Core PowerShell Cmdlets

PowerShell Cmdlets

- Designed to work with objects

- Designed to work with a variety of providers
  - Some help examples vary by provider
  - Use same alias for all providers

- Common cmdlets you will use all the time

- Provide core functionality

- Read full help and examples for all cmdlets in this lesson

Get-ChildItem

Commonly used command in FileSystem
- Aliases: dir, ls, gci

Retrieves child objects in a hierarchy
- Active Directory Organizational Unit
- Certificate store
- Registry key

Use filtering options when possible
- –Include, –Exclude, –Filter
- These parameters vary in use by provider

```
PS C:\> dir c:\scripts –file -recurse
```

Where-Object

Used for filtering objects in the pipeline
- Aliases: where, ?

Traditional syntax uses a script block and $_
- PS C:\> get-service m* | where {$_.status –eq 'running'}

New syntax can be simpler
- PS C:\> get-service m* | where status –eq 'running'

New syntax won't work for complex filtering

Select-Object

Select just what you need
- Aliases: Select, slo

Select object properties
- Filter properties with –Exclude or –Include
- Expand nested object properties
- Writes a custom object to the pipeline

Select a subset of objects
- First or Last X number of objects
- Unique objects
- Skip first X number of objects (starting at 1)

Many of these parameters can be combined

Selecting is not the same as formatting

```
PS C:\> get-process | select
id,name,workingset,path –first 5
```

Sort-Object

Sort objects on property value, usually numeric
- Alias: sort

Specify a comma separated list of property names
- Can sort on a custom property

Default order is ascending
- Or use –Descending
- Can only sort multiple properties in one direction unless you use custom properties

Not case-sensitive by default

Get objects with a unique value for property name

```
PS C:\> get-process | sort WorkingSet
–descending | select –first 10
```

Group-Object

Organize objects into groups based on a property
- Alias: group

Creates a new object:
Microsoft.PowerShell.Commands.GroupInfo
- Count, Name, Group (array of grouped objects)
- Use –NoElement to only get Count and Name

Can create a hash table instead

Can group on a custom property

```
PS C:\> get-process| group company -noelement | sort
count,@{Expression="name";Descending=$False}
–descending
```

Measure-Object

Alias: measure

Writes a new object to the pipeline:
Microsoft.PowerShell.Commands.GenericMeasureInfo

Perform quick measurements based on a numeric property
- Always gets Count
- Sum, Minimum, Maximum, Average

Perform quick measurements based on words and characters
- Character, Line, Word

```
PS C:\> dir c:\scripts –file | measure length –sum
-average
```

Get-Content

Read text file contents
- Aliases: cat, type, gc

Improve performance for very large files with –ReadCount

Get the "head" of a file (–TotalCount)
- –parameter aliases: –head, –first

Get the "tail" of a file (–Tail)

Read complex text files
- –Delimiter
- –Encoding
- –Raw
- –Stream

```
PS C:\> get-content computers.txt | foreach
{get-service wuauserv –comp $_}
```

Core Cmdlet Demonstrations

Lab

1. List all .txt files under C:\Windows and all subfolders

2. Repeat and measure how many txt files and their total size

3. Get all processes that have a company name property, group them by company name and sort the count in descending order

4. Get processes that have a StartTime property, sort on it then select the ID, Name, WorkingSet and StartTime properties, but only display the first 10 processes.