Working with Objects

TRAINSIGNAL

THE GLOBAL LEADER IN PROFESSIONAL COMPUTER TRAINING

The PowerShell Paradigm

- PowerShell is an object based management engine
- Cmdlets work with objects in the pipeline
- Objects = no text parsing
- Always think about working with objects, their properties and methods

What is an Object?

- An object is a software construct
- It has properties that describe it
- It has methods that it can do or be done to it
- Most cmdlets handle the "dirty work"

```
PS C:\> help about_objects
```

What is an Object?

Class:
Automobile
Parent
Class:
Vehicle

Some properties can be changed (R/W)

Some properties are read-only

Properties:
Color: Blue
Owner: Jeff
Weight: 1245
Doors: 2
Automatic: True
MaxSpeed: 120
VIN: ABC1234DEF
Engine: EngineObject

Methods:
Start()
Stop()
ChangeOwner(newOwner)
Accelerate(EndSpeed)

Some actions need parameters

Some properties can be nested objects

TRAINSIGNAL

What is an Object?

Class: System.ServiceProcess.ServiceController

Properties:
Name: Spooler
DisplayName: Print Spooler
Status: Running
Machinename: .
DependentServices: Fax

Methods:
Start()
Stop()
Pause()

TRAINSIGNAL

Using Get-Member

An objects properties and methods are referred as its *members*

Get-Member is the discovery cmdlet
- Alias: gm

Don't assume that cmdlet output shows all or actual properties
- Some properties are calculated or custom
- PowerShell attempts to be IT Pro friendly

Pipe any command to Get-Member to learn about objects exiting the pipeline

TRAINSIGNAL

Get-Member

AliasProperty

Property (Get;Set)

PropertySet

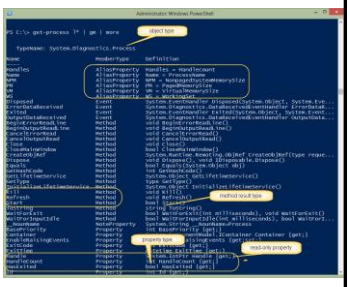
ScriptProperty

NoteProperty

Method

ScriptMethod

Event



Using Object Members

Display all object properties

```
PS C:\> get-process lsass | select *
PS C:\> get-process lsass | format-list *
```

Tip: Use the alias fl

Once you know properties you can use them

```
PS C:\> get-process | Sort pagedmemorysize -desc | Select
ID,Name,*MemorySize -first 5
```

Use object methods only if there isn't a cmdlet

```
PS C:\> get-wmiobject win32_process | foreach {
$_.GetOwner()} | Select User -unique
```


Creating Custom Properties

Create custom properties with Select-Object, Format-List or Format-Table

```
@{Name=<property name>;Expression= {<code>}}
```

Expression scriptbl[illegible]ple [illegible]as complex as necessary

You can also use shortcut of N for Name

And E for Expression

Use $_ to reference current object in the pipeline in code

Properties will be piped to next cmdlet

Subtract each object's StartTime property from the current date

```
PS C:\> get-process | select
ID,Name,StartTime,@{Name="RunTime";Expression={(get-date)
-$_.StartTime}} | sort RunTime -desc | select -first 10
```

Adding Members

Extend an object with Add-Member

NoteProperty	CodeProperty	ScriptProperty	Property
AliasProperty	ScriptMethod	Method	PropertySet
CodeMethod	MemberSet	Dynamic	Event

Assign a new type name

Extensions are temporary

Only works with PSObject types

Add-Member

```
PS C:\> $files = dir c:\scripts –file

PS C:\> $files | Add-Member -MemberType AliasProperty -Name
Size -Value Length

PS C:\> $files | Add-Member ScriptProperty -Name FileAge
-Value {(Get-Date) - ($this.LastWriteTime)}

PS C:\> $files | Add-Member ScriptProperty -Name
FileAgeDays -Value {[int]((Get-Date)
- ($this.LastWriteTime)).TotalDays}

PS C:\> $files | Add-Member Noteproperty Computername
$env:computername

PS C:\> $files | sort Size -Descending | Select
FullName,Size,FileAge*,Computername -first 10
```


Object Members in Action

Creating Objects

You can create your own objects or objects "on-the-fly"

Use Type Accelerators

```
PS C:\> [adsi]$admin="WinNT://CHI-FP01/Administrator,User"
```

PowerShell creates an object based on a type

```
PS C:\> $xmas = [datetime]"12/25/2014"
PS C:\> [datetime]$xmas = "12/25/2014"
PS C:\> [xml]$config = get-content c:\work\config.xml
```

Or use –AS operator

```
PS C:\> $i = 34.567 -as [int]
```

Common accelerators and types:
[wmi] [wmiclass] [adsi] [datetime] [xml]

Discover them on your own:

```
PS C:\> [psobject].assembly.gettype("System.Management.Automation.TypeAccelerators")::Get
```

TRAINSIGNAL

New-Object

Create a new object based on its class name

```
PS C:\> $word = new-object
Microsoft.Office.Interop.Word.ApplicationClass
```

Or create a custom object using a hash table of properties

```
PS C:\> $h = @{Name="Jeff"; Computer=$env:computername;
Time=(Get-Date)}
PS C:\> $o = New-Object PSObject –property $h
```

Or use a type accelerator

```
PS C:\> [pscustomobject]$h
PS C:\> $o = [pscustomobject][ordered]@{Name="Jeff";
Computer=$env:computername; Time=(Get-Date)}
```

TRAINSIGNAL

COM Objects

- Created with New-Object -COM
- Pipe to [illegible] to discover how to [illegible]
- Not all [illegible] properties are usable


```
PS C:\> $wsh = New-Object -COM wscript.shell
PS C:\> $wsh.Popup("Good afternoon
$env:username",5,"Hello")
```

TRAINSIGNAL

Objects in Action

Lab

1. What type of objects do you get when you run: Get-Eventlog -list

2. What do you think is the actual property name for the Max(K) column?

3. Using Get-Eventlog display the 100 most recent events in the System event log, but only show the TimeGenerated, a property that shows how old the event is, its source, entry type and a property that displays Computername instead of Machinename.