Pipeline Input Workbook
http://MonthOfLunches.com

This worksheet is designed to help you figure out how two commands will connect to one another in the pipeline. Start by filling in your two commands in the boxes below. You'll fill in the same command names in the same boxes throughout this worksheet.

Tip: If you use Adobe Reader to fill in this form, it'll auto-populate all but the first two boxes.

Command 1:		Command 2:
Test	|	

Refer back to these boxes to remind yourself which command is Command 1, and which is Command 2, in the following pages.

Plan A: ByValue

Pipe the first command to **Get-Member.** Note that the command *will run,* so if it's a potentially dangerous command, be sure you're doing this in a test environment.

Command 1:

```
Test | Get-Member
```

Look at the first line of output for the type name, and write it here:

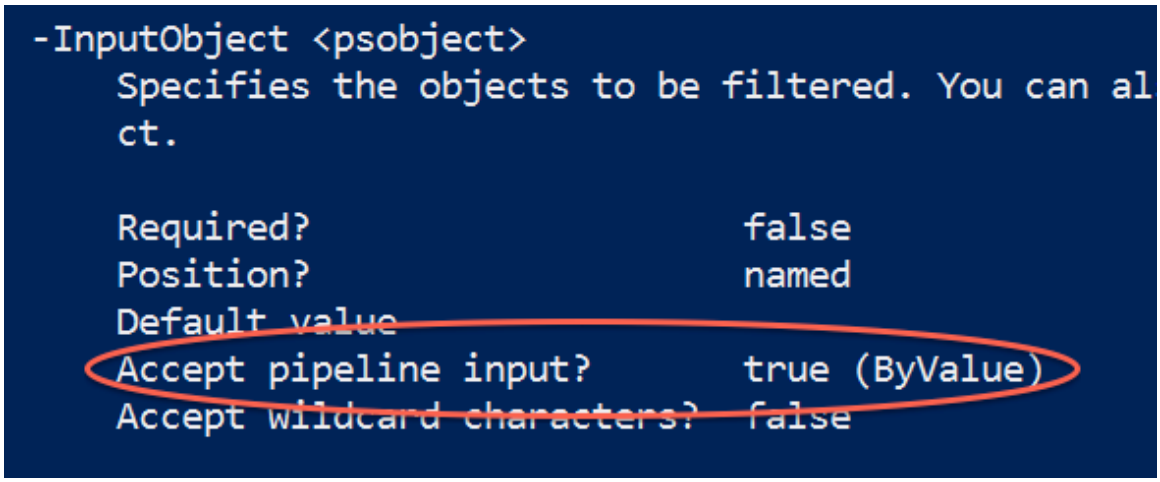
TypeName of Command 1 Output:

Now look at the *full* help for Command 2:

Command 2:

```
Get-Help [                    ] -Full
```

Do you see any parameters of Command 2 that can accept the type of object produced by Command 1, from the pipeline, ByValue?



For example, if the TypeName of the Command 1 output is System.Diagnostics.**Process**, then in the help file for Command 2 you would look for a parameter that can accept <**Process**>.

Note: If you find a parameter that accepts the generic <Object> or <PSObject> type, from the pipeline, ByValue, then that is the parameter you want.

If you find such a parameter, write the parameter name here:

Parameter of Command 2 that accepts the object type produced by Command 1:

Did you find a parameter?

YES → ByValue will work. The objects produced by Command 1 will attach, or *bind*, to the parameter you identified. Read the help for Command 2 to see what that parameter will do. ● **Stop – you're done.**

NO → ByValue will not work. → **Continue to the next page.**

Plan B: ByPropertyName

Continue to look at the *full* help for Command 2.

Command 2:

```
Get-Help                          -Full
```

You are looking for parameters that are listed as accepting pipeline input ByPropertyName. Make a list of all such parameters in Column A on the next page.

```
Required?                    false
Position?                    named
Default value                Localhost
Accept pipeline input?       true (ByPropertyName)
Accept wildcard characters?  false
```

Now look again at the **Get-Member** output from Command 1.

Command 1:

```
Test                              | Get-Member
```

Look specifically at anything labeled as a Property, ScriptProperty, AliasProperty, or NoteProperty. In Column B on the next page, write down any property names that are already listed in Column A.

```
Name                 MemberType
----                 ----------
Name                 AliasProperty
RequiredServices     AliasProperty
CanPauseAndContinue  Property
CanShutdown          Property
CanStop              Property
Container            Property
DependentServices    Property
DisplayName          Property
```

In other words, if you see a property named “ComputerName,” and Column A already contains “-ComputerName,” then write “ComputerName” in Column B, on the same line where Column A contains “-ComputerName.”

Column A Candidate Parameters		Column B Matching Properties

The values from the properties in Column B will be passed to the parameters listed in Column A. Look at the help file for Command 2 to determine what these parameters will do.

Stop – you're done.