1. Right-click in the Database Browser pane and click **Create a New Connection**.
2. Enter a name for the database connection. "New Connection" is not a valid name.



3. The Data Link Properties dialog will open:

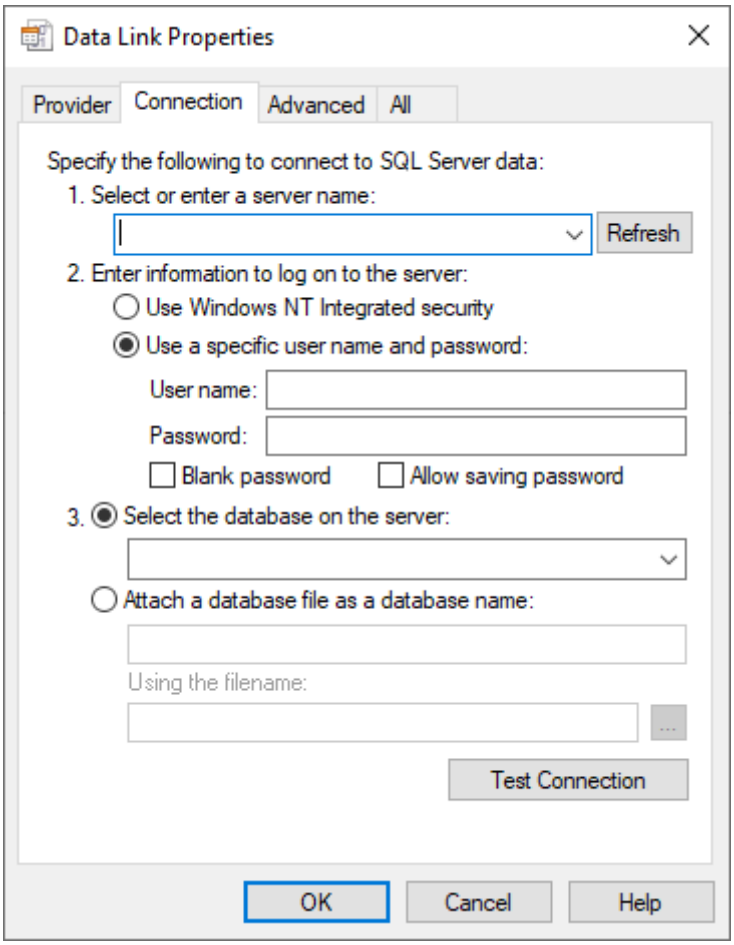


4. Click the **Provider tab** > **select a provider** > then click **Next >>**:



5. On the **Connection tab** > enter the **provider-specific connection settings** > then click **OK**:

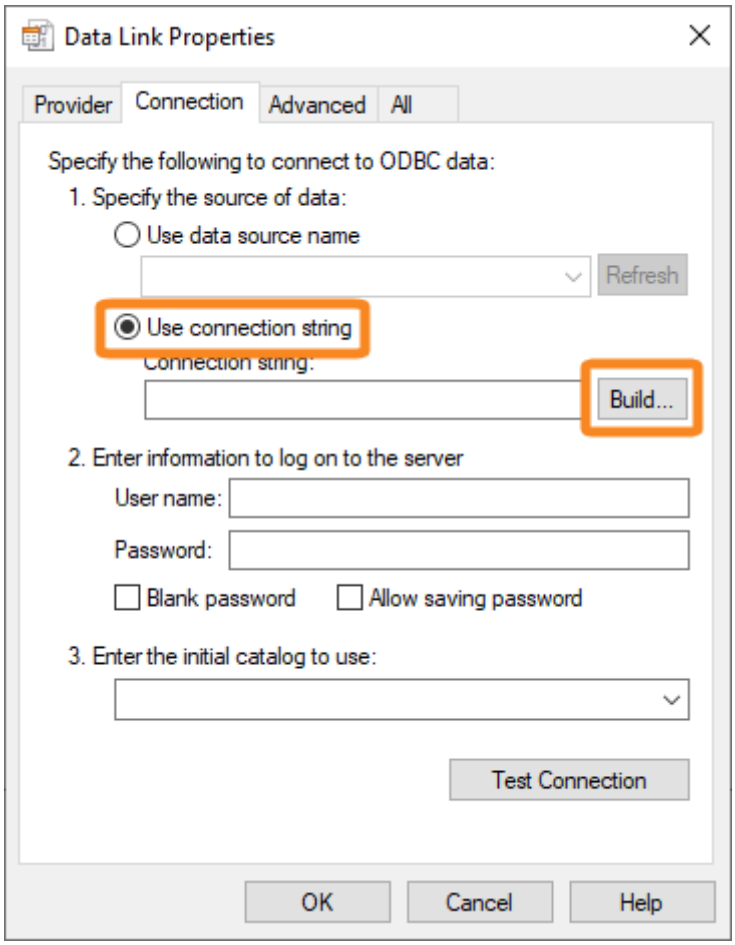
   The options on the Connection tab will vary depending on the selection you made on the Provider tab.



If you already have an ODBC connection setup you can select ODBC on the Provider tab, then on the Connection tab select the preconfigured ODBC connection from the **Specify the source of data** option.

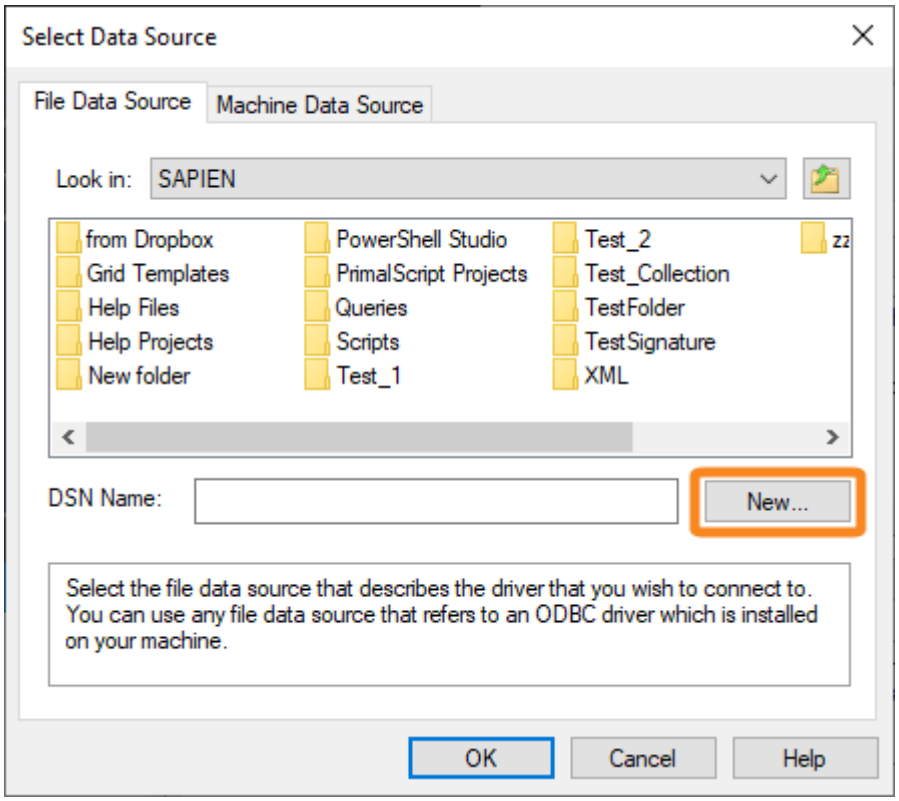
To create a new ODBC connection

- Click **Use connection string** > and then click **Build...**:



To start the ODBC connection wizard

- After selecting **Use connection string** (see above), click **New...**:

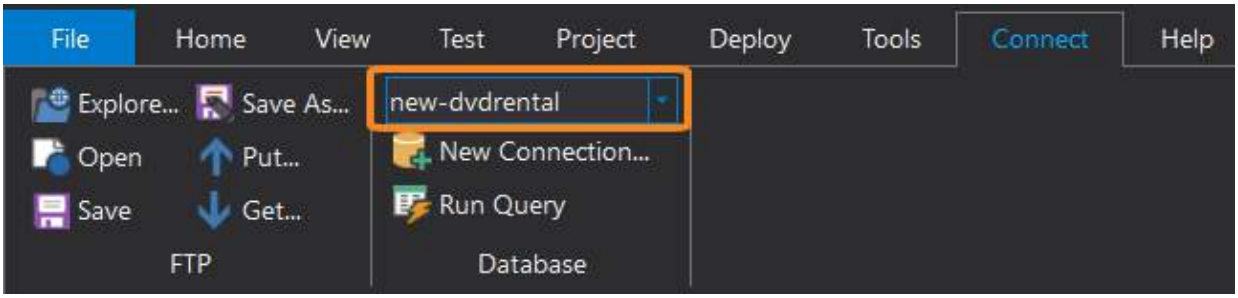


Run SQL Queries

There are many ways to run SQL queries in PrimalScript—all include the requirement to select a database connection. After you select the connection, it is used by default for all queries until you change it.

To run a SQL query

1. On the ribbon, click **Connect** > in the Database section, **select a connection** from the connections drop-down list:

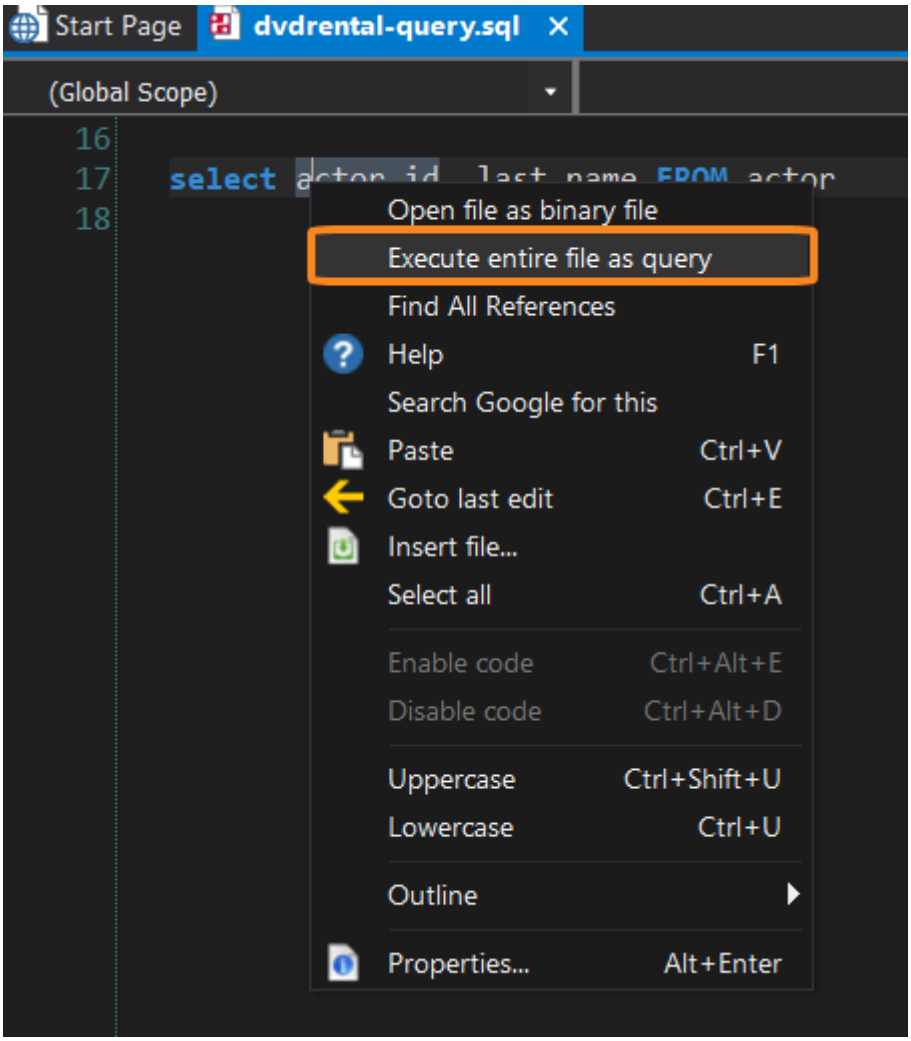


   PrimalScript uses this connection for all queries until you select a different one.

2. Click **Run Query**.

-OR-

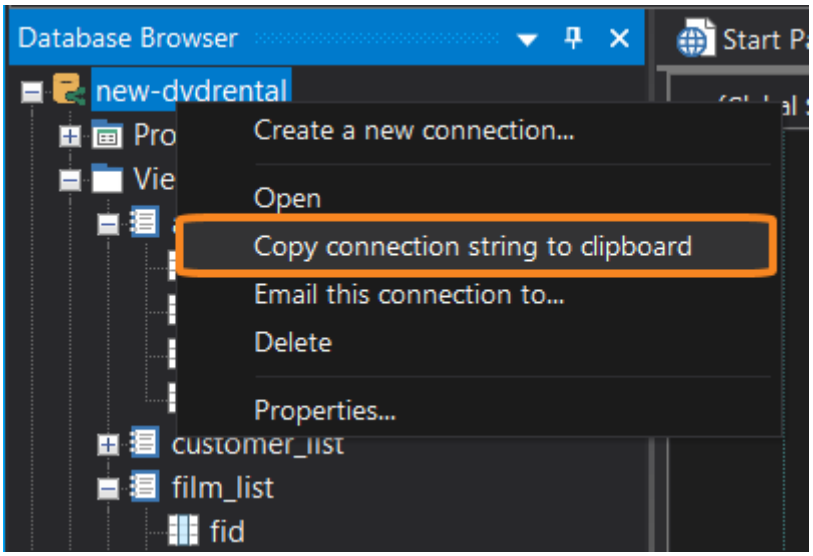
1. On the ribbon, click **Connect** > in the Database group, **select a connection** from the drop-down list.
2. In the editing pane, right-click and then select **Execute entire file as query**.



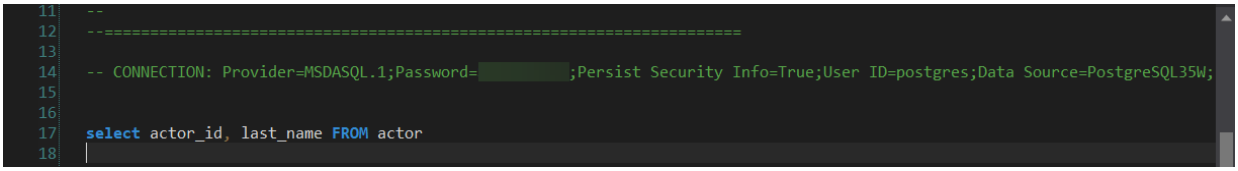
This option appears in the context menu only after you have selected a connection.

-OR-

1. In the Database Browser, right-click a connection and then click **Copy connection string to clip-board**.



2. **Paste the connection string on the first line of your script**. The comment must begin with "**Connection:**":



A connection string in a script takes precedence over the selection in the Connection box on the Tools tab.

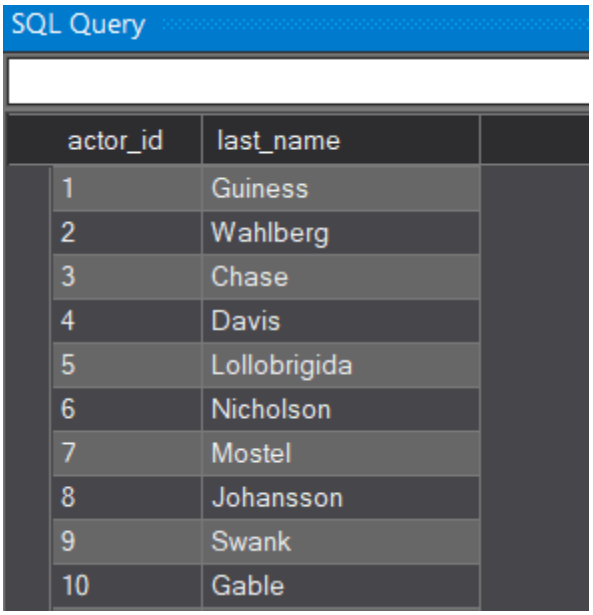
To run the queries in a SQL file

1. **Select a connection** by using any of the methods described above.
2. Right-click each query and then click **Execute select as query**.

You can run only one SQL query at a time.

Query Results

Query results are displayed in the SQL Query pane:



actor_id	last_name
1	Guiness
2	Wahlberg
3	Chase
4	Davis
5	Lollobrigida
6	Nicholson
7	Mostel
8	Johansson
9	Swank
10	Gable

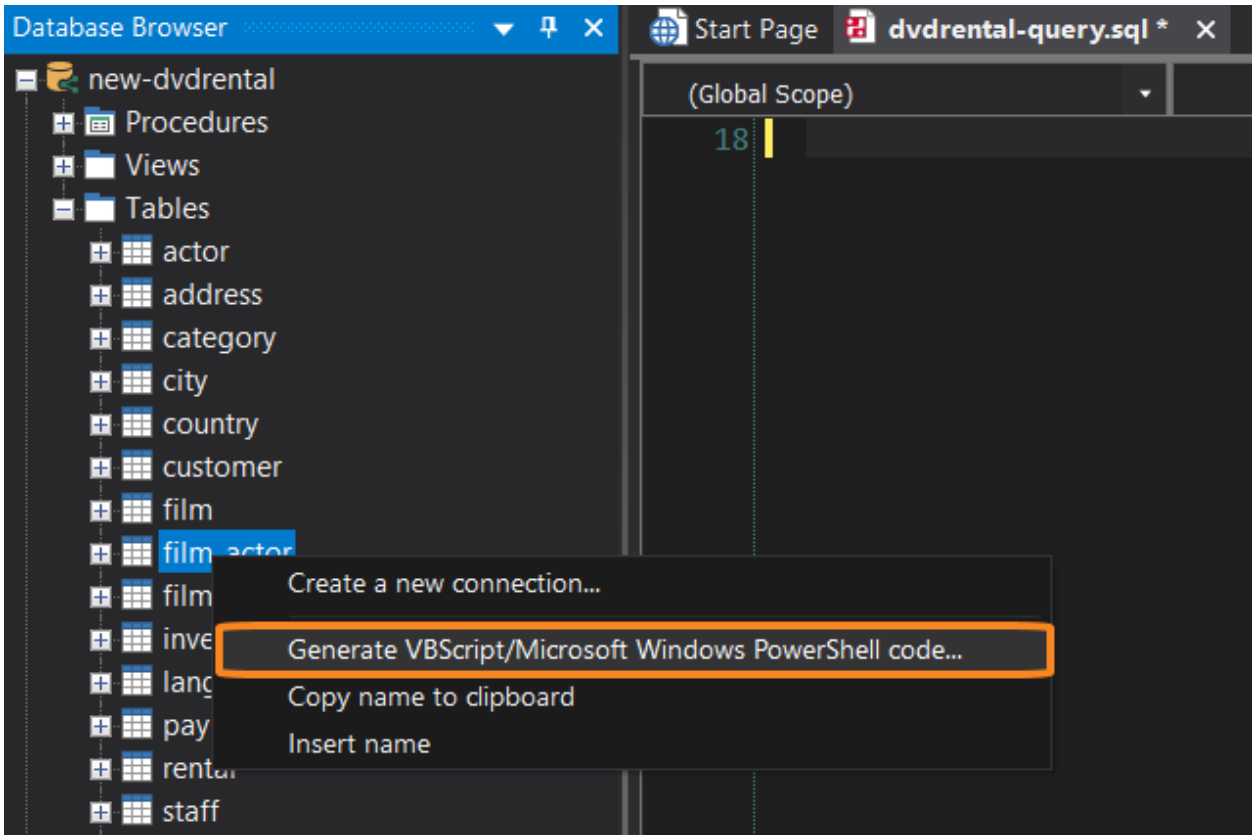
ADO Wizard

The ADO Wizard generates VBScript or Windows PowerShell script for your database connection.

```
' #region Database query, generated code

On Error Resume Next
'Database connection
Dim objCn
Set objCn = CreateObject("ADODB.Connection")
objCn.Open "Provider=MSDASQL.1;Password=          ;Persist Security Info=True;User ID=postgres;

'Query data
Const ADO_Dynamic = 2
Const ADO_FwdOnly = 0
Const ADO_Keyset = 1
Const ADO_Static = 3

Dim rs_film_actor, strSQL
strSQL = "Select * from film_actor"

Set rs_film_actor = CreateObject("ADODB.RecordSet")
rs_film_actor.Open strSQL, objCn, ADO_FwdOnly

'Go through data
Do Until rs_film_actor.EOF
    WScript.Echo rs_film_actor("actor_id")
    WScript.Echo rs_film_actor("film_id")
    WScript.Echo rs_film_actor("last_update")
    rs_film_actor.MoveNext
Loop
'Close database connections
rs_film_actor.Close
objCn.Close
On Error GoTo 0
' #endregion
```

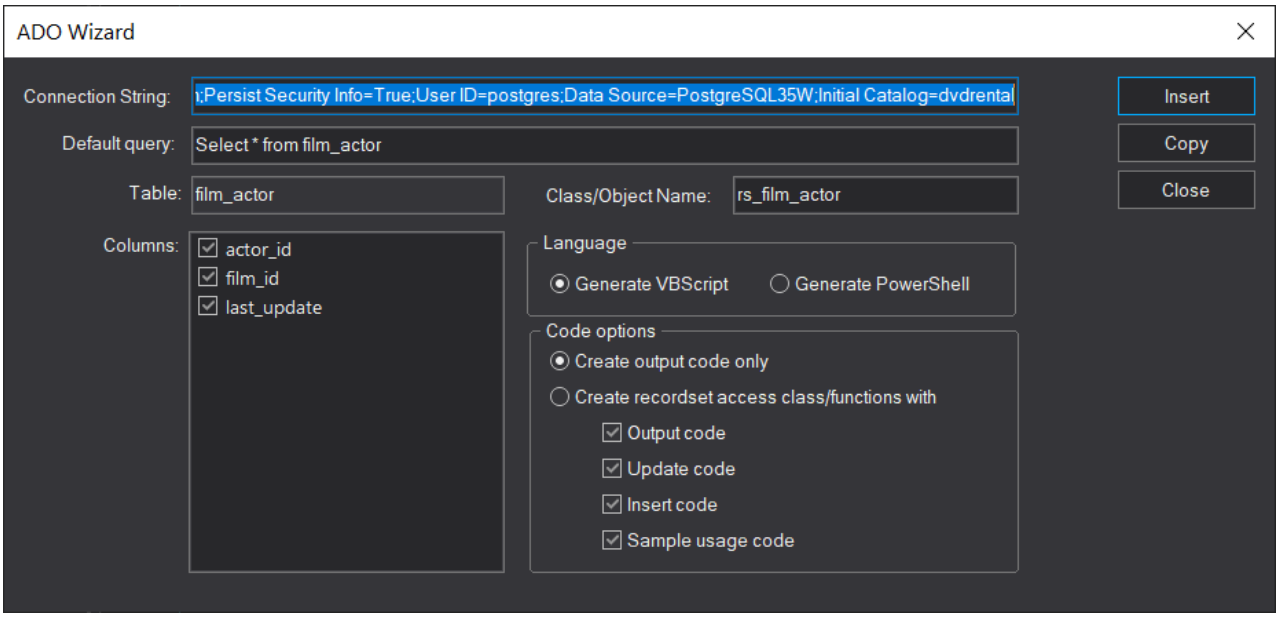
To start the ADO Wizard

- In the Database Browser, under the Tables node, right-click a table name and then click **Generate VBScript/Powershell code**.



Tips for using the ADO Wizard

- After the code is inserted, you can change connection string details in the wizard or in the editing pane.
- The default query uses the table you clicked to start the wizard. You can change the table name, but you cannot refresh the Columns list to reflect the change. To generate columns a different table, close the wizard, click the new table, and open the wizard again.
- You can shift-click to select multiple columns from the Columns list.



Use explicit column names

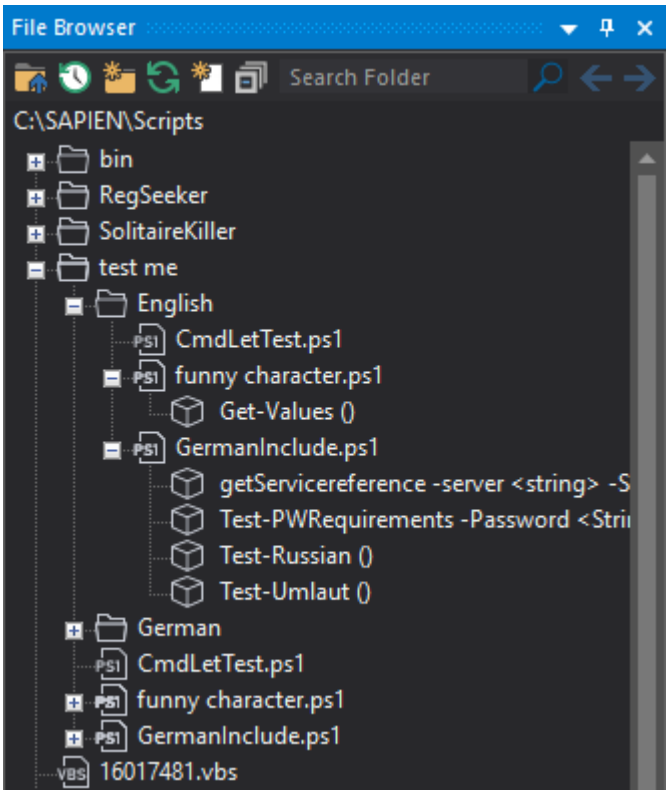
Although the ADO wizard makes it easier to create SQL connections in your VBScript or Windows Powershell scripts, there's no substitute for solid coding standards. Querying a database properly is very important in any scenario.

The ADO Wizard creates queries with " select * ". Always replace the " * " with an explicit list of the columns that you need. An explicit list insures that the DBAs can index your query properly and your code will be more reliable and perform better. Also, If columns are added to the table, they are not automatically added to your query. This is especially important when extra columns have a large data type like image or varchar(max) that can add a significant delay to your query with no benefit. Avoiding " select * " is considered good database practice in every relational database engine.

For a more feature rich database browser and visual query builder, try PrimalSQL from SAPIEN Technologies.

6.4 File Browser

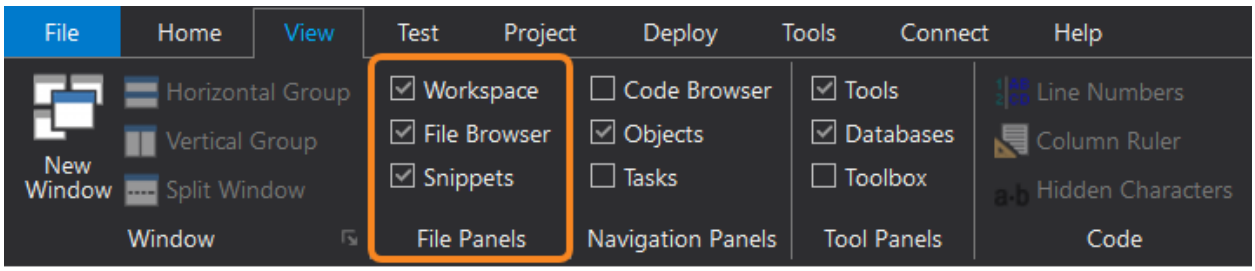
The File Browser provides direct access to files and folders on your hard drive. You can create new files and folders, search for files, open files in PrimalScript, and easily navigate between frequently used locations. You can even view and manage the security state 93 of your files.



Accessing the File Browser

To show / hide the File Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):

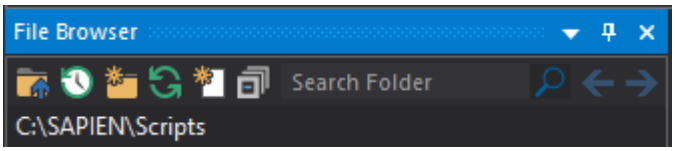


OR-

- Execute the keyboard shortcut ***Ctrl*** + ***Alt*** + ***F***.

File Browser - Buttons and Search

There are eight buttons, a search box, and a location field at the top of the File Browser:



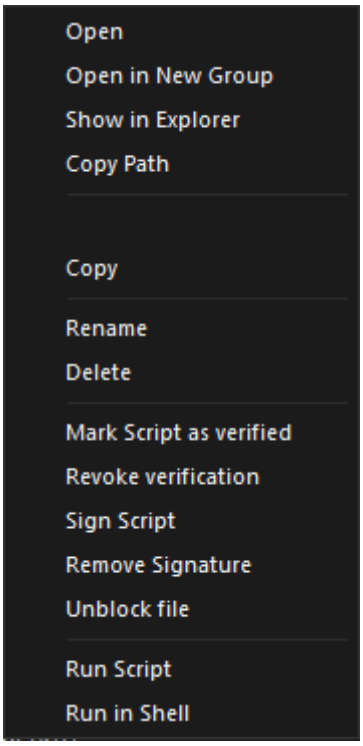
From left to right:

- **Select Folder**
  Select a different folder in your environment as the current folder.
- **Previous Folder**
  Select a previously used folder as the current folder.
- **New Folder**
  Create a new empty folder in the current directory.
- **Reload**
  Refresh the current folder.
- **New File**
  Create a new file.
  If the new file name has a known extension, the associated template is automatically applied.
- **Collapse Nodes**
  Collapse all expanded nodes.
- **Search Folder**
  Scans the current directory and displays all of the matching results.
- **Previous Item**
  Go to the previous item matched by the current search.
- **Next Item**
  Go to the next item matched by the current search.
- **Location**
  Displays the current directory.

File Browser - Context Menu Options

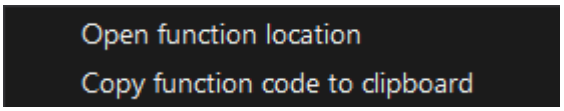
The context menu options will vary depending on what is selected in the File Browser (a folder, a file, a function, etc.) If a file is selected, the options will depend on the file type.

Right-click on a folder or file to display the following options:



- **Open**
  Opens the file in the Editor.
- **Open in New Group**
  Opens the file in a new Editor group.
- **Show in Explorer**
  Opens the corresponding file or folder location in File Explorer.
- **Copy Path**
  Copies the full file or folder path to the clipboard.
- **Copy**
  Copies the highlighted file to the clipboard.
- **Rename**
  Highlights the file or folder name for editing.
- **Delete**
  Deletes the highlighted file or folder.
- **Mark Script as Verified, Revoke Verification, Sign Script, Remove Signature, Unblock File**
  See Security State of Scripts and Files 93.
- **Run Script**
  Runs the script without loading it in the editor.
- **Run in Shell**
  Runs the script in the Windows shell.

Right-click on a function to display the following options:

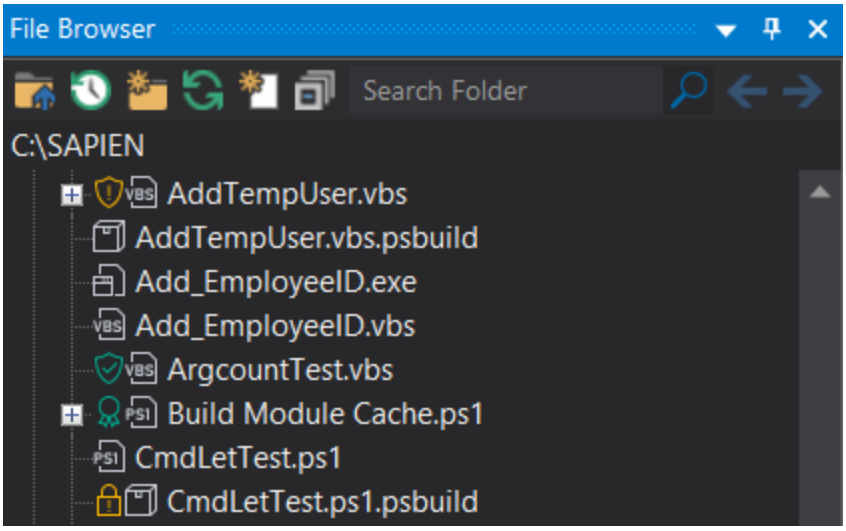


- **Open function location**
  Opens the file in PrimalScript at the function's location.
- **Copy function code to clipboard**
  Copies the function code from the file to the clipboard.

  You can also drag and drop the function code directly to your open script.

Security State of Scripts and Files

The File Browser in PrimalScript and the SAPIEN Script Explorer both indicate the state of your scripts and other files by displaying a color-coded icon before the file name:



Security Icons:

- **Blocked**
  Recently downloaded from the internet or other source.

  Windows applies an extended attribute to mark the file as "blocked".
- **Signed**
  Signed with a certificate trusted by your computer. If there is a time stamp, it has a date and time before the certificate expires. The stored checksum of the file matches the current file.
- **Verified**
  Marked as *verified* 94.
- **Modified**
  Modified after being *signed* or *verified* 94.

- **Blank (no icon)**
  The file does not have an extended attribute. The file does not have a valid signature, has not been verified, and has not been blocked.

These color-coded security icons are also included in SAPIEN's Script Explorer, which is also a free download from your SAPIEN Account page.

About Script Verification

Script *verification* creates a checksum over your script and stores it in an extended attribute. If someone else modifies the script, the checksum no longer matches.

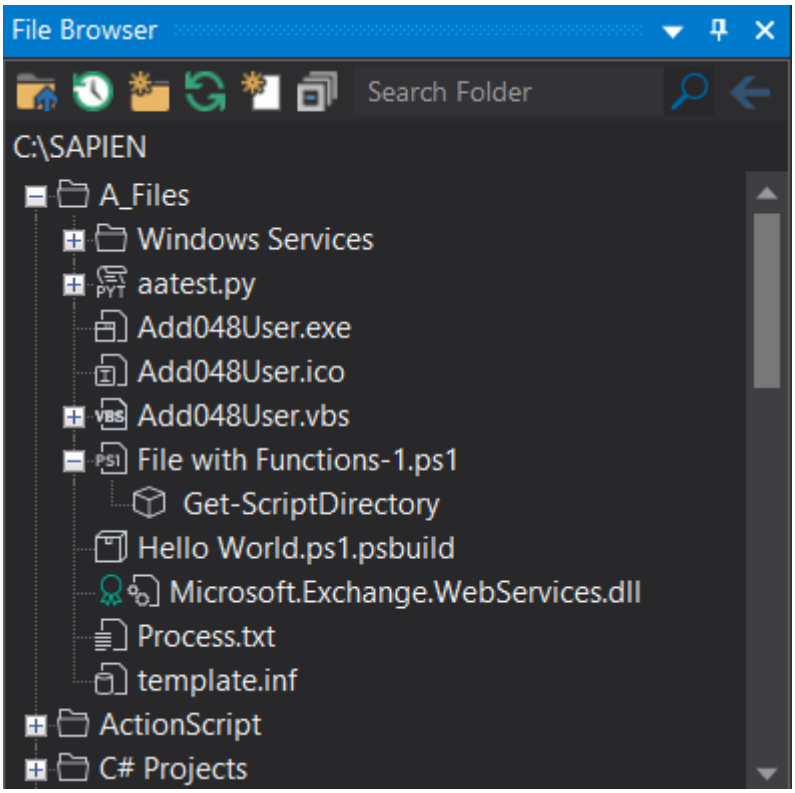
Script verification has three states:

- **Not Verified**
  The file does not have an extended attribute.
- **Verified**
  You have marked the file as *Verified*.
- **Modified**
  A file marked as *Verified* has subsequently been modified.

Files that have been marked as "verified" essentially have the same checksum feature of a digital signature, without the actual signature. Therefore, although script verification does not have the trusted standard of a digital certificate, it is the next best.

Using the File Browser

The File Browser has node indicators in front of some files that designate the file type. PrimalScript will also parse the files in your selected folder in the background and list any contained classes, methods, or functions:



This applies to all file types PrimalScript knows, not just PowerShell files.

To open a file

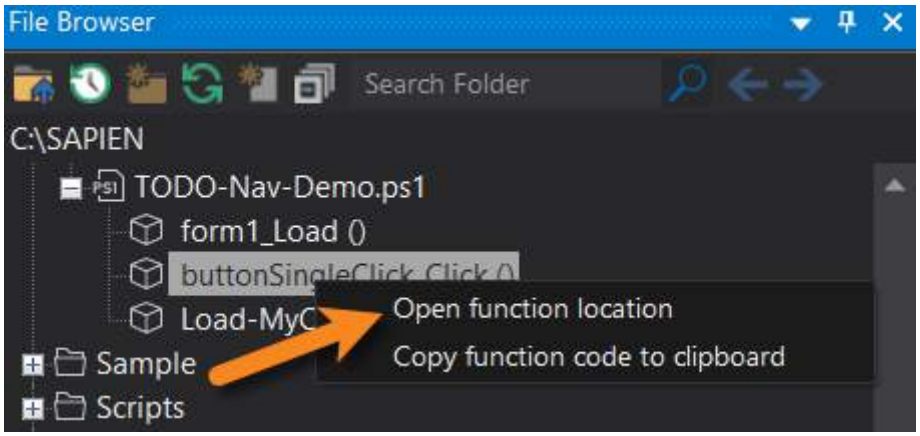
- Double-click a file in the File Browser to open it in PrimalScript.

-OR-

- Drag a file from the File Browser and release it anywhere on the PrimalScript document or ribbon area.

To open a file at the function's location

- Right-click a function in a file and select **Open function location**

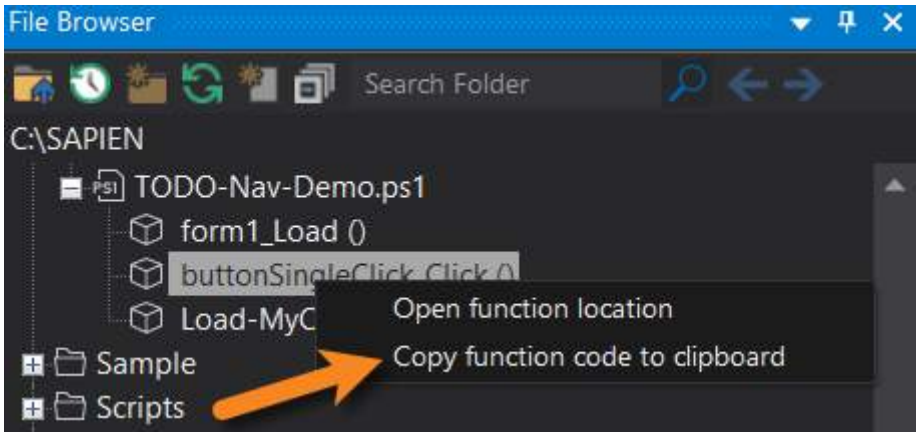


-OR-

- Double-click a function in a file.

To copy the function

- Right-click and select **Copy function code to clipboard**



The function is copied from the file into the Windows clipboard.

You can also drag the function code from the File Browser and drop it into your open file in the PrimalScript Editor.

PrimalScript will not interfere if you drag and drop between languages.

To copy the full path name

- Hold the *Shift* key, then drag and drop a file from the File Browser into your open script to drop the full file path.

If you drop a PowerShell file into your script it is enclosed in quotes and dot-sourced.

To run a script

You can run scripts directly from the File browser without having to load them in the editor first.

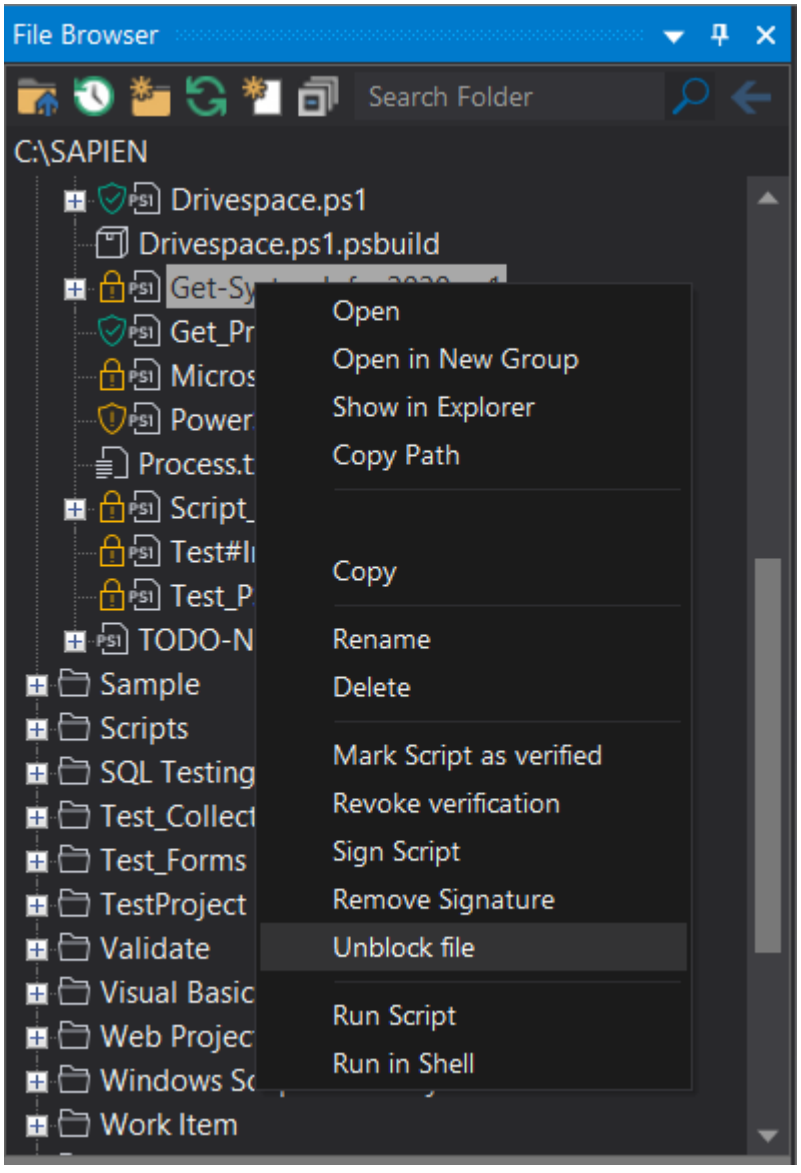
- Right-click and select **Run Script**

You can also run a script in the Windows shell.

To unblock a file

- Right-click and select **Unblock file**

Unblocking a file removes the extended attribute Windows added to the file when it was downloaded from a public or unsecure location or when it was extracted from a zip file with the same extended attribute.

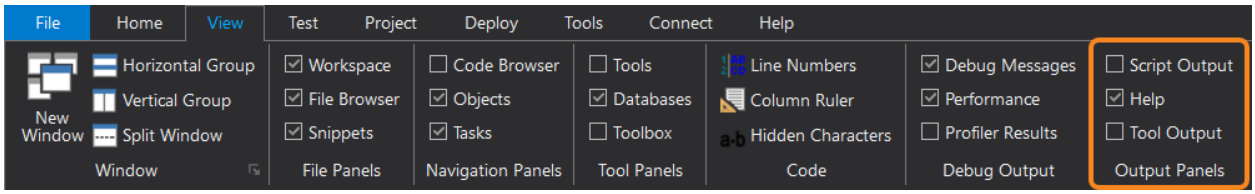


6.5 Help Panel

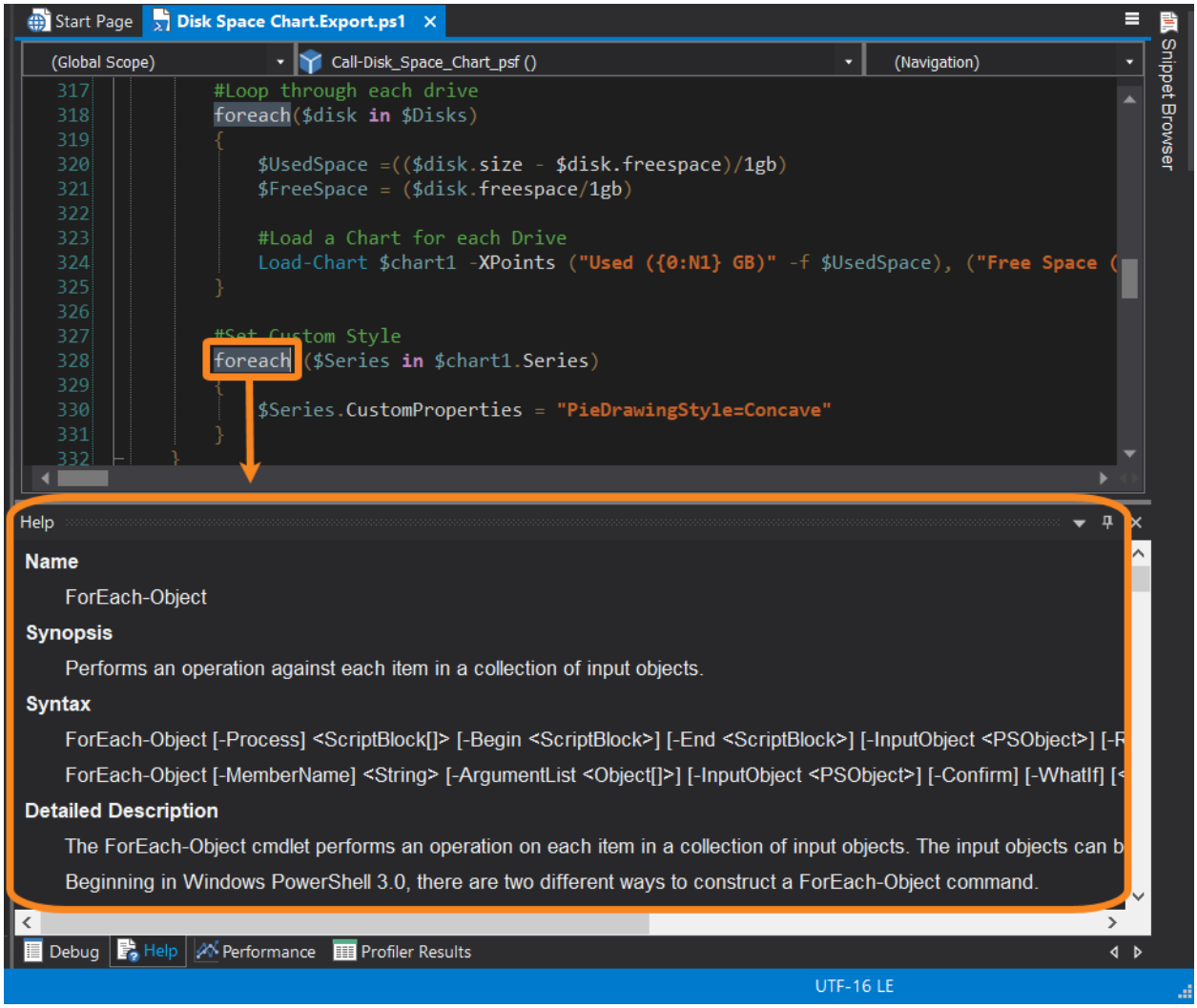
The Help panel will automatically display the help information for any item you select. This works for commands in the Object Browser, editor windows, and more.

To show / hide the Help panel

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



The context sensitive Help is displayed for the selected item:



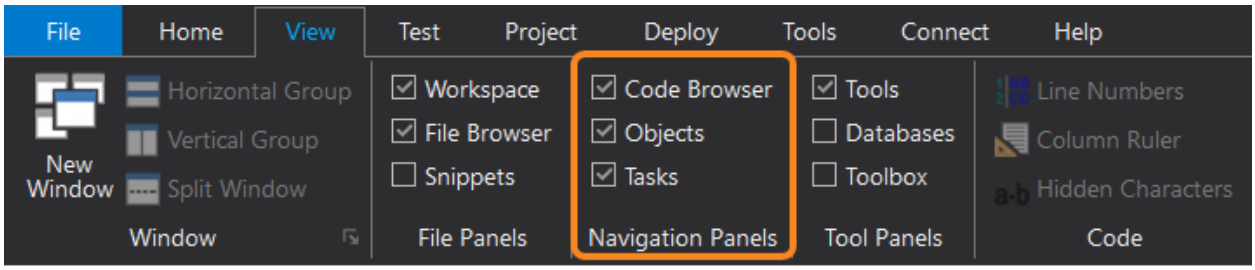
6.6 Object Browser

The Object Browser allows you to search, copy, and research COM, Windows PowerShell, WMI Classes, and Microsoft .NET Framework objects.

Accessing the Object Browser

To show / hide the Object Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):

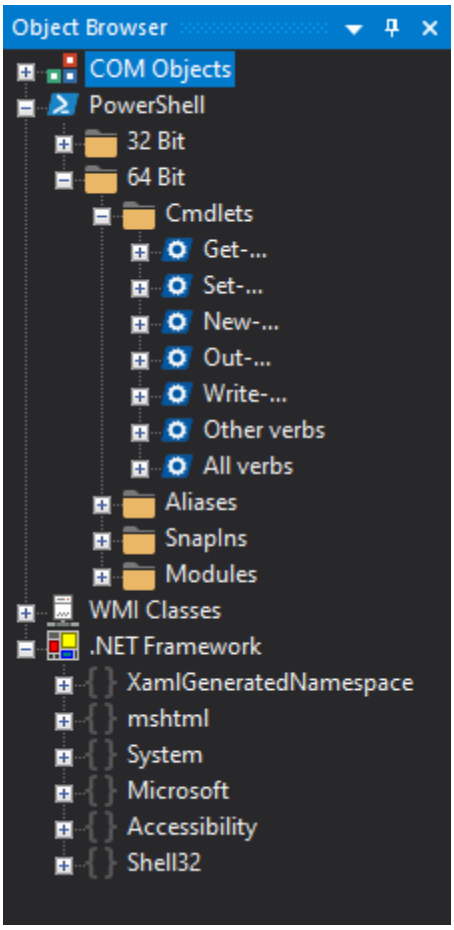


-OR-

- Execute the keyboard shortcut ***Ctrl*** + ***Alt*** + ***J***.

Using the Object Browser

Objects are displayed in collapsing menus:



To add an object to your code

- Drag an object from the Object Browser to the code editor.

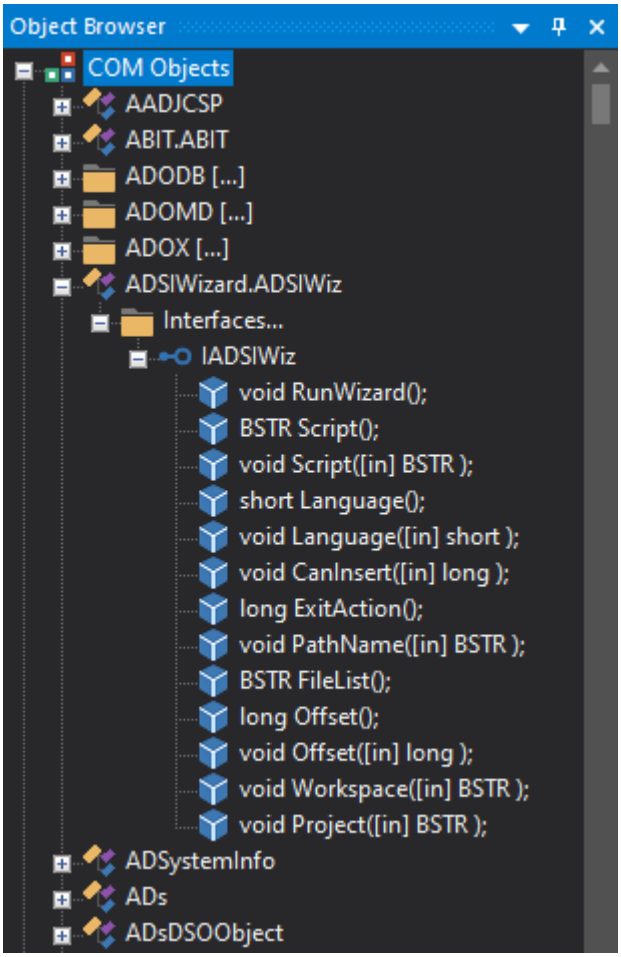
To search MSDN (Microsoft Developer) for an object

- Right-click an object and then click **MSDN Help**.
  You can drill down until you arrive at the object that you need.

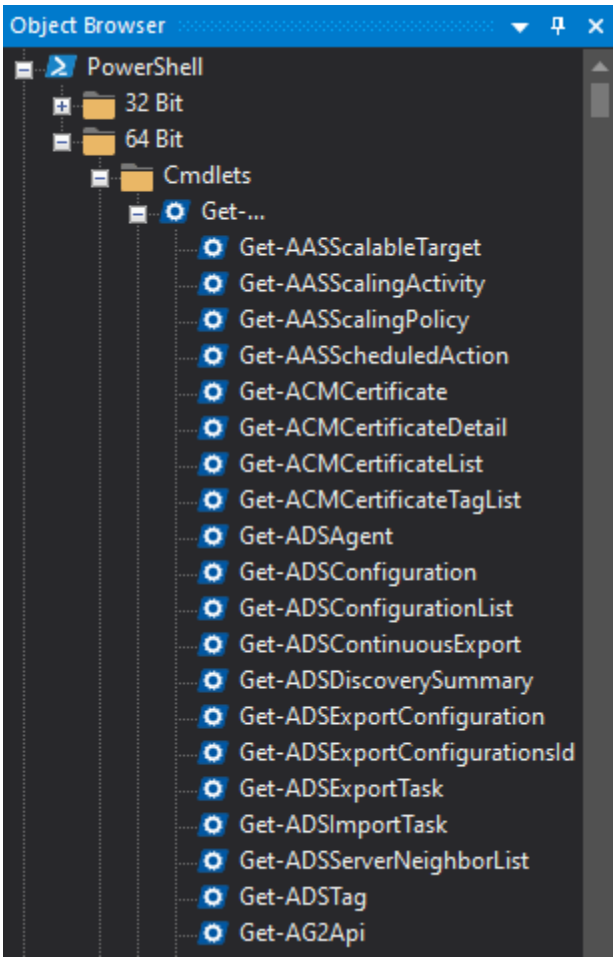
To do a Google search about an object

- Right-click an object and then click **Google this**.

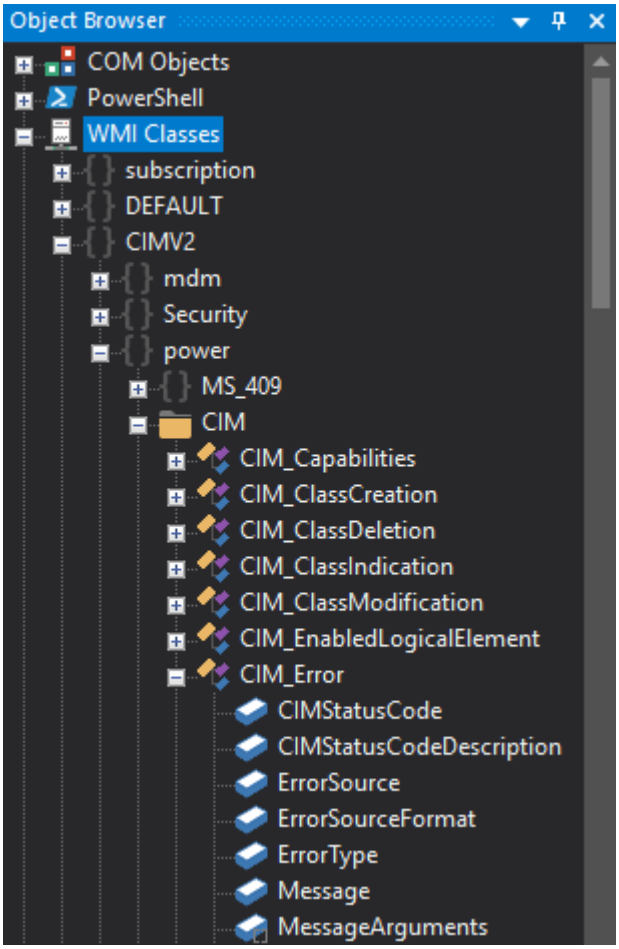
COM Objects



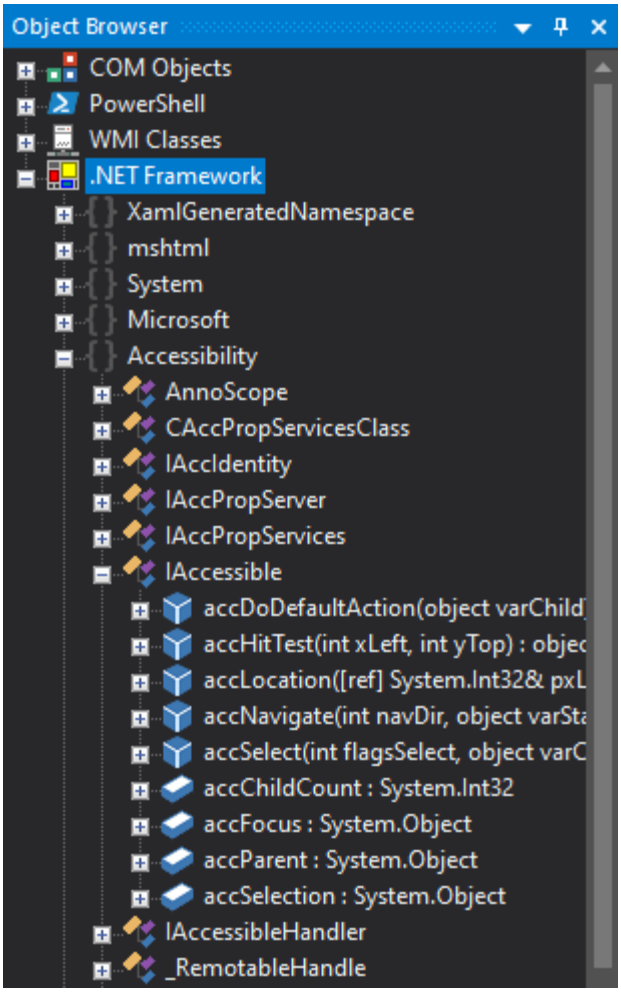
PowerShell Objects



WMI Class Objects



.NET Framework Objects

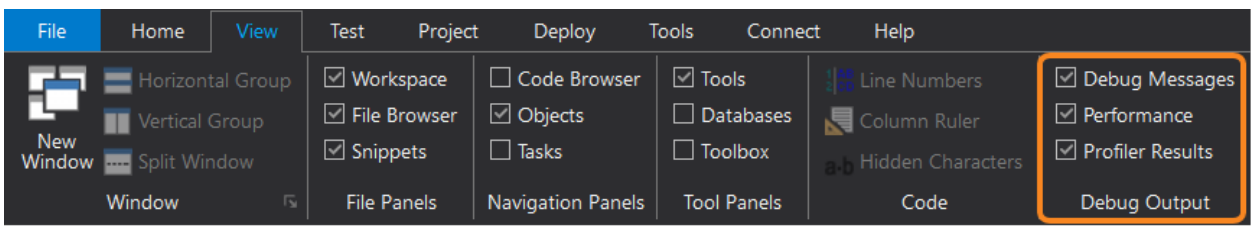


6.7 Performance Panel

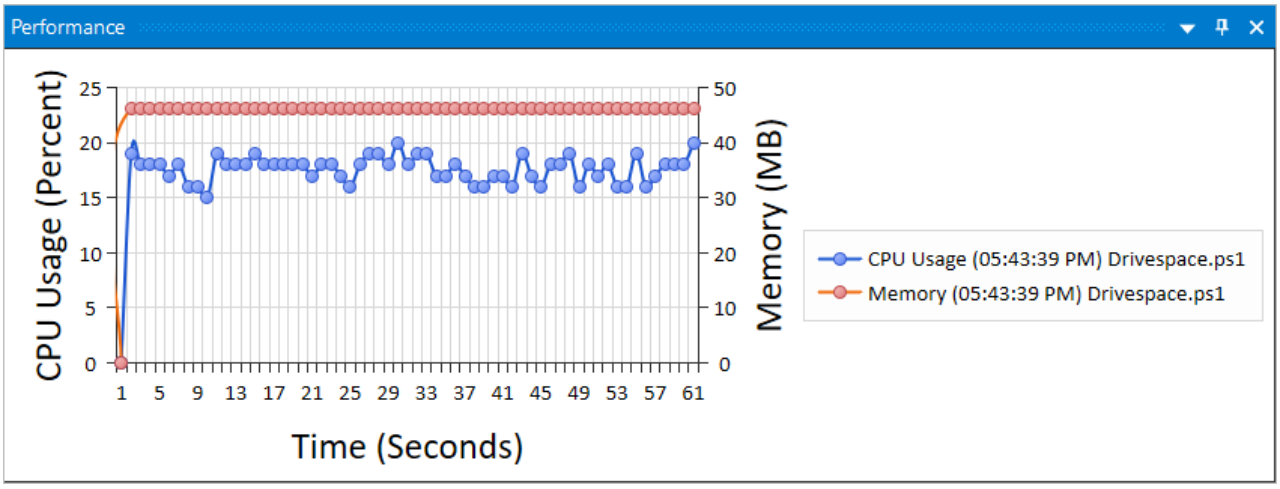
Performance data can be collected on any script run from within PrimalScript as long as you enable output redirection—this applies to PowerShell, VBScript, Perl, or any other script that you run from within PrimalScript.

To show / hide the Performance panel

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



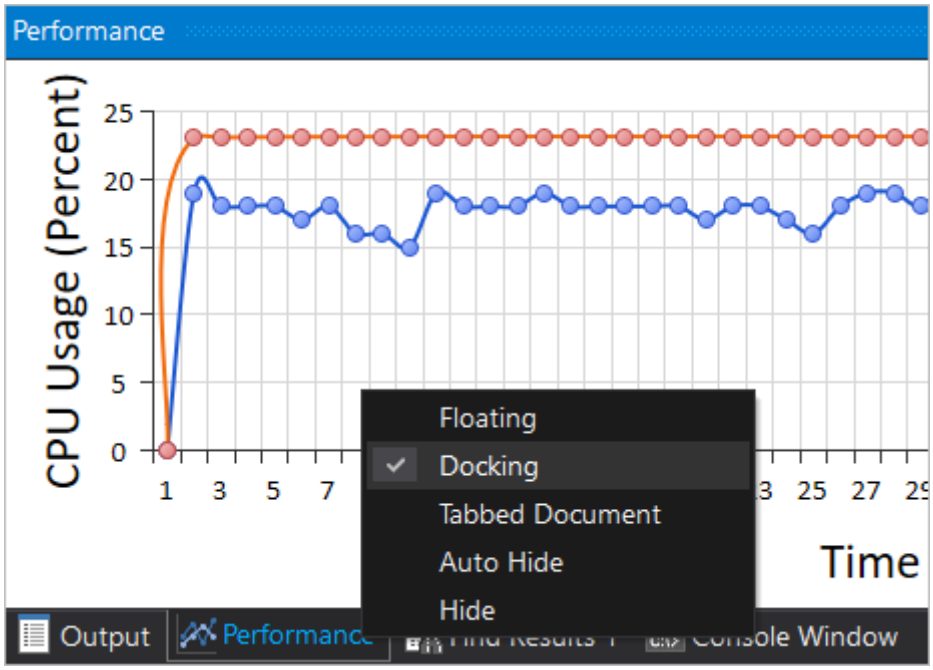
The graph in the docked Performance panel scales automatically as the script runs:

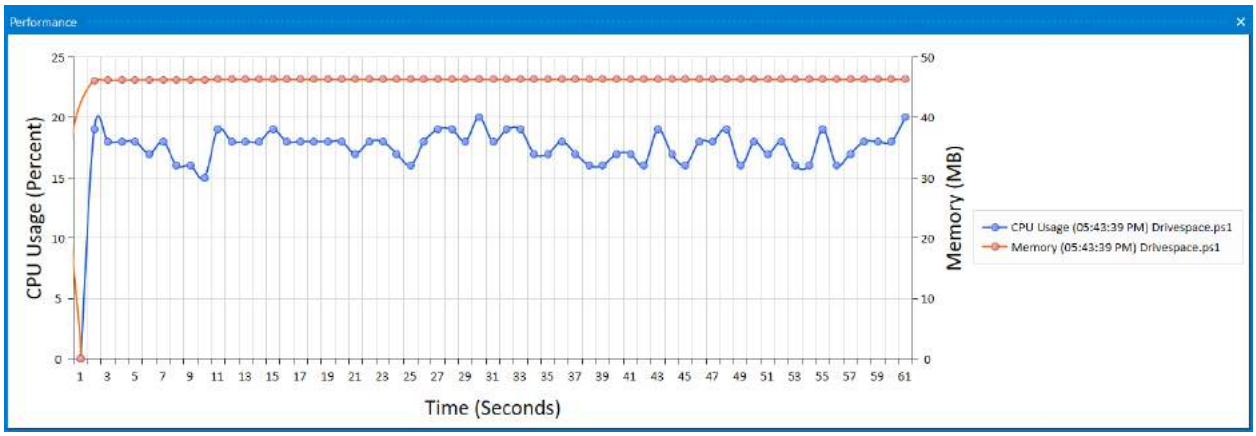


Because of the generally small height of the docked panel, some details may not be visible in the graph.

To see the performance data in greater detail

- Right-click on the Performance tab and select either **Floating** or **Docking**:

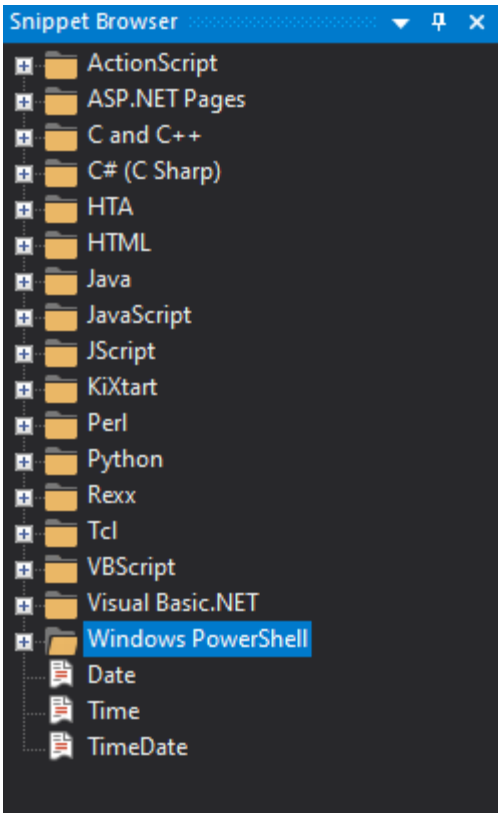




When your script finishes, the Output panel will display the peak values during script execution.

6.8 Snippet Browser

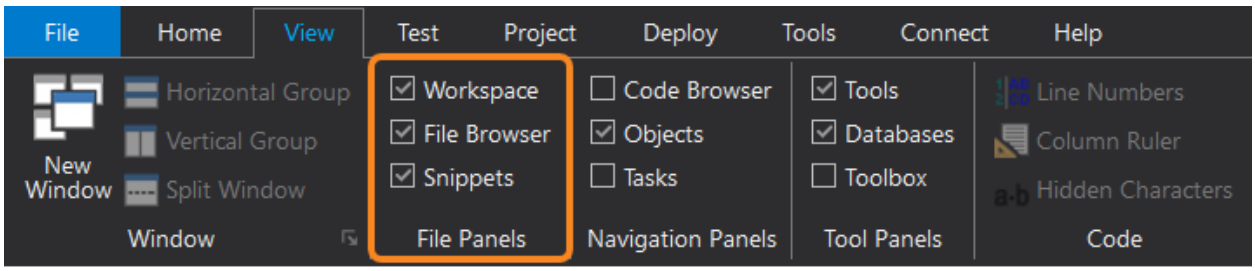
The Snippet Browser provides access to a library of built-in code snippets, organized by language. Snippets are a great way to dramatically speed up scripting time by never having to write the same script code twice. Instead, save commonly-used code as Snippets where it can be easily reused.



Accessing the Snippet Browser

To show / hide the Snippet Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



Using the Snippet Browser

To insert a snippet from the Snippet Browser

- Drag a snippet from the Snippet Browser into the code editor.

To insert a snippet from the code editor

- Place your cursor in the code editor and then double-click a snippet in the Snippet Browser.

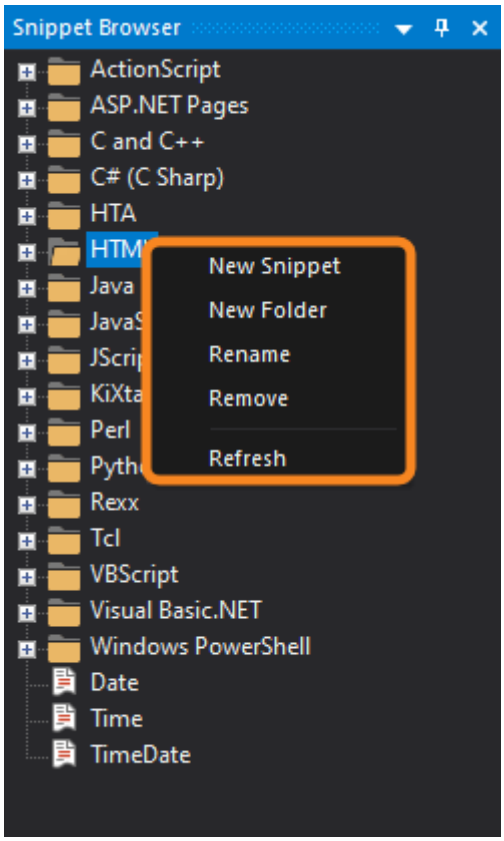
-OR-

- Place your cursor in the code editor, type the Snippet name and then press *Ctrl+J*.

  Snippets are language-specific. When using the *Ctrl+J* technique, only Snippets in the current scripting language are inserted.

To add, remove, or rename a snippet or snippet folder

- Right-click the Snippet Browser pane (not the title bar), a snippet, or a snippet folder > then select the appropriate option:

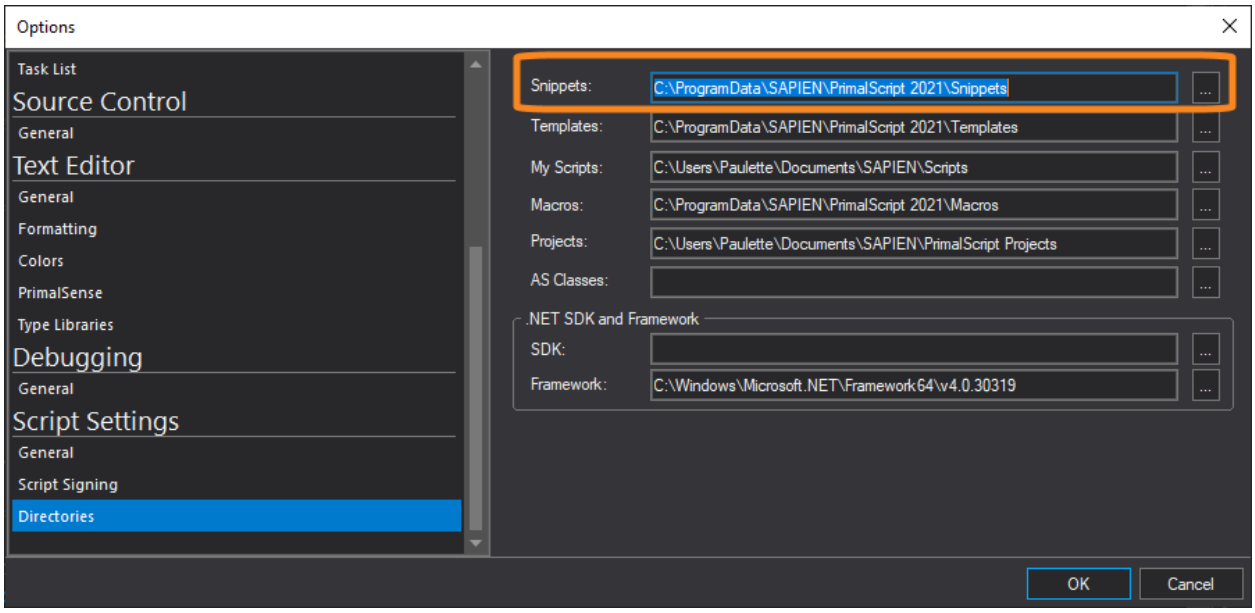


Snippet Browser - Default Source Directory

Change the snippet source directory

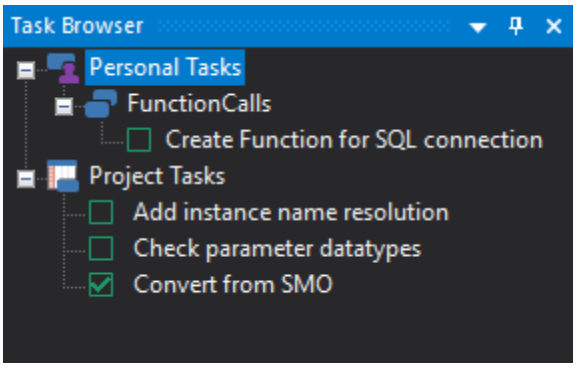
By default, PrimalScript keeps its Snippets in the **%ProgramData%\SAPIEN\PrimalScript yyyy\Snippets** directory. However, you can change the directory where PrimalScript gets the snippets that it displays in the Snippet Browser.

- Click **File** > **Options** > **Script Settings** > **Directories**, and then enter an alternate path in the Snippets field, including a UNC path.



6.9 Task Browser

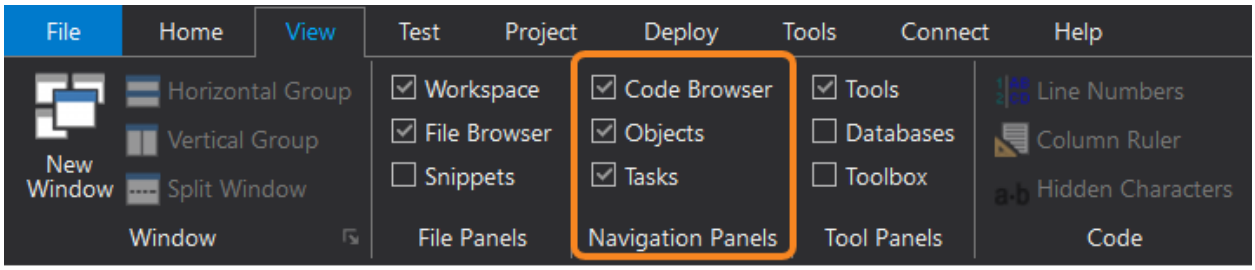
The Task Browser automatically creates a "To Do" list based on code comments and tasks that you add to your files; these tasks are listed under Project Tasks. You can also create and manage Personal Tasks within the Task Browser.



Accessing the Task Browser

To show / hide the Task Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



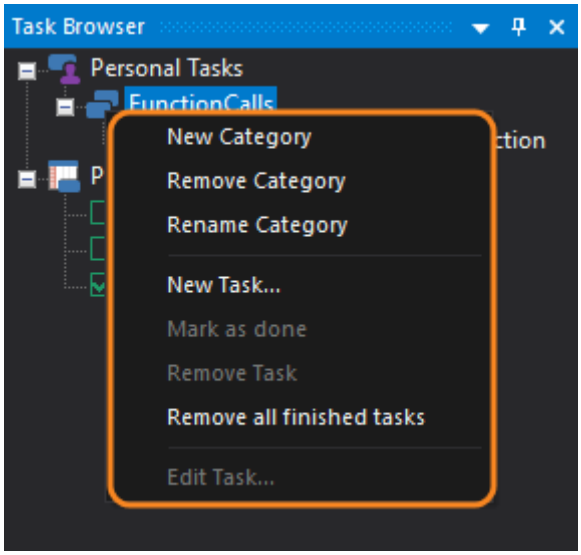
Using the Task Browser

Personal Tasks

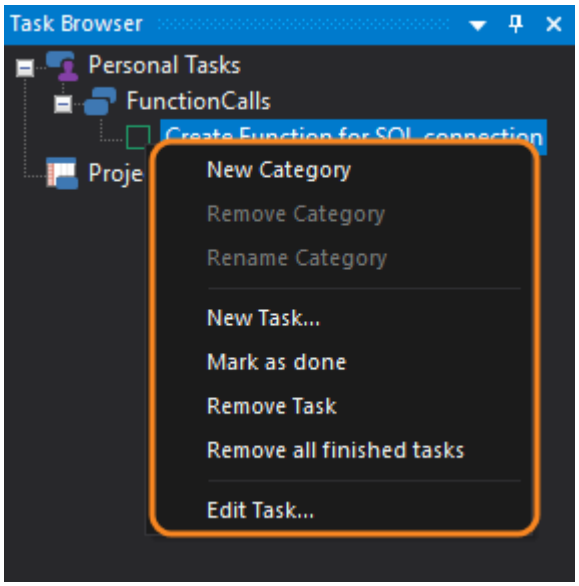
To create and manage Personal Tasks

- Right-click Personal Tasks and create a New Category, then right-click the category and create a New Task. To manage your categories and tasks, right-click a category or task and select an option from the context menu.

  The context menu options will vary depending on whether you right-clicked a category or a task.



Category Context Menu



Task Context Menu

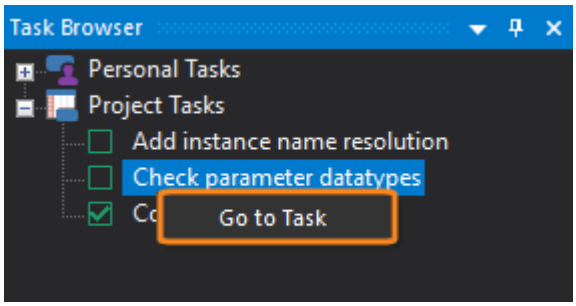
Project Tasks

PrimalScript collects tasks for the **Project Tasks** list from code comment tokens, such as TODO and DONE:

```
4
5  #TODO: Add instance name resolution
6  #TODO: Check parameter datatypes
7  #DONE: Convert from SMO
8
```

To go to a task

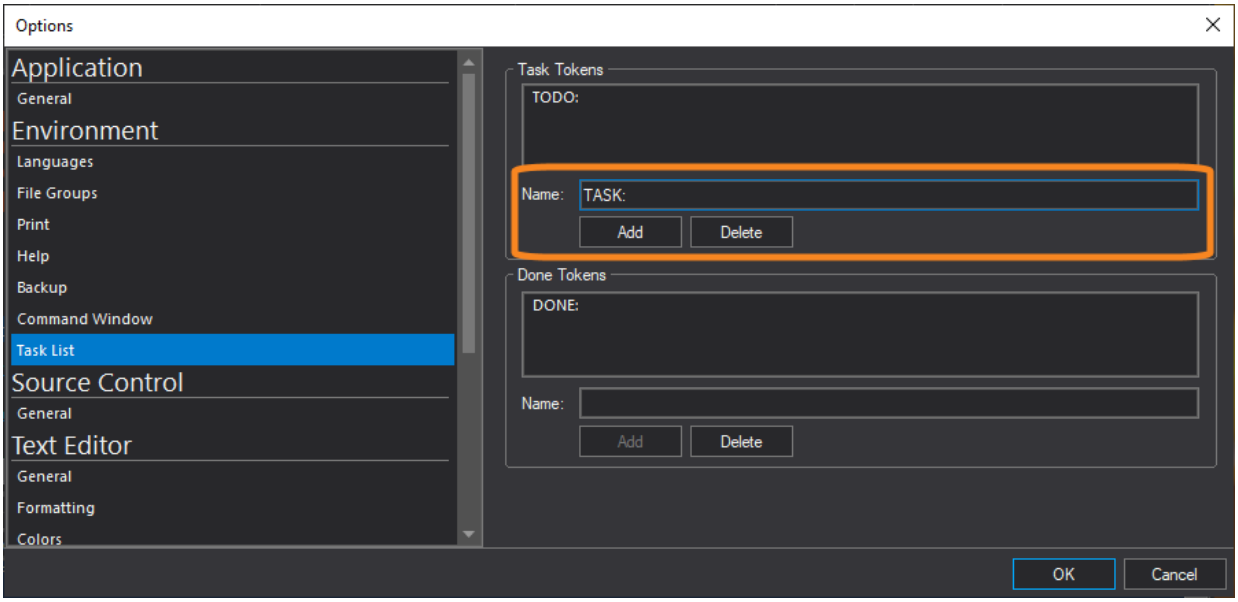
- In the Task Brower, under Project Tasks > **double-click a Task** or **right-click a Task** and select **Go to Task**.



PrimalScript opens the file and places your cursor on the line with the task token.

To add a task token

1. Click **File** > **Options** > **Environment** > **Task List**.
2. In either Name box, **type a name**, click **Add**, then click **OK**.

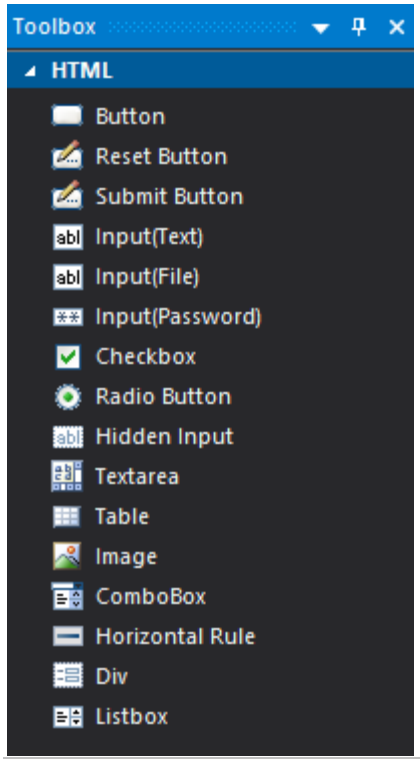


To delete a task token

1. Click **File** > **Options** > **Environment** > **Task List**.
2. Select a token, click **Delete**, and then click **OK**.

6.10 Toolbox

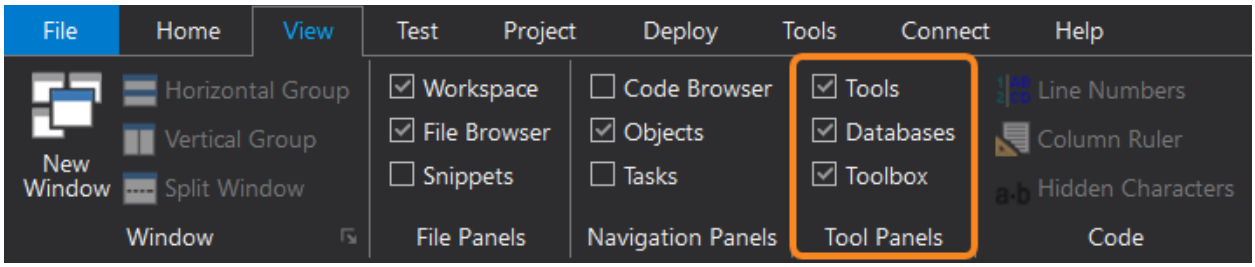
The Toolbox offers a set of HTML constructs that can be inserted into your current HTML work area.



Accessing the Toolbox

To show / hide the Toolbox

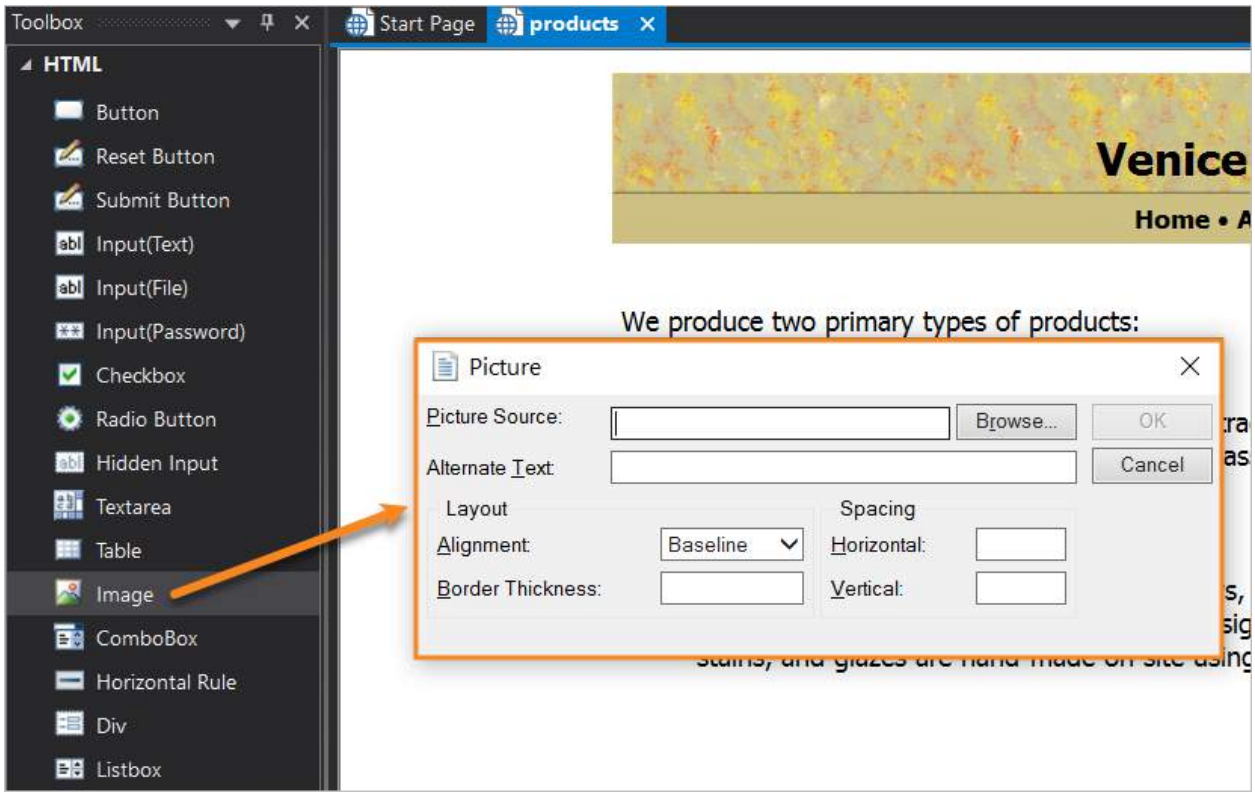
- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



Using the Toolbox

To insert an element from the Toolbox

- Place your cursor in the HTML page and then, in the Toolbox, click the object:



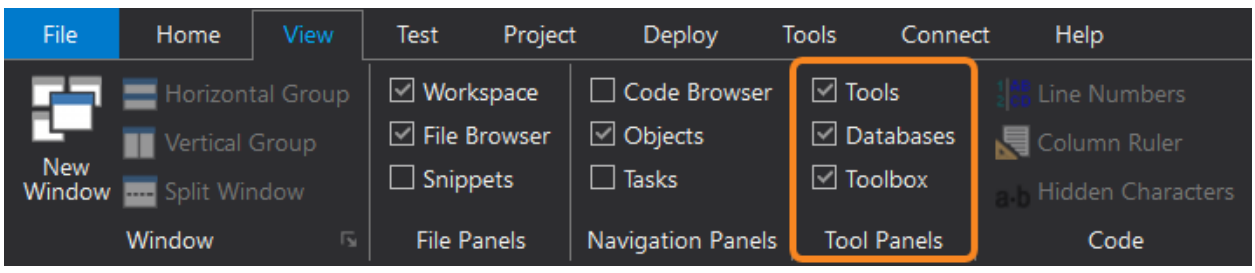
6.11 Tools Browser

The Tools Browser provides access to commonly used tools of various types, and provides convenient access to tools you might need while working on projects. The tools that appear in the Tools Browser are script-related tools that were added by scanning your system when PrimalScript is first installed.

Accessing the Tools Browser

To show / hide the Tools Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



OR-

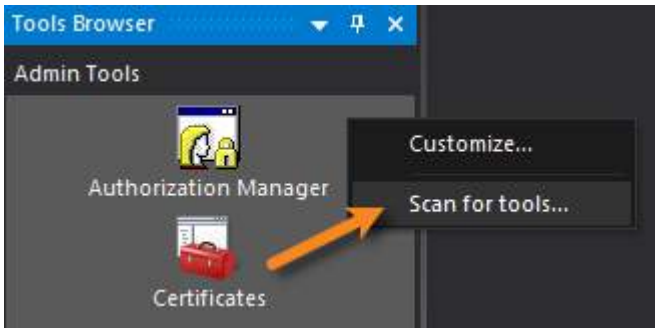
- Execute the keyboard shortcut ***Ctrl*** + ***Alt*** + ***X***.

Scanning for tools

The Tools Browser is populated by scanning your system when PrimalScript is installed, but you can rescan at any time. Rescanning will add additional tools to the Tools Browser after the initial scan.

To rescan for tools

- **Right-click** anywhere in the Tools Browser panel (not the title bar) and then click **Scan for tools...**.

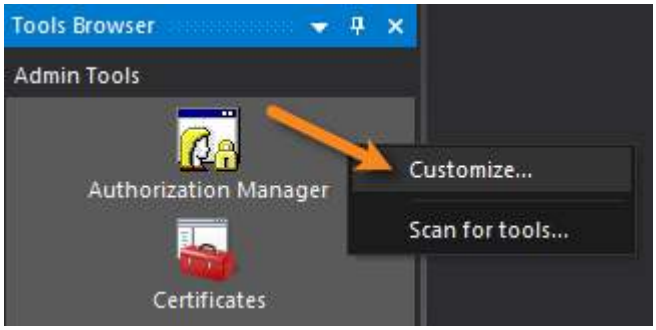


Customizing the Tools Browser

You can customize the Tools Browser in a number of ways. In addition to adding and removing tools, you can also add, remove, and rename groups. You can also change the tool display.

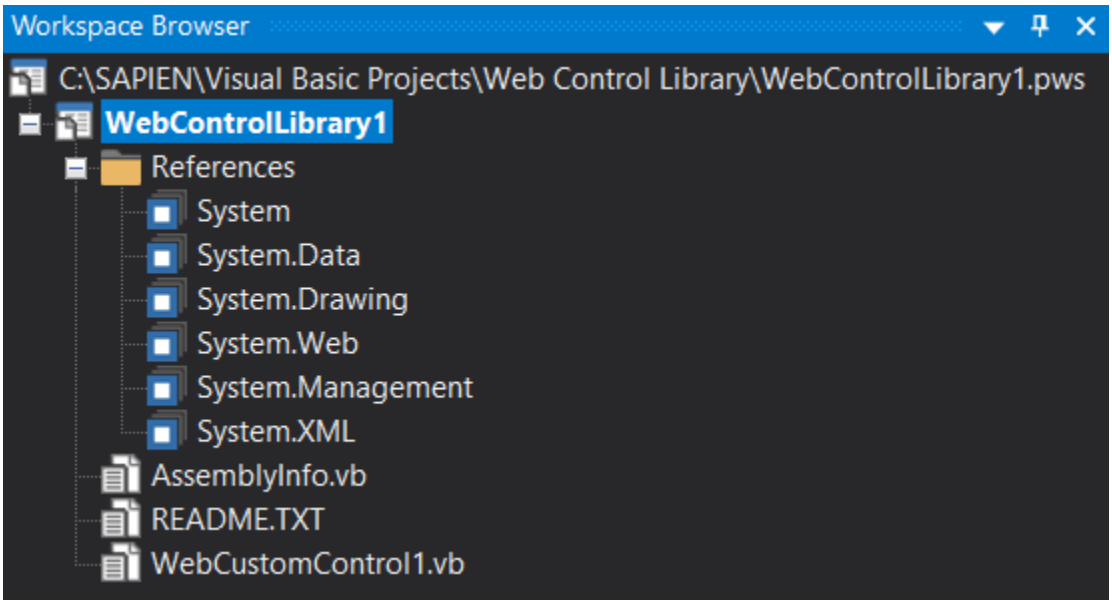
To customize the Tools Browser

- **Right-click** in the the browser panel (not the title bar) and then click **Customize...**:



6.12 Workspace Browser

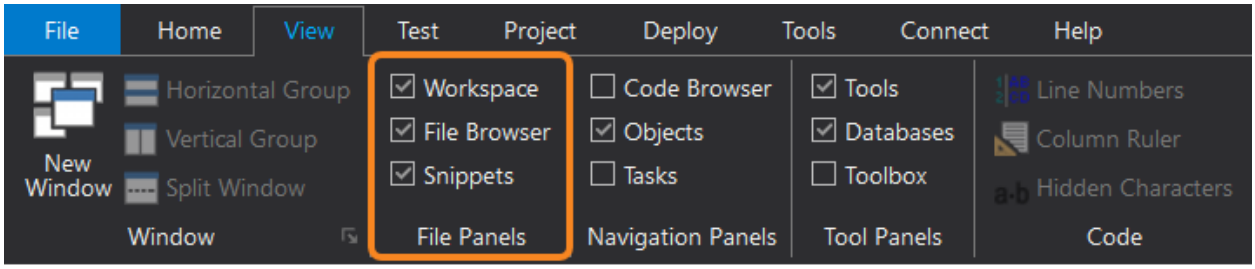
The Workspace Browser helps you to keep projects organized by displaying all of the items associated with a project, including supporting files.



Accessing the Workspace Browser

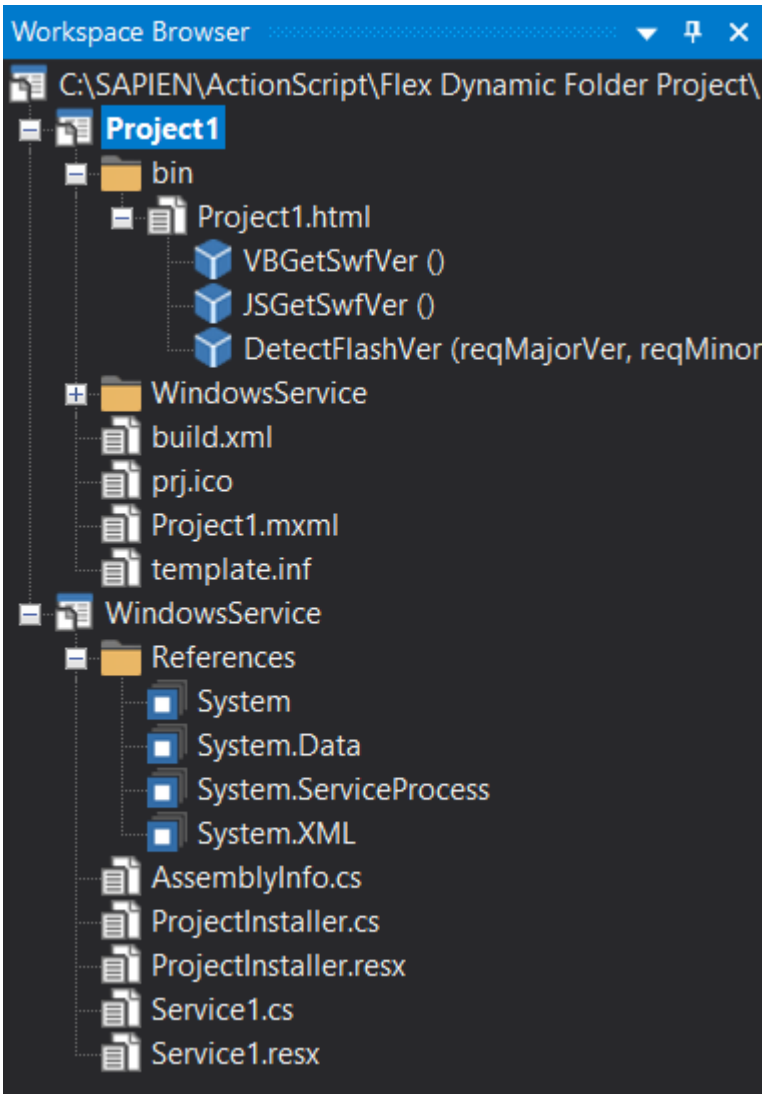
To show / hide the Workspace Browser

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):

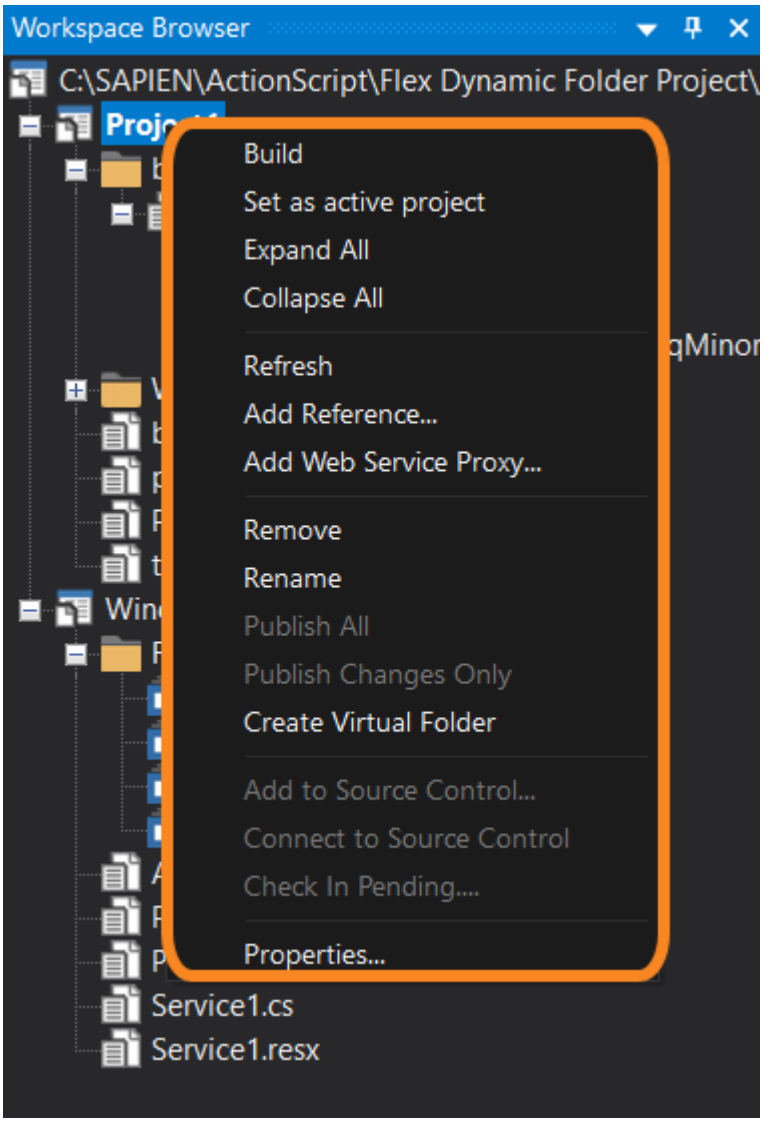


Using the Workspace Browser

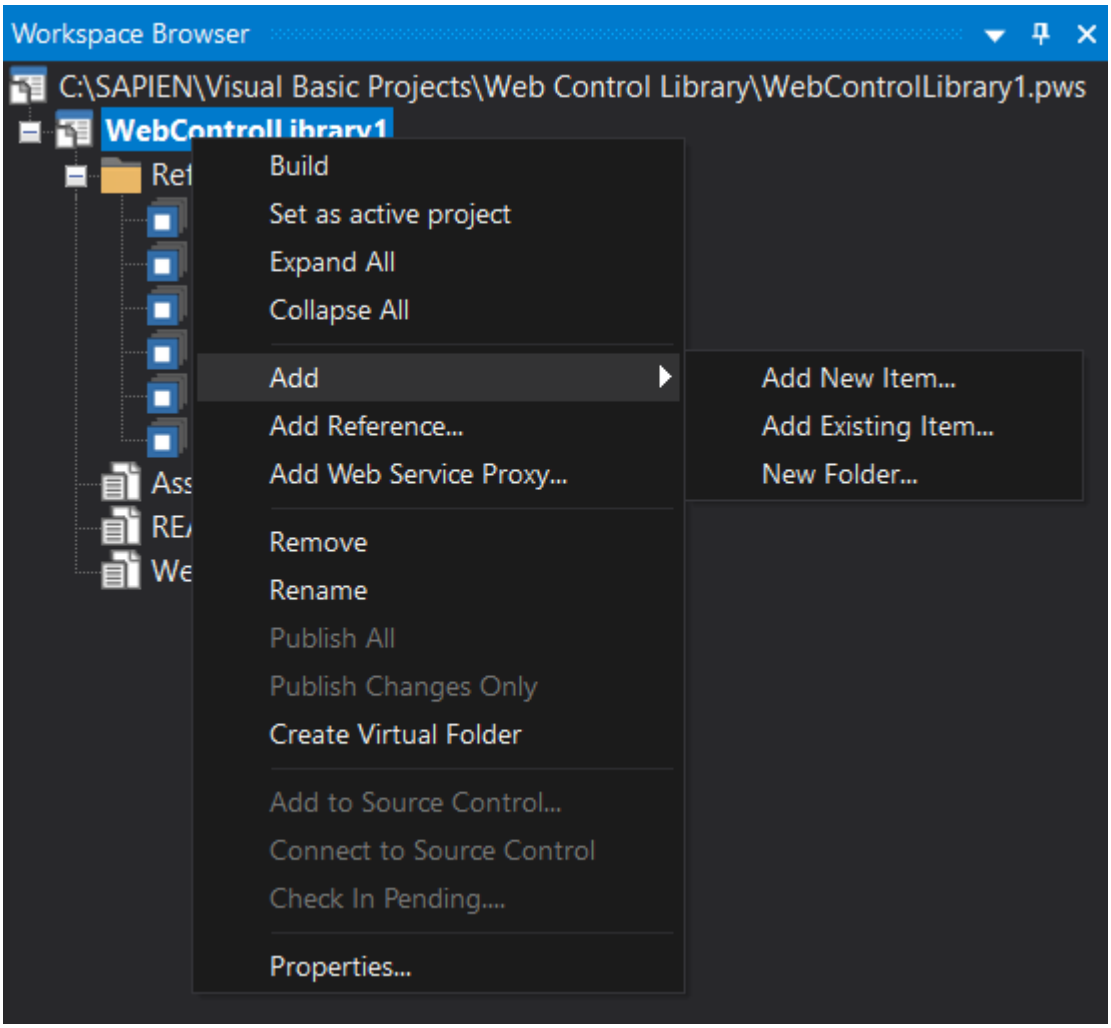
The Workspace Browser can contain multiple projects, allowing you to work with different projects simultaneously. Each project is listed in a separate portion of the Browser's hierarchical tree.



You can right-click various elements to work with them, such as modifying their properties, etc.



The context menu options will vary depending on whether a folder, file, or project is selected. If a project is selected, the options will depend on the project type; for example, Windows Script Host WSF and WSC projects offer different options than ASP or ActionScript projects.



For most project types, you can right-click the project itself to publish it.

7 Project Management

This section shows you how to create and work with projects.

PrimalScript projects group related files and settings, such as all files associated with a Web site, an AJAX project, an ActionScript project, or a .NET project. PrimalScript provides support for two special types of projects related to Windows scripting: WSF and WSC projects.

7.1 Creating Projects

This topic shows you how to create, back up, and run projects.

To create a project

1. Click **File** > **New Project**.
2. Select the project type and enter a name and location.
3. Create a new workspace for the project or add it into the current workspace.

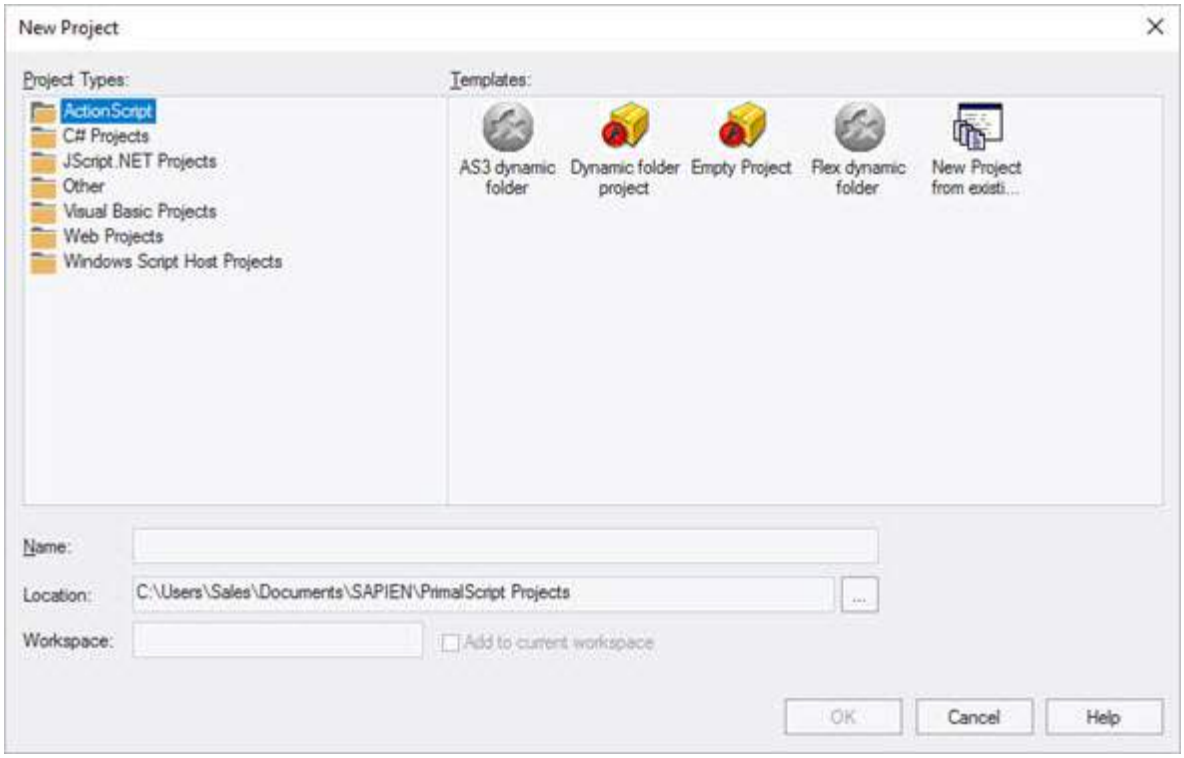
A workspace is a collection of projects that are defined by a .pws file.

To open a workspace (and its projects)

- In File Explorer, **double-click the .pws file**.

-OR-

- In PrimalScript, click **File** > **Open** and select a .pws file:



To back up a project

- Click **Project** > in the Deploy section click **Backup**.

To run a project

- Click **Project** > in the Build & Run section, click **Run** (*Ctrl+F6*).

Dynamic Folder Projects

Most projects are designed to work only with files that you use PrimalScript to add. However, Dynamic Folder Projects detect and incorporate all files in the project directory on disk, no matter how those files were added. Dynamic Folder Projects are useful when you're creating files in another application and then using PrimalScript to edit those files.

Dynamic Folder Projects are available in selected project types, such as Action Script and Web Project.

7.2 Projects and Your Workflow

One very valuable use for projects is in managing development workflow. Using projects, you can develop entirely on your local machine, ensuring that your work-in-progress doesn't affect production users. When you're ready, the project can be deployed-by PrimalScript, and as a single unit-to a production server where the project goes "live."

This contrasts with the common technique of editing files directly in production such as on a Web server. With this technique, changes are seen immediately by users-but so are mistakes. Editing "live" files directly is a very poor development practice and PrimalScript projects mean you don't ever need to. Instead, projects allow you to work on your local machine which serves as a development "sandbox" or testbed. You then deploy completed, tested, debugged files into production.

Workspace Management

Workspaces form the basis for projects; all projects are contained within a workspace and a workspace can contain multiple projects. PrimalScript can only have one workspace open at a time. A workspace helps to organize related projects. For example, you might have one project for a Web site and another for an ActionScript project that is used in the Web site. The workspace allows them to be open at the same time within PrimalScript.

The **Workspace** area of the **Project** tab provides functionality for working with workspaces:

- **Open**
  Opens a previously-saved .pws file.
- **Save** and **Save As**
  Saves the current workspace to a .pws file.

- **Close**
  Closes the current workspace.

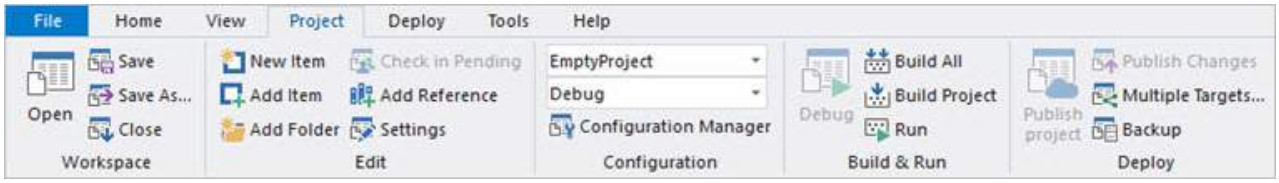
Workspaces can also include contacts such as the developer responsible for the workspace. Simply right-click the workspace name in the Workspace Browser to add a contact. You can also:

- Add new projects or insert existing projects into the workspace.
- Add configurations besides the default Debug and Release configurations.
- Open the workspace .pws file as a text file.
- Connect the entire workspace to source control and then add the entire workspace to your source control solution.
- For workspaces under source control, perform check-in and check-out operations for the entire workspace.

Managing Project Items

Projects can consist of multiple files and folders.

The **Project** tab, or right-clicking on a project in the Workspace Browser, provides options for adding new items, adding existing items (that is, items which already exist but aren't yet part of the project), and adding folders. You can right-click a project item in the Workspace Browser to remove it (without deleting it) from the project, or to permanently delete it.



When adding new items (as opposed to existing items), the dialog box restricts you to those file types which are valid for the type of project you're working on.

7.3 Projects and Source Control

Projects can be managed as a unit through PrimalScript's source control integration (source control integration 136 must be configured first).

To work with source control

- Right-click the project name in the Workspace Browser to access the source control options:

  Get Latest Version
  Gets a read-only copy of a file.

  Check In
  Checks files in to source control.

  Check Out

Checks files out of source control.

Undo Check Out
Undoes the last source control check out.

Add to source control
Adds the project to source control.

Check in pending
Checks in all project files which have changed or not yet been checked in.

Connect to source control
Connects the project folder to a folder within your source control solution, providing a place for the project to reside within the source control solution.

Individual files within the project can be checked in or out independently, although checking files in together as a project helps to simplify file and source control management.

7.4 Project Properties

You can view and change the properties of a project.

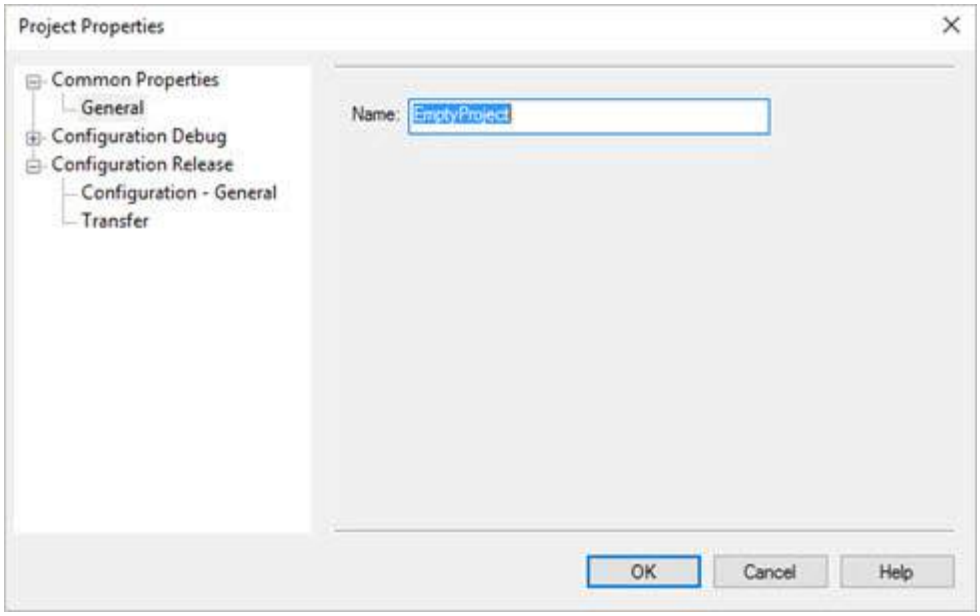
Project Properties Dialog

To view project properties

- In the **Workspace Browser**, right-click the project name and then click **Properties**.

-OR-

- Click **Project** > in the Edit section, click **Settings**.



A project's Common Properties include its name, in the **General** section. The **Debug** and **Release** configuration sections each have a **General** section which allow you to specify an external program or URL to run or debug the project. They also have a **Transfer** section which allows you to specify transfer options for publishing the project, either in debug mode or release mode. Transfer can be made via ftp or network file copy and you can specify destination paths and credentials. You can also set the project mode to Local or Master which will be discussed shortly.

This separation between debug and release modes allows you to specify (for example) a local server for debugging the project and a production Web server for releasing the project.

Managing Configurations

You can modify the workspace itself to provide additional configurations aside from Release and Debug or even to modify those two names. Doing so can provide you with additional transfer destinations, if needed.

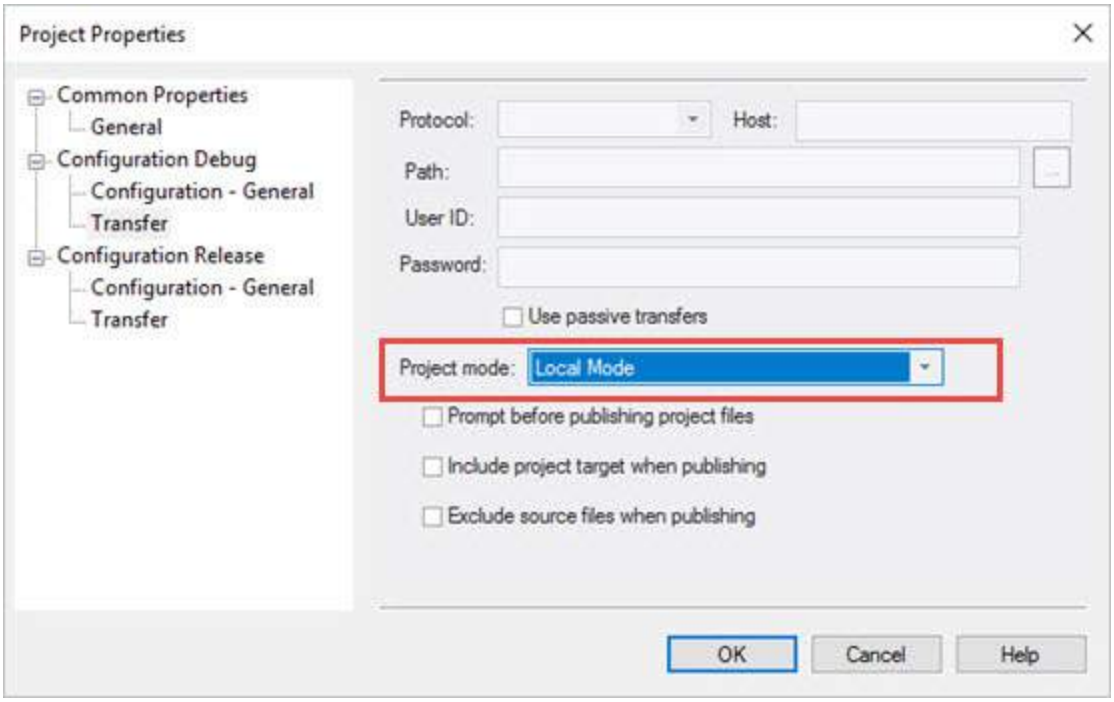
For example, you might create an additional configuration called "Test." This would allow you to have a Debug configuration where the project runs locally, a Test configuration that deploys to a test server, and finally a Release configuration that deploys to your production server. You can create as many additional configurations as you need.

To do so, right-click the workspace and select Configuration Manager (or simply click Configuration Manager on the Project tab). Click **New...** to add a new configuration. When adding a new configuration, you can copy the configuration's settings from an existing configuration, if desired-this is a good shortcut when the new configuration won't differ much from an existing one.

Then, go back into the project properties and you'll see your new configuration listed.

Local Mode vs. Master Mode

If you open a project's configuration (any of them—Debug, Release, or one you've defined), you'll notice a "project mode" setting. Projects can be set to either Local Mode or Master Mode.



Local Mode

The default is Local Mode, which allows you to work on all project files locally. Files aren't published until you specifically do so (discussed in the next section). When you publish, files are published to the target defined in the currently-selected configuration, meaning you can have a separate publishing target when you're in Debug, Release, or another configuration you've defined.

Master Mode

With Master Mode, changed files are automatically published to the defined target for the currently-selected configuration.

How is this useful? Let's say that you don't have a Web server on your local computer and, while in Debug mode, you want to test from a testing Web server on your local network. However, for Release, you want to publish to a production Web server. Configure the Debug configuration to be in Master mode and configure it with the appropriate settings to publish files to your testing Web server. Each time you save a file locally, it'll be published to your testing server immediately. Keep the Release configuration in Local mode and switch to Release mode when you're ready to publish all changed files to your production Web server.

7.5 Publishing Projects

You can publish your project after you have defined transfer destinations in the project's configurations. You can publish all project files or only files that have changed since you last published.

To publish a project

1. Click **Project** > the Configuration section, select **Debug** or **Release** (or, your custom configuration).
2. In the Deploy section, click **Publish Project**.

-OR-

- In the **Workspace Browser**, right-click the project name and then click **Publish All**.

To publish changes

1. Click **Project** > the Configuration section, select **Debug** or **Release** (or, your custom configuration).
2. In the Deploy section, click **Publish Changes**.

-OR-

- In the **Workspace Browser**, right-click the project name and then click **Publish Changes Only**.

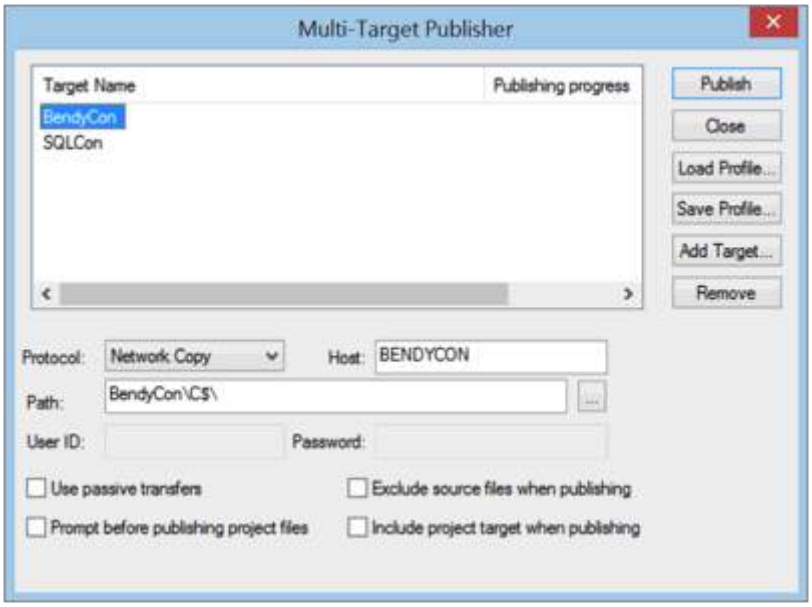
Multi-Target Publishing

PrimalScript includes multi-target publishing capabilities. This allows a project to be published to multiple targets at the same time, such as publishing a Web project to multiple servers in a Web farm.

To use the multi-target publisher, a project must be open in PrimalScript. You cannot publish individual files to multiple targets; only files that are part of a project can be published to multiple targets.

To start the multi-target publisher

- Click **Project** > in the Deploy section click **Multiple targets**.



To operate the multi-target publisher

- **Publish**
  Publishes the current project to all listed targets.
- **Close**
  Closes the multi-target publisher.
- **Load Profile**
  Loads a previously-saved list of targets.
- **Save Profile**
  Saves the current list of targets.
- **Add Target**
  Adds a new target. You must do this before you can configure the properties for a target.
- **Remove**
  Removes a target from the list.

After adding a target you can specify its name. This name appears only in the list and has no relationship to the target's server name or other information. For each target configure:

- **Protocol**
  Select FTP or Network Copy.
- **Host**
  This is the target's server name. Do not include "\\" or other characters when using network copy, and do not include ftp:// when using FTP.
- **Path**

The path to where the project should be copied. For a network copy, specify "share\path\path" without any leading backslash; for FTP, enter the path from the FTP root and do not specify any leading slash.

- **User ID and Password**
  Available only for FTP operations. For network copy, your current logon credentials must have sufficient rights on the destination (or, manually execute a NET USE command from the command-line to specify alternate credentials).
- **Options**
  Select any of the four options, as appropriate. These options are configured per-target, not for the entire multi-target publishing process.

To publish to a mapped network drive

- In the Path field, enter the complete drive letter and path (e.g. "M:\MyFolder"). Leave the **Host** field blank.

8 Backup and Restore Files

Nothing replaces a professional source control system, but PrimalScript has Oops Resilience™, which includes several features to help you restore your script to a previous state.

For information about using PrimalScript with a source control system, see *Source Control Integration* 136

8.1 Infinite Undo™

This topic covers Undo and Redo options.

PrimalScript maintains an extensive "undo list" that lets you undo and redo actions even after you have saved your file. Because the undo list is stored in a separate stream, this feature works only when the file is saved to a Windows NTFS volume.

Undo and redo work like a last-in-first-out stack. When you press Ctrl+Z to undo, it undoes the most recent action. Press Ctrl+Z again, PrimalScript undoes the previous (second most-recent) action. Keep pressing Ctrl+Z until the file reaches the desired state. Similarly, redo redoes the most recent action and you can press it repeatedly.

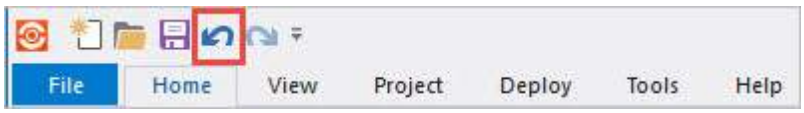
Unlike conventional undo and redo, PrimalScript saves undo stack with the file and maintains even after the file is saved—even months later, after the file has been repeatedly opened, saved, and closed.

To undo an action

- Press ***Ctrl*+*Z*** (or ***Alt*+*Backspace***) repeatedly, if necessary.

-OR-

- On the Quick Access toolbar (upper-left corner of the screen), click **Undo**:



To redo an action

- Press ***Ctrl*+*Y*** (or ***Alt*+*Insert***) repeatedly, if necessary.

-OR-

- On the Quick Access toolbar (upper-left corner of the screen), click **Redo**:

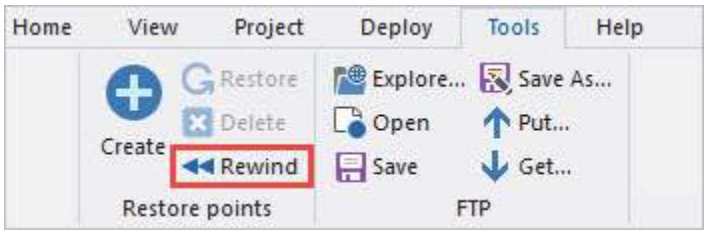
8.2 TempPoint Files

This topic explains how to work with TempPoint files.

As soon as you edit a script file, PrimalScript saves the original (unedited) version of the script in a hidden TempPoint file. By default, PrimalScript saves the TempPoint file until you close PrimalScript or reopen the file, but you can change the defaults to prevent PrimalScript from saving or deleting TempPoint files.

To restore the current file to its original state

- Click **Tools** > in the Restore points section, click **Rewind**:



To use the TempPoint file without changing the current file

1. In Windows > Folder Options > on the View tab > in Hidden Files and Folders, check **Show hidden files, folders, and drives**.
2. In File Explorer > copy and rename the TempPoint file and turn off the hidden attribute.

   You can also do this programmatically. For example, in Windows PowerShell:

   ```
   Get-ChildItem <file> -Hidden | Copy-Item -Destination <file> -PassThru | ForEac
   ```

To prevent PrimalScript from creating TempPoint files

- Click **File** > **Options** > **Environment** > **Backup** > and uncheck **Create a restore point as soon as a file is modified**.

To prevent PrimalScript from deleting TempPoint files

- Click **File** > **Options** > **Environment** > **Backup** > and uncheck **Remove restore points when application closes**.

8.3 Restore Points

This topic explains how to work with Restore Points.

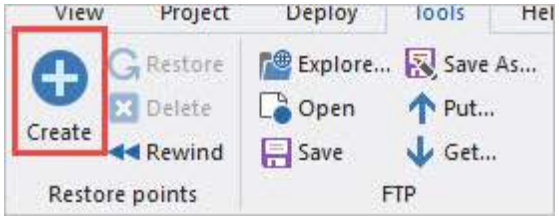
The PrimalScript Restore Points feature lets you save a version of your file as you work. You can experiment with different coding strategies knowing that a last-known-good copy of your script is saved on disk. Experienced developers and scripters typically "take a checkpoint" or update a restore point each time they complete a new feature and the script runs without error.

Restore points are saved as hidden files in the script directory as .RestorePoints.<filenameExtension> file name.

Unlike TempPoint files and backups—you control restore points. You create them, restore from them, and delete them explicitly.

To create or update a restore point

- Click **Tools** > in the Restore points section, click **Create**:



To revert the current file to its last restore point

- Click **Tools** > in the Restore points section, click **Restore**.

To delete a restore point

- Click **Tools** > in the Restore points section, click **Delete**.

  You can delete and recreate a restore point at any time.

8.4 Backup Files

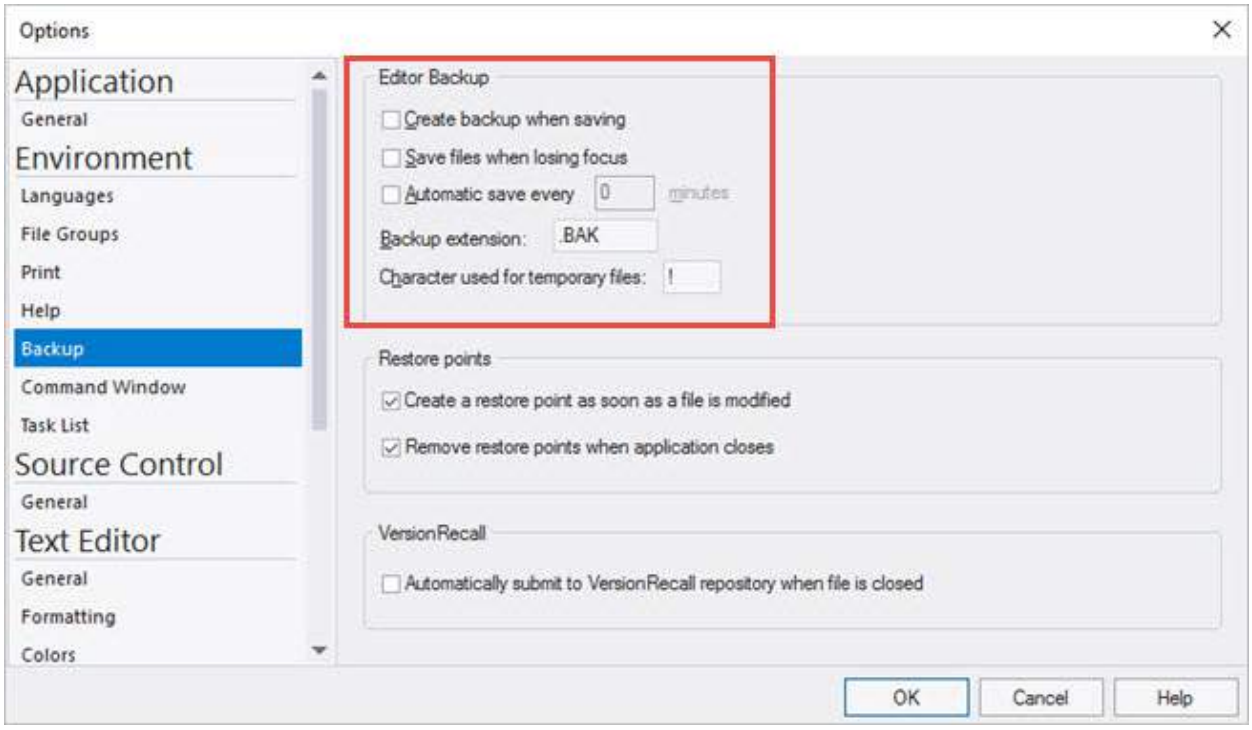
PrimalScript has optional backup features that create and update a backup copy of your script:

- Every time your save.
- When you click a pane other than the editor.
- At predetermined time intervals.

By default, the backup file has a .BAK file name extension, but you can customize the extension.

To enable and configure backup files

- Click **File** > **Options** > **Environment** > **Backup** > in the Editor Backup section, **set your preferences**:

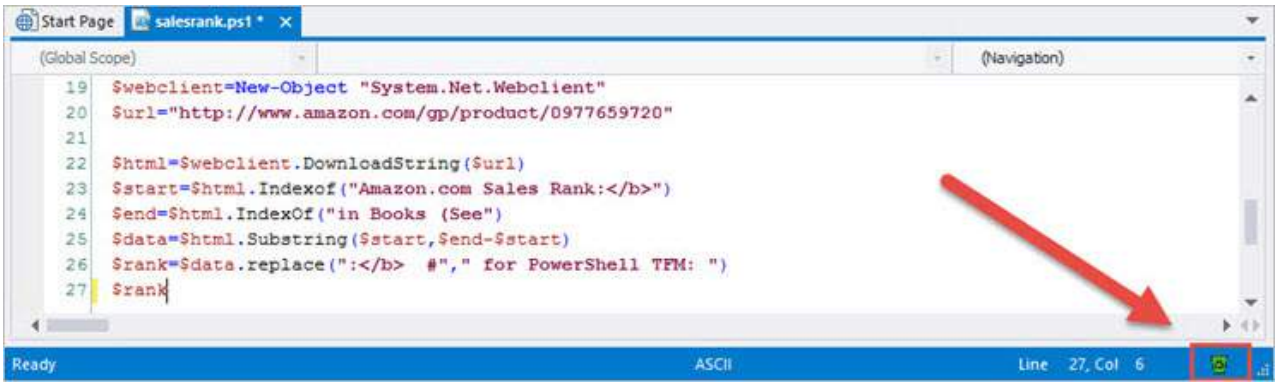


8.5 Recycle Bin

This topic explains the Recycle Bin functions.

The Recycle Bin—which appears on the PrimalScript status bar at the bottom of the window—stores the text that you delete from files, even after you close and reopen PrimalScript. You can review these deleted segments and remove them from the Recycle Bin, copy them to the Clipboard, or reinsert them into your script.

The Recycle Bin is not file-specific. It contains deleted segments from all files. If it fills up, the oldest items are deleted to make room for newer ones.

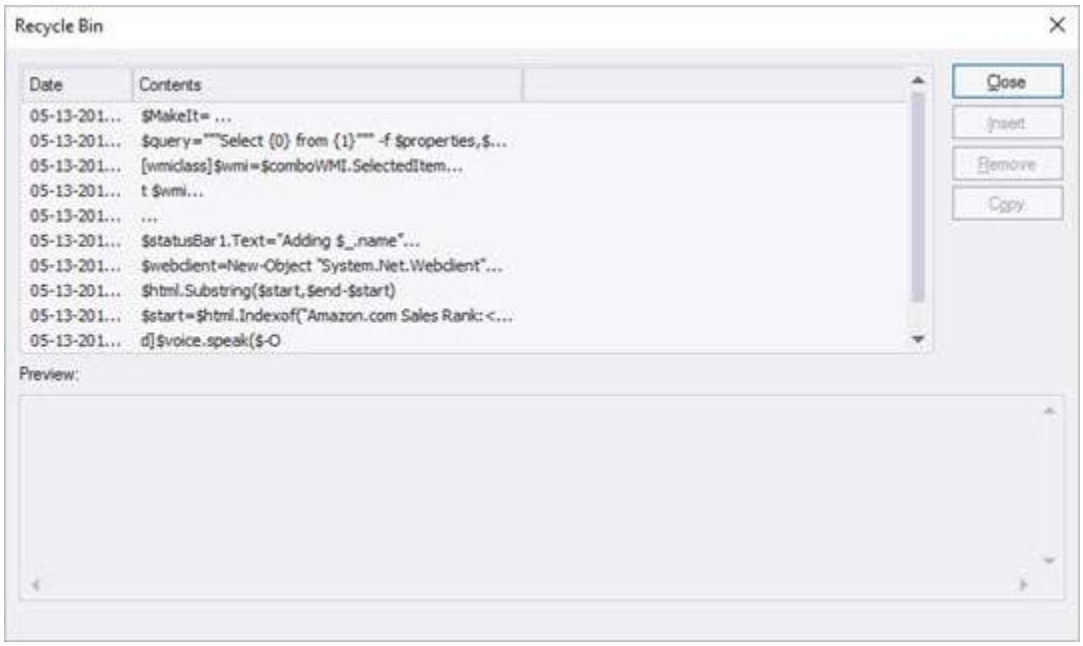


To open the Recycle Bin

- Double-click the Recycle Bin icon.

-OR-

- Right-click the Recycle Bin icon, and then click **Open**.



To insert a recycled item into your script

1. Place your cursor at the insertion point in the script.
2. Open the Recycle bin.
3. Click an item date to select the item.
4. Click **Insert**.

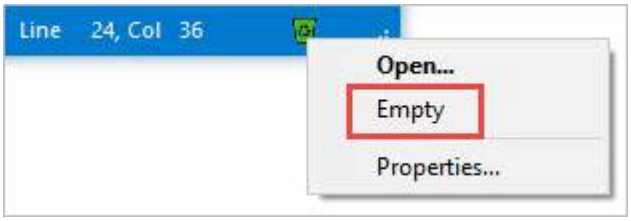
To delete items from the Recycle Bin

1. Open the Recycle bin.
2. Click an item date to select the item.

3. Click **Remove**.

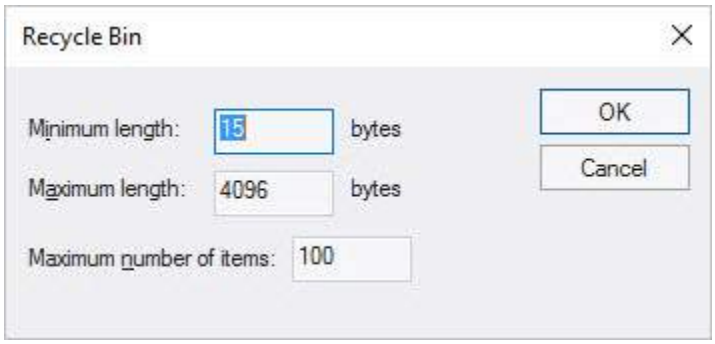
To clear (delete all items) from the Recycle Bin

- Right-click the Recycle Bin icon, and then click **Empty**.



To make the Recycle Bin larger or smaller

1. Right-click the Recycle Bin icon > click **Properties**.
2. Change the **Maximum number of items** value. The default is 100 items.



9 Source Control Integration

PrimalScript provides a number of source control options, including a Universal Version Control system that integrates with command-line tools such as Git, or integrating with a Microsoft Source Code Control Integration (MS SCCI) software provider.

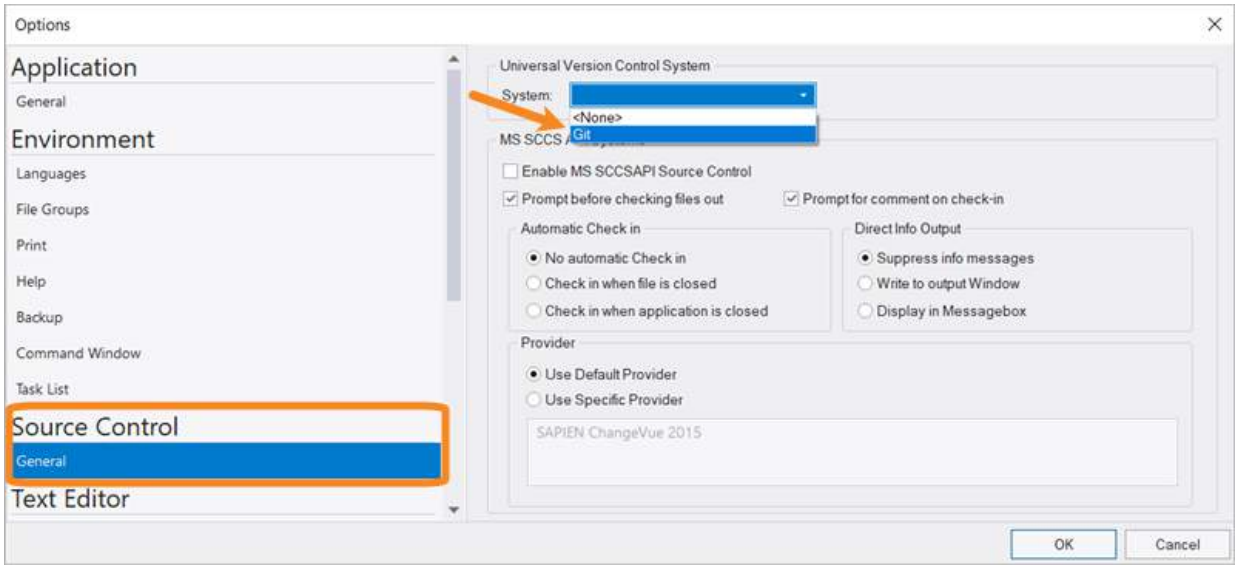
9.1 Universal Version Control

The Universal Version Control system allows configuration of any source control provider with command-line tools. The current support scope includes the Git source control system. Support will be expanded to include other providers.

Using Git

To enable Git support

1. Go to **File** > **Options** > **Source Control** > **General**, then select **Git** in the System drop-down list:

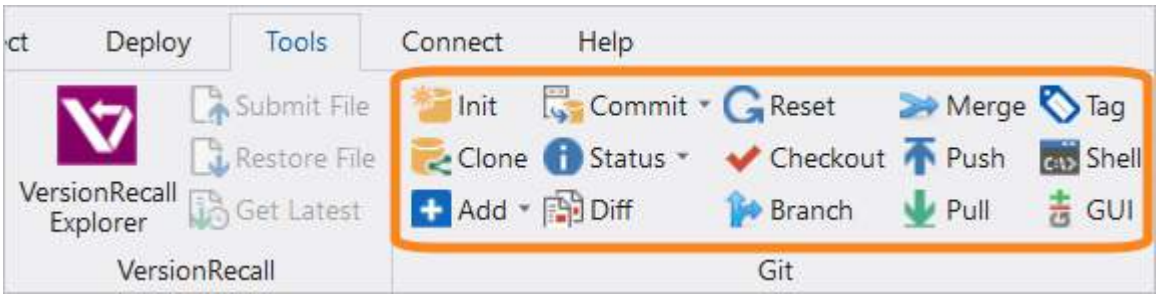


2. Click **OK**, and then PrimalScript will prompt you to restart.

To disable the Universal Version Control feature, select **<None>** from the System drop-down.

Git Commands

When you open a file, the preconfigured Git commands will appear on the **Tools** tab, in the **Git** section:

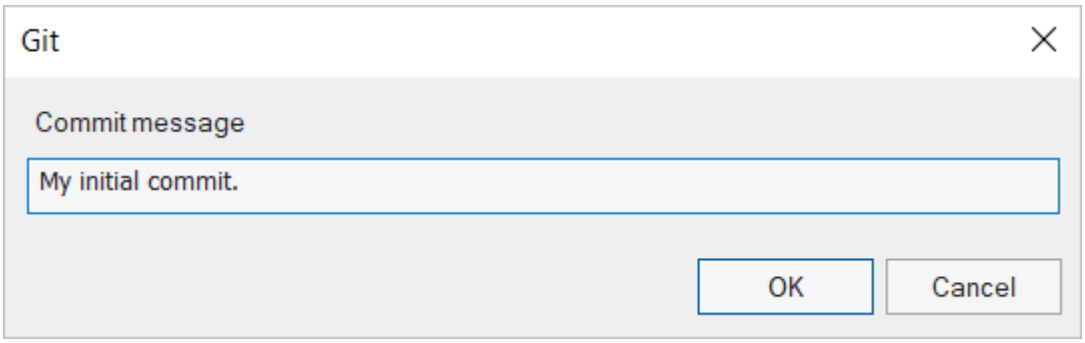


- **Init**
  Initialize a Git repository in the current folder.
- **Clone**
  Create a clone of a remote repository.
- **Add**
  Add a file to a repository.
- **Add All**
  Add all files in the folder to a repository.
- **Commit**
  Commit a change to a repository.
- **Commit all files**
  Commit all changes to a repository.
- **Status**
  Get the status of the current file.
- **Status All**
  Get the status of all files in the folder.
- **Diff**
  Show the difference for the current file.
- **Reset**
  Rewinds history (files + commits) back to the previous commits.
- **Checkout**
  Switch branches or restore working tree files.
- **Branch**
  Create a new branch.
- **Merge**
  Merge the specified branch.
- **Push**
  Upload the local repository content to a remote repository.
- **Pull**
  Fetch and download content from a remote repository.
- **Tag**

Create a tag for the current repository.

- **Shell**
  Launch a Git command shell.
- **GUI**
  Launch the Git GUI tool.

You will be prompted if a value is required to execute the command. For example, when you select the Git **Commit** command, a commit message is required:



Output from Git is displayed in the Tool output pane.

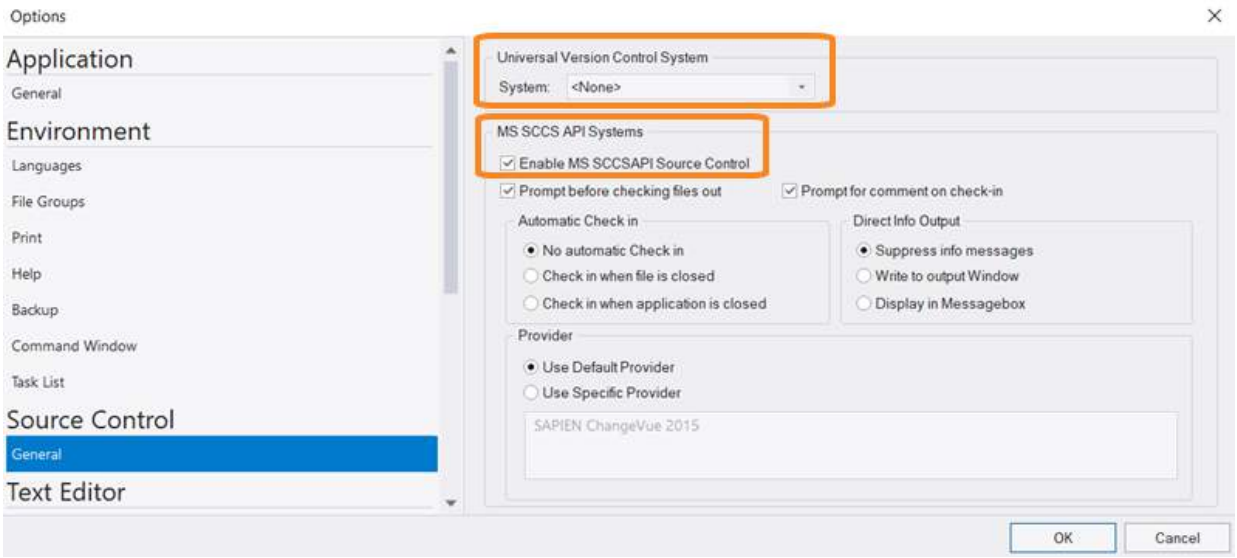
9.2 Microsoft Source Code Control Integration

Your source control software must either be VersionRecall from SAPIEN Technologies, or your source control provider must provide an SSAPI-compatible client, such as Microsoft Visual Source Safe.

Before configuring PrimalScript for source control, you must install your source control software's client.

To configure source control integration

1. Go to **File** > **Options** > **Source Control** > **General** and make sure that the Universal Version Control feature is disabled (the **System** menu should be **blank** or **<None>**.
2. Select **Enable MS SCCSAPI source control**, and click **OK**.

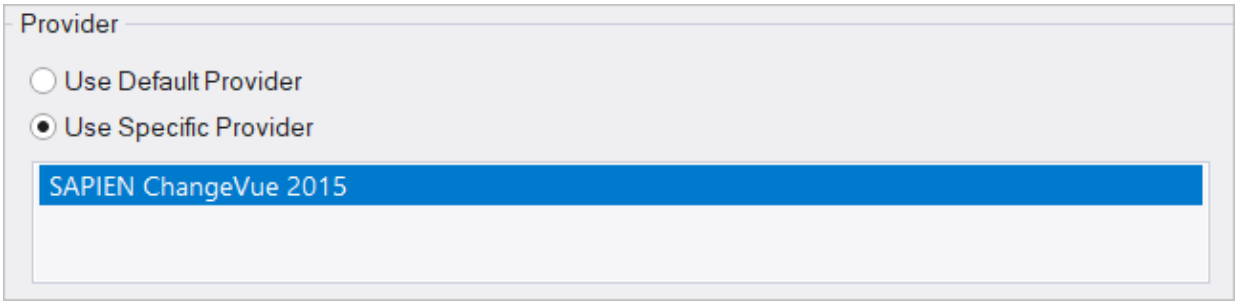


PrimalScript will require a restart, and then it will automatically detect the presence of the source control client and display it in the Provider list box.

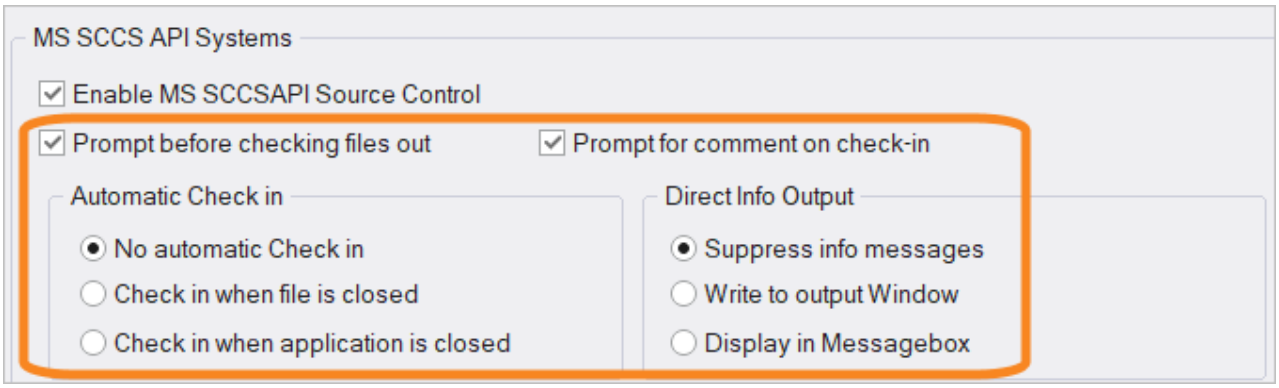
After you enable MS SCCSAPI and restart PrimalScript, your source control provider must be displayed in the Provider list; if it is not, then source control is not properly installed and will not be available to PrimalScript.

To select a source control provider

1. Go to **File** > **Options** > **Source Control** > **General**, then click **Use Specific Provider** in the Provider section.
2. Select your provider from the list, then click **OK** and restart PrimalScript.



After enabling source control you can configure the options however you like, including prompting before checking out files, and automatic check-in options:



Using Source Control

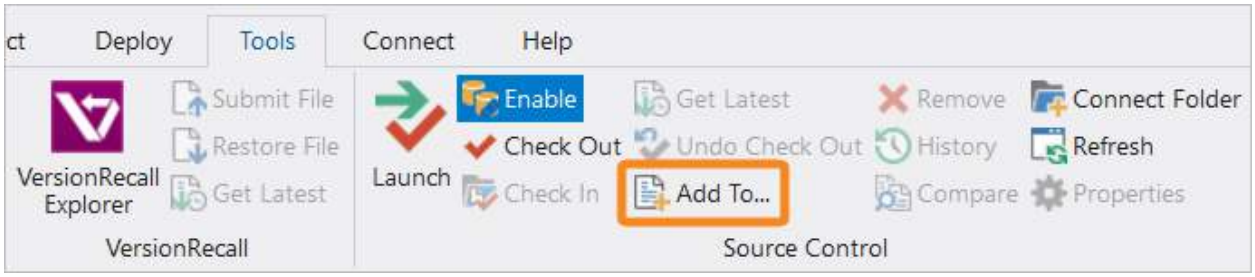
PrimalScript does not provide source control capability—it simply integrates with the features of your compatible source control software. Some features described here may not be available in your software, or may work somewhat differently.

Before a script can be managed through source control, it must first be added.

You must first save unsaved scripts before they can be added. If you do not, PrimalScript will prompt you to save the file first.

To add a file to source control

- With the file open, on the **Tools** tab > in the **Source Control** section > select **Add To...**:



Your source control software governs the add process and may prompt you for login credentials, a location for the script, or other information.

Once added, scripts can be checked in or out using the **Source Control** menu on the Tools tab:

Source Control Commands

Some of these source control options may not be available, or may work differently, depending on your source control provider:

- **Launch**
  Launches your source control software's user interface.
- **Enable**
  Enable / disable source control integration.
- **Check Out**
  Check out the current file from source control.
- **Check In**
  Submit and check in the current file into source control.
- **Get Latest**
  Get the latest version of the current file from source control.
  Retrieves a read-only copy of a file.
- **Undo Check Out**
  Reverse a previously performed checkout on the current file.
- **Add To...**
  Add the current file to the associated source control project.
- **Remove**
  Remove the current file from source control.
  Does not automatically delete any local copies of the file.
- **History**
  Show the current file's history.
- **Compare**
  Compare the current file to a previous version.
- **Connect Folder**
  Connect a local folder to source control.
  This makes it easier to work with groups of files since they can be more easily checked in and out as a unit, and since they'll be conveniently located in a single local folder on your computer when you're working with them.
- **Refresh**
  Refresh source control status.
- **Properties**
  Show source control properties.

10 Universal Help

PrimalScript provides truly universal, integrated help for most scripting languages, providing you with access to the manufacturer's documentation and with context-sensitive help.

10.1 Add External Help

PrimalScript has the capability to link to external help.

If you install the manufacturer's language documentation before installing PrimalScript, PrimalScript can automatically configure itself to use the documentation. However, in some cases, you need to manually configure PrimalScript to provide full help capabilities.

When properly configured, PrimalScript help is context sensitive. For example, if you open an ActionScript document, select a keyword, and then press *F1*, Macromedia's HTML help should appear and display help for that keyword.

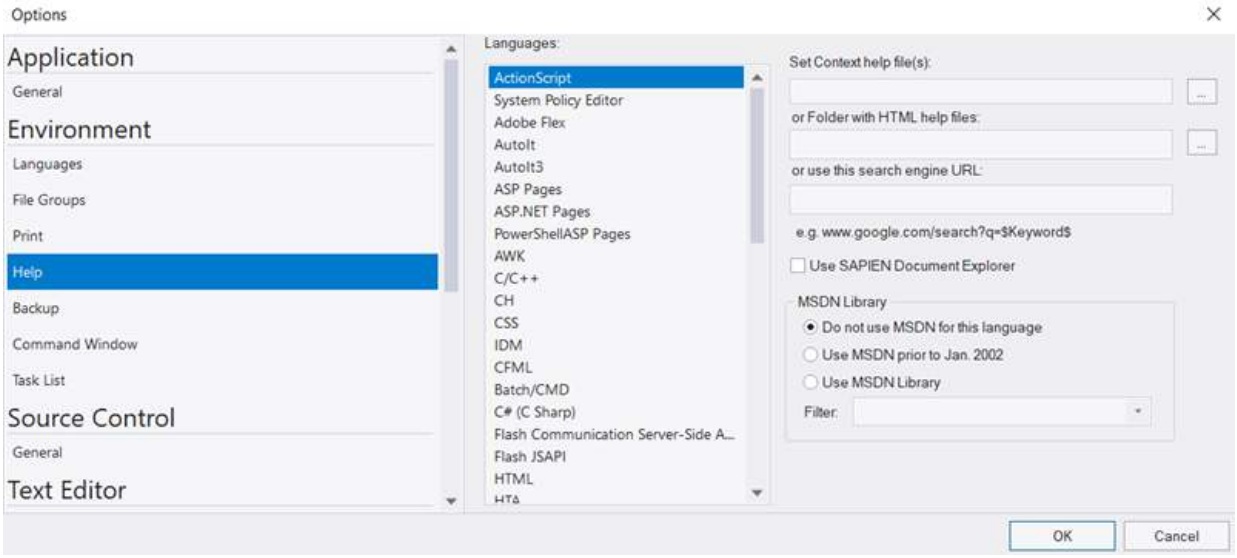
PrimalScript Help is language-sensitive. If you're editing a JScript document and haven't configured Help settings for the JScript language, pressing *F1* may not do anything, even though you may have configured Help for the similar JavaScript language that PrimalScript treats as a distinct, independent language.

PrimalScript provides special integrated help capabilities for VBScript and Windows PowerShell, which are enabled by default. In either language, moving your cursor to a keyword and pressing *F1* will display help.

- **For Windows PowerShell**
  PrimalScript uses the XML help files designed for the Get-Help cmdlet. Help is available only for modules that include an XML help file.
- **For VBScript**
  Help is provided as an electronic edition of *WSH and VBScript Core: TFM* by Jeffery Hicks. PrimalScript extracts keyword help from the text and displays it in the Help pane.

To link PrimalScript to an external help source

1. Click **File** > **Options** > **Environment** > **Help**.

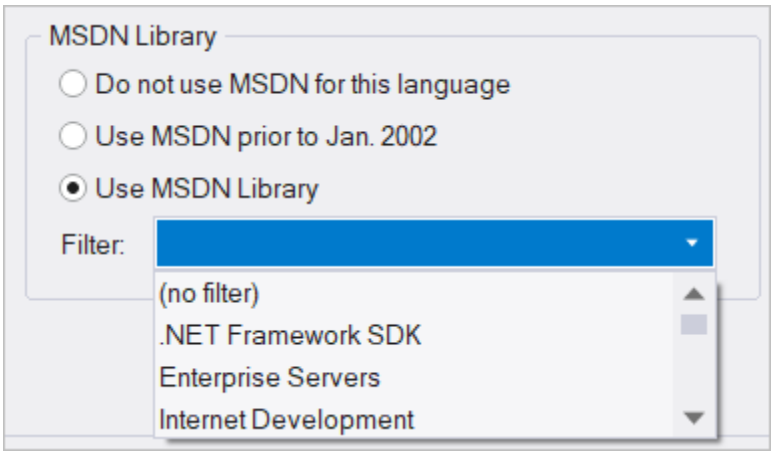


2. From the Languages list, select a language.
3. Navigate to the directory that contains the files for each type of help.

   - **Context-sensitive help files**
     Typically supplied in HLP or CHM files. Enter the full path to the files.
   - **HTML-based help**
     Enter the path to the directory that contains the help files.
   - **Internet-based help**
     Enter the URL of the Internet-based help search engine. You can also provide a URL such as www.google.com/search?q=$Keyword$, which uses the Google search engine. When typing the URL, use to represent the search keyword, use **$Keyword$**.

MSDN Library Integration

PrimalScript can integrate with Microsoft MSDN Library—either the current edition or the pre-Visual Studio .NET edition issued before January 2002. Simply select the appropriate setting (this is obviously most useful for Microsoft languages like VBScript, VB.NET, and so forth). You can also select a search filter which will be passed to MSDN Library.



The proper edition of MSDN Library must be installed on your computer in order for MSDN Library integration to function. You can download MSDN Library for free from Microsoft's Downloads Web site.

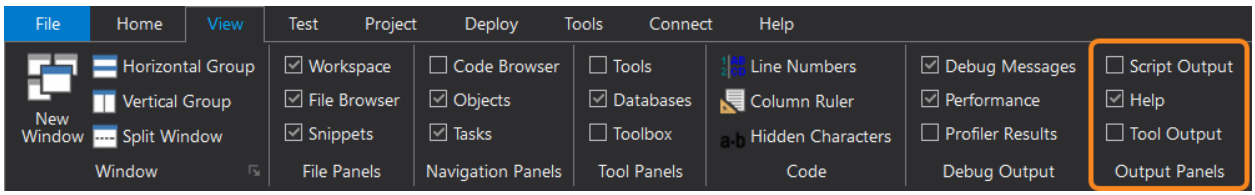
When properly configured, PrimalScript Help is context sensitive. In other words, if you open a VB.NET document and select the word InputBox, and then press *F1*, MSDN Library should display assistance for that function.

10.2 Help Panel

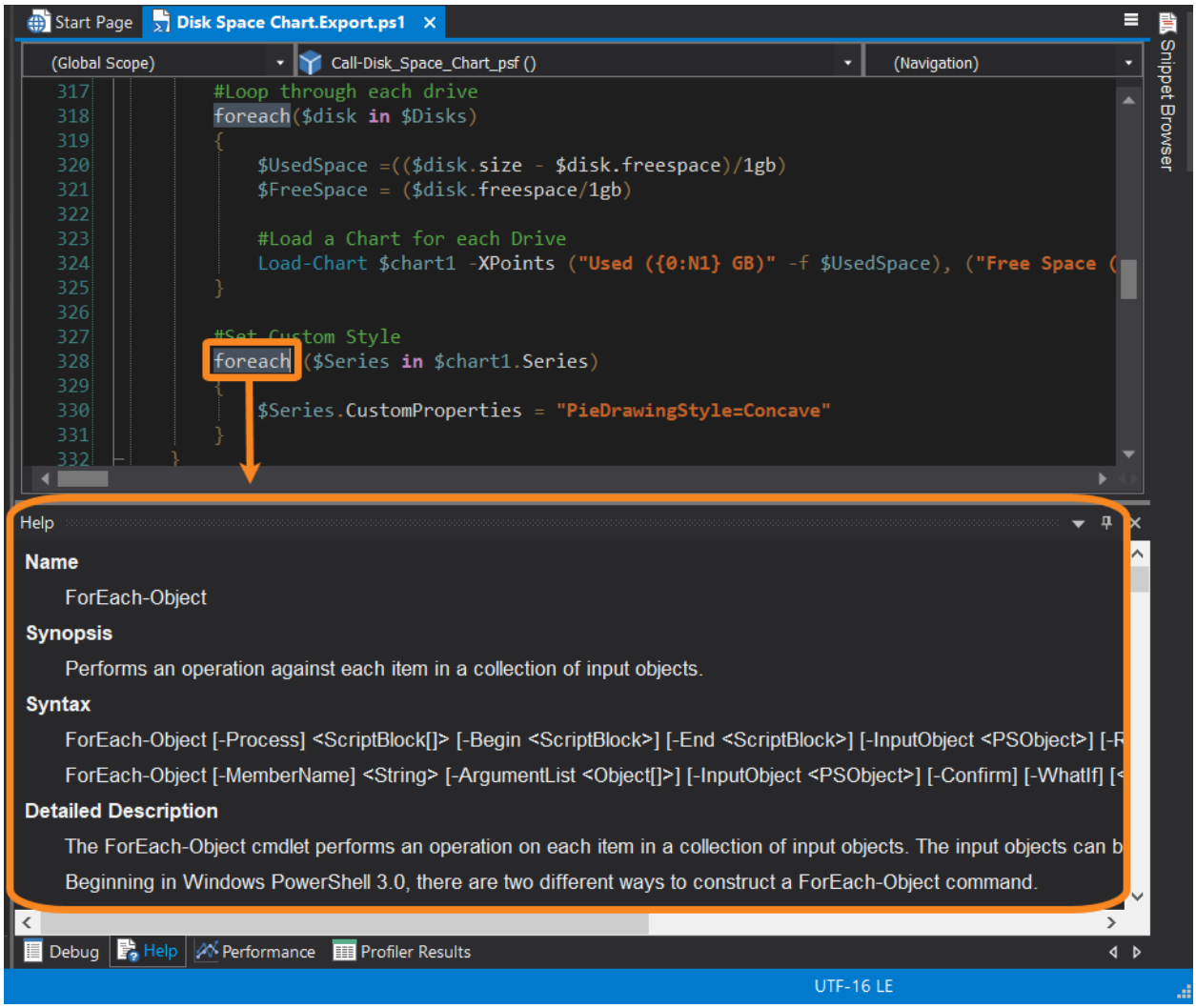
The Help panel will automatically display the help information for any item you select. This works for commands in the Object Browser, editor windows, and more.

To show / hide the Help panel

- On the ribbon, click the **View** tab and then **check** (to show) or **uncheck** (to hide):



The context sensitive Help is displayed for the selected item:

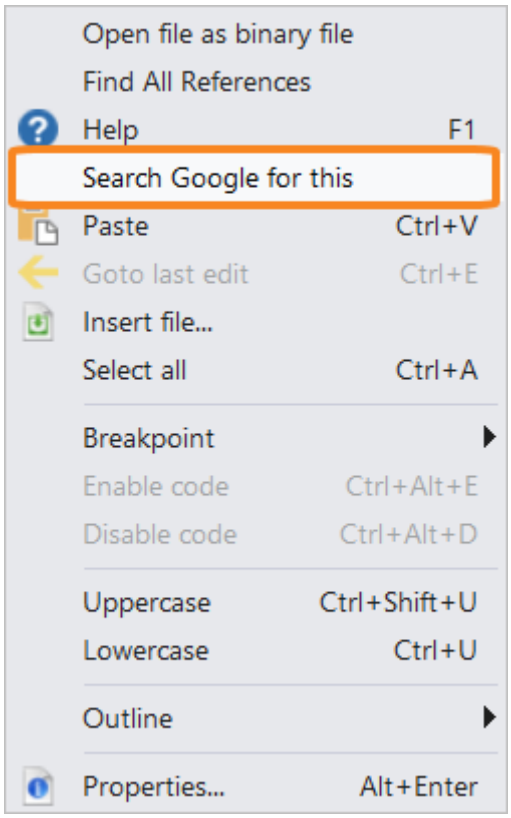


10.3 Google This

PrimalScript has a special feature that uses the Google search engine.

To google a term in your script

- Select and right-click the term and then click **Search Google for this**:



11 Windows Script Host Features

PrimalScript provides features designed to make Windows Script Host (WSH) scripting-in VBScript and JScript easier and more efficient.

11.1 Windows Script Files

A Windows Script File (.wsf) is a special, XML-formatted file that can contain multiple scripts (referred to as jobs), define command-line arguments, and more. Using the WSF format, you can essentially write your own command-line tools using VBScript or JScript. Although the XML format is complex, you don't need to worry about it because PrimalScript handles it all behind the scenes—allowing you to focus on your scripts.

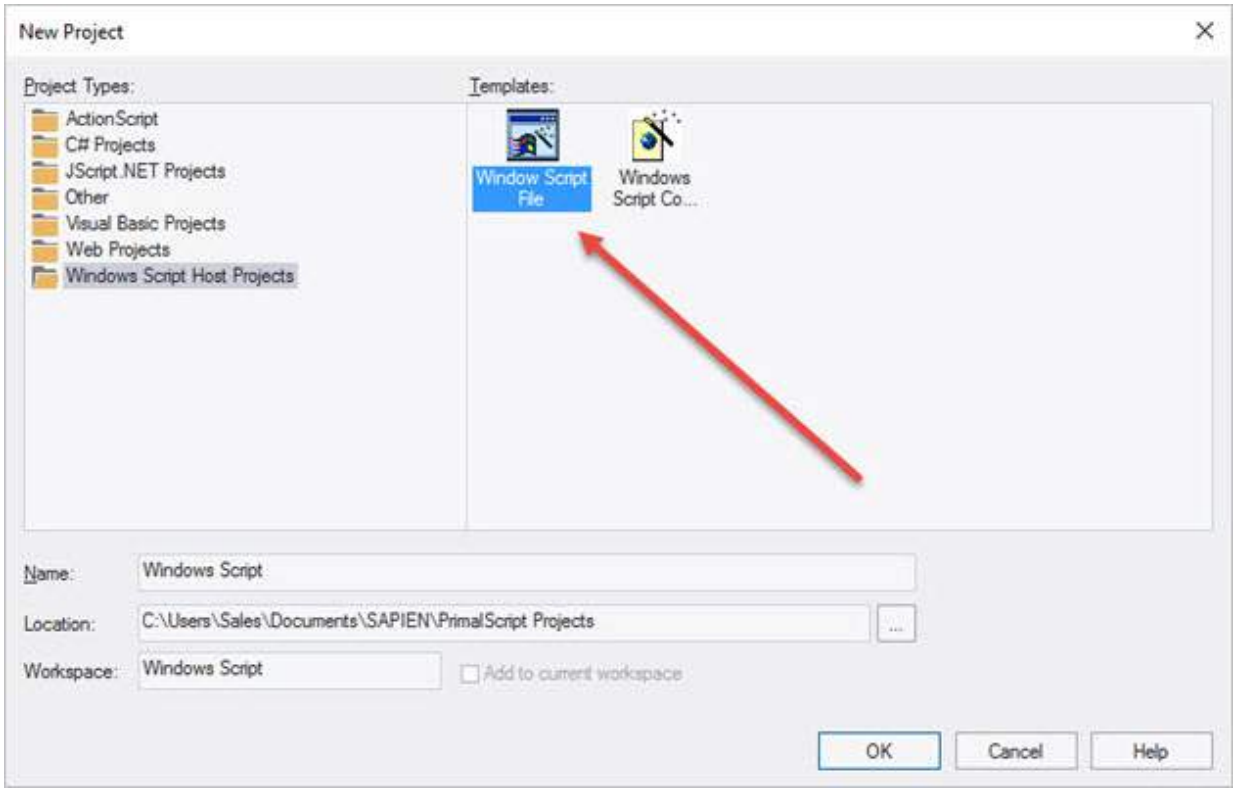
In this topic we will show you how to create a WSF project 147, and explain the WSF workspace 150, properties 153, and code 154.

Starting a WSF

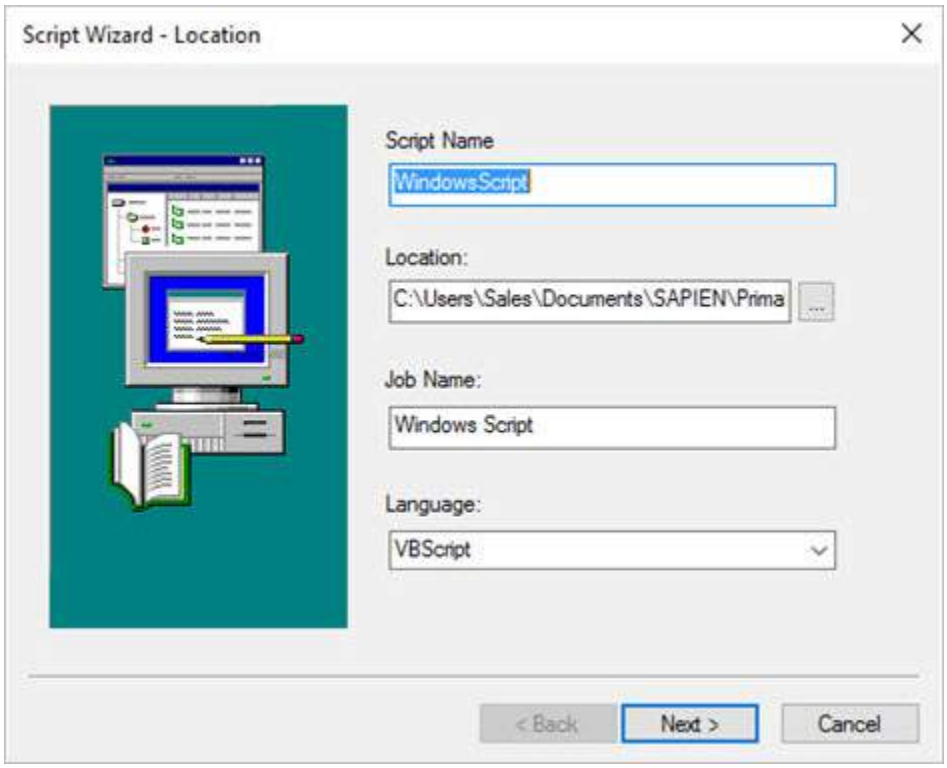
A WSF is a special kind of PrimalScript project.

To create a WSF project

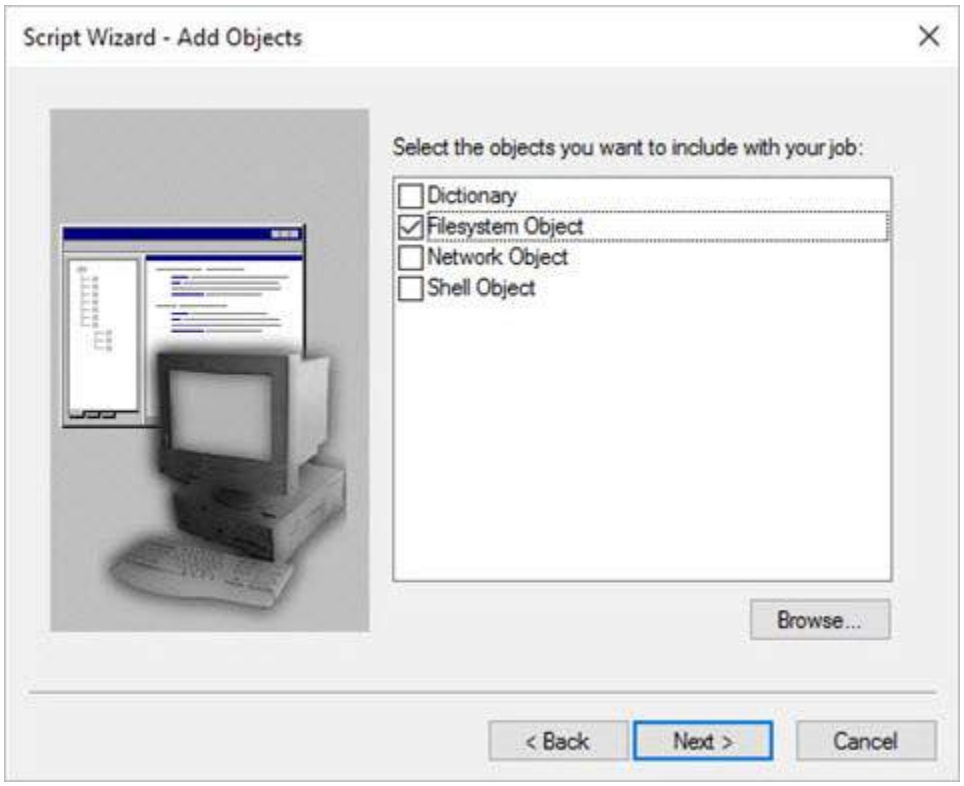
1. Click **File** > **New Project** > **Windows Script Host Projects** > **Windows Script File**.
2. Enter a name and location for your new file and specify a name for the workspace.



3. Specify the name of the script, its location, the name of the first job in the WSF, and the scripting language.



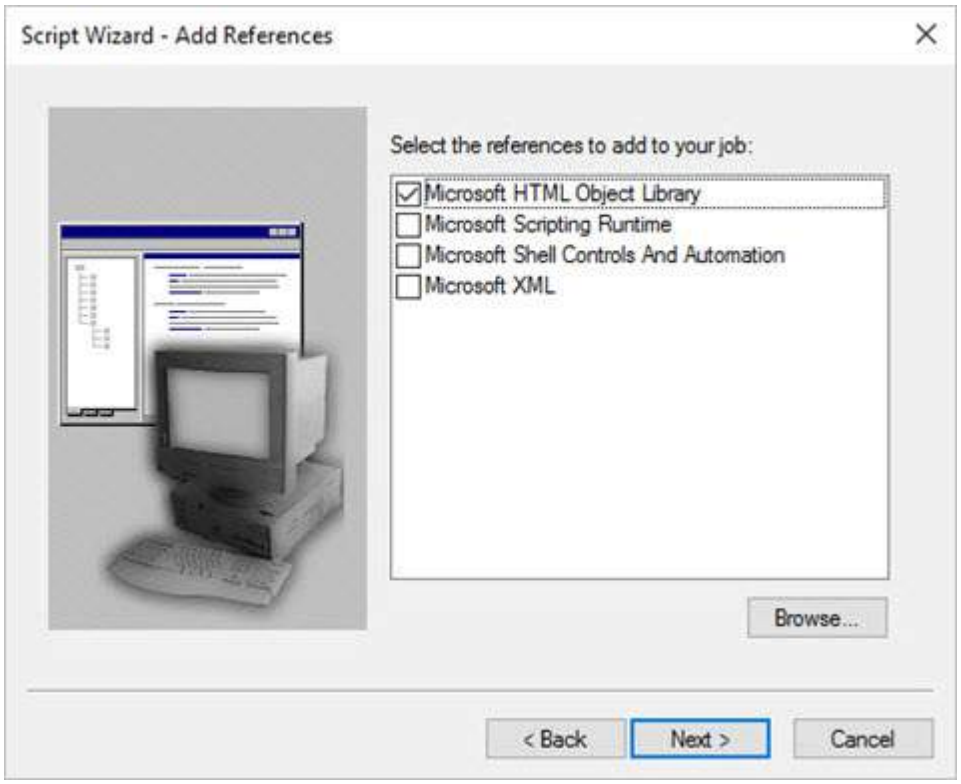
4. Add objects to the job.



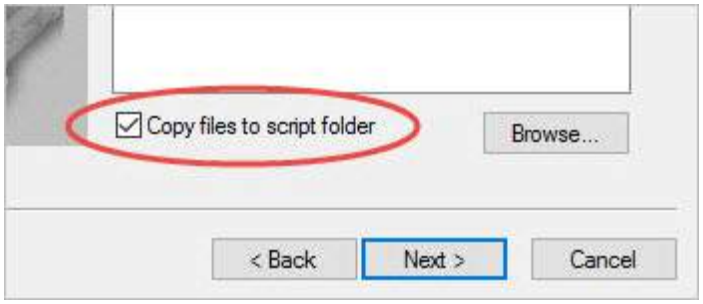
To add objects that are not on the default list, click **Browse**.

By referencing the objects that your scripts will use, you can use any constants defined in the objects' type libraries.

5. Add other references.



6. Add additional files to the job. To direct PrimalScript copy these files to the folder for you, click **Copy files to script folder**.

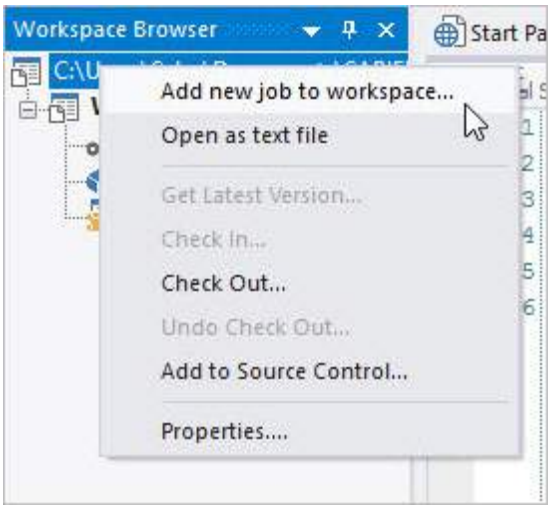


WSF Workspace

The WSF is part of a special PrimalScript workspace. You can add additional scripts—or jobs—to the workspace.

To add additional scripts or jobs to the workspace

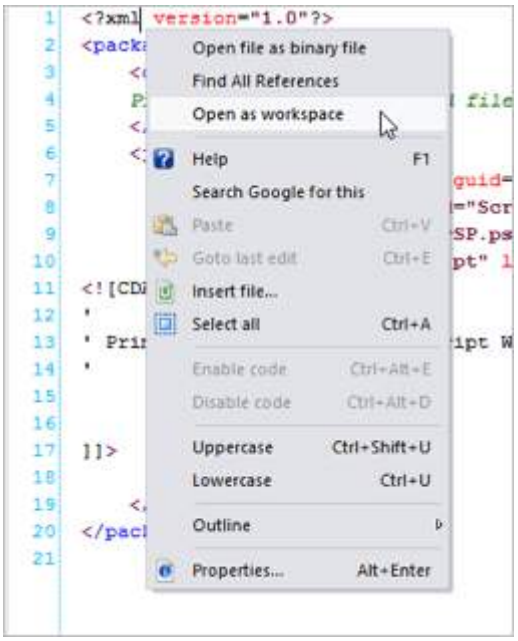
- Right-click the top-level workspace item in the **Workspace Browser** and select **Add new job to workspace**.



You can also open the workspace (that is, the WSF file) as a text file. Doing so provides access to the raw XML formatting as well as the script code of the jobs contained in the workspace.

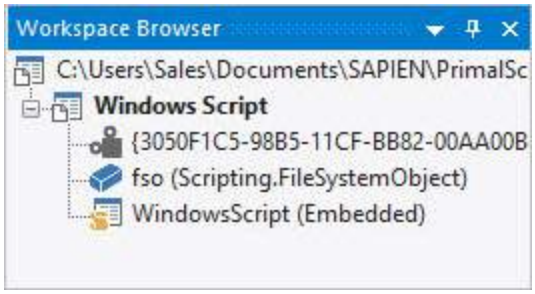
```
<?xml version="1.0"?>
<package>
    <comment>
    PrimalCode wizard generated file.
    </comment>
    <job id="Windows Script">
        <reference id="MSHTML" guid="{3050F1C5-98B5-11CF-BB82-00AA00BDCE0B}" version="4.0"  />
        <object id="fso" progid="Scripting.FileSystemObject"  />
        <script id="C:\PS\AlterSP.ps1" src="C:\PS\AlterSP.ps1" language="Unknown" />
        <script id="WindowsScript" language="VBScript" >
<![CDATA[
'
' Primalscript 2007 Windows Script Wizard generated
'


]]>
        </script>
    </job>
</package>
```

You can also switch back to the workspace by right-clicking in the editing pane.

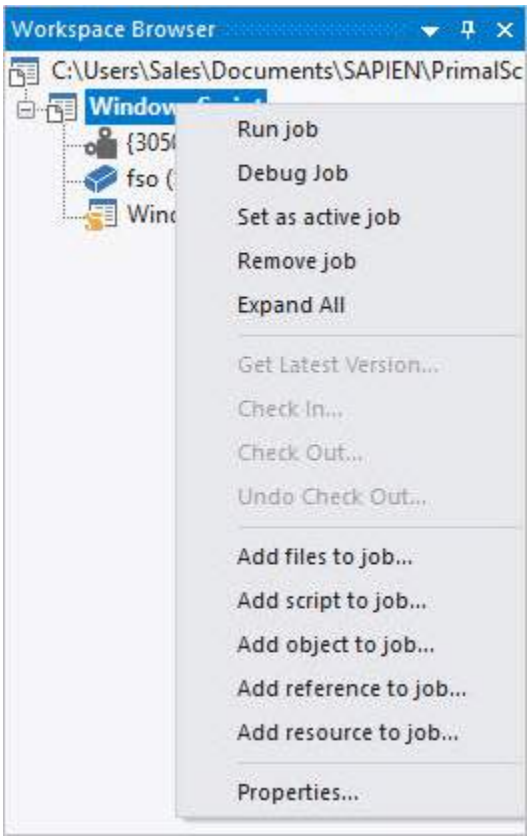


The context menu also offers options to add the workspace to source control (or, if it's already been added, check it in or out).

The **Workspace Browser** provides the key to managing the file. In addition to the actions available by right-clicking the top-level workspace item, you can also right-click a job (the next level of item), or any of the items within a job.



Right-clicking a job (such as "WindowsScript," shown here) gives you more options:



Right-clicking other objects—such as objects, references, or scripts—allows you to remove or rename them, as appropriate.

Each WSF file or workspace can contain multiple jobs. Each job can contain one or more scripts, files, references, objects, and so forth.

To execute the final WSF and run a specific job

- Run `filename.wsf //job:jobid`.

WSF Properties

Each job within a WSF workspace has a set of properties. These include the job's name (or ID) which is used to reference the job when running the final WSF.

The job can also have a text description which describes what the job does. If the WSF is run with only a **/?** argument, the description is part of what is automatically displayed.

A job can also have a usage explanation and an example. Both are textual fields that embed information within the job for future reference. Finally, jobs can have zero or more arguments. These are defined command-line arguments which can be accessed within the job's scripts via the **WScript.Arguments** object.

Arguments have a description, or name. For example, the argument shown here would be accessed from the command line like this: **`multicomputer.wsg /list:listname`**, using the argument's name. Arguments can also have a line of help text which is displayed as part of the automatic **`/?`** feature. Finally, arguments have a type. The type can be string, Boolean (meaning True or False) or simple. Simple arguments are simply specified or not; they are not given a value. For example, **`/verbose`** is an example of a simple argument: either it's specified or it isn't.

WSF Code

Script code is included within the WSF normally using the PrimalScript code window. PrimalScript handles the XML formatting necessary to enclose the script within its job, and the job within the overall WSF.

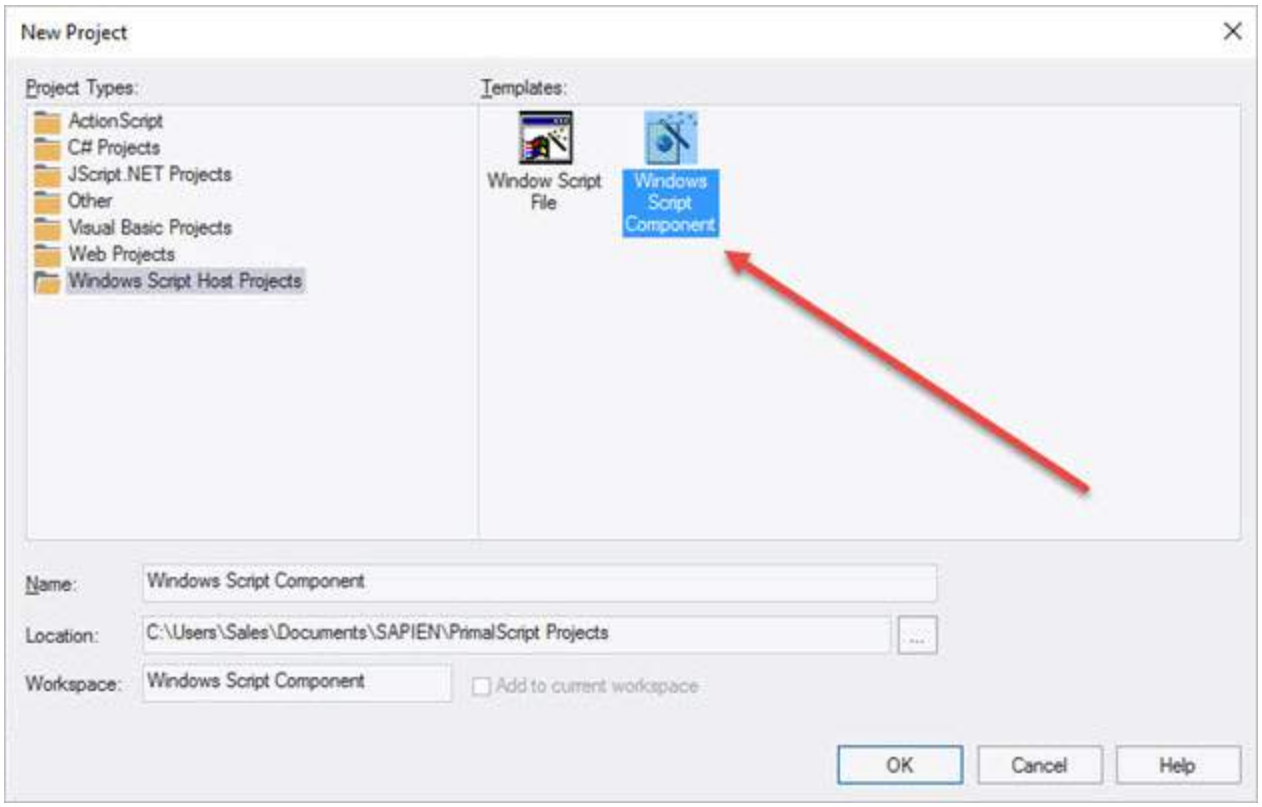
11.2 Windows Script Components

A Windows Script Component (.wsc filename extension) is a special, XML-formatted file that allows a script to function as a COM object much like a DLL. Using the WSC format, you can essentially write your own modular COM components using VBScript or JScript. Although the XML format is complex, you don't need to worry about it because PrimalScript handles it all behind the scenes, allowing you to focus on your scripts.

In this topic we show you how to create, configure, and use a WSC project.

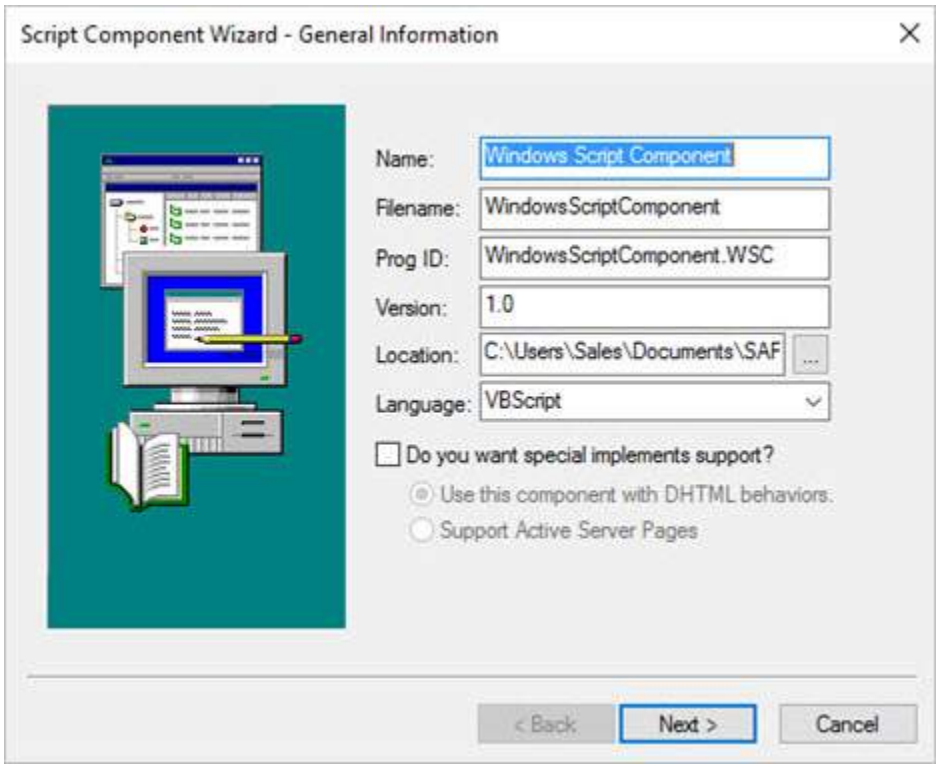
Starting a WSC

A WSC is a special kind of PrimalScript project. To begin one, select **New Project**, from the **File** menu. Under **Windows Script Host Projects**, select **Windows Script Component**. Provide a name and location for your new file and specify a name for the workspace.

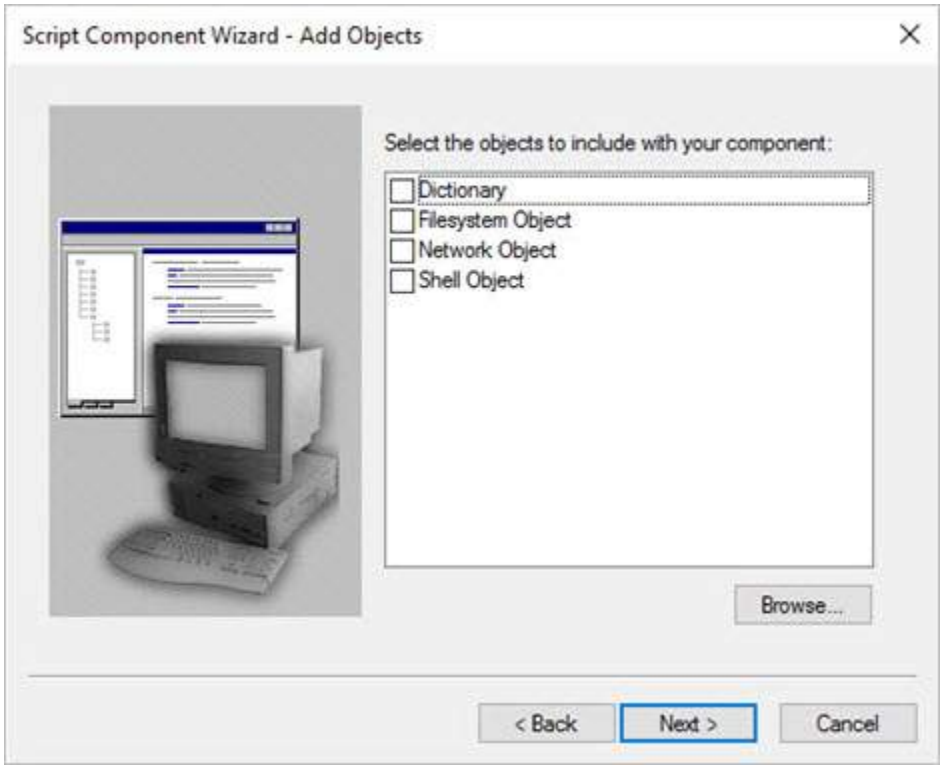


Next, specify the basic properties:

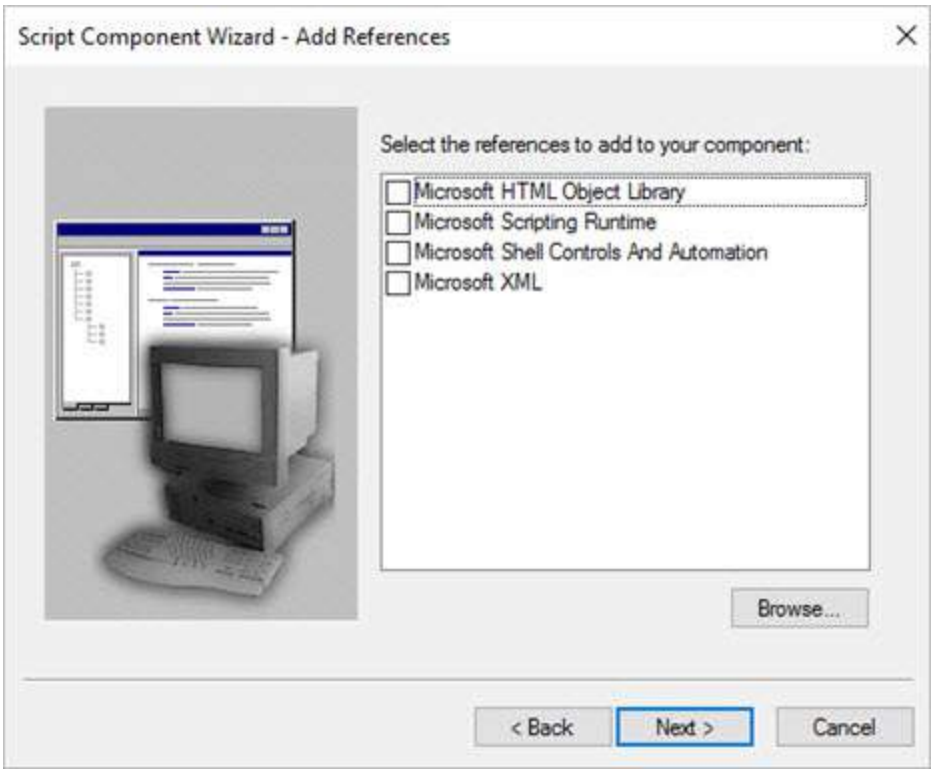
- Name of the script
- Location where the script will be stored
- ProgID which will be used to reference the component
- Scripting language



Next, select any objects that will be used within the component. You can click **Browse** to add objects which aren't on the default list. By referencing the objects that your scripts will use, you can use any constants defined in the objects' type libraries.



Similarly, you can add various other references in the Add References dialog.



Finally, you can add additional files to the component. These can be automatically copied to the script folder if desired.

After completing the Wizard, your new WSC will be ready. You can begin managing it by using the Workspace Browser.

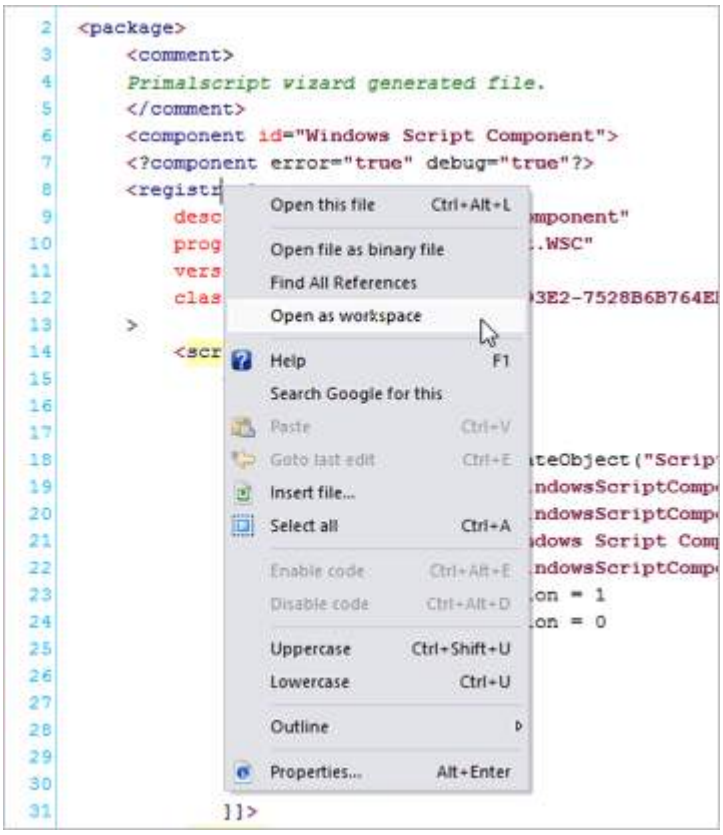
WSC Workspace

The WSF is part of a special PrimalScript workspace. You can add additional components and their scripts to the workspace. To do so, right-click the top-level workspace item in the Workspace Browser and select **Add new component to workspace**. The context menu also offers options to add the workspace to source control (or, if it's already been added, check it in or out).

You can also open the workspace (that is, the WSC file) as a text file. Doing so provides access to the raw XML formatting as well as the script code of the jobs contained in the workspace.

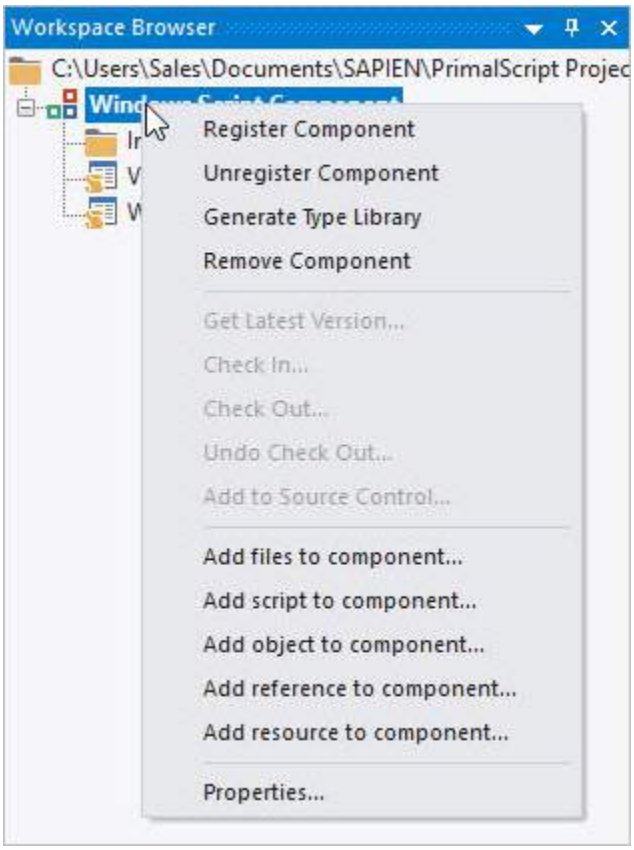
```
<?xml version="1.0"?>
<package>
    <comment>
    Primalscript wizard generated file.
    </comment>
    <component id="Windows Script Component">
    <?component error="true" debug="true"?>
    <registration
        description="Windows Script Component"
        progid="WindowsScriptComponent.WSC"
        version="1.0"
        classid="{3D6B06C6-5C7C-426A-93E2-7528B6B764EB}"
    >
        <script language="VBScript">
            <![CDATA[
                Function Register()
                    Dim TypeLib
                    Set TypeLib = CreateObject("Scriptlet.TypeLib")
                    TypeLib.AddURL "WindowsScriptComponent.WSC"
                    TypeLib.Path = "WindowsScriptComponentWSC.tlb"
                    TypeLib.Doc = "Windows Script Component"
                    TypeLib.Name = "WindowsScriptComponentWSC.tlb"
                    TypeLib.MajorVersion = 1
                    TypeLib.MinorVersion = 0
                    TypeLib.Write
                End Function

                Function Unregister()

                End Function
            ]]>
        </script>
```

Right-clicking in the editing pane allows you to switch back to the workspace.



The Workspace Browser provides the key to managing the file. In addition to the actions available by right-clicking the top-level workspace item, you can also right-click a component (the next level of item) or any of the items within a job.

Right-clicking a component (such as "Windows Script Component," shown here) provides a number of options:



Modifying the component's properties allows you to change its ProgID, description, and other details.

Right-clicking other objects—such as objects, references, or scripts—allows you to remove or rename them, as appropriate.

Remember, each WSC file, or workspace, can contain multiple components. Each component can contain one or more scripts, files, references, objects, and so forth.

WSC Code

Script code is included within the WSC normally using the PrimalScript code window. PrimalScript handles the XML formatting necessary to enclose the script within its job and the job within the overall WSC.

WSC Properties, Methods, and Events

Just as with regular COM components, a WSC can have members—properties, methods, and events, these are added by right-clicking the Interface item in the Workspace Browser and selecting the appropriate menu item.

When adding a method, you'll specify its name and any parameters it will accept.

PrimalScript will create the necessary function shell to implement the method. Similarly, creating a property allows you to indicate if it is a read/write property, a read-only property, or a write-only property, and PrimalScript creates the appropriate routines to handle the getting (reading) and setting (writing) of the property.

Note that members can have different external names (the names used when programming with the component from within another script) and internal names (the name the member is known by within its own script).

```
dim _IOMode
_IOMode = 0

function get__IOMode()
    get__IOMode = _IOMode
end function

function put__IOMode(newValue)
    _IOMode = newValue
end function
```

Using a WSC

Before a WSC can be used, it must be *registered*, just like a COM DLL.

To register a WSC

- Select **Script** > **Components** > **Register active component**, then click the **Register Component** button on Script toolbar or right-click the component in the Workspace Browser and select **Register Component**.

The component can be unregistered in the same way.

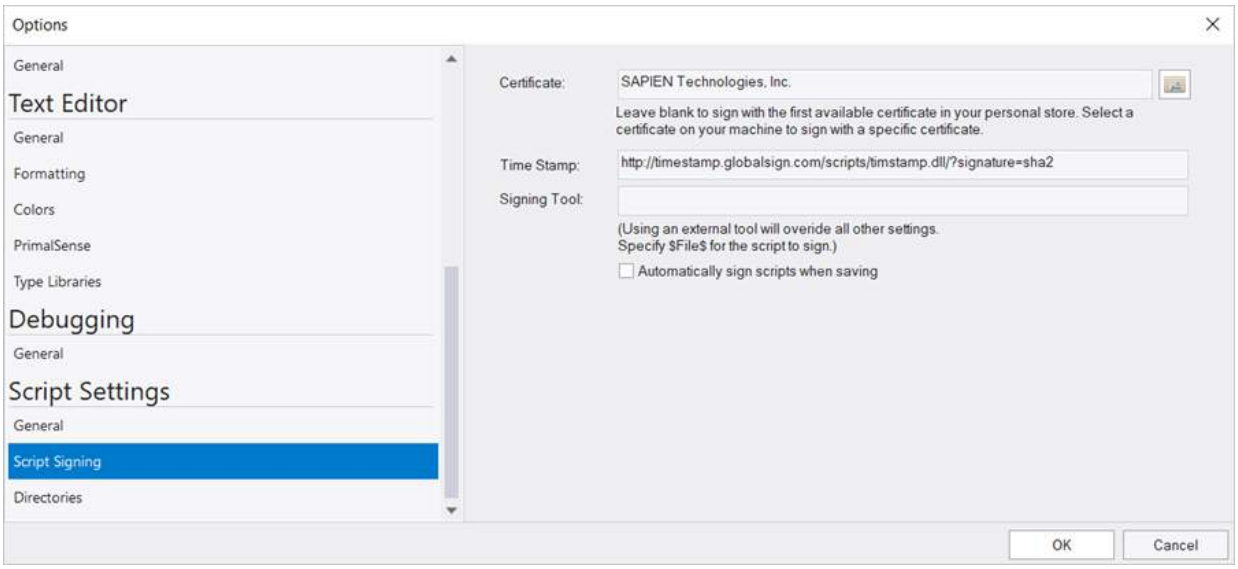
You can also generate a type library (.tlb file) for the component which allows features like PrimalSense to function for the component's members.

11.3 Script Signing

PrimalScript includes the ability to digitally sign VBScript, JScript, WSF, and Windows PowerShell scripts, provided you have installed a valid code-signing certificate on your computer. The certificate should be installed to the default personal store. First, you need to configure PrimalScript with the name of the certificate.

To configure PrimalScript to sign scripts

1. Select **File** > **Options** > **Script Settings** > **Script Signing**:



2. Leave the Certificate field blank to sign with the first available certificate in your personal store, or select a certificate on your machine.
3. Select **OK** to save.

Select **Automatically sign scripts when saving** to have PrimalScript automatically sign all scripts when those scripts are saved.

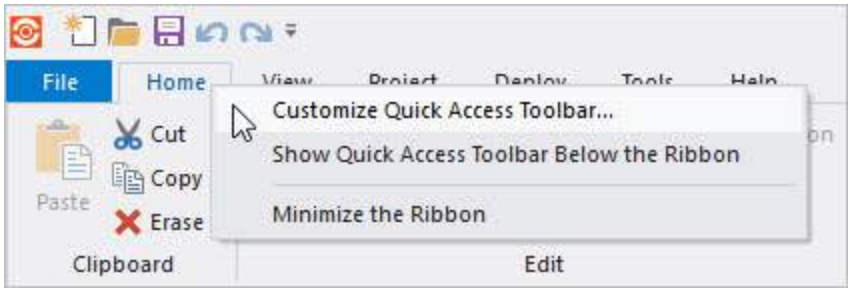
Different certificate settings are available for Windows PowerShell and Windows Script Host (VBScript and JScript).

11.4 Script Encoding

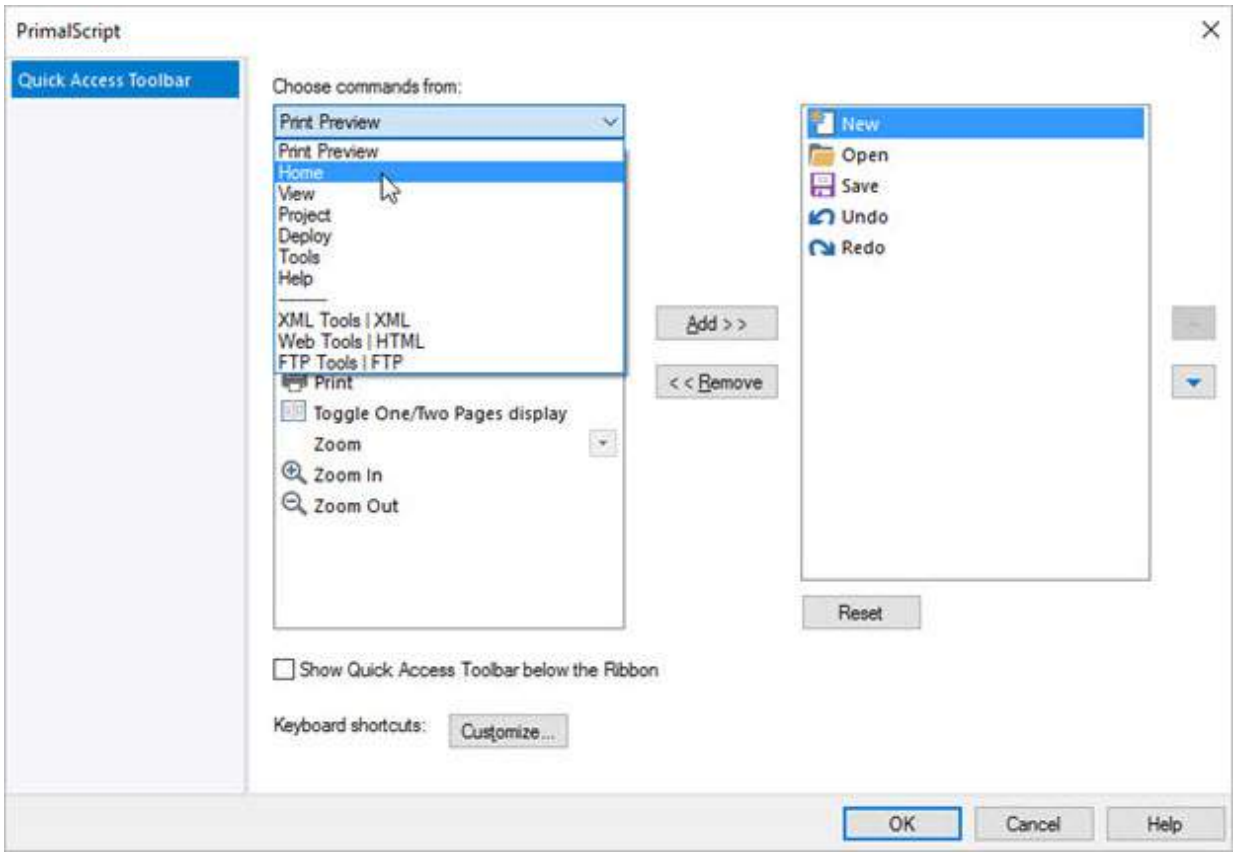
PrimalScript can save VBScript and JScript scripts that are encoded using the Windows Script Encoder. The Encoder helps to protect the source code of your script while still allowing WSH to execute it. However, because numerous Decoders exist on the Internet, you should not rely on Encoding to protect sensitive information—such as credentials—that are hardcoded into your scripts.

To encode a script

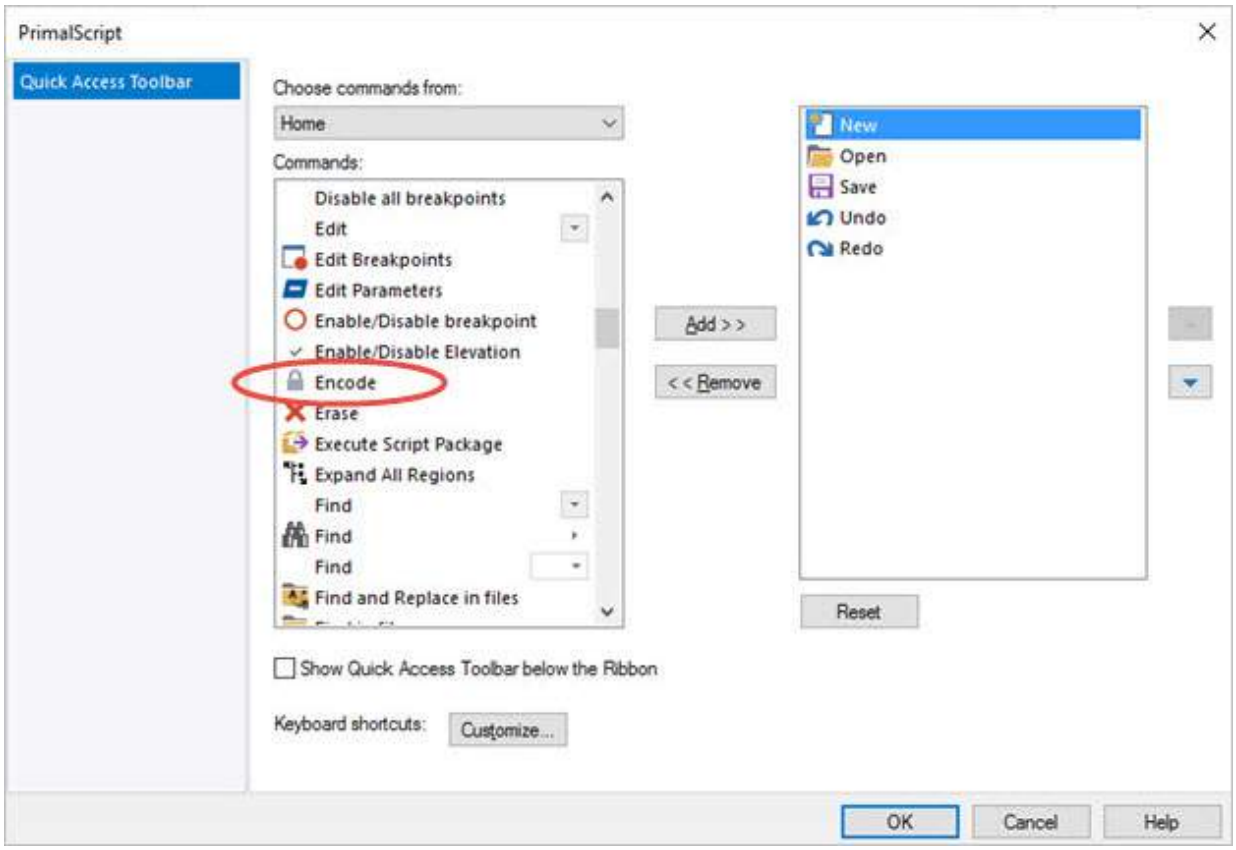
1. Right-click on the **Home** tab > **Customize Quick Access Toolbar**.



2. Choose **Home** from the drop-down.



3. Select **Encode** from the Commands list.



4. Click **OK**.

11.5 Windows Script Host TrustPolicy Settings

PrimalScript includes support for Windows Script Host TrustPolicy which is a feature built into WSH that can help prevent untrusted scripts from running.

PrimalScript provides access to the per-user TrustPolicy configuration through the **Options** dialog.

On Windows XP and later computers, you must first manually set the registry key **HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows Script Host\Settings\UseWINSAFER** to **0** in order for TrustPolicy to take effect. Doing so allows TrustPolicy to override Software Restriction Policies.

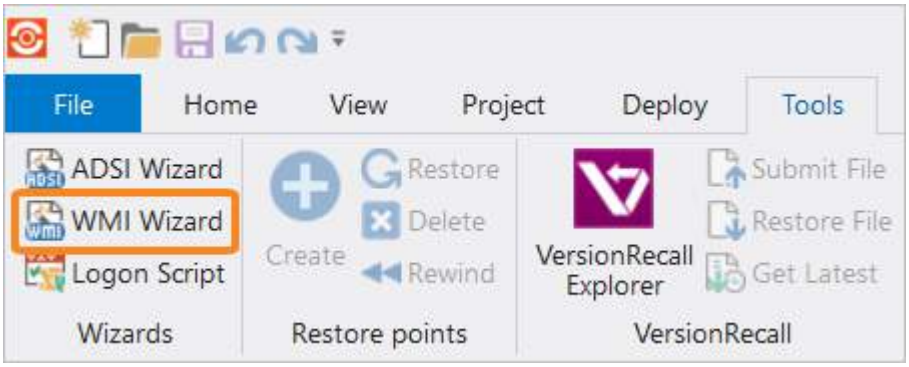
Be aware that TrustPolicy can also be configured on a machine-wide basis in a way that overrides the settings PrimalScript configures, and that these settings can be deployed in a non-overridable fashion from Group Policy.

11.6 WMI Wizard

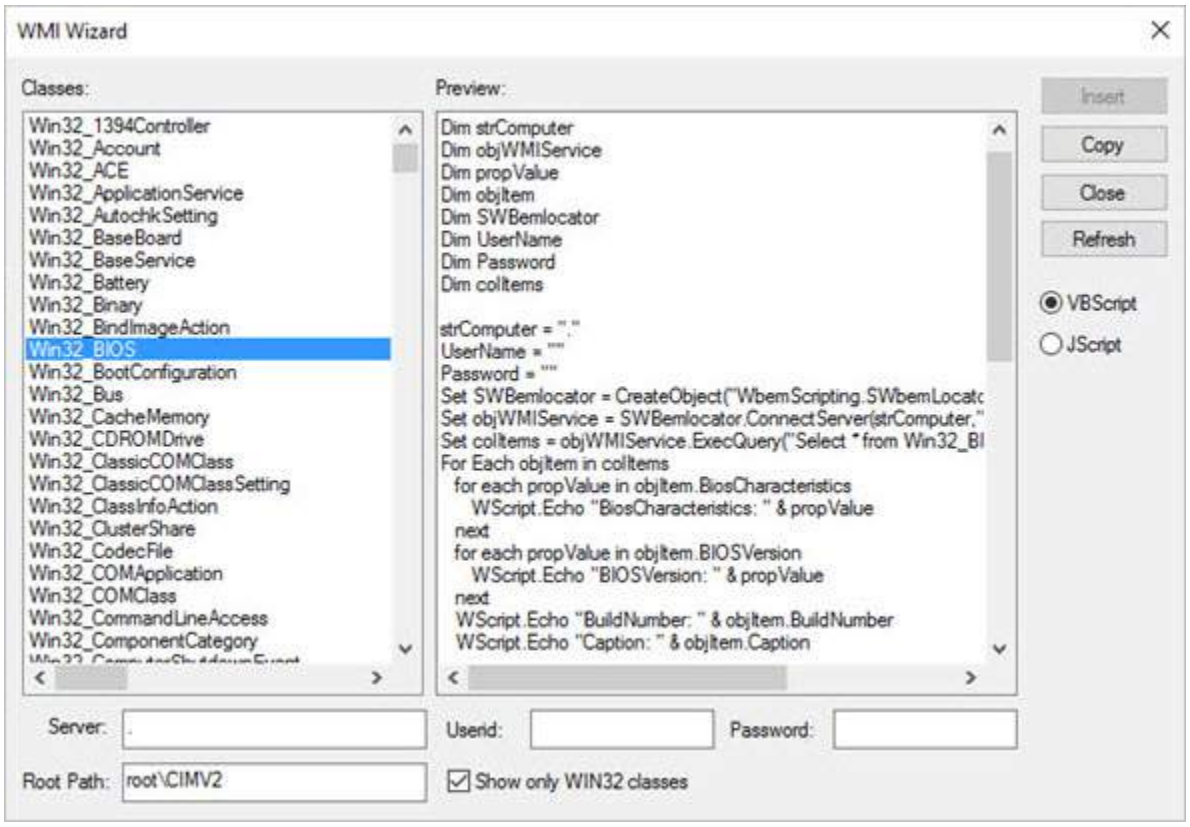
PrimalScript includes a convenient WMI Wizard that writes short Windows Management Instrumentation scripts for you.

Using the WMI Wizard

Launch the WMI Wizard by selecting the **Tools** ribbon > **Wizards** section > **WMI Wizard**:



The Wizard can query most available WMI classes. By default, it looks only at classes beginning with "Win32" in the **root\cimv2** namespace of your local computer.



The Wizard can produce scripts in either VBScript or JScript, although it defaults to the language currently in use if you open it while a VBScript or JScript file is open and active.

To query the WMI namespaces on another computer

1. Specify that computer's name in place of "." and, if necessary, provide proper credentials for the remote computer.
2. To display classes other than those in the **\root\cimv2** namespace, type the new namespace (such as **\root\MicrosoftIISv2**).
3. Finally, uncheck the **Show only WIN32 classes** checkbox to display classes whose names do not begin with "Win32". After specifying these options, click **Refresh** to refresh the class list.
4. Select any class to see its sample script and click **Insert** to insert the script into your current file or click **Copy** to copy the WMI script to the Clipboard.

11.7 ADSI Wizard

PrimalScript includes a convenient ADSI Wizard that makes writing Active Directory Services Interface scripts easier for you. The ADSI Wizard only produces code in VBScript.

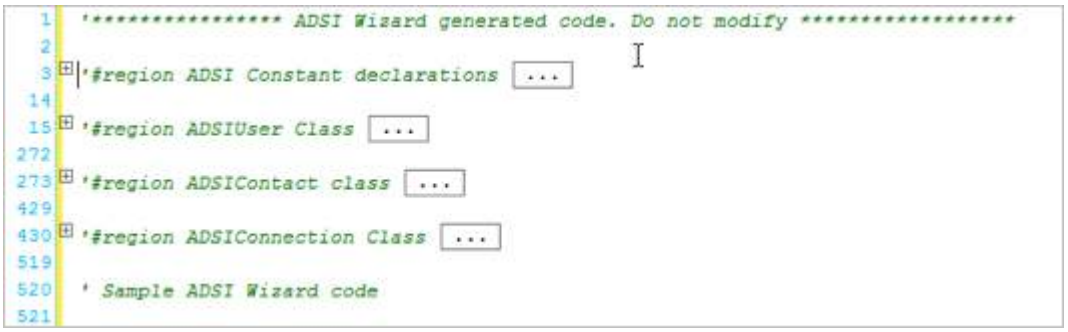
Using the ADSI Wizard

Launch the ADSI Wizard by selecting the **Tools** ribbon > in the Wizards section, select **ADSI Wizard**.

In order to open the ADSI Wizard you must have a .vbs file currently open.

Tell the Wizard which ADSI objects you want to work with: Users, Contacts, Computers, Organizational Units, or Groups. For each type of object, you can have the Wizard produce sample code showing how to add, delete, or modify objects.

The Wizard creates a new class for each type of object selected as well as a generic ADSIConnection class which connects to ADSI. The code added by the Wizard is pre-folded.



Additional sample code demonstrates how to use the new classes to query and work with ADSI objects.

```
    ' Additional Wizard Sample Code
    Call objADSI.GetUser(objADSIUser,"cn=Users","Hooten")
    call objADSI.GetContact(objADSIContact,"ou=HR,ou=Depts","Gates")
    objADSI.CreateContact "ou=HR,ou=Depts","Gates"
End If
' End sample ADSI Wizard code
```

To begin

- Declare a variable which will represent a retrieved ADSI object. Set the variable equal to a new instance of the appropriate class (e.g., Set objUser = New ADSIUser)

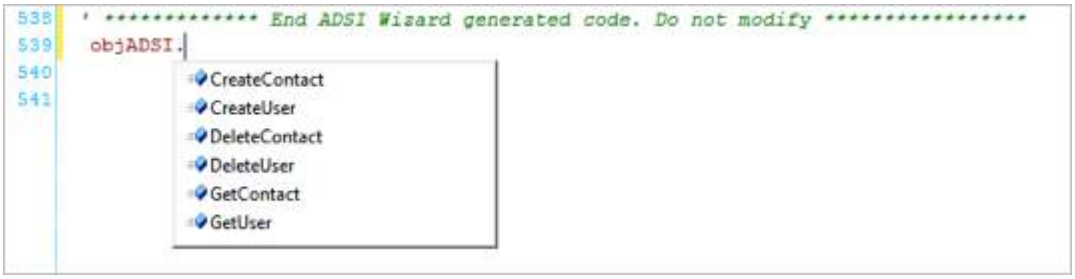
To retrieve an object

- Use the appropriate method of objADSI. The first argument of the "Get" method (such as GetUser) should be an object variable that will represent the retrieved object.

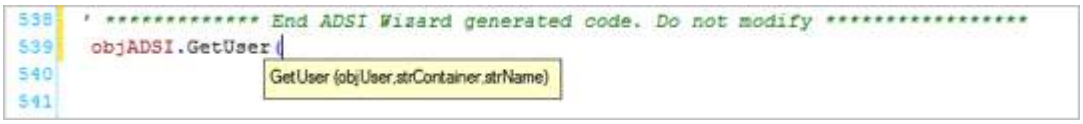
To work with an object

- Use the methods of the newly-retrieved object.

To assist you, the classes added by the ADSI Wizard all support PrimalSense[68]. For example, the objADSI variable supports methods for dealing with all of the object types you selected in the Wizard.



Pop-up tool tips help remind you of the correct syntax for using each class.

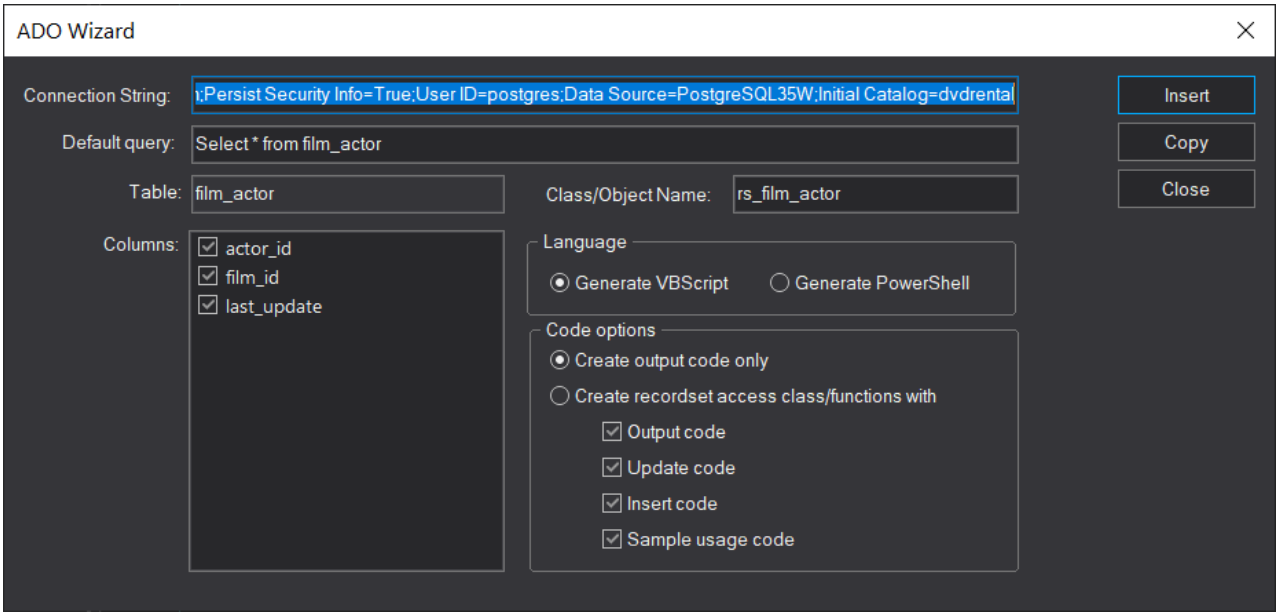


The objects will also display PrimalSense code hinting to help you select the appropriate property or method.

Rather than attempting to write scripts for you, the ADSI Wizard creates code that makes ADSI scripting easier by providing objects which represent specific classes, including full PrimalSense support and makes ADSI scripting more intuitive and direct.

11.8 ADO Wizard

The ADO (**A**ctiveX **D**ata **O**bjects) Wizard is accessible from the Database Browser[86].



12 Script Debugger

PrimalScript includes a highly functional integrated debugger for Windows scripts (VBScript, Jscript and Windows PowerShell). You can also access an external debugger from PrimalScript. This section covers both the integrated and external script debuggers.

12.1 Integrated Script Debugger

This section covers PrimalScript's integrated Script Debugger, which provides built-in debugging for PowerShell (PS), Windows VBScript (VBS), and JScript (JS) files.

12.1.1 Debugger Security

The information in this section is important for understanding how PrimalScript's debugger works, including its requirements and limitations.

The integrated debugger relies on Microsoft Windows components, including the Machine Debug Manager and Process Debugger. These are already available on most Microsoft Windows computers although they may not be properly registered with the operating system. In addition, the debugger is designed to obey Windows' security infrastructure regarding debugging. As a result, the following conditions apply to its use:

- The debugger may need to be used, for the first time on each computer, by a user who has permissions to install and start new services (specifically, the Microsoft Machine Debug Manager which may already exist but which may need to be registered and started). You can see if this will be necessary by examining the list of services installed on the computer; if the Machine Debug Manager service exists and is started, the debugger will not need to register and start it.
- The debugger can only be used by user accounts possessing the **Debug** Windows user right. This is by design and cannot be circumvented.
- If the debugger is unable to access the Machine Debug Manager service, or if it is unable to obtain Debug permissions from the operating system, it will display an error message.
- Some Microsoft software may create a "debugging" or "debugging-users" group. The user running PrimalScript needs to belong to this group in order for the debugger to work properly.

Some organizations restrict use on their computers by using Group Policy objects, user rights assignment, and security templates. If your organization has done so in such a way that your user account is unable to properly register or start the Machine Debug Manager service, or if your user account does not have the Debug user right, then *the debugger will not function*. The debugger's inability to function is a consequence of the security decisions your organization has made, and SAPIEN Technologies cannot provide a workaround for your organization's security policies.

These conditions are imposed because the debugger uses many of Windows own services and capabilities to provide advanced debugging features and those services and capabilities are integrated with Windows' own security infrastructure.

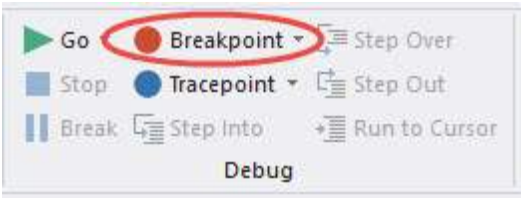
12.1.2 Setting Breakpoints

Breakpoints instruct the debugger to stop on a specified line of code. This allows you time to review what the script is doing at that point.

In PrimalScript, breakpoints are toggled switches; click to create it, click again to delete it.

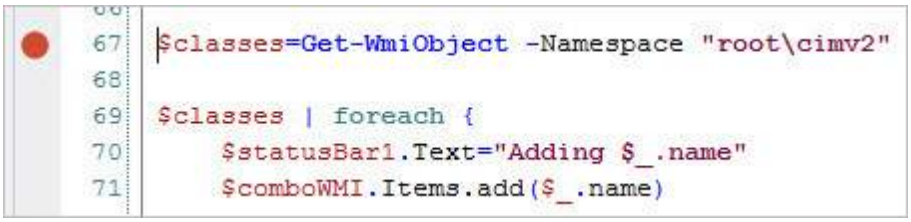
To set a breakpoint

1. Place your cursor on the line where you want the breakpoint.
2. Click **Home** > in the Debug section, click **Breakpoint** (*F9*):



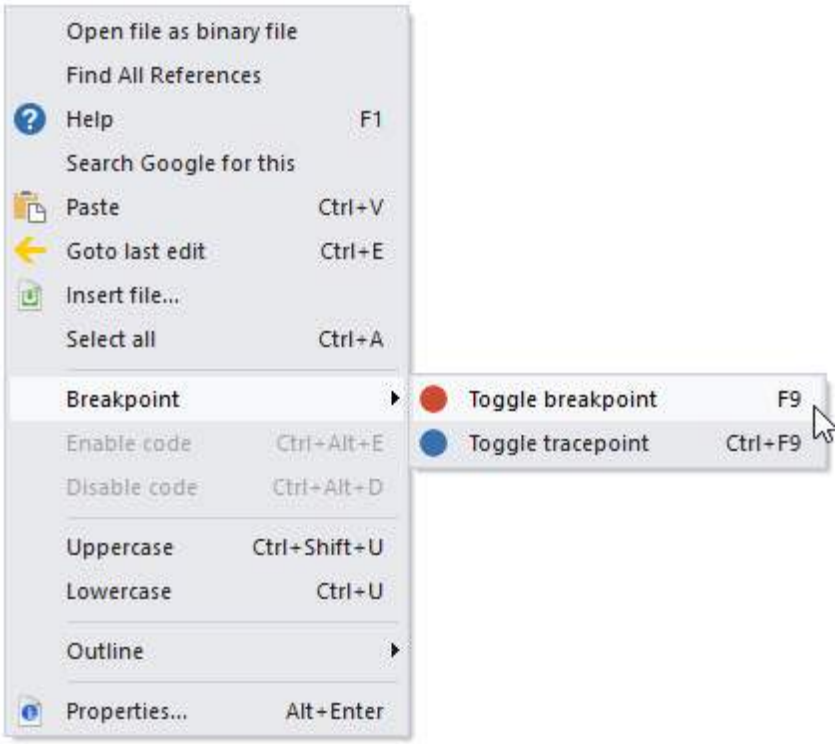
*-**OR**-*

- Click the margin to the left of the line numbers:



*-**OR**-*

- Right-click the line, point to Breakpoint, and then click **Toggle breakpoint**:



To disable a breakpoint

A disabled breakpoint (unfilled red circle) appears in the script, but does not break into the debugger.

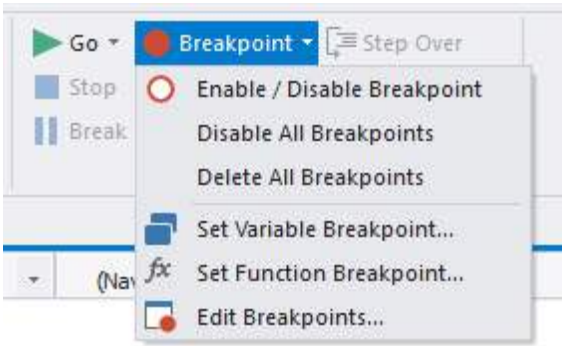
1. Place your cursor on the line with the breakpoint.
2. Click **Home** > in the Debug section, click the **Breakpoint** menu > then click **Enable/Disable**.

To delete a breakpoint

- Repeating any of the steps used to create a breakpoint, such as pressing ***F9***.

To disable or delete all breakpoints

- Click **Home** > in the Debug section, click the **Breakpoint** menu > then click **Disable all Breakpoints** or **Delete all Breakpoints**:

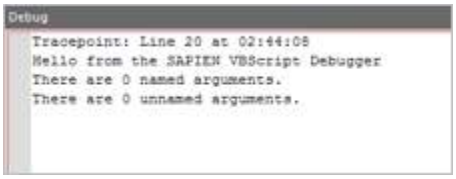


12.1.3 Setting Tracepoints

Tracepoints are designed to provide to VBScript users a feature much like debugging statements (Write-Debug) in Windows PowerShell. Tracepoints are an alternative to adding and then deleting or commenting WScript.Echo calls.

When you execute the script in the PrimalScript debugger, tracepoints log text to the Debug pane. When you run the script, they have no effect.

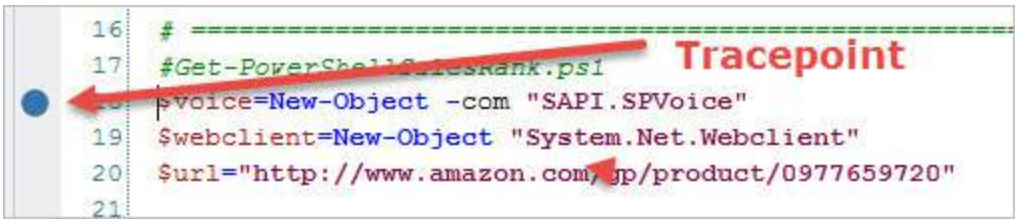
Think of it as a WScript.Debug. You can use the same syntax and expressions you would normally use in a WScript.Echo call.



For example, to determine why the output from a SQL query fails after a few hundred records, instead of stepping through the records, add a TRACE statement that records the value after each query. When the script stops, you can use the trace output to diagnose the problem.

To add a tracepoint

- Click **Home** > in the Debug section, click **Tracepoint** (*Ctrl+F9*):



To delete a tracepoint

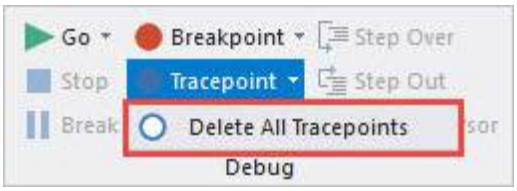
- Click the tracepoint.

*-**OR**-*

- Click the tracepoint line, click **Home** > in the Debug section, click **Tracepoint**.

To delete all tracepoints

- Click **Home** > in the Debug section, click the **Tracepoint** menu > then click **Delete all Tracepoints**:

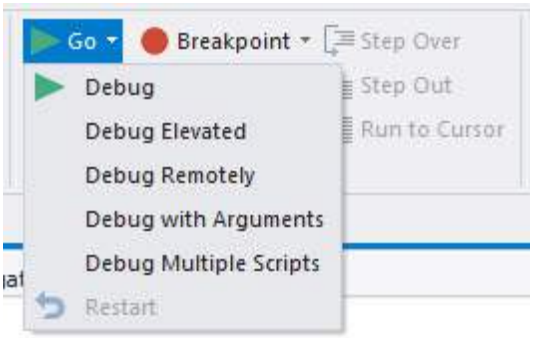


12.1.4 Debugging Scripts

When you debug a script, you control the execution of statements. The debugger stops at breakpoints and writes trace statements at tracepoints.

To start the debugger

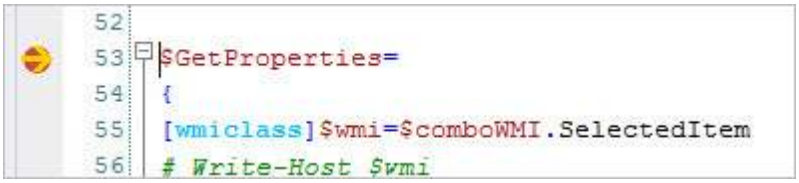
- Click **Home** > in the Debug section, click **Go** (*F5*):



To debug a script in elevated mode

- Click **Home** > in the Debug section, click the **Go** menu > then click **Debug elevated**.

  The script runs and writes output to the Output pane until it reaches a breakpoint. When it stops at a breakpoint, a yellow arrow in the margin indicates the line of code executes next:



  When the script is stopped at a breakpoint, you can add or remove breakpoints, but you cannot edit the script.

To execute the next line of code (step into)

- Press ***F11***.

  The yellow arrow advances to the next line of code.

To stop the debugger

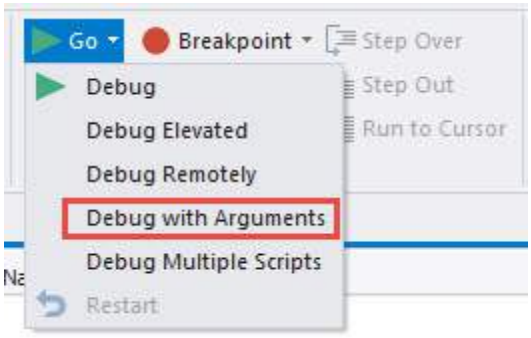
- Click **Home** > in the Debug section, click **Stop** (*Shift+F5*).

12.1.5 Adding Arguments

You can write scripts that require command-line arguments or parameter values. To test these scripts in the debugger, you need to pass test arguments to your script.

To debug a script that requires arguments

- Click **Home** > in the Debug section, click the **Go** menu > then click **Debug with arguments**:



This feature will allow you to specify script command-line arguments while debugging.

You can also store the debug parameters within the script by adding **%DebugArguments%** comments to the code.

```
# %DebugArguments%=Server01
```

or in VBScript

```
' %DebugArguments%=Server01
```

To specify an argument with spaces or special characters, enclose it in quotation marks.

```
# %DebugArguments%="New York"
```

To specify multiple arguments, enter them in the order that they are specified in the script.

```
%DebugArguments%=Servername Username password
```

12.1.6 Using Meta-Comments

PrimalScript lets you use meta-comments to determine how and where your scripts are run. You can run them in 32 or 64-bit mode, elevated or not, remotely or locally, and so on.

Meta-comments override your current platform settings.

Syntax for meta-comments

- **%ForcePlatform%=32 | 64**
  # %ForcePlatform%=64 (*runs the script in 64-bit mode*)
- **"%ForceElevation%"=true | false**
  # %ForceElevation%=yes (*runs the script elevates*)
- **%ForceShell%=Name**
  # %ForceShell%=PowerShell 64 bit (*runs the script in 64-bit Windows PowerShell*)
  # %ForceShell%=PowerShell Elevated (*runs a Windows PowerShell script elevated*)
- **%ForceHost%=Hostname[,User[,password]]0**
  # %ForceHost%=JABBA (*runs the script on a remote computer named JABBA*)
- **%ForceCodePage%=Codepage**
- **%ForceSTA%=true | false**

You can combine meta-comments. For example, the following meta-comments run the script in 64-bit mode *and* elevated:

You can also use meta-comments to specify optional credentials. For example, the following meta-comments force the script to run on the JABBA computer and prompt for user name and password:

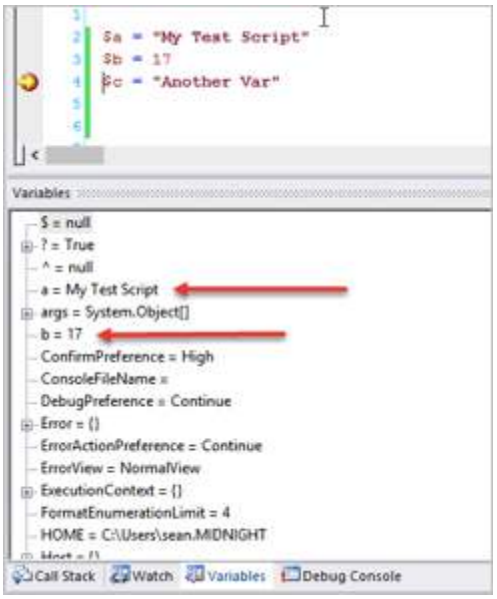
```
# %ForceHost%=JABBA,Prompt
```

The following meta-comments force the script to run on JABBA, but prompt for a password only when the user name is "Alex":

```
# %ForceHost%=JABBA,Alex,Prompt
```

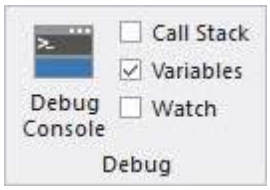
12.1.7 Examining Variables and Object Properties

The **Variables** pane displays a dynamic view of variable values and object property values that changes as the script runs. Only objects and variables defined thus far in the script are displayed.



To display the Variables pane

- Click **Home** > in the Debug section, check **Variables**:



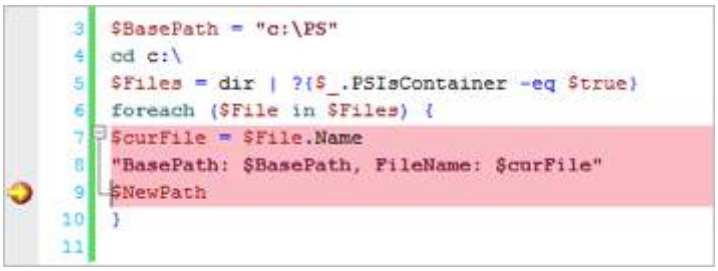
The Variables pane is updated whenever a line of script code is executed.

12.1.8 Evaluating Expressions

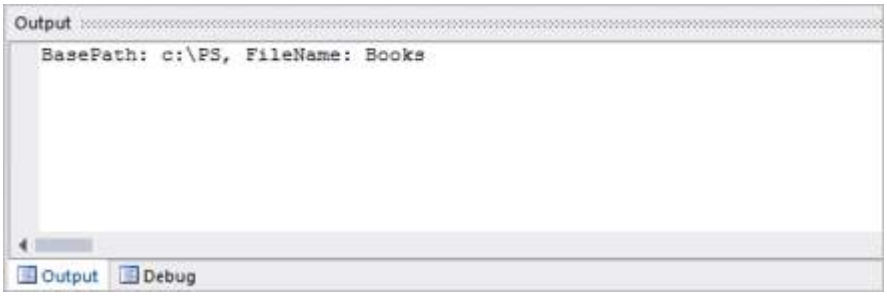
The debugger can evaluate complex expressions and show how the result of the expression changes as the script runs.

To evaluate an expression

1. Run the script to a breakpoint.
2. Click the expression and drag it to the **Output** pane.



PrimalScript re-evaluates the expression each time a line of script is executed so you can monitor the expression's result.



You can add multiple expressions to the list.

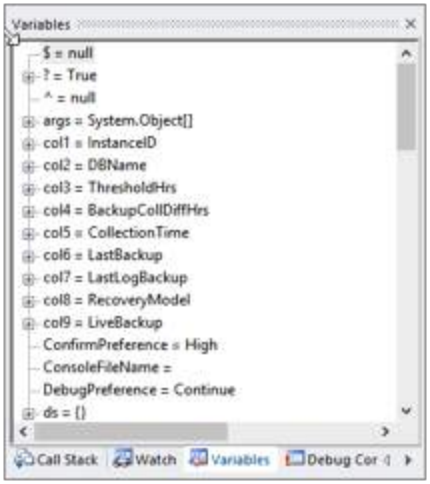
To delete an expression

- In the **Output** pane, select the expression and press **Delete**.

12.1.9 PowerShell Debugging Console

PrimalScript includes a special debugger console for Windows PowerShell scripts. It lets you debug in 32-bit and 64-bit mode and you can elevate the debugger to run as administrator.

The Variables pane displays the values of the Windows PowerShell automatic variables, as well as variables that are defined in the script:

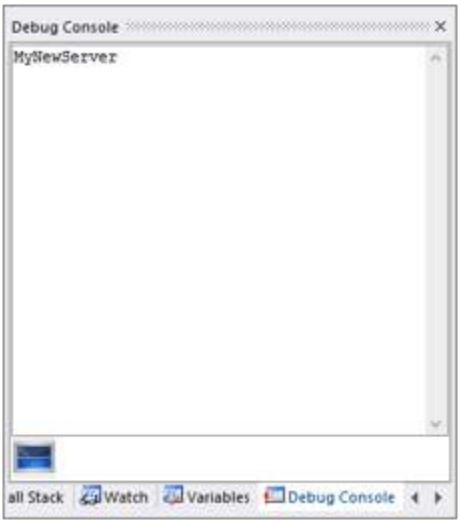


The Watch pane displays selected variables and their values, rather than looking at all the variables at once.

To add a variable to the Watch pane

- In the code editor, **click the variable**, and then **drag it into the Watch pane**:

The Debug Console lets you interact with the script runspace while you are stopped at a breakpoint:



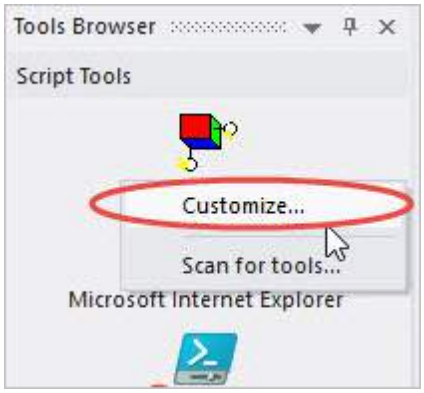
You can run commands to provide additional information, change the value of variables, or experiment with what-if conditions.

12.2 External Debuggers

You can add an alternate debugger (or any other external application) to PrimalScript.

To add an external debugger

1. Click **View** > in the Panels section, click **Tools**.
2. Right-click the Tools Browser and then click **Customize**:



To use an added external debugger

1. Click **View** > in the Panels section, click **Tools**.
2. In the Tools Browser, click the **icon for the debugger**.

13 FTP Client

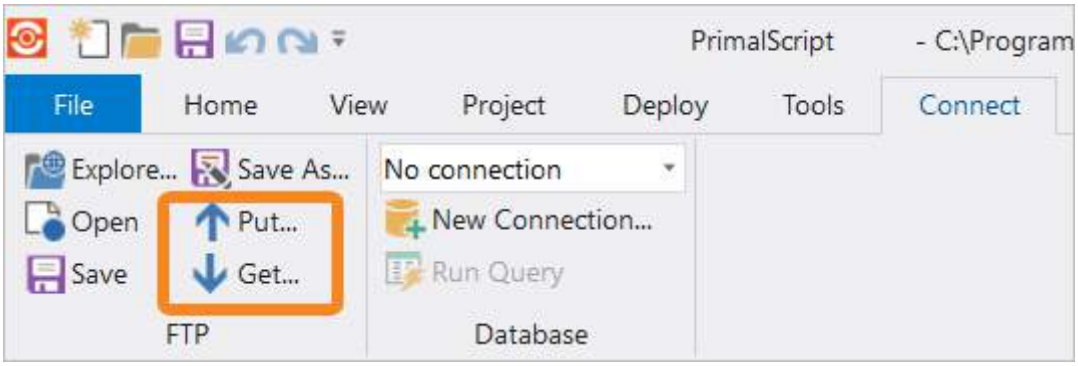
PrimalScript includes basic built-in FTP functionality to make sending and receiving files easier. This topic shows you how to send and receive files via FTP, and how to connect to an FTP site.

13.1 Sending and Receiving Files via FTP

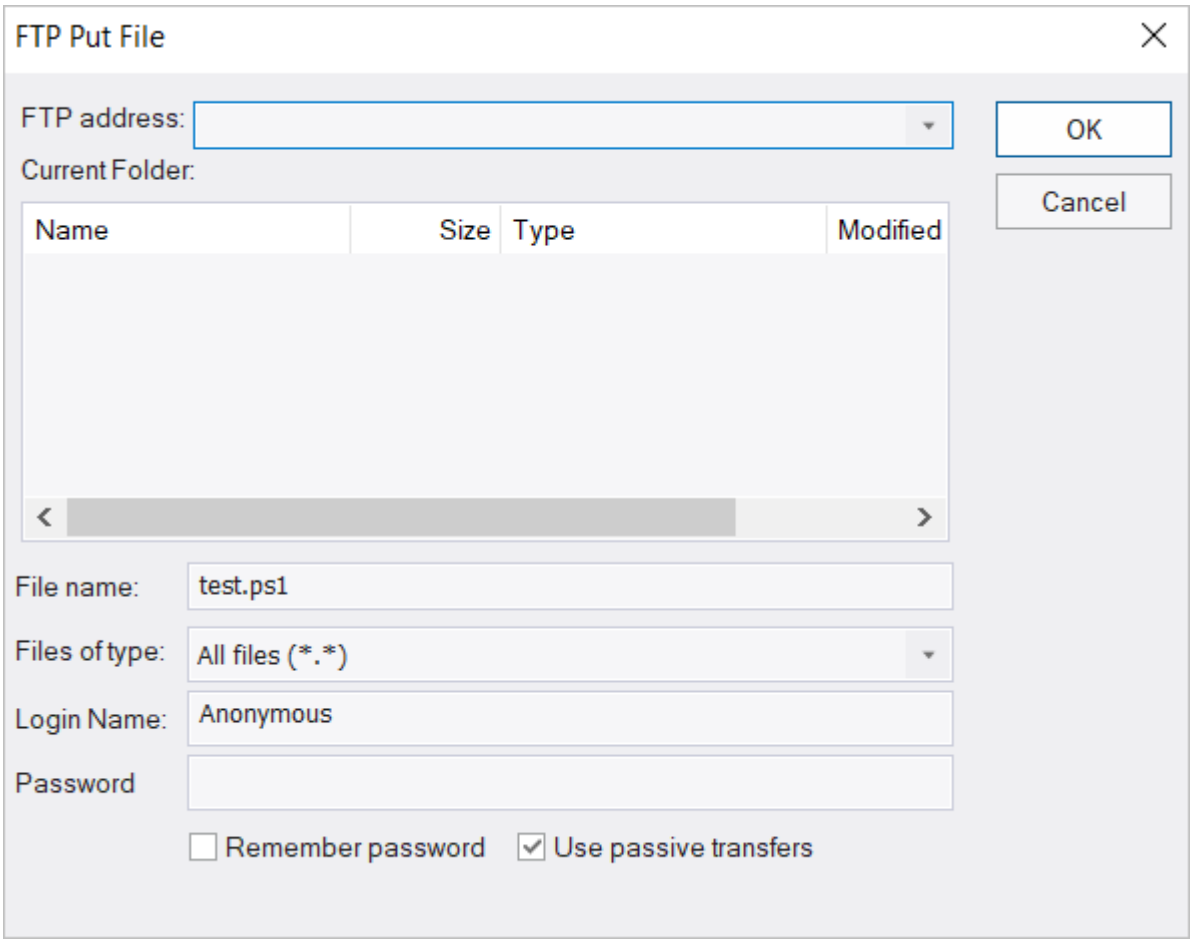
This topic shows you how to send and receive files via FTP.

To send or receive by FTP

1. On the **Connect** tab > in the FTP section, select **Put** or **Get**:



2. Enter the data for your files and then click **OK**:

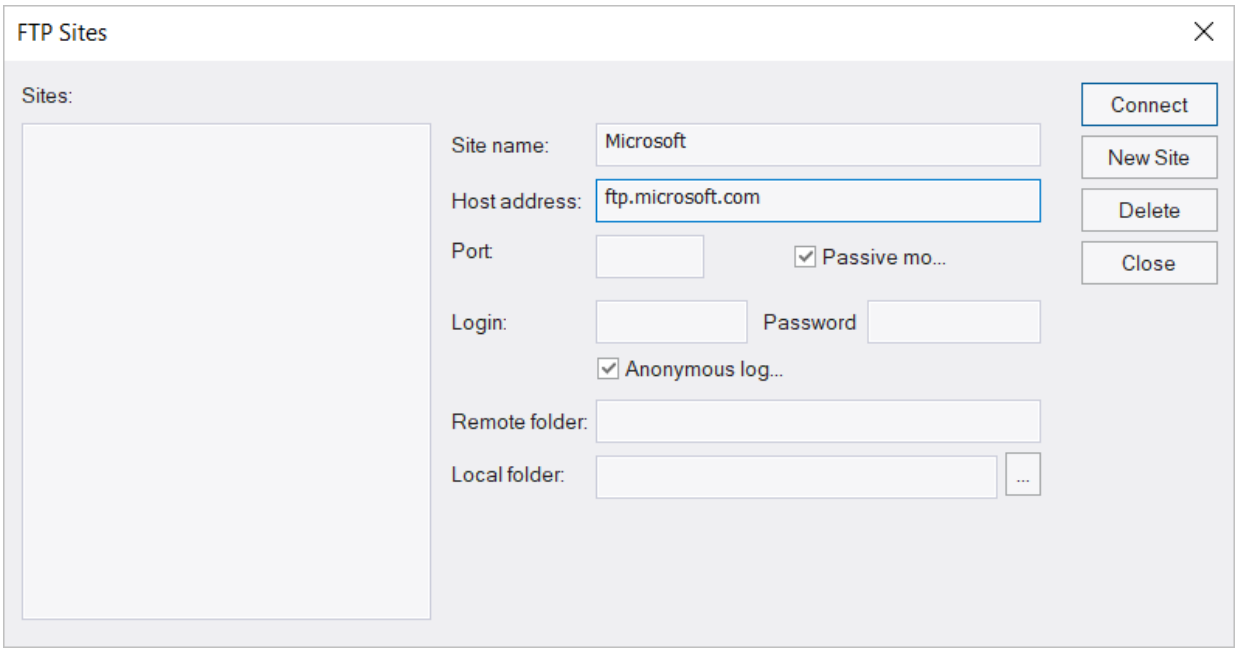


13.2 Exploring FTP Sites

This topic shows you how to connect to and explore an FTP site.

To define a named FTP site

1. On the **Connect** tab > in the FTP section, select **Explorer**.
2. Enter the FTP server and login credentials:



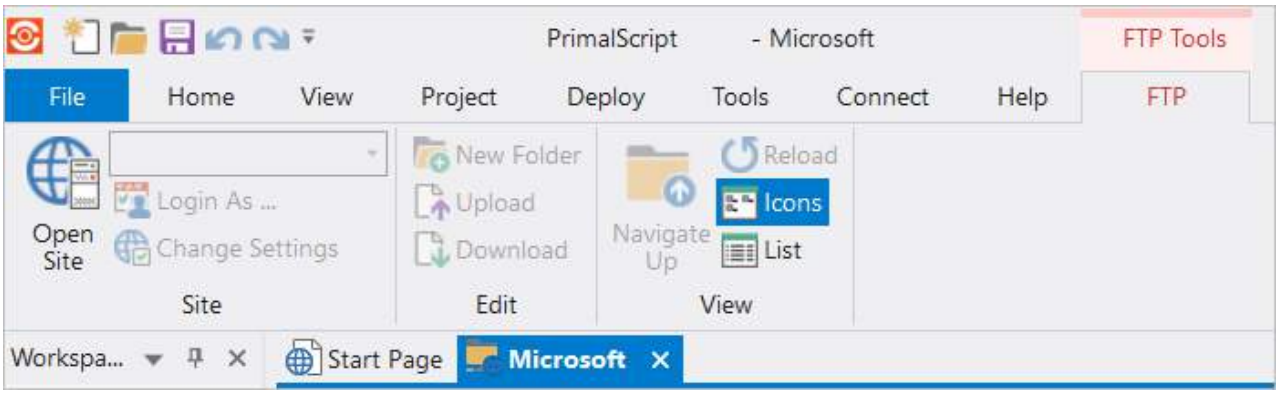
PrimalScript doesn't support SSL for FTP.

3. Click **Connect**.

   When you connect to an FTP site, an **FTP Tools** context ribbon and a tab for the FTP connection appear. The tab displays a file explorer for the FTP site.

On the tab for your FTP connection you can browse the files and folders and files on the site, and drag-and-drop files between your machine and the FTP server.

You can also drag and drop files from the **PrimalScript File Browser** to your FTP site, as well as upload and download files via the FTP tab.



14 Packaging Scripts

PrimalScript contains the Script Packager™, which can package single or multiple scripts, supporting files, and COM components into a single, standalone executable file (.exe).

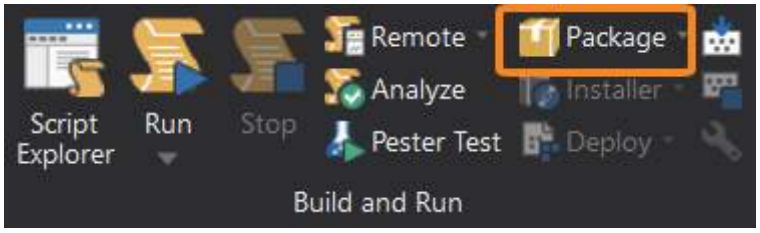
14.1 Creating a Script Package

This topic shows you how to create a script package.

If this is your first package, begin by setting up the Script Packager 184.

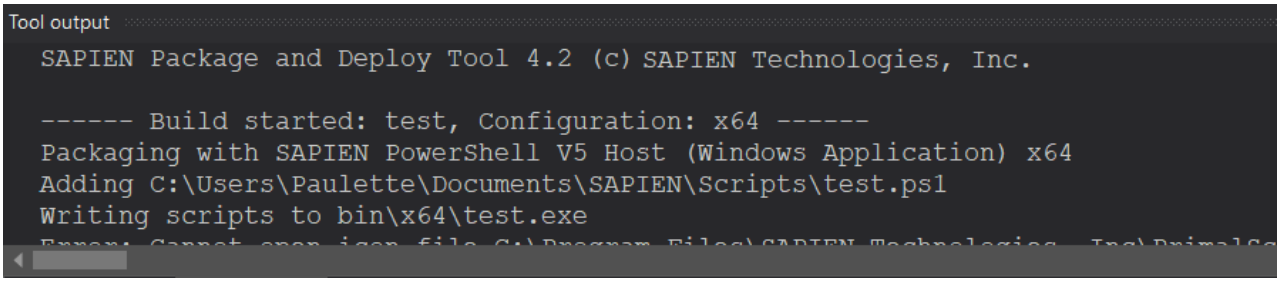
To create a package

- Click **Home** > and then in the Build and Run section click **Package** (*Ctrl+F7*):



PrimalScript checks the syntax of the designated files and packages them into an executable file (.exe).

If your build is successful, information about the new executable file is displayed in the Tool output pane:

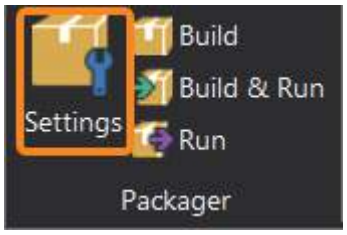


14.2 Setting up the Script Packager

The Script Packager contains everything you need to customize your executable files and create a package.

To open Packager Settings and configure a script package

1. Click **Deploy** on the ribbon, then click **Settings** in the Packager section to open the Script Packager interface:



2. Select the desired settings in the Script Packager interface (see details below):
   - Script Engine 185
   - Output Settings 186
   - Restrictions 192
   - Version Information 192
   - Build Commands 193

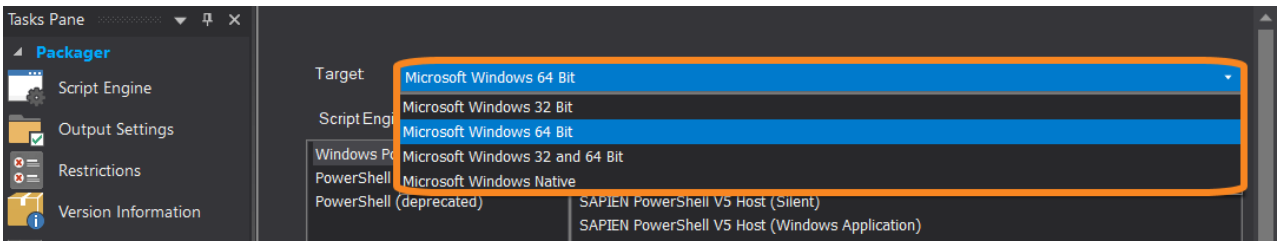
Script Engine

Target Platform

The Script Packager provides four options for building executables:

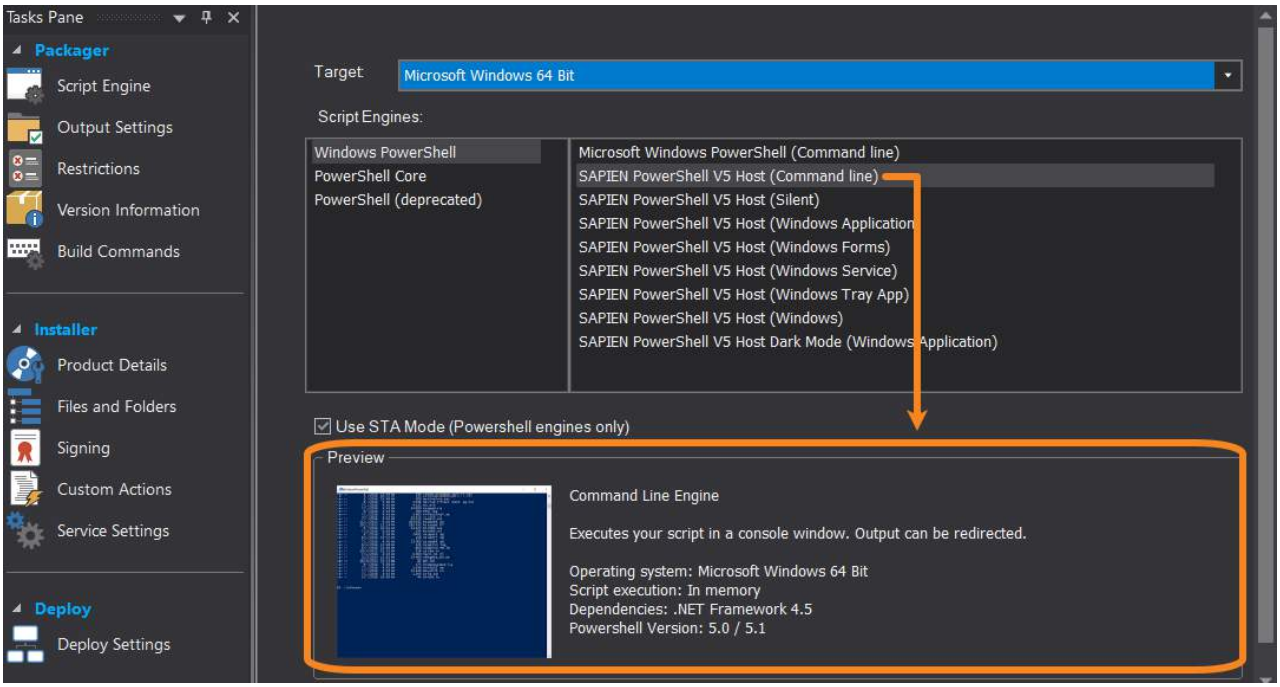
- **Microsoft Windows 32 Bit** will generate a 32 bit excecutable.
- **Microsoft Windows 64 Bit** will generate a 64 bit executable.
- **Microsoft Windows 32 and 64 Bit** will generate a 32 bit and a 64 bit executable.
- **Microsoft Windows Native** will create a starter executable which will launch the correct version depending on the current platform.

Select the desired platform from the options in the Target drop-down list:



Script Engines

Each script engine option provides a preview of what the selection will do:



Each package contains only one engine type. To include more than one script type in an executable file, create an MSI file.

STA Mode

Use STA Mode (Powershell engines only)

STA (Single Threaded Apartment) Mode allows you to start your script in single threaded mode. This is essential when your script uses forms to interact with the Windows GUI. Some GUI controls require STA mode in order for them to function correctly.

Output Settings

Output Settings Options

File name

Filename of the executable.

Folder

Folder for the executable.

It is recommended that you leave the common folder default name of "bin" for consistency. *Learn more* 188.

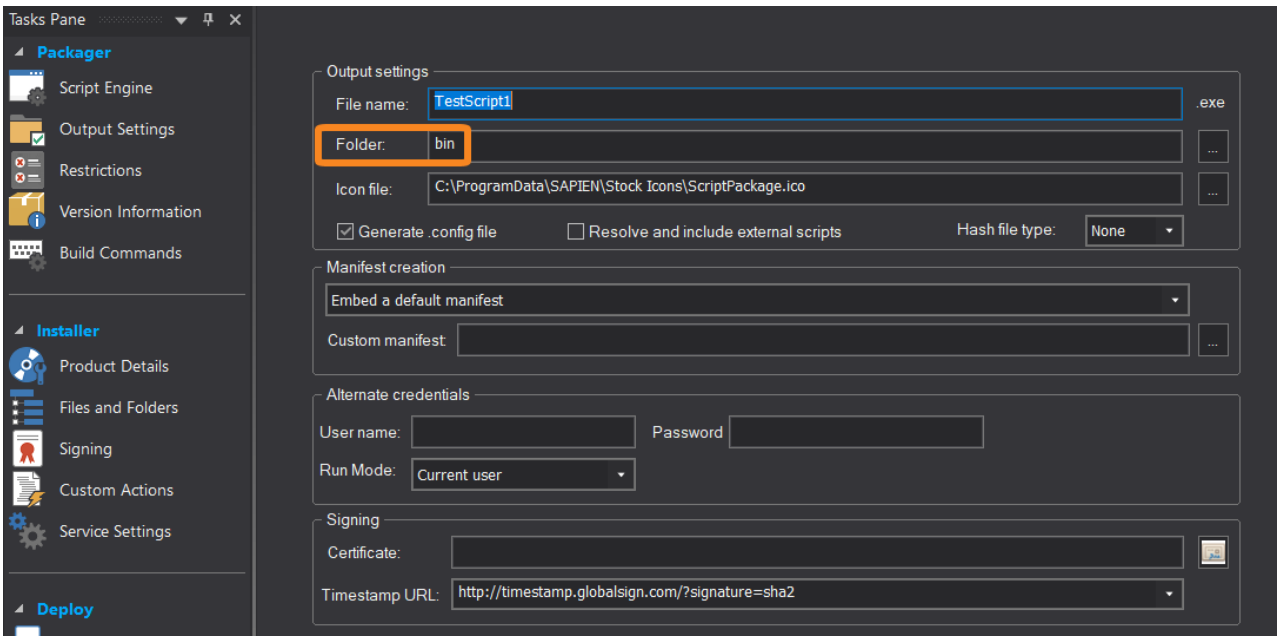
Icon file (optional)

A custom icon (.ico) for the executable.

Generate .config file	Generates a .config file. If you select *Windows Native*, .config files will be generated for all three of the .exe files.
Resolve and include external scripts	The code of external scripts will get injected into the packaged script when building the executable. Enable this option to resolve dot sourced files while packaging. *Learn more* 189.
Hash file type	Options for the Hash file type: *None*, *MD5*, *SHA1*, *SHA256*.
Manifest creation	Options for the manifest file, including a custom manifest. (This is an executable manifest, not a Windows PowerShell module manifest.)
Custom manifest	Opens a file to the specified line.
Alternate credentials	Uses the credentials of the specified user to run the scripts in the executable file. *Learn more* 190.
Run mode	***Current user***: Runs scripts with the permissions of the user who runs the executable file. ***Impersonate user***: Switches to the security context of the specified user, but uses the environment (e.g. network profiles, mapped drives, environment variables) of the current user. ***RunAs user***: Runs scripts with the permissions of the specified user in the specified user's environment. *Learn more about the Run Mode options* 191.
Signing	Specify the code signing certificate to sign your executable. If you specify a PFX file that requires a password, include it here. The Timestamp URL creates a timestamp for the signature used to sign the file, allowing the signature to remain valid even after the certificate expires.

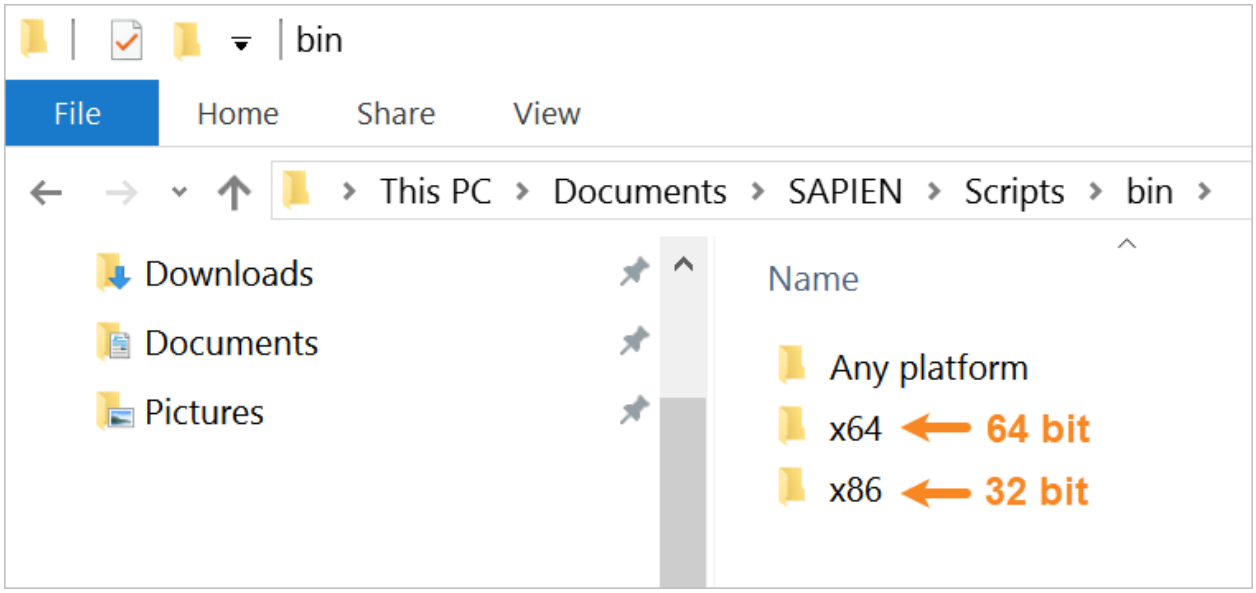
Engine Settings

The packaged executable files are generated in a platform specific folder under a common folder. It is recommended that you leave the common folder default name of "bin" for consistency:

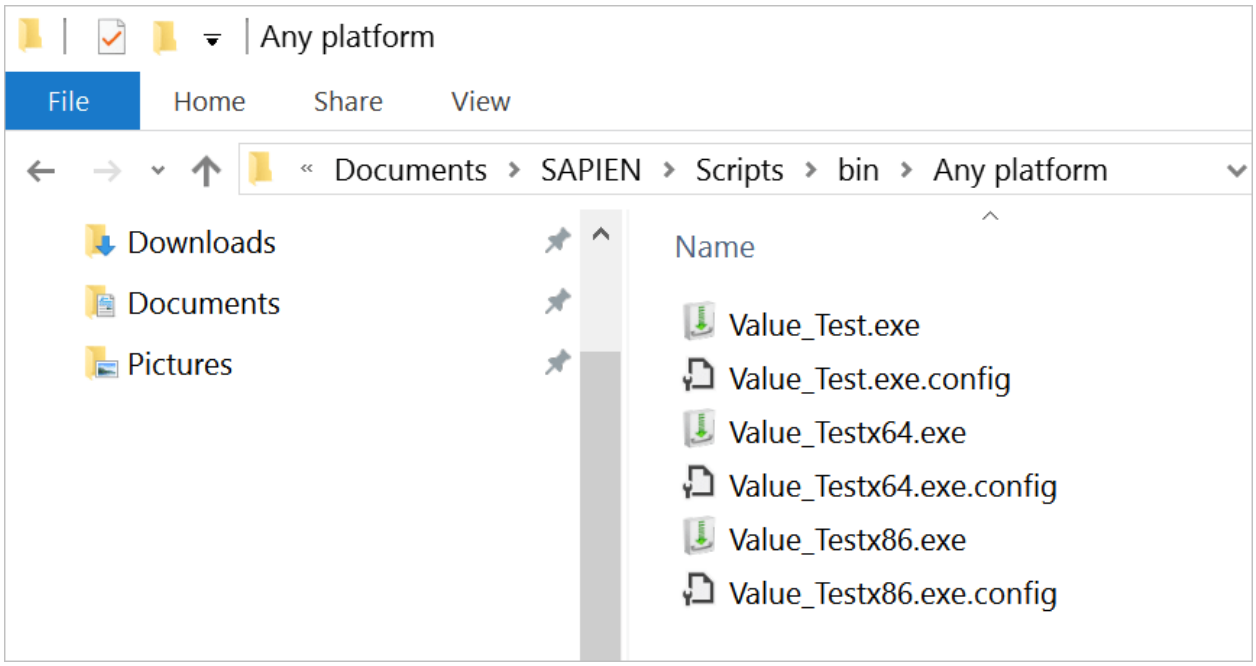


The build target you select will determine the platform specific folder that the packaged file(s) are generated in:

- **32 bit** files will be in ***bin\x86***
- **64 bit** files will be in ***bin\x64***
- **32 bit and 64 bit** files will be in their respective folders (***bin\x86*** and ***bin\x64***)



- **Windows Native** executables will be in ***bin\Any platform***



Choosing the *Windows Native* option will generate three .exe files:

- o **<app>x86.exe** and **<app>x64.exe** are your actual packaged script.
- o **<app>.exe** is a starter application that will execute the right package for the current platform.

You must install or deploy all three files together for your application to work. The starter application will receive the same icon, digital signature, and manifest as the packaged files, so a shortcut to **<app>.exe** will create the same experience.

If you select both *Windows Native* and *Generate .config file*, then .config files will be generated for all three of the .exe files.

External Scripts

Select ***Resolve and include external scripts*** to deploy dot sourced files with the executable. If this option is enabled, the code of the external scripts will get injected into the packaged script when building the executable.

- Files specified with or without single and double quotes are supported. Files that do not exist will issue a warning. If you have a dot source statement inside a comment block, the file will be inserted into the comment block.
- Using a line comment will prevent a file from being resolved.
- If you need to resolve only some but not all external files, you can use a different case for the file extension:
  - o **./include/lib.ps1** will be resolved by the packager.

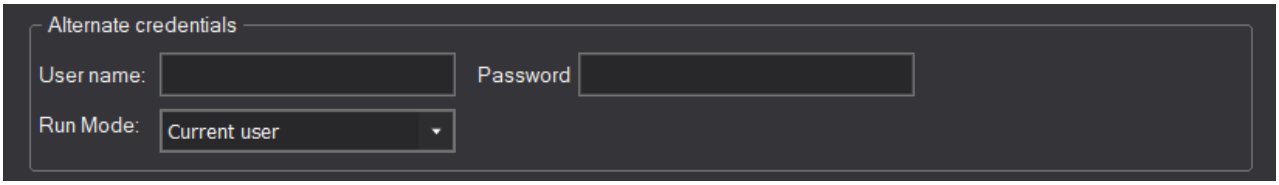
o **./include/lib.PS1** will *not* be resolved.

In other words, the statement is case sensitive; the actual filename's case is not relevant.

```
#>

# ."/include/oldhello.ps1"

#. "./oldinclude/someotherfile.ps1"

."./includes/hello.ps1"

Write-Host "Yes, it works"

.$PSScriptRoot\includes\functions.ps1

get-platform

$args

```

Alternate Credentials

By default, the scripts in a package run in the security context of the user who runs the package. You can specify alternate credentials (a username and password) that will be used to run the scripts.



The alternate credentials you supply must be available (either as local or domain accounts) on any computer where the packaged executable will run. Also, the credentials must generally have local administrator privileges on the computer where the package will run.

Alternate Credentials options:

- **Username**
  Username of the specified user that will run the scripts in the package.
  To specify a domain, use username@domainname format, not domain\user format. Do not specify a domain or computer name for local accounts.
- **Password**
  Password of the specified user that will run the scripts in the package.

- **Run Mode**
  Select the user profile that will run the scripts in the package.
  - o ***Current user***
    Runs scripts with the security context of the current user, in the current user's environment.
  - o ***Impersonate user***
    Runs scripts with the security context of the specified user, in the current user's environment.
  - o ***RunAs user***
    Runs scripts with the security context of the specified user, in the specified user's environment

Elevate Regular User to Full Administrator

This section explains how to package a script as an executable, with the objective of allowing a regular user to accomplish a task that requires full administrator privileges.

Some background:

Since Windows Vista, the Administrator security token is split—you cannot simply logon as Admin and do anything you need to do. An Admin must *elevate* in order to accomplish certain tasks (e.g., when accessing or modifying certain system areas). This has ramifications for packaging executables—you cannot successfully use a run mode of *RunAs* or *Impersonation*, and **also** *elevate* at the same time.

When selecting *RunAs* or *Impersonation*:

- The specified credentials are stored inside the packaged executable, encrypted.
- When the packaged executable is launched, it uses certain API calls to create a new security token (*Impersonation*) or run itself with the specified credentials (*RunAs*). The executable needs to load and execute in order for this to happen.

When selecting a *manifest for elevation*:

- The manifest is embedded in the executable—unencrypted—because Windows needs to read this information.
- Windows will load and evaluate this manifest **before** any code is executed. If you run this from a regular user, you will be prompted for Admin credentials and also to verify elevation. The credentials stored inside the package have no effect at this point because they would only be applied after the fact.

Essentially, due to the way Windows evaluates manifests, elevation happens **before** *RunAs* / *Impersonation*—but it needs to be the other way around to avoid prompts and to not give regular users Admin privileges. The Script Packager accomplishes this via a two-step process:

1. Starter.exe—a simple script packaged as an executable that includes; the Admin credentials, a run mode of either *RunAs* or *Impersonation*, and instructions to launch your script.
2. Your script—packaged as an executable, with a manifest for elevation.

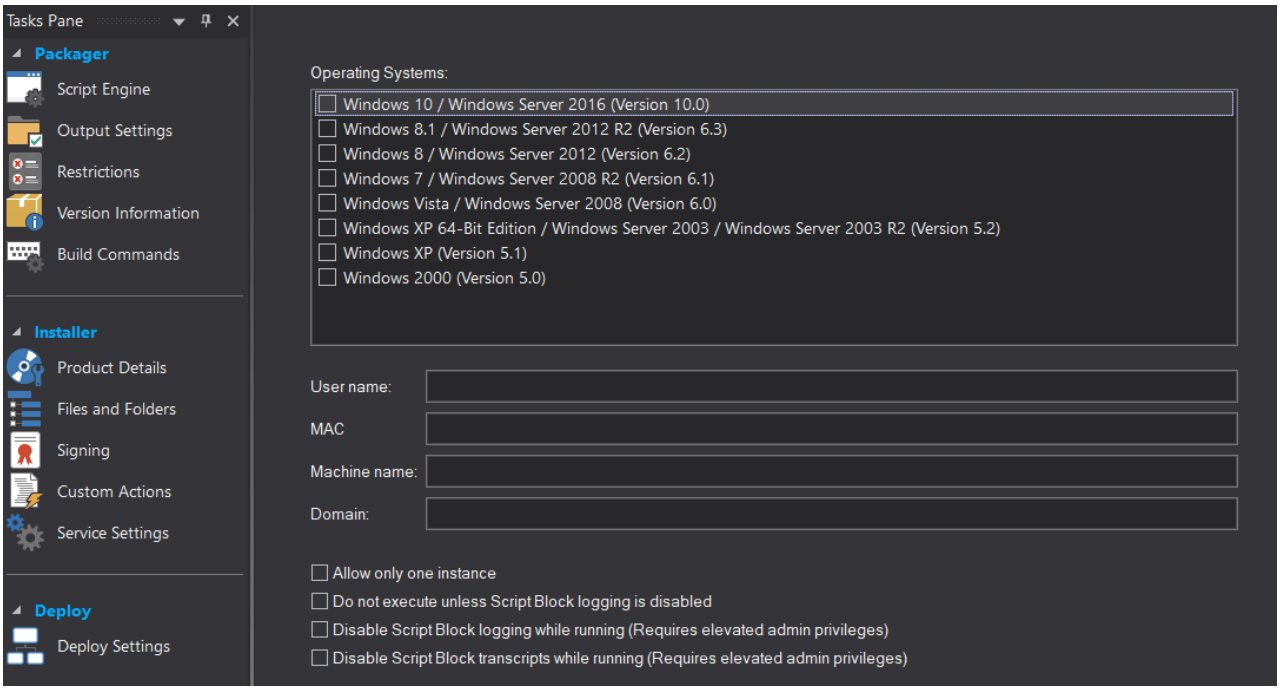
Using this process, Starter.exe will launch and use the specified Admin credentials, and then your

script will run with elevation.

Depending on your local settings, you may get a prompt to allow your script to modify your system, but it will not prompt you for actual credentials.

Restrictions

Use the Restrictions to limit the environment in which the package runs.

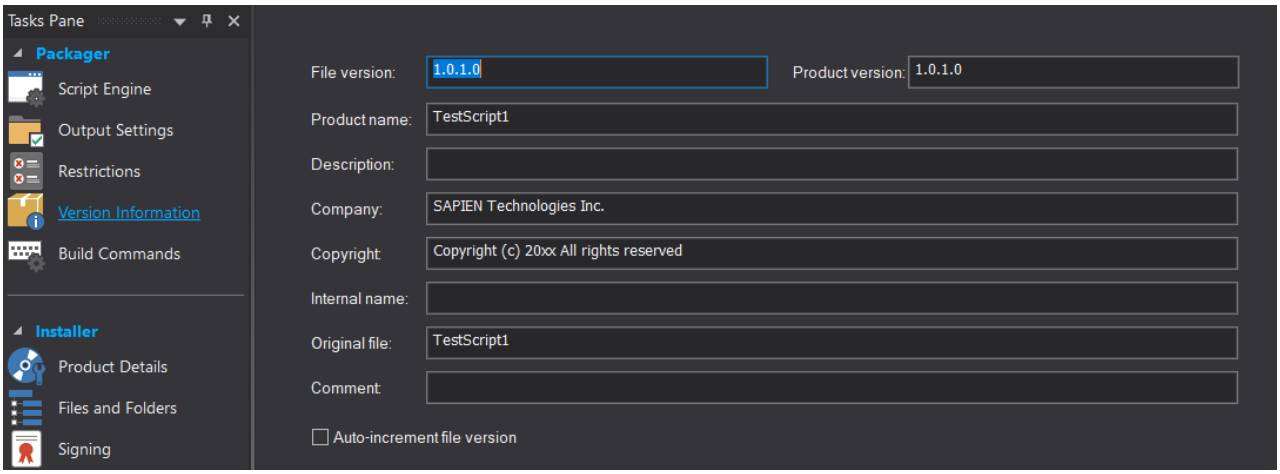


When restricted to a specific version, the executables display the expected and encountered versions in the error message.

Version Information

Use the Version Information settings to specify characteristics of the current version of the executable file.

The version number must be in #.#.#.# format.



Build Commands

Use the Build Commands to define custom commands to run before or after packaging.

The commands will be executed in the sequence defined; one after the other, rather than in parallel.

Use the four buttons at the top-right of each section to manage the pre- and post-packaging commands:

From left to right:

-  **Add File** - Browses for a file / exe.
-  **Remove** - Removes the command.
-  **Move Up** - Moves the command up in the order.
-  **Move Down** - Moves the command down in the order.

15 Remote Script Execution Engine

The **Remote Script Execution Engine**™ (**RSEE**™) is an enterprise-level remote script execution environment.

RSEE Overview

RSEE consists of two components: The client, which is built into PrimalScript and PowerShell Studio, and a remote service that must be deployed to each computer where you will remotely run scripts. RSEE is capable of deploying a script from within PrimalScript and PowerShell Studio, out to remote computers where the script is executed, and bringing the scripts' output and results back to PrimalScript or PowerShell Studio for your review.

RSEE is a complex tool and it interacts closely with Windows' security subsystems. RSEE is recommended for use only by experienced Windows administrators who fully understand service deployment and management, cross-computer security and authentication and, in the case of domain environments, Group Policy objects and Active Directory administration. Apart from the guidelines in this manual, SAPIEN Technologies cannot assist you with security issues caused by improper configuration nor can we assist with Active Directory, Group Policy, or local computer configuration tasks.

RSEE is designed only for Windows Script Host (WSH) scripts in VBS (VBScript) or JS (JScript) files. It is not designed for other WSH scripts (including WSFs) nor is it designed for scripts written in other languages (such as batch, KiXtart, and so forth).

RSEE Deployment

RSEE's service component is packaged in a **Microsoft Windows Installer** (**MSI**) file and is suitable for deployment via Group Policy. You can also manually install it on individual machines. Keep in mind that, once installed, the service needs to be started in order to be useful. This will occur automatically after restarting the computer on which the service is installed (the service is set to start automatically by default).

After deploying the service, there are a number of configuration steps that you must take in order to properly configure RSEE in your environment.

Identity

RSEE installs, by default, to log in under the privileged LocalSystem account. This may be sufficient for your purposes. However, when deploying scripts in PrimalScript and PowerShell Studio, be sure not to specify any credentials in the Launch dialog box. Also be advised that the LocalSystem account may not be able to execute some scripts, depending on their security requirements.

We recommend that you configure the RSEE service to run under a user account that has administrative privileges on the local computer. In a workgroup environment this would be a local account, and we recommend creating the same local account (with the same password) on all of your com-

puters, for consistency. In a domain environment, we recommend creating a single domain account which has local administrative rights on all computers in the domain, and using this account to run the RSEE service. Whenever the RSEE service is running under a user account, you must specify that account (and its password) when deploying scripts in PrimalScript.

When using RSEE, you have the option to specify the credentials under which the script should execute. Generally speaking, you need to provide the same credentials that the RSEE service is using to log on.

TCP Port

The RSEE service defaults to using TCP port 9987 for incoming connections, and TCP port 9988 for outgoing connections. It is your responsibility to ensure that any local firewalls will permit incoming traffic on this port. Keep in mind that the Windows Firewall (Windows XP SP 2 and later, and Windows Server 2003 SP 1 and later) can be centrally configured via a domain Group Policy object.

To specify a different port

- You can specify a different port via the registry key **HKEY_LOCAL_MACHINE\Software\Policies\SAPIEN**. The Value name is InPort (for the incoming port) and OutPort (for the outgoing port). Note that these values are most easily configured by means of a Group Policy object (GPO), and we provide a template (ADM file) that can be imported into a GPO to configure RSEE.

The RSEE service *and* both PrimalScript and PowerShell Studio (as the RSEE client) utilize InPort and OutPort. The service listens to InPort for incoming connections and uses OutPort to send script output back to the client. The client reverses this: scripts are sent via InPort and results are received on OutPort. The registry key above configures these ports for both clients and the service.

Domain Tips

While manually configuring a few computers in a workgroup is not a hardship, manually configuring an entire domain of computers can be burdensome. An Active Directory domain environment provides a number of capabilities for centralizing and automating this configuration, however. While this section is not intended as a comprehensive tutorial in Active Directory (we recommend that you consult an experienced Active Directory administrator or the appropriate documentation if you need more assistance), the following tips should help you configure RSEE more easily:

- **Create a domain account**
  Name this account something like "RSEEUser" and provide it with a strong password per your organization's password policies.
- **Deploy the RSEE service**
  This can be done by means of a Group Policy object (GPO) linked to the appropriate levels in the domain. The RSEE service defaults to running under the LocalSystem account and it defaults to port 9987. The service's MSI is located in the RSEE folder under your PrimalScript Enterprise install-

ation folder.

- **Make the RSEE service account a local Administrator**
  You can do this in a Group Policy object (GPO). Browse to Computer Configuration > Security Settings > Restricted Groups. Add a group ("Administrators") and then add your RSEE domain account (and any other appropriate accounts) to the group.
- **Configure the RSEE service**
  You need to configure the RSEE service to log on with the user account (and password) you created. This can either be done manually or using a script. The book Windows Administrator's Automation Toolkit, for example, contains a script that can set the logon account and password used by services running on multiple computers. Utilities like Service Explorer (www.scriptlogic.com) can perform the same task.
- **Select the TCP port**
  We provide a Group Policy object (GPO) administrative template (ADM file) that you can import into a GPO and use to centrally configure the TCP port used by the RSEE service. This ADM file is located in the RSEE folder under your PrimalScript Enterprise installation folder.

Using RSEE

RSEE now supports Powershell. To deploy the current script (only VBS and JS files are currently supported) to one or more remote computers that have the RSEE service installed, click the RSEE button on the Script toolbar, or select Run Script on Remote Computer from the Script menu.

RSEE performs a quick scan of your script to look for commands that might create a graphical user element such as the VBScript MsgBox() function. If it finds any of these functions, it displays a warning message. Keep in mind that scripts will not normally be able to interact with the desktop environment on remote computers, meaning there would be no way for someone to respond to graphical elements such as MsgBox() or InputBox(). As a result, these elements can cause the script to "hang" and stop responding. RSEE does not perform an exhaustive check for graphical elements; it is your responsibility to ensure they're not used in your scripts. RSEE will allow you to continue with graphical elements because you may have configured the RSEE service to interact with the desktop of the remote computer. It's your decision.

RSEE Launch dialog

The Launch dialog lists the computers where your script will be deployed. Note that the Launch dialog always preloads a default list of computer names at startup. Here's what you can do:

- Click Launch to run the script on the computers which have a checkmark next to their name.
- Set or clear the checkbox next to one or more computer names. You can leave names in the list but clearing their checkbox will prevent RSEE from attempting to run the script on them.
- Click Close to close the Launch dialog. If you've changed the list of computer names, you'll be prompted to save your changes.

- Use Load List and Save List to load an alternate list of computer names (from a text file) or save the current list to a text file. By default, PrimalScript will look for a text file called Default.clt in the \SAPIEN\RSEE Lists folder under your Documents folder. You will need to create the file yourself if you want a pre-loaded list when you launch RSEE.
- Use Select All and Unselect All to set or clear the checkbox next to all computer names currently in the list.
- Select a computer name and click Remove to remove it from the list.
- Type a computer name (must be resolvable to an IP address by your computer) or IP address and click + to add that computer to the list.
- Specify a username (user ID) and password. These will be used to run the script on the remote computer, and should generally match the username that the remote RSEE service is using to log in. Note: if the username you specify is a local account on the remote computer(s), then just type the username. If the username is a domain account, specify the name in the format user@domain. The older domain\user format is not supported.

When you click Launch, RSEE will execute the script on the remote computer(s). Any output produced by the script will be displayed in the Output pane within PrimalScript or PowerShell Studio. Note that the message "Socket connection failed" indicates that RSEE was unable to connect to the RSEE service on a specified computer (either because the computer is not connected to the network, has a firewall blocking the RSEE service ports, or the RSEE service is not installed).

RSEE deploys scripts asynchronously. That is, RSEE sends the scripts out to the remote computers you've selected and then displays whatever results come back. If your scripts produce no output then you won't see any results in PrimalScript or PowerShell Studio.

It's possible for the RSEE service on a remote computer to run into a problem (particularly security-related ones) that it can't report back; in these instances, it will seem to you (looking at PrimalScript or PowerShell Studio) as if nothing has happened. Whenever possible, your scripts should incorporate error-checking and -trapping, and should produce appropriate output so that you get some results back if the script executes correctly.

Note that RSEE cannot be used to deploy a script for later execution. If you need to schedule a script to execute on a remote computer at a particular time, use Windows' built-in Task Scheduler instead of RSEE. You can even write a script utilizing the SCHTASKS.EXE command line tool that creates remote scheduled tasks on multiple computers.

Also note that, if an **Output** pane is already open in PrimalScript or PowerShell Studio, RSEE will utilize it rather than creating a new one. You will need to manually select the tab to view any RSEE results or error messages.

RSEE Restrictions

In order to bring the output of remote scripts back to your computer, the remote RSEE service cap-

tures the standard command-line output of your scripts. That means any script output must be created using the WScript.Echo method. ***Do not*** use graphical user interface functions such as MsgBox() or InputBox(). Because the RSEE service doesn't interact with the desktop, nobody will ever see these functions' dialog boxes and the script will hang.

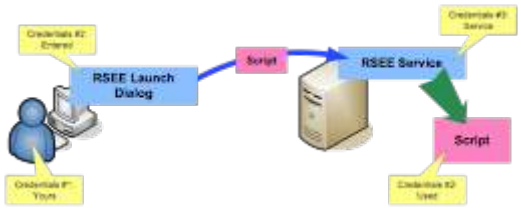
It is possible, if the RSEE service is running under the LocalSystem account, to configure Windows to allow the service to interact with the desktop. You may wish to experiment with this configuration, but it is not a recommended configuration because of the usual security restrictions on the LocalSystem account.

Also avoid any object methods-such as WScript.Popup-that create graphical elements.

Any objects referenced by a script must be installed, registered, and available on the remote machine where RSEE executes the script.

At this time, RSEE can only be used to execute Windows Script Host scripts. RSEE explicitly launches scripts under CScript.exe which must be available on the remote computers.

Most other restrictions in RSEE are actually Windows security restrictions. When the RSEE service launches, it does so using the credentials you configure in Windows' service manager. When the RSEE service receives a script, it creates a brand-new process using whatever credentials you enter into the RSEE Launch dialog. The following figure illustrates this process and the three sets of credentials involved:



RSEE Credentials and Execution Process

Always bear in mind that your scripts execute under the security credentials you provide (Credentials #2 in the diagram). This process does require your attention, as several things can go wrong if you're not careful:

- If you specify credentials in the Launch dialog (#2 in the diagram) that the RSEE service account (#3 in the diagram) doesn't have permission to use in a new process launch, then script execution will fail.

Practically speaking, the credentials you provide in the Launch dialog (#2 in the diagram) need to be the same as the credentials the RSEE service uses to log in (#3 in the diagram).

- If the RSEE service account (#3 in the diagram) doesn't have appropriate rights (including "Log on as a service"), then the RSEE service will not be able to start.
- If your script tries to do something that the Launch credentials (#2 in the diagram) don't have permission to do—such as log into a database or access a file share—then you'll receive an error. Depending on the exact situation, this may or may not be communicated back to you in Prim-

alScript or PowerShell Studio.

- If your script tries to perform an illegal operation—such as specifying alternate credentials in a WMI connection (which is illegal because the script is executing locally on the remote machine, and local connections to WMI aren't allowed to use alternate credentials)—you'll receive an error. Again, depending on the exact circumstances, this error may or may not be fed back to you in PrimalScript or PowerShell Studio.

These and other similar situations are not problems with RSEE; they are inherent conditions of the Windows operating system and its security subsystems. Whenever you encounter an error with RSEE, bear these conditions in mind and think about the possible security ramifications of what your script is trying to do.

RSEE Notes

RSEE encrypts scripts during transmission to help keep them secure.

RSEE does not implement any sort of IP filtering capability (which might, for example, allow you to ensure that only your computer can utilize RSEE on remote servers). Instead, we recommend using Windows' own built-in IP filtering (available as part of Windows' IPSec features). Using this filtering, you can ensure that only specified IP addresses are allowed to communicate on the TCP ports used by the RSEE service, thus restricting who can contact that service and utilize RSEE.

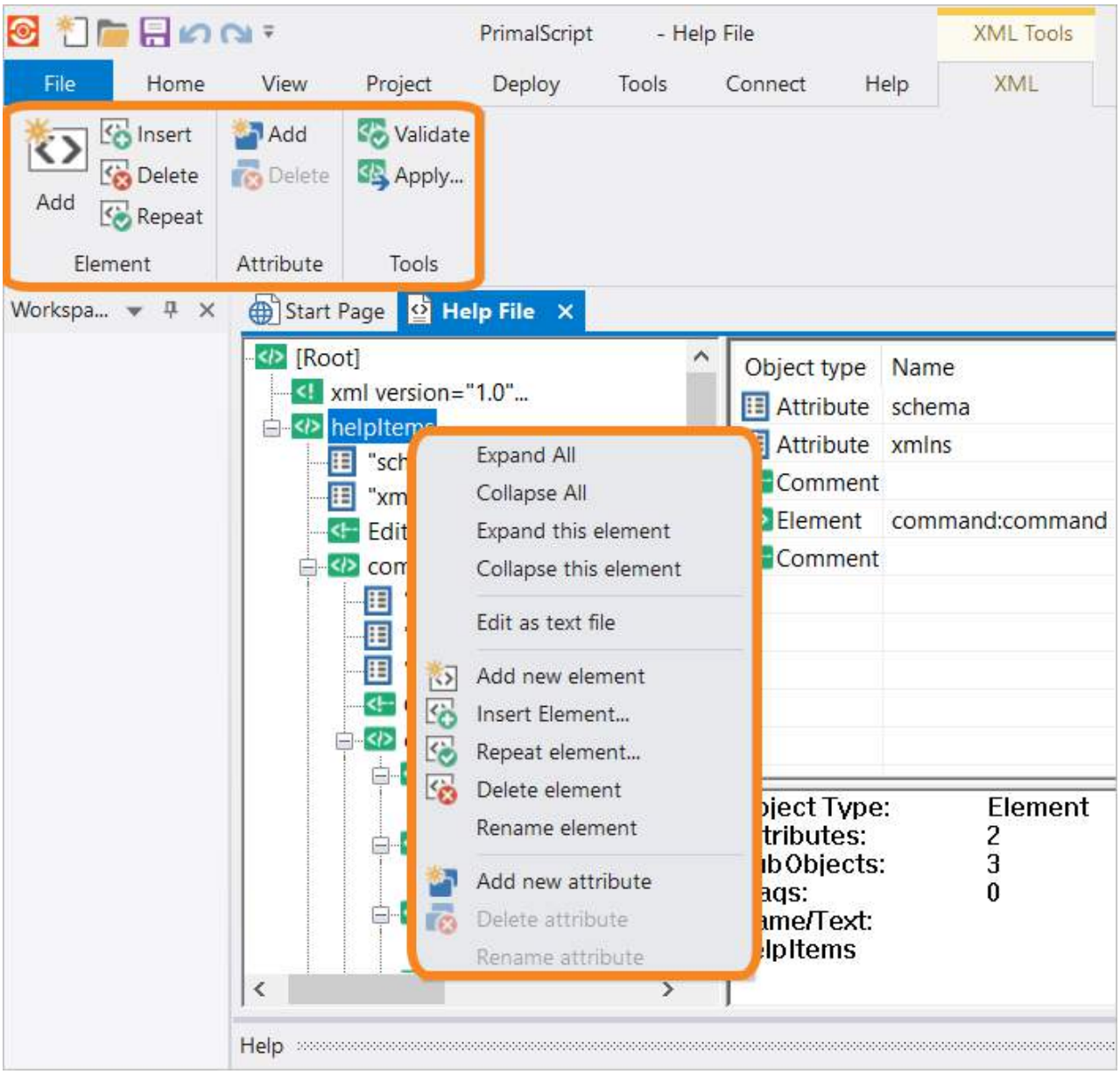
16 Visual XML Editor

The Visual XML Editor provides a graphical user interface for working with XML-formatted documents. You can also work on XML documents in the standard text editor, and switch back and forth between the two modes.

16.1 Using the Visual XML Editor

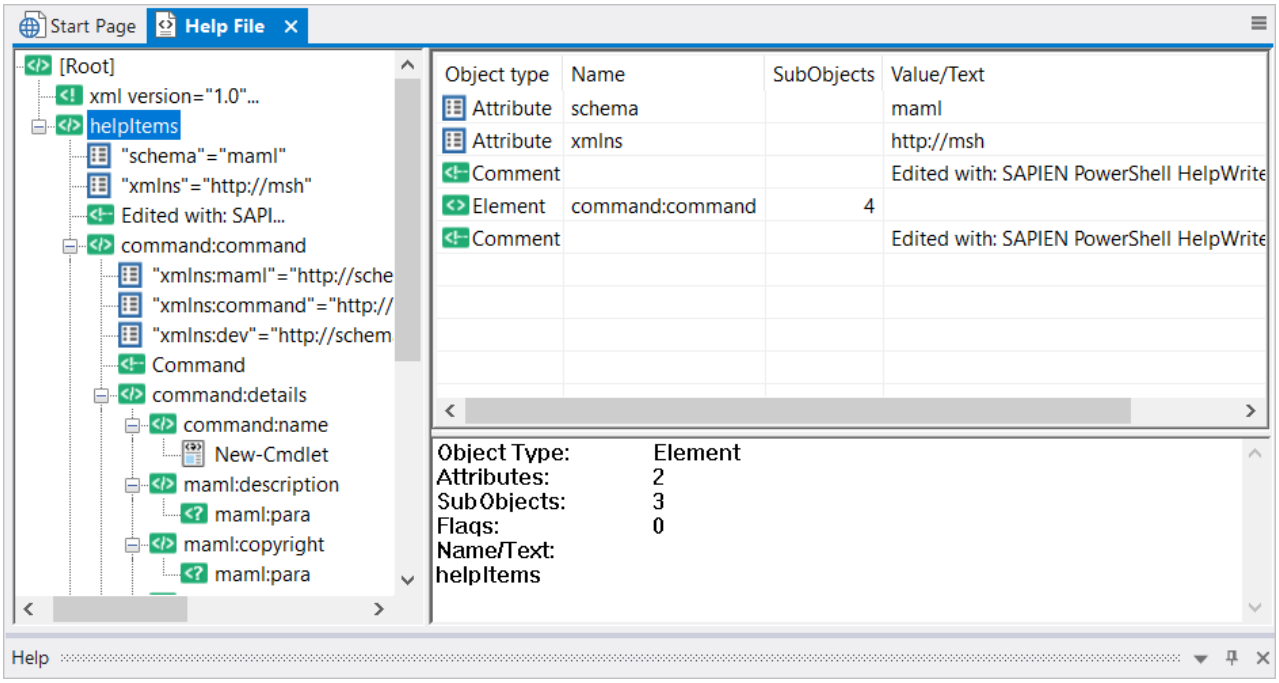
When you create a new XML document or open an existing XML document, PrimalScript opens the Visual XML Editor.

Most of the actions discussed here can be activated from the XML toolbar, the XML menu, or by right-clicking the editor:



XML documents consist of elements which can have *attributes*. The primary function of the Visual XML Editor is to allow you to manipulate these elements and attributes visually. You can add elements and attributes, remove elements and attributes, insert elements, repeat an existing element (essentially a shortcut for copying and pasting it, allowing you to repeat it however many times you want), and so forth.

When you select an element, its attributes and comments are displayed in the right-hand pane where you can edit their values:

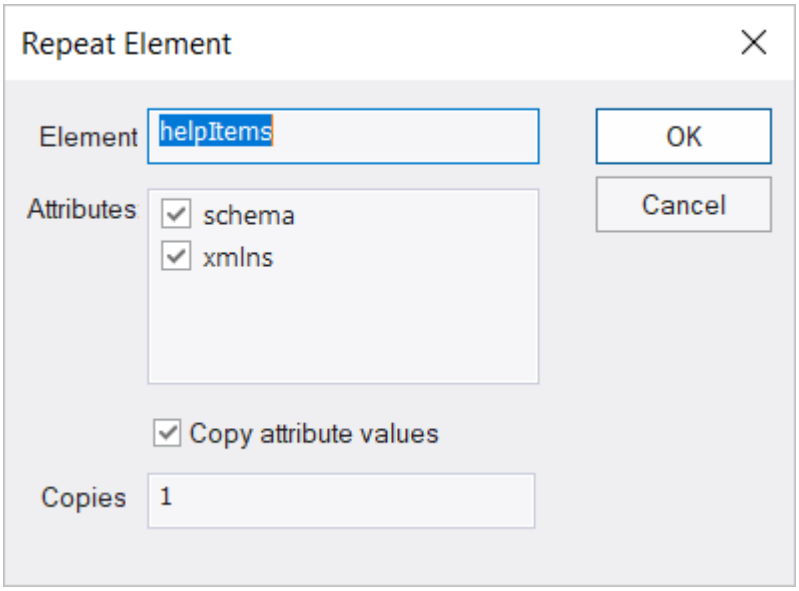


16.2 Repeating Elements

One of the Visual XML Editor's most useful features is the ability to repeat existing elements.

To repeat an element

1. Right-click the element, and then click **Repeat element**.
2. Select the attributes to repeat and the number of repetitions.



You can also change the name of the repeated element (does not change the original).

16.3 Switching to Text

You can use the PrimalScript XML editor or view XML in a text editor, and also switch between them.

To switch to text mode

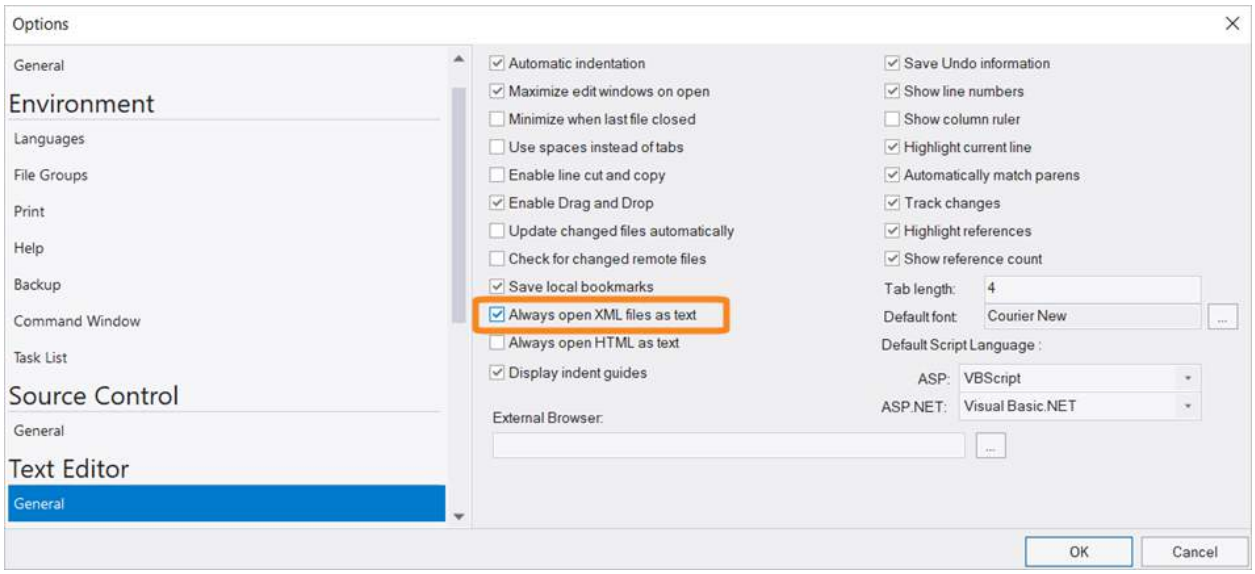
- Right-click the XML document root, and then click **Edit as text file**.

To switch to the XML Editor

- Right-click in code editor, and then click **Open as XML file**.

To open all XML documents in a text editor

- Click **File** > **Options** > **Text Editor** > **General** > then check **Always open XML files as text**:



This setting changes the default, but it doesn't prevent you from switching between XML and text modes.

17 ScriptMerge

ScriptMerge is a stand-alone application shipped with PrimalScript that compares files and folders and applies differences to either of the two compared items.

17.1 Running ScriptMerge

ScriptMerge can be started from the Windows Start Menu, or from within the PrimalScript and PowerShell Studio applications.

To start ScriptMerge from the Windows Start Menu

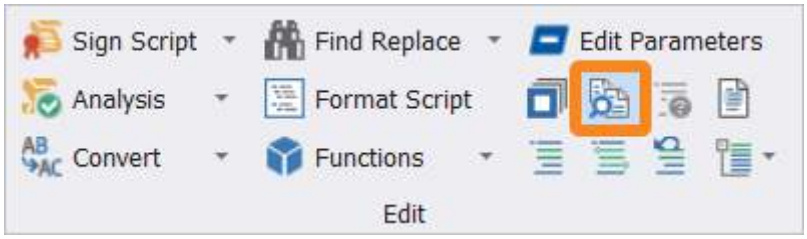
- In the Windows Start Menu, select **SAPIEN Technologies, Inc.** > **ScriptMerge**:

To start ScriptMerge from PrimalScript

1. In **PrimalScript** click the **View** tab > then in the Panels section, check the **Tools** box.
2. In the **Tools Browser** click **SAPIEN Tools** > then click the **ScriptMerge** icon.

To start ScriptMerge from PowerShell Studio

- In **PowerShell Studio**, open two files to compare > then click **Home** > in the Edit section, click the **Compare Files** button:



17.2 Comparing Files

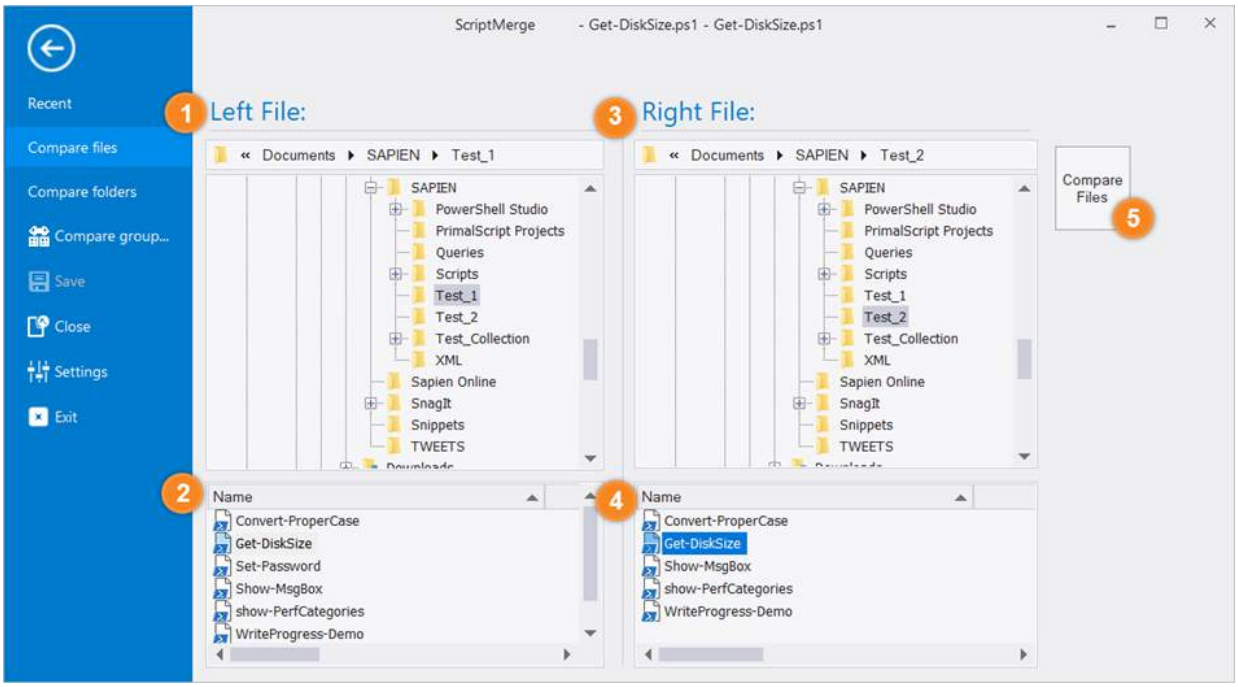
ScriptMerge compares files side-by-side and highlights the differences.

To compare files

Click **File** > **Compare files**:

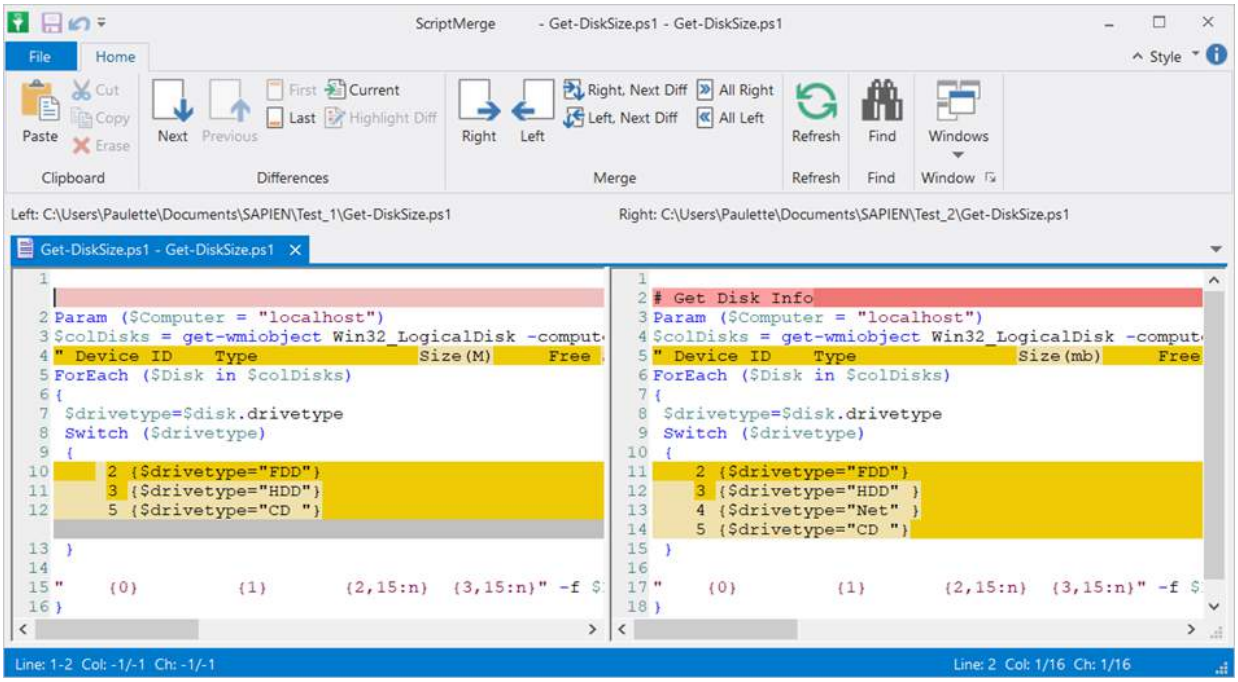
1. In the Left File window, navigate to the folder.
2. Select a file in the Name window below.
3. In the Right File window, navigate to the folder.
4. Select the other file to compare in the Name window below.

5. Click **Compare Files**.



When the files are first opened, ScriptMerge displays the differences as gray, light yellow, and dark yellow colored lines. The current difference—in this case the first difference—is highlighted in varying shades of red:

- Light yellow indicates words that have changed.
- Dark yellow indicates a line that contains a change.
- Dark grey lines indicate a line that was deleted.

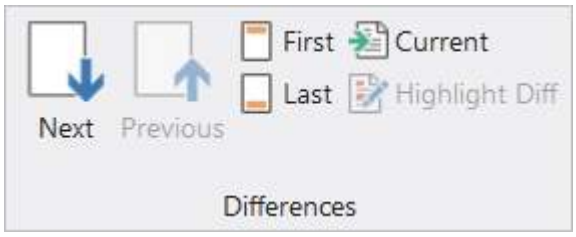


To see the highlighting in action, change a line and save it. The changes will be reflected in the comparison.

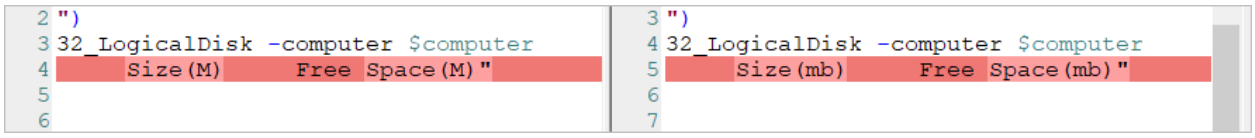
You can customize the comparison differences coloring in File > Settings > Merge Options > Color Options.

To step through the differences

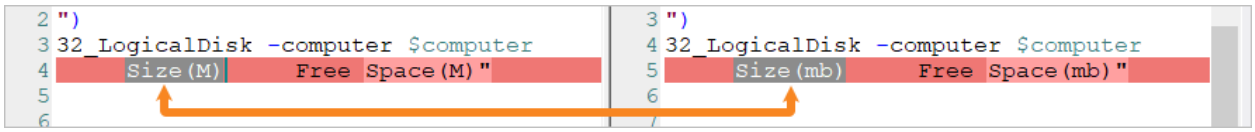
- In the Differences section, click **Next** and **Previous** to go back and forth through the differences:

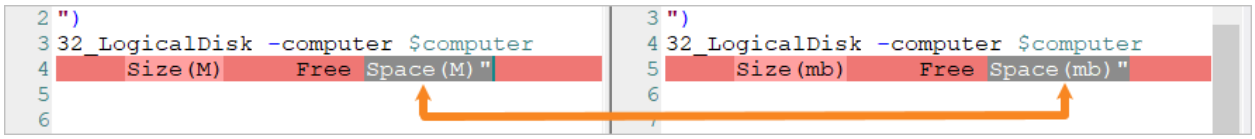


The current difference is highlighted in different shades of red:



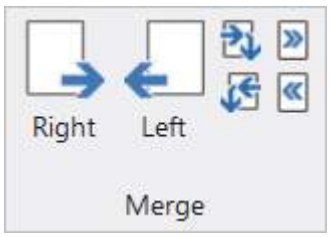
Click **Highlight Diff** to add extra emphasis to the changed elements for the current difference:





To merge the differences

- In the Merge section, select **Right** or **Left**:



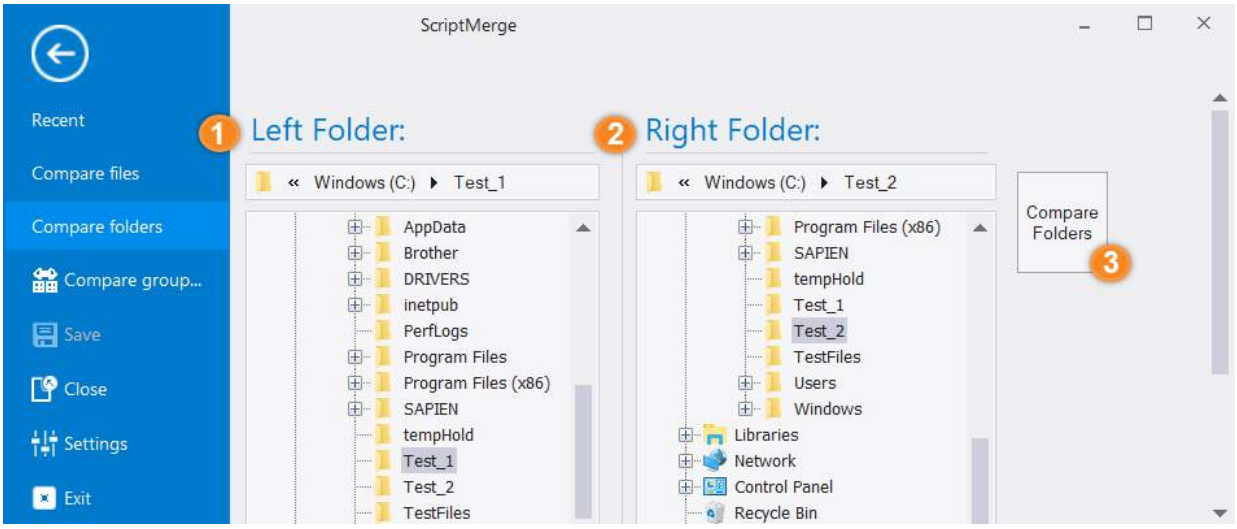
17.3 Comparing Folders

ScriptMerge compares folders side-by-side and highlights the differences.

To compare folders

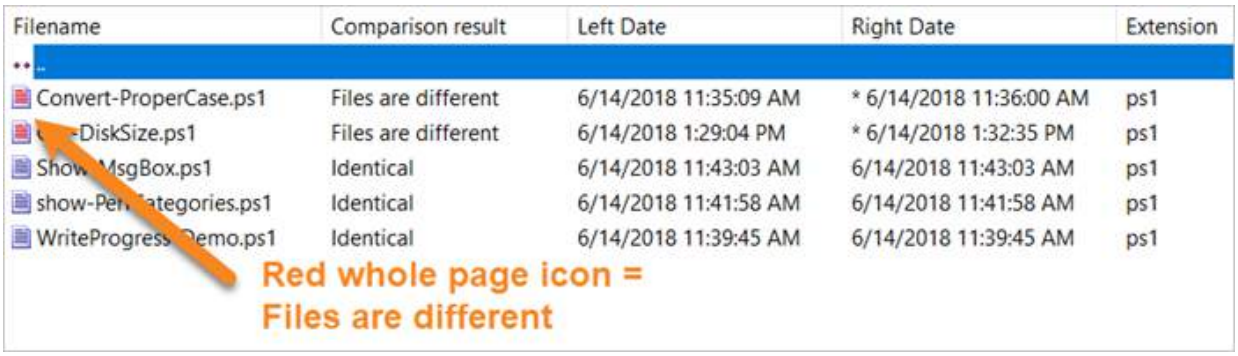
Click **File** > **Compare folders**:

1. In the Left Folder window, select a folder.
2. In the Right Folder window, select a folder.
3. Click **Compare Folders**.

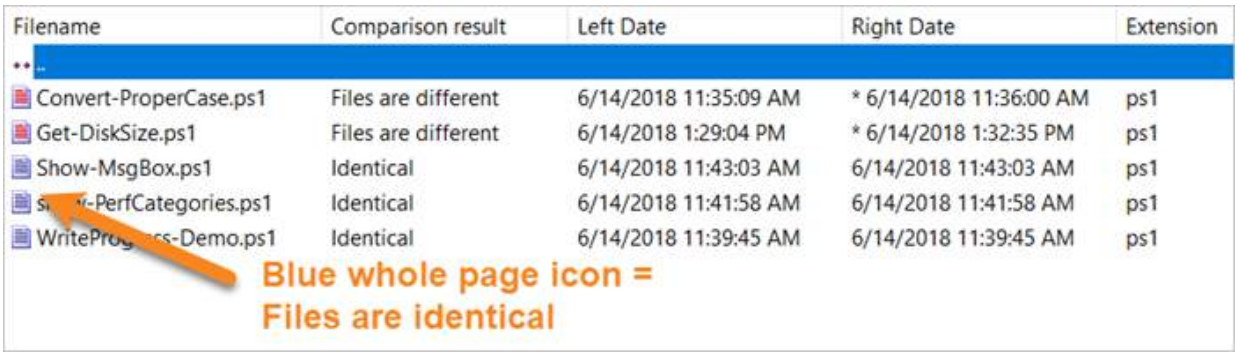


ScriptMerge compares the files in each folder and their contents. The results show which folders have files in both locations, or if the folders have files in only one location. If the folders have files in both locations, ScriptMerge indicates if the files are different or identical.

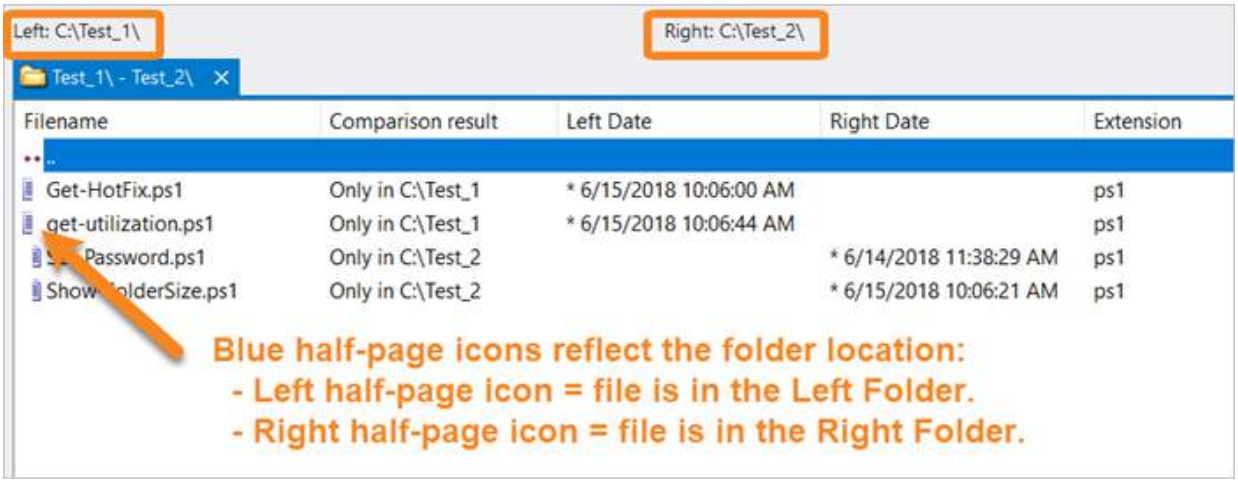
Files that exist in both locations but are different are marked with a red whole page icon:



Files that are identical in both locations are marked with a blue whole page icon:



Files that are only in one folder are marked with a blue half-page icon. The icons reflect the folder location: left half-page icons are in the Left Folder; right half-page icons are in the Right Folder:

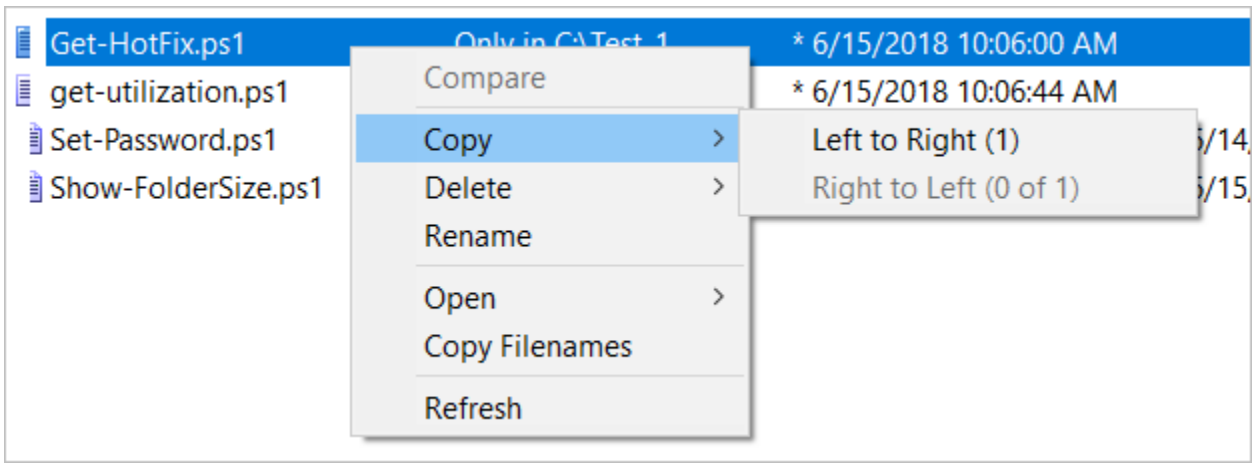


To replace a file in one folder with the file in the other folder

- In the Merge section, select **Right** or **Left**.

-OR-

- Right-click the file and select **Copy** (**Left to Right** or **Right to Left**):



17.4 Comparing Groups

You can group pairs of files and then easily open the group to compare. This feature is useful for repeated comparison of the same files.

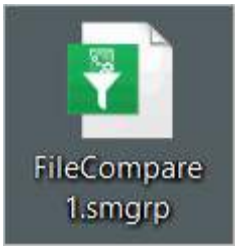
To create and open a group

1. Create a text file with the file pairs listed as follows:

   File1|File2 separated by the pipe symbol (**|**).

   Example: **C:\Users\Me\Documents\SAPIEN\script1.ps1|C:\Users\Me\Documents\SAPIEN\script2.ps1**

2. Save the file as <***filename***>**.smgrp** (smgrp = ScriptMerge Group):

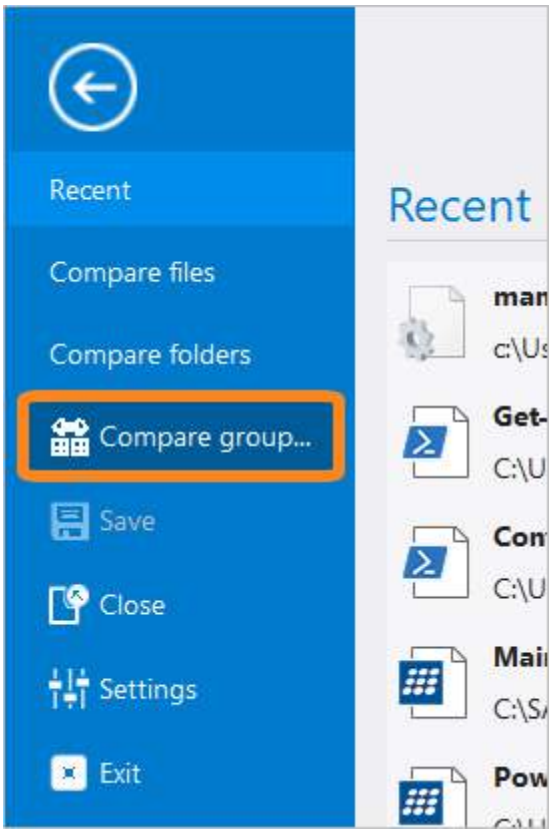
   

3. Open the group file and ScriptMerge will open the contained pairs at the position of the first difference:

   - Double-click the group file.

   -OR-

   - In ScriptMerge select **File** > **Compare group**, then navigate to the group file location:



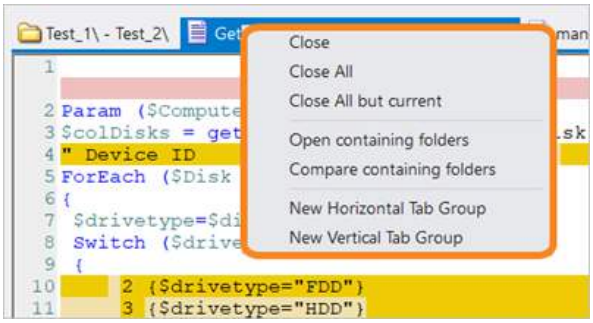
17.5 Context Menu Options

The ScriptMerge context menu options will vary depending on if you are comparing files or folders.

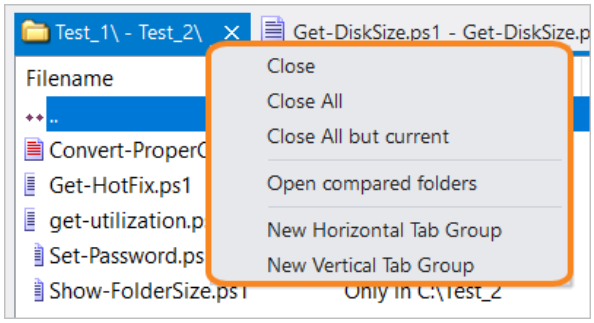
To access the context menu options

- **Right-click** on the file comparison or folder comparison tab:

Compare Files - Context Menu



Compare Folders - Context Menu



- **Close**

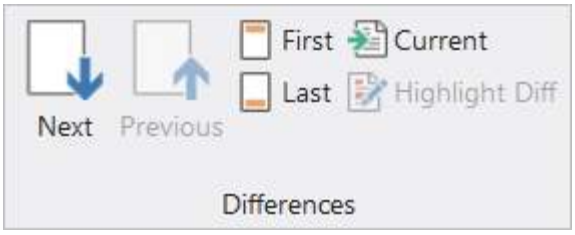
Closes the highlighted tab.

- **Close All**
  Closes all of the tabs.
- **Close All but current**
  Closes all tabs except for the highlighted tab.
- **Open containing folders (file comparison only)**
  Opens two Windows Explorer instances with the compared files selected.
- **Compare containing folders (file comparison only)**
  Compares the files in each underlying folder, and also the file contents.
- **Open compared folders (folder comparison only)**
  Opens two Windows Explorer instances, one for each compared folder.
- **New Horizontal Tab Group**
  Moves the selected tab to a separate horizontal tab group.
- **New Vertical Tab Group**
  Moves the selected tab to a separate vertical tab group.
- **Move to Previous Tab Group**
  Moves the selected tab back to the original tab group.

Tab groups are especially useful when you have a folder comparison open and also a number of files compared. Move the folder comparison to it's own tabbed group so that it remains visible while you compare the files in the folders.

17.6 Navigating Between Differences

The **Differences** section of the ribbon provides buttons to help you move between differences in a file or folder:

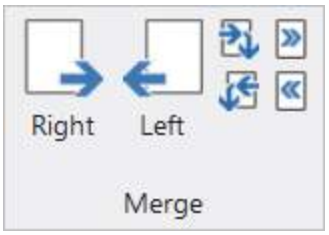


- **Next** (*Ctrl+Down*)
  Moves forward to the next difference.
- **Previous** (*Ctrl+Up*)
  Moves back to the previous difference.
- **First** (*Ctrl+H*)
  Moves to the first difference.

- **Last** (*Ctrl+E*)
  Moves to the last difference.
- **Current** (*Ctrl+Enter*)
  Scrolls the code window to the current difference.
- **Highlight Diff**
  Adds extra emphasis to the changed elements in code files.

17.7 Reconciling Differences

The **Merge** section of the ribbon provides buttons to help you copy from one file or folder to another:



- **Right** (*Ctrl+Right*)
  Copies the current selection from the left to the right file or folder.
- **Left** (*Ctrl+Left*)
  Copies the current selection from the right to the left file or folder.
- **Right**, **Next Diff**
  Copies the current selection from the left to the right and advances to the next difference.
- **Left**, **Next Diff**
  Copies the current selection from the right to the left and advances to the next difference.
- **All Right**
  Copies all differences from the left to the right file or folder.
- **All Left**
  Copies all differences from the right to the left file or folder.

17.8 Signing Scripts

ScriptMerge allows you to to handle digital signatures in your files.

You can re-sign scripts or remove signatures from the ribbon buttons:

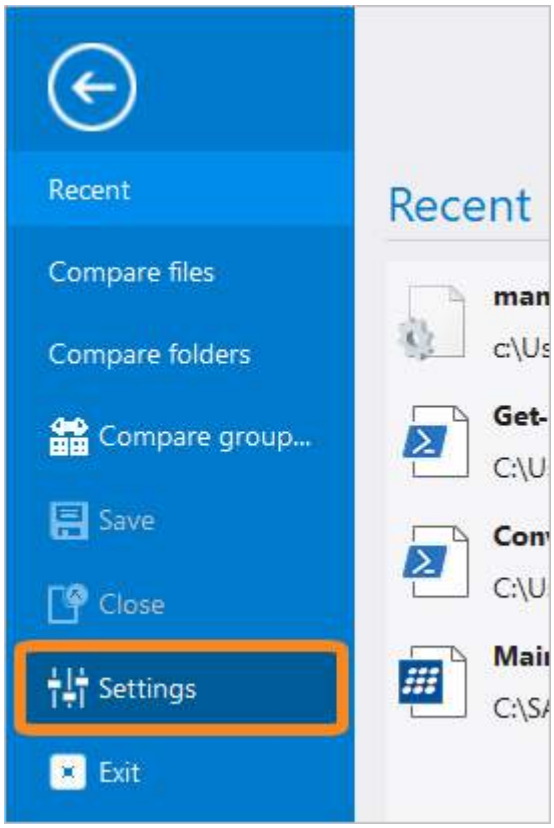
- **Sign Left**
  Sign the script file displayed on the left.
- **Sign Right**
  Sign the script file displayed on the right.
- **Sign Both**
  Sign both script files.
- **Remove Left**
  Remove the signature from the script file displayed on the left.
- **Remove Right**
  Remove the signature from the script file displayed on the right.
- **Remove Both**
  Remove the signatures from both script files.

17.9 ScriptMerge Settings

You can adjust some ScriptMerge tool settings, such as keyboard shortcuts and Quick Access Toolbar buttons. You can also change the highlight colors for comparisons, and toggle some compare options.

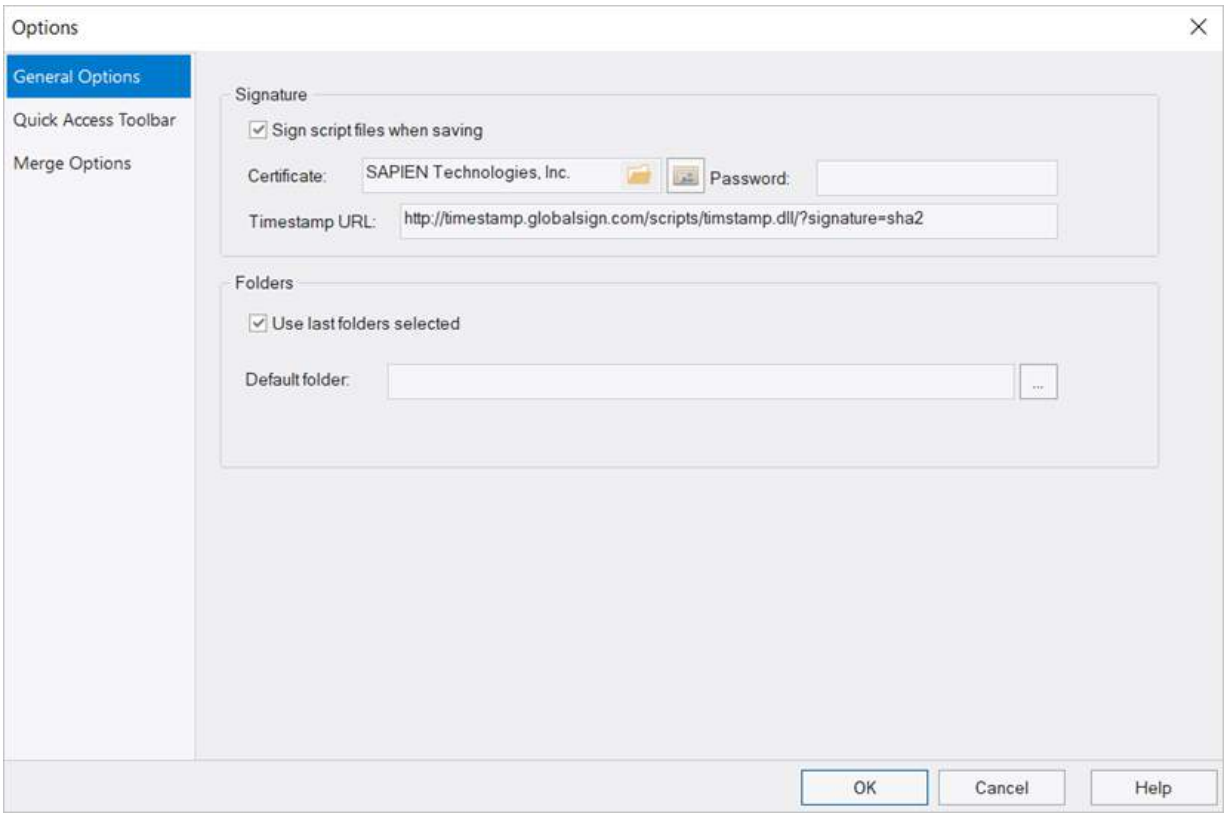
To access the ScriptMerge options

- Select **File** > **Settings**:

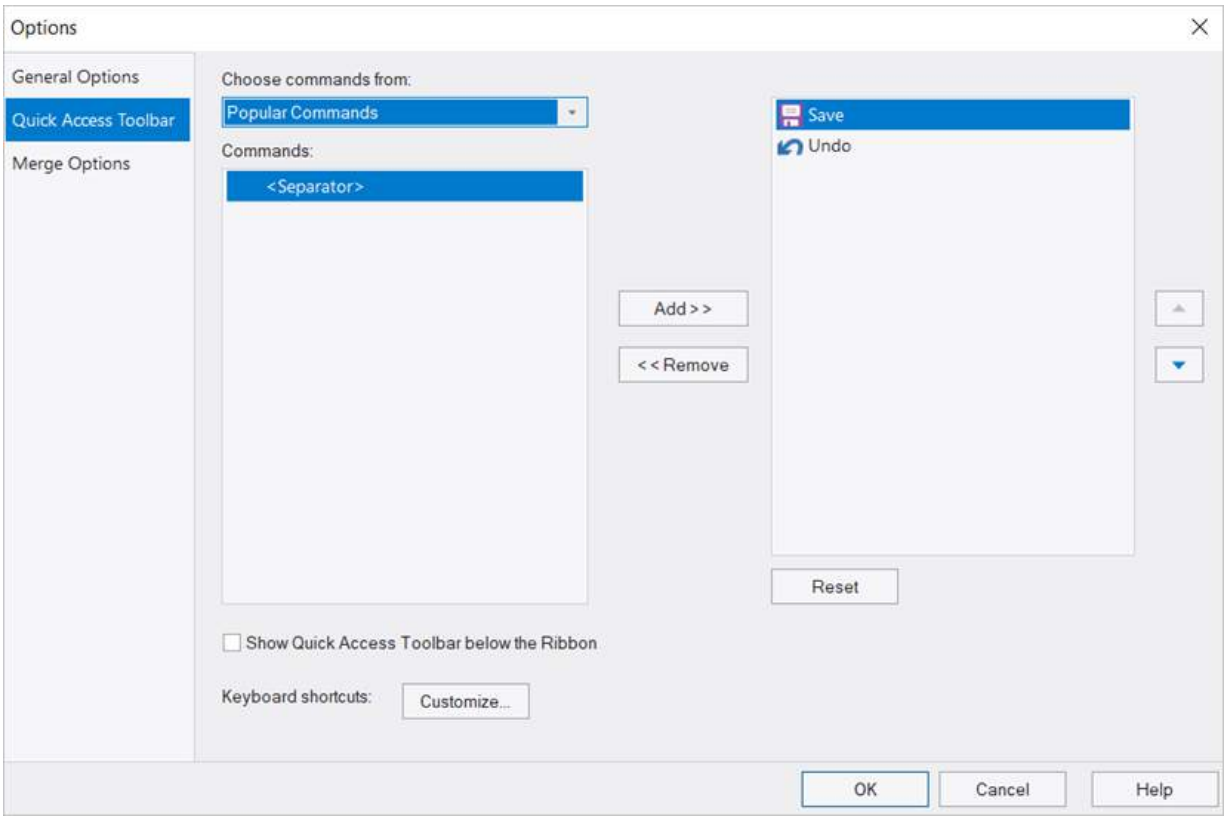


The following settings can be adjusted:

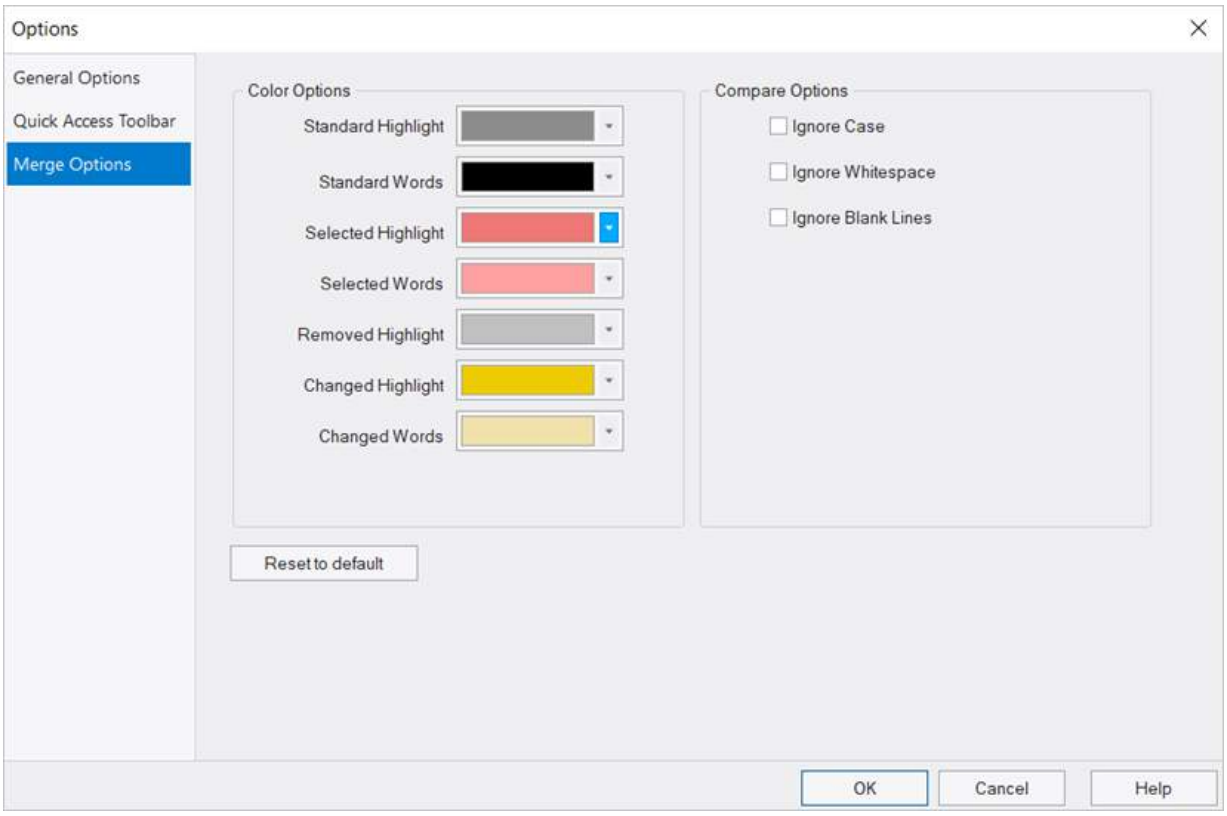
- **General Options**
  - Sign script files when saving.
  - Use last folders selected.



- **Quick Access Toolbar**
  - o Add, Remove, Reset toolbar buttons.
  - o Show Quick Access Toolbar below the Ribbon.
  - o Customize Keyboard Shortcuts.



- **Merge Options**
  - o Select comparison color options.
  - o Toggle compare options for Case, Whitespace, and Blank Lines.



18 Snippet Editor

PrimalScript and PowerShell Studio provide a collection of snippets to help you complete common coding tasks quickly. You can use the Snippet Editor to easily edit and create snippets.

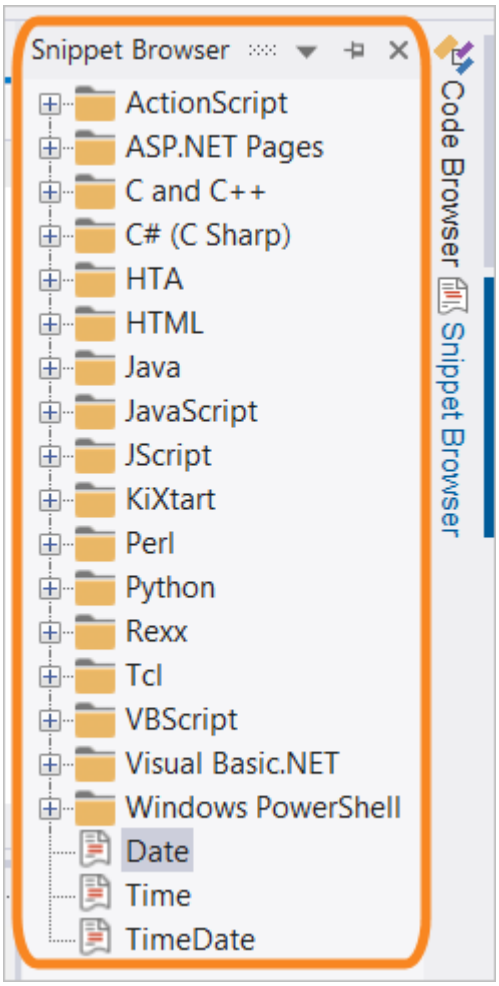
About Snippets

Snippets are small pieces of reusable code that can be quickly inserted into your scripts, thus saving you time and reducing errors. This piece, or "snippet" of code, can vary from a full-fledged function to a simple single line statement. Snippets come in a variety of languages such VBScript, PowerShell, C#, etc.

PrimalScript and PowerShell Studio come with extensive libraries of reusable code snippets. You can also save any text or code block as a snippet to automate code development. Snippets can include placeholders; PrimalScript and PowerShell Studio will prompt you to supply values for these when you use the snippet.

Snippet Browser

Use the Snippet Browser to access and manage snippets:



To view the Snippet Browser

- On the View ribbon, in the Panels section, select Snippets.

-OR-

- On the docking area to the right of the Code Editor, hover over the Snippet Browser tab to un-hide the Snippet Browser.

To create a snippet

- Highlight the code you want, right-click and choose Save as Snippet....

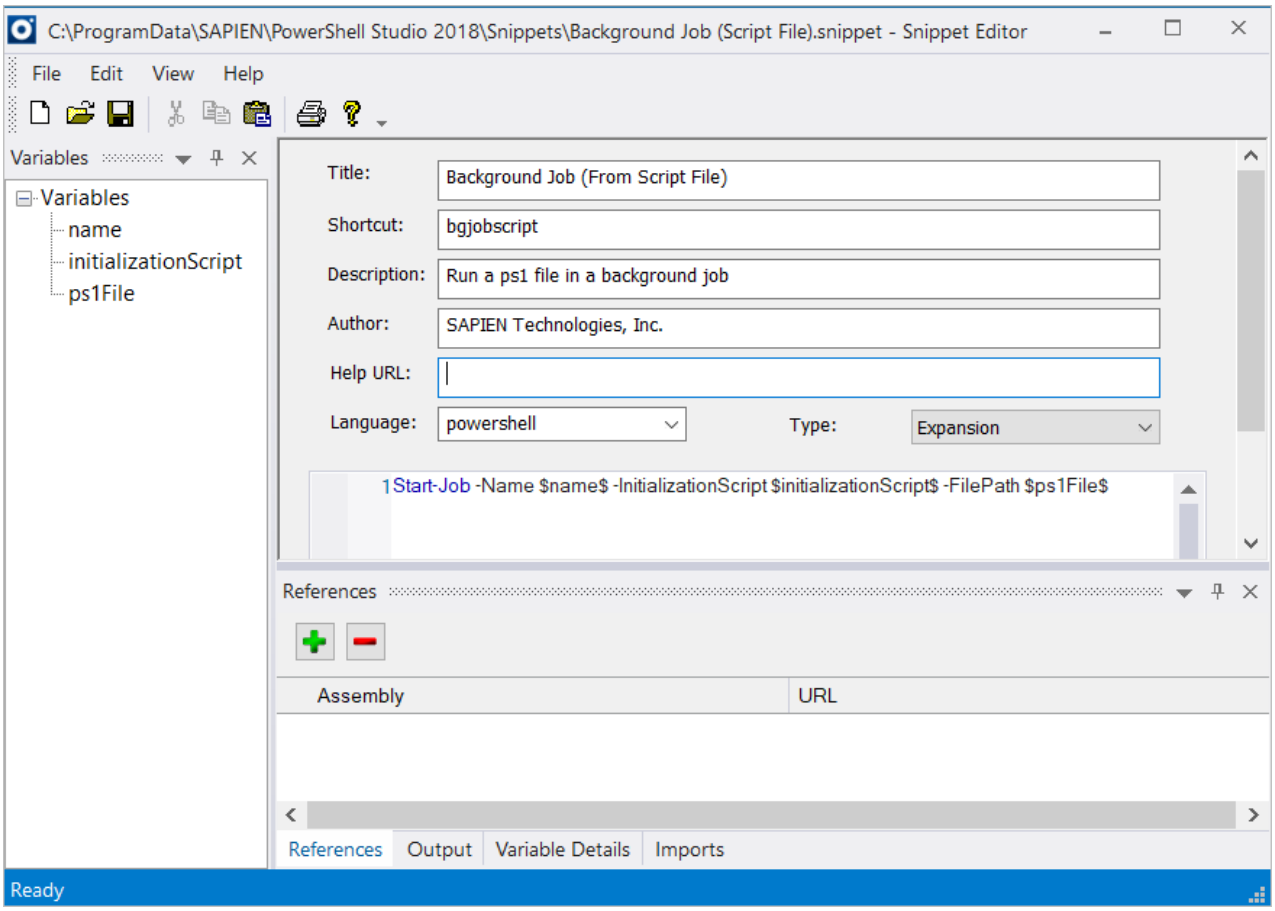
To insert a snippet

- In the Snippet Browser, locate the desired snippet and either double-click or right-click and select Insert Snippet. The snippet will be inserted in the location of the cursor.

Snippet Editor

The Snippet Editor is a self-contained program within PrimalScript and PowerShell Studio that supports multiple programming languages. Using the Snippet Editor is a fast and easy way to edit existing snippets and to create your own.

The Snippet Editor will launch when you edit an existing snippet or create a new snippet:



The Snippet Editor

Snippet Properties

The top section of the Snippet Editor allows you to enter the following snippet properties:

- **Title**
  The name of the snippet.
- **Shortcut**
  The text you need to type in the code editor to invoke the snippet.
- **Description**
  A short description of the snippet explaining what it does.
- **Author**

The snippet author details.

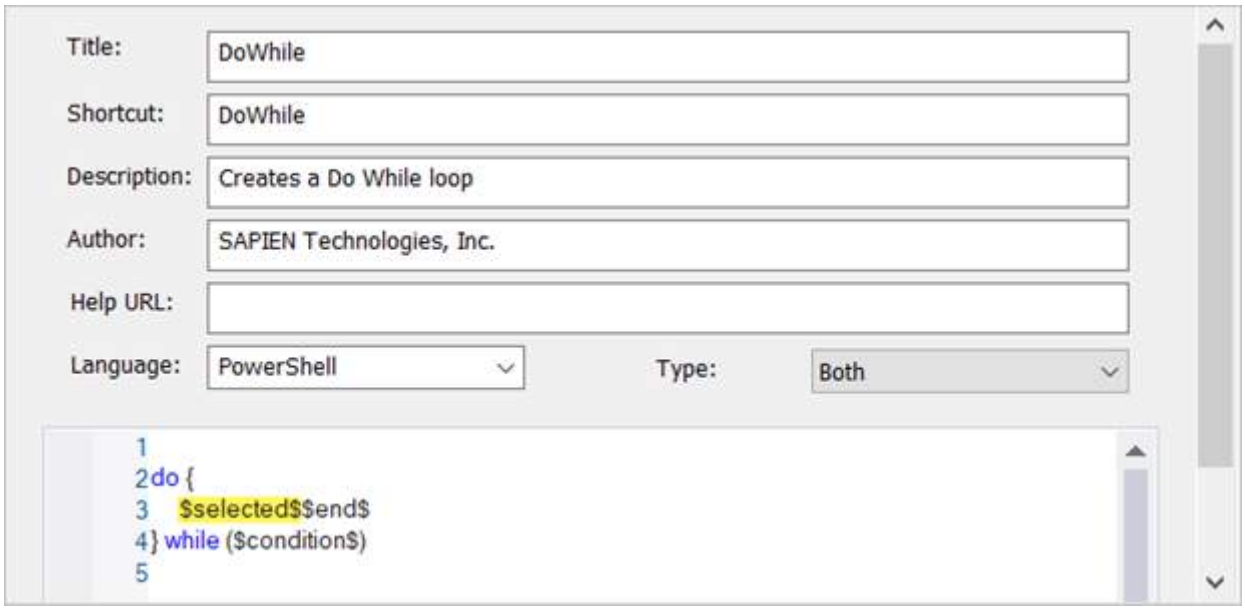
- **Help URL**
  A link to help information. This will be displayed in the code editor.
- **Language**
  Set to 'powershell' for snippets used in PowerShell Studio.
  Set to the appropriate language for snippets used in PrimalScript.
- **Type**
  This setting defines how the snippet will be displayed in the code editor (inserted into the code, or surrounding existing code). The options are:
  - ***Expansion***
    Select this if your snippet is intended to be simply inserted into code.
  - ***Surrounds With***
    Select this if your snippet can surround existing code.
  - ***Both***
    Select this if your snippet can be used both ways.

The selection you choose for the **Type** property will dictate the menu options available when you insert the snippet in the code editor:

- **Insert Snippet...**
  Only displays snippets where the 'Type' property is defined as ***Expansion*** or ***Both***.
- **Surround With Snippet...**
  Only displays snippets where the 'Type' property is defined as ***Surrounds With*** or ***Both***.

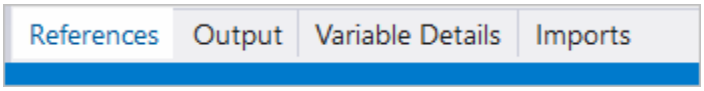
If you choose a 'Type' value of ***Surrounds With*** or ***Both*** you must include the **$selected$** placeholder variable* somewhere in your snippet code body, otherwise you may overwrite user code when your snippet is used:

(* Refer to the Built-in Placeholder Variables 225 section below for more information about the placeholder variables provided with the Snippet Editor.)

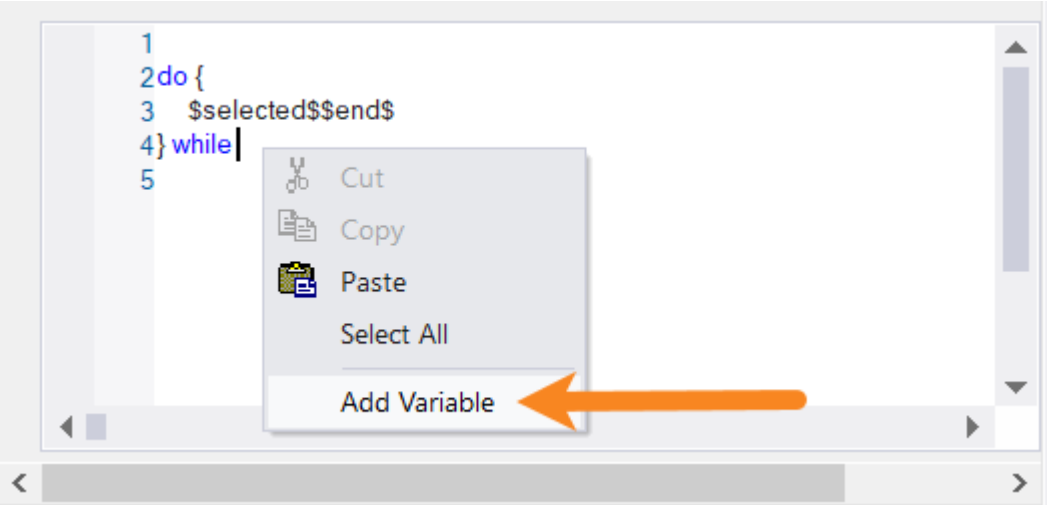


Snippet Windows

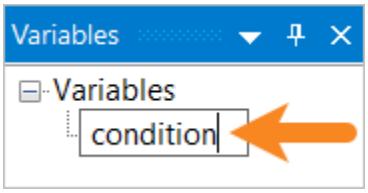
The tabs at the bottom of the Snippet Editor provide more configuration options for your snippets:



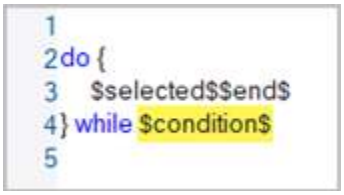
- **References**
  This section tells PrimalScript or PowerShell Studio what dependencies your snippet has. The assemblies you list here will be loaded into PrimalScript or PowerShell Studio when you use the snippet.
- **Output**
  This section is not used for PowerShell snippets.
- **Variable Details**
  Before you configure the variable details you must add a placeholder variable to a snippet:
  1. Position your cursor in the snippet code editor where you want to insert a variable, then right-click and select **Add Variable**:



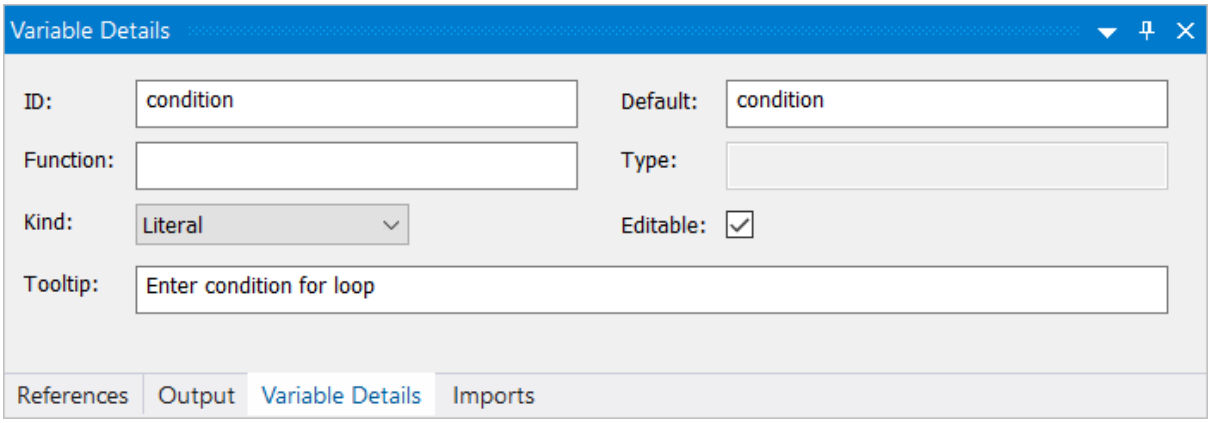
2. Name the variable:



3. The variable will appear in the snippet as **$<variable>$** (e.g., **$condition$**):



4. Configure additional variable properties:



- **ID**
  The variable name.
- **Default**
  A default value if required.

- **Function**
  Not used in PowerShell snippets.
- **Type**
  Not used in PowerShell snippets.
- **Kind**
  Not used in PowerShell snippets.
- **Editable**
  Not used in PowerShell snippets.
- **ToolTip**
  Provides some text explaining the purpose of the variable. This helps the snippet user understand how to complete the snippet. PrimalScript or PowerShell Studio will display these tooltips as the user navigates between the placeholders in the code editor.

- **Imports**
  This section is not used for PowerShell snippets.

Built-in Placeholder Variables

The Snippet Editor provides two built-in placeholder variables:

- **$selected$**
  Allows you to merge code from the code editor into your snippet when it is used. For example, you could create a snippet called ExtractFunction containing this code:

```
1function $function-name$()
2{
3    $selected$
4}
```

  Now you can highlight lines of code and use this snippet to refactor them into a reusable function.

- **end**
  Specifies where the cursor should be placed when a snippet is inserted into the code editor.

19 Reference

This section provides an overview of the SAPIEN Updates tool, and lists the keyboard shortcuts available in PrimalScript.

19.1 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.

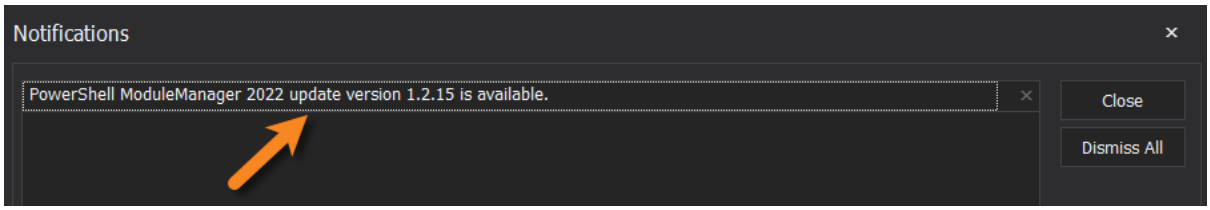
Every SAPIEN product has a built-in update tool—**SAPIEN Updates**—which will check for updates on all current activations and unexpired trial versions of our products. Available product updates are indicated in the SAPIEN Updates tool and also in the Notifications dialog 226 (see below).

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.

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

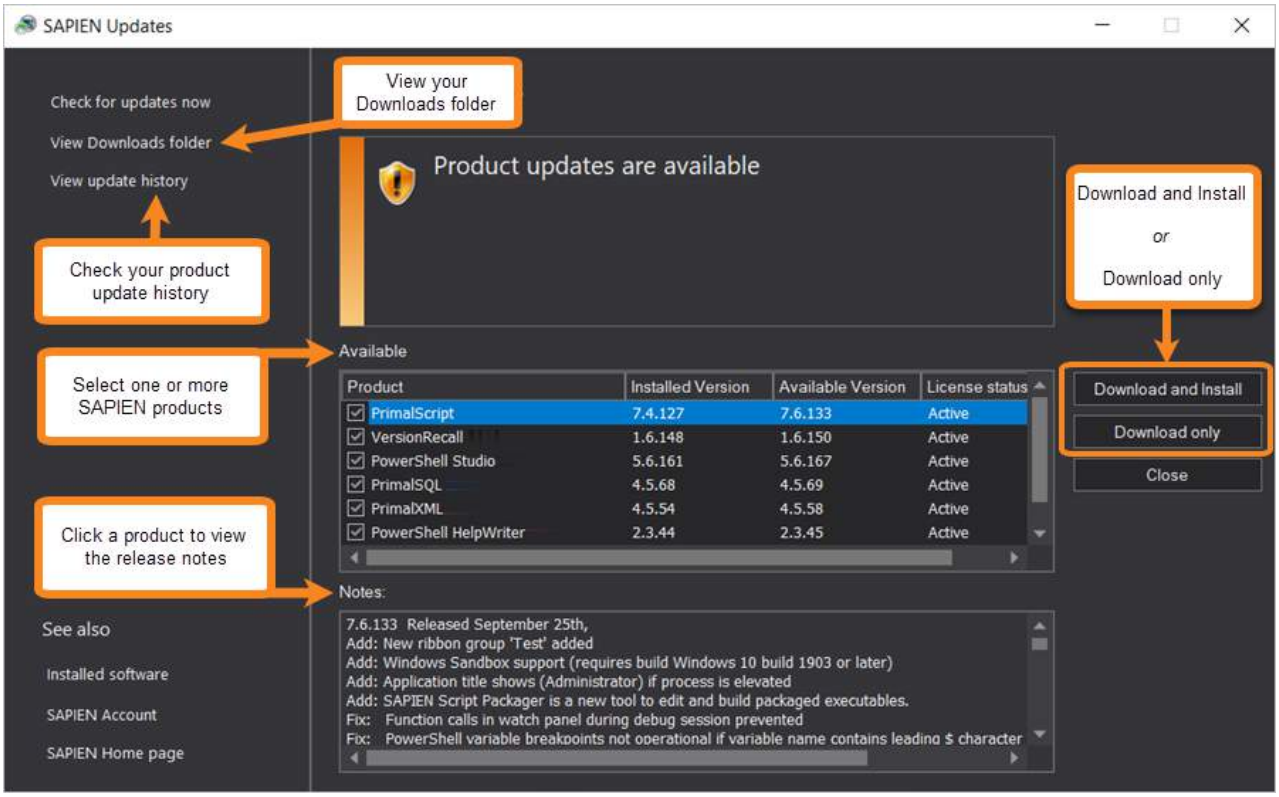
How to access the SAPIEN Updates tool

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

-OR-

- Click the notification icon 226 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	Displays a selectable list of available product updates. Select one or more products to Download or Download and Install.
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	Displays a brief synopsis of what was changed, added, or fixed for the products selected in the **Available** window. The build history for all SAPIEN products is available here.

Update On-Demand

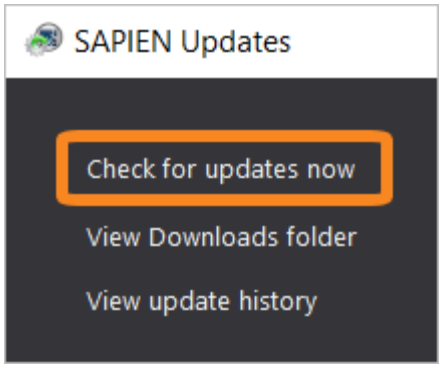
You don't need to wait to be notified when an update is available; you can check for updates at any time. This is particularly useful if you've heard about a new update and want to install it immediately, or if you are ready to start a new project and want to complete all updates before you begin.

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 SAPIEN Updates tool, or if you 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.

19.2 Keyboard Shortcuts

This section covers the keyboard shortcut commands available in PrimalScript.

General Commands

Copy	*Ctrl + C* *Ctrl + Insert*
Paste	*Ctrl + V* *Shift + Insert*
Cut	*Ctrl + X* *Shift + Delete*
Select All	*Ctrl + A*
Delete	*Del*
Undo	*Ctrl + Z* *Alt + Backspace*

Redo	*Ctrl + Y* *Alt + Insert*
New File	*Ctrl + N*
Open File	*Ctrl + O*
Open Project	*Ctrl + Shift + O*
Save	*Ctrl + S*
Print	*Ctrl + P*
Help	*F1*
Switch to Next Document Tab	*Ctrl + Tab*
Switch to Prev Document Tab	*Ctrl + Shift + Tab*
Access Ribbon Key Shortcuts	*F12*
Focus on search combo	*Ctrl + /*
View Console (Shell)	*Ctrl + Alt + A*
View File Browser	*Ctrl + Alt + F*
View Object Browser	*Ctrl + Alt + J*
View Tools Browser	*Ctrl + Alt + X*
New Project	*Ctrl + Shift + N*
Open related file	*Ctrl + Alt + O*
Rebuild last package	*Ctrl + Alt + P*
Run last package	*Ctrl + Shift + P*
View Startpage	*Ctrl + Alt + S*
Save All	*Ctrl + Shift + S*
Cycle clipboard	*Ctrl + Shift + V*
Global bookmarks	*Alt + F2*
File Properties	*Alt + Return*
View fullscreen	*Shift + Alt + Return*
Close file	*Ctrl + F4*
Find in Files	*Ctrl + Shift + F*
Replace in files	*Ctrl + Shift + H*

Document Commands

Package File	*Ctrl + F7*
Debug	*F5*
Run	*Ctrl + F5*
Stop Script	*Shift + F5*
Run in Console	*Ctrl + Alt + F5*
Run Selection	*Alt + X*
Run Selection in Console	*Ctrl + Shift + R*
Open code behind file (ASPX)	*Ctrl + Shift + B*
Load Header file (C / C + +)	*Ctrl + Shift + M*
Open file under cursor	*Ctrl + Alt + L*

Project Commands

Debug Project	*F6*
Run Project	*Ctrl + F6*
Add new item	*Ctrl + Shift + A*
Add existing item	*Alt + Shift + A*
Build project	*Ctrl + F8*
Build all	*Ctrl + Shift + F8*
Stop build	*Ctrl + Esc*

Debugging Commands

Debug Project	*F6*
Debug Document	*F5*
Resume	*F5*
Step Into	*F11*
Step Over	*F10*
Step Out	*Shift + F11*
Run to Cursor	*Ctrl + F10*
Toggle Breakpoint	*F9*
Delete all Breakpoints	*Ctrl + Shift + F9*
Toggle Tracepoint	*Ctrl + F9*
Restart debugger	*Ctrl + Shift + F5*

Editor Commands

Goto Line	*Ctrl + G*
Find	*Ctrl + F* *Alt + F3*
Replace	*Ctrl + H*
Find Next	*F3*
Find Previous	*Shift + F3*
Find Selection	*Ctrl + F3*
Comment Line	*Ctrl + Q*
Un-Comment Line	*Ctrl + Shift + Q*
Goto Next Bookmark	*F2*
Goto Previous Bookmark	*Shift + F2*
Toggle Bookmark	*Ctrl + F2*
Toggle Collapsed Code	*F8*
Collapse All Code Nodes	*Shift + Alt + F8* *Ctrl + Shift + Minus*
Expand All Code Nodes	*Shift + F8* *Ctrl + Shift + Plus*
Add Bold Tag	*Ctrl + B*
Capitalize text	*Alt + C*
Disable code	*Ctrl + Alt + D*
Enable code	*Ctrl + Alt + E*
Find Incremental	*Ctrl + I*
Double quote string	*Ctrl + Q*
Single quote string	*Alt + Shift + Q*
Replace word	*Ctrl + R*
Copy word	*Ctrl + Alt + W*
Cut word	*Ctrl + Shift + W*

Return Commands

Insert Line Break	*Enter*
Delete	*Del*
Delete Line	*Ctrl + Shift + L*
Delete To Next Word	*Ctrl + Del*
Backspace	*Backspace*
Backspace	*Shift + Backspace*
Backspace To Previous Word	*Ctrl + Backspace*
Copy To Clipboard	*Ctrl + C* *Ctrl + Insert*
Cut To Clipboard	*Ctrl + X* *Shift + Delete*
Paste From Clipboard	*Ctrl + V* *Shift + Insert*
Undo	*Ctrl + Z* *Alt + Backspace*
Redo	*Ctrl + Y* *Alt + Insert*
Move Down	*Down*
Move Up	*Up*
Move Left	*Left*
Move Right	*Right*
Move To Previous Word	*Ctrl + Left*
Move To Next Word	*Ctrl + Right*
Move To Line Start	*Home*
Move To Line End	*End*
Move To Document Start	*Ctrl + Home*
Move To Document End	*Ctrl + End*
Move Page Up	*PgUp*
Move Page Down	*PgDn*
Move To Matching Bracket	*Ctrl +]*

Move To Next Modified Line	*Ctrl + Shift + Down*
Move To Prev Modified Line	*Ctrl + Shift + Up*
Go To Last Edit Position	*Ctrl + E*
Go To Function Declaration	*Ctrl + F12*
Next paragraph	*Alt + Down*
Previous paragraph	*Alt + Up*
Next error	*F4*
Previous error	*Shift + F4*
Previous function	*Ctrl + PgUp*
Next function	*Ctrl + PgDn*
Previous occurrence	*Ctrl + Shift + I*
Next occurrence	*Ctrl + Alt + I*

Scroll Commands

Scroll Down	*Ctrl + Down*
Scroll Up	*Ctrl + Up*
Current caret position	*Ctrl + Shift + E*

Indenting Commands

Indent	*Tab*, *Alt + Right*
Outdent	*Shift + Tab* *Alt + Left*

Selection Commands

Select Down	*Shift + Down*
Select Up	*Shift + Up*
Select Left	*Shift + Left*
Select Right	*Shift + Right*
Select To Previous Word	*Ctrl + Shift + Left*
Select To Next Word	*Ctrl + Shift + Right*
Select To Line Start	*Shift + Home*
Select To Line End	*Shift + End*
Select To Document Start	*Ctrl + Shift + Home*
Select To Document End	*Ctrl + Shift + End*
Select Page Up	*Shift + PageUp*
Select Page Down	*Shift + PageDown*
Select All	*Ctrl + A*
Select Word	*Ctrl + W*
Select To Matching Bracket	*Ctrl + Shift +]*
PrimalSense Complete Word	*Ctrl + Space*
Show parameters	*Ctrl + Alt + Space*
List members	*Ctrl + Alt + T*

Other Commands

Change Character Casing (to uppercase)	*Ctrl + Shift + U*
Change Character Casing (to lowercase)	*Ctrl + U*
Toggle Overwrite Mode	*Insert*
Transpose Characters	*Ctrl + T*
Insert Snippet	*Ctrl + J*

19.3 Appendices

Appendices for PrimalScript Help Manual

19.3.1 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 PrimalScript articles on the SAPIEN blog.

Release details

To view a brief description of what was changed, added, or fixed in the most recent PrimalScript builds, view the product version history.

Need more help?

Please direct your product related questions to the PrimalScript support forum, and your scripting questions to the appropriate Scripting Answers forum.

19.3.2 Appendix B: Icon License Attribution

Appendix B

Icon License Attribution

Some of the icons used in this manual were made by Freepik at www.flaticon.com and are licensed under CC BY 3.0: