**Language of PowerShell Lab Answers**

**Command Syntax**

In this lecture I’m gonna give you the answers to the questions, from the Command Syntax Section.

**We’ll take a look at the command syntax from get service**

The Answer is in 6 parts, we’ll start from the left and work to the right

1. Every cmdlet is based upon the same **structure** A name for this **structure** is? Verb - Noun
2. A dash tells PowerShell that a Parameter is being used.
3. Two angle brackets tell’s me that this a Argument.
4. The word string is located between two angle brackets, string is called a Value Type.
5. When two square brackets are surrounded by two angle brackets this tells us that this can take multiple arguments separated by a comma.
6. These three groups are called \_Parameter sets.

**Question 2 video and tablet**

**The question was:**

* Take a look at the parameter sets from get-service. Can you tell me which parameters are unique in each set? Now tell me what parameters are common in each set.

**The answer is in three parts.**

* **In order to have a parameter set each set must have at least one unique parameter. Our unique parameters in this case are:**
* Displayname from set #1
* Inputobject from set # 2
* Name from # 3

**Question 2 Part 2**

* Question was:
* Now tell me what parameters are common to each set.

**The Common Parameters in Set # 1,2, 3 are:**

* Computername, DependentServices, Exclude, Include, RequiredServices, Common Parameters

**Question 2 Part 3**

* The Question was:

You probably noticed that commonparameters are common to all three sets.

* What command would you use to checkout the help for commonparameters?
* Type **get-help about\_common\_parameters.**

Scroll to the top.

* Some of the highlights are **debug, outvariable, verbose, whatif and confirm**
* If you want to know more notice there are descriptions for each parameter.
* Most of the time these parameters work with every cmdlet.

**Question # 3**

* The Question was:
* If I type **get-service**, then press return.

Why does the command run without using any parameters?

Type CLS

* Type **get-help get-service -full.**
* Now using the syntax, explain why getservice **ran** without using any parameters.

**The answer is:**

* Because all the parameters in parameter set # 2 and 3, all have square brackets around them.
* That means that all the parameters are optional and not needed.

And that means that get-service will run by itself without any parameters.

**Question #4**

* The question was:

**Type get-eventlog, then press return.**

**Notice that geteventlog is asking for a value or a logname**

Press Ctrl-C

* Using the syntax, explain why geteventlog requires you to type a value or a logname, while get-service ran without a value or even a parameter
* Type get-help get-eventlog -full. Scroll up to the Syntax section.
* Notice that every parameter is **surrounded by square brackets.**
* That means that all the parameters for the command get-eventlog are optional and not needed.
* Notice that -**logname** is surrounded by square brackets but the argument is not.

That means that -logname is **optional** (you don’t have to type -logname)

but you have to type a string (which in this case is a text string)

* So that’s why when you type get-eventlog and you press return

PowerShell asks you for a value.

* Let’s go ahead and type **security** (which is one of the Windows log files.

**Question #5**

* The Question was:
* Using help get-eventlog -showwindow.
* Take a look at the parameter -instanceID .
* Explain why -instanceID should be second in the order of parameters?

-InstanceID <int64> is **position 1** so it’s second in the lineup of parameters.

-logname <string> is **position 0** so that makes it first in the lineup

* So, the thing to remember is that PowerShell starts the lineup of parameters at position 0

**Question #6**

Let’s scroll down to newest.

* The question was:
* Why can you place –newest, anywhere, in the order of parameters?

- Newest <Int32> The **position for -newest is named.**

* That means you can put it anywhere in the lineup.

Thanks for checking out the questions and answers in this section.