

[**WINDOWS AND USER PRODUCTIVITY**](https://www.itprotoday.com/windows-and-user-productivity)**>**[**POWERSHELL**](https://www.itprotoday.com/windows-and-user-productivity/powershell)

**PowerShell One-Liner: Creating and Modifying an Environment Variable**

In this article, I'll show you how to quickly and easily create and modify environment variables.

[Rod Trent](https://www.itprotoday.com/author/Rod-Trent) | Jul 12, 2013

In [PowerShell One-Liner: Getting Local Environment Variables](http://windowsitpro.com/powershell/powershell-one-liner-getting-local-environment-variables), I showed you how to use a single PowerShell command to get the full list of environment variables known to the operating system, and also how to retrieve just a single value. Once you have that mastered, it's time to move on to bigger and better things.

In this article, I'll show you how to quickly and easily create and modify environment variables. In a future article I'll wrap up this series on environment variables by working on deleting them.

**Creating an Environment Variable**

There are a couple different types of environment variables that you can create. The first one is a temporary variable that lasts only as long as your PowerShell session, allowing you to create a pointer to work with for the rest of your PowerShell script.  You create this type of variable using the *$env:*operation.

The following one-liner performs this function:

*$env:MyTestVariable = "My temporary test variable."*

Using the $env function, I assigned the new environment variable a name (MyTestVariable) and supplied it with some data ("My temporary test variable.").

Make sense?

But, what if you want to create a more permanent environment variable that is accessible anytime? What if an application requires the environment variable be present, or you plan to utilize the contents of the environment variable across your entire Enterprise for future scripts?  In that case, you'll want to use the *SetEnvironmentVariable*method that is provided as part of the .NET Framework integration.

To invoke this method, run the following one-liner:

*[Environment]::SetEnvironmentVariable("TestVariableName", "My Value", "<option>")*

In the one-liner above, "TestVariableName" is the name I gave to the environment variable and the "My Value" is the data that I assigned to it. The "*<option>*" parameter allows you to tell PowerShell the type of environment variable should be created. You can assign the new environment variable to the *User*level (per Windows user profile), *Machine*level (for anyone logging onto the computer), or *Process*level (another way to create a temporary variable like above, using the *$env*operation).

Here's another one-liner example modified to show you how to assign the new environment variable to the Machine level.

*[Environment]::SetEnvironmentVariable("TestVariableName", "My Value", "Machine")*

**Modifying an Environment Variable**

Now that you understand how to create new environment variables and comprehend (hopefully) how to configure the different types of environment variables, let's move on to another pretty simple concept: modifying environment variables.

There's really not much more you need to do, or know beyond this: If an environment variable with the name you provide already exists, PowerShell will simply take the data you are assigning to the variable and update the existing one with the new information.

So, modifying an existing variable is easy, however, this is where our first one-liner example comes into play and becomes very important. You want to be a bit careful when modifying environment variables since some applications, and even the OS, rely on the information they provide. So, it helps to know which variables already exist and what information they are providing. So, if you haven't yet, go and read: [PowerShell One-Liner: Getting Local Environment Variables](http://windowsitpro.com/powershell/powershell-one-liner-getting-local-environment-variables)

**Part 3:** [PowerShell One-Liner: Deleting Environment Variables](http://windowsitpro.com/powershell/powershell-one-liner-deleting-environment-variables)