**Praca z potokami – info**

Co to jest potok?

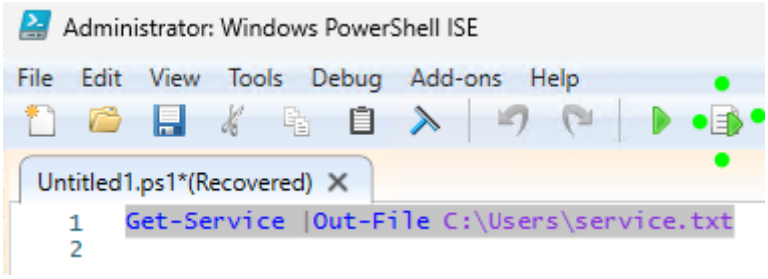
Jak zapisać dane w pliku

Jak przeglądać dane w GridView

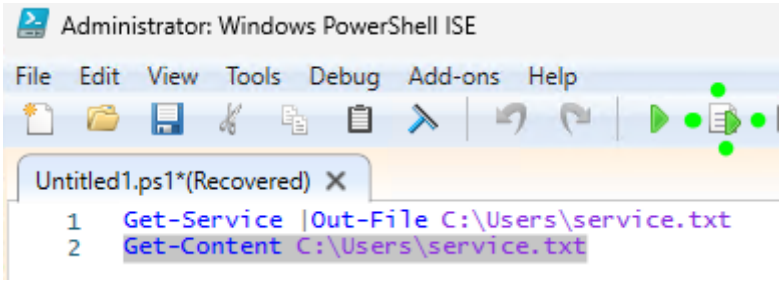
Jak działa potok

Do czego służy Get-Member

Możliwości PowerShell widać, gdy komendy łączymy w potokach.



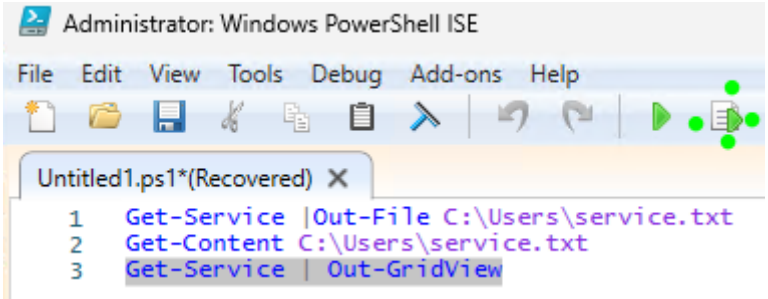

lub F8




lub F8

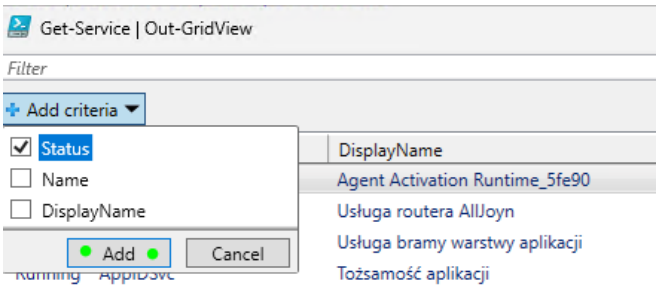
Umieszczanie danych w pliku tekstowym jest konieczne, jeżeli wynik zachowasz do dalszej analizy w przyszłości.

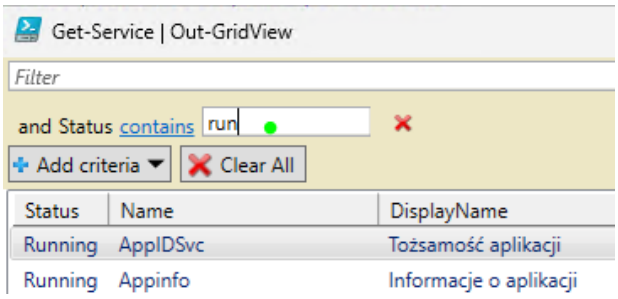
Jeżeli chcesz zobaczyć dane teraz że wygodniej jest użyć instrukcje jak poniżej



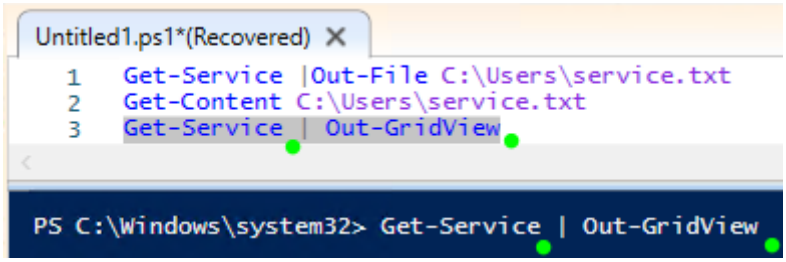

lub F8

Uruchomi graficzną przeglądarkę do przeglądania wyniku



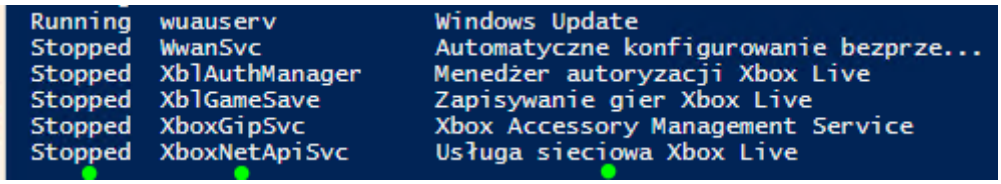




Popatrzcie na wynik niżej.



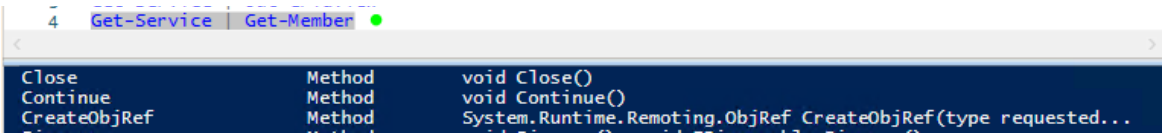

Jak działa przekazywanie danych z jednej komendy do drugiej przez potok.

W skryptach unix lub dos jeżeli druga komenda miała przetworzyć dane z pierwszej komendy to musiała umieć przetworzyć tekst zwracany przez pierwszą komendę,




co wiązało się z parsowaniem tekstu tak aby kolejna komenda mogła go dalej przetwarzać.

W PowerShell tekst, który widać na ekranie to tylko reprezentacja obiektów, na których pracują wszystkie polecenia. Poniżej dowód.



F8

Jego głównym zadanie jest odpowiadać na pytanie z jakim obiektem pracujemy, jakie ma właściwości ten obiekt lub jakie metody można wywołać na tym obiekcie wystarczy jak powyżej dowolna komenda (tu get-service) i jej wynik przesłać potokiem do get-member

Przeanalizujmy wynik

Można ustalić o jaki obiekt chodzi tu:

```
PS C:\WINDOWS\system32> Get-Service | Get-Member

   TypeName: System.ServiceProcess.ServiceController
```

Znając tę nawę możesz szukać opisu tego typu na stronach www np. msdn

Widać, że w tym typie mamy zdefiniowane aliasy

```
   TypeName: System.ServiceProcess.ServiceController

Name                      MemberType     Definition
----                      ----------     ----------
Name                      AliasProperty  Name = ServiceName
RequiredServices          AliasProperty  RequiredServices = ServicesDependedOn
Disposed                  Event          System.EventHandler Disposed(System.
```

Obiekty mają dużo właściwości

```
WaitForStatus             Method         void WaitForStatus(System.ServicePr
CanPauseAndContinue       Property       bool CanPauseAndContinue {get;}
CanShutdown               Property       bool CanShutdown {get;}
CanStop                   Property       bool CanStop {get;}
Container                 Property       System.ComponentModel.IContainer Co
DependentServices         Property       System.ServiceProcess.ServiceContro
DisplayName               Property       string DisplayName {get;set;}
MachineName               Property       string MachineName {get;set;}
ServiceHandle             Property       System.Runtime.InteropServices.Safe
ServiceName               Property       string ServiceName {get;set;}
ServicesDependedOn        Property       System.ServiceProcess.ServiceContro
ServiceType               Property       System.ServiceProcess.ServiceType S
Site                      Property       System.ComponentModel.ISite Site {g
StartType                 Property       System.ServiceProcess.ServiceStartM
Status                    Property       System.ServiceProcess.ServiceContro
ToString                  ScriptMethod   System.Object ToString();
```

Dzięki CanStop dowiadujemy się, że ta usługa może być zatrzymana.

Właściwość może być typu string jak MachineName co pozwala określić na jakim komputerze pracuje usługa.

Bardziej złożona ServicesDependedOn określa jakie inne usługi muszą być uruchomione wcześniej, aby ta usługa zadziałała.

Ta właściwość jest kolekcją elementów typu ServiceController który teraz omawiamy

Widać tu również metody którymi dysponuje ten typ

```
   TypeName: System.ServiceProcess.ServiceController

Name                      MemberType    Definition
----                      ----------    ----------
Name                      AliasProperty Name = ServiceName
RequiredServices          AliasProperty RequiredServices = ServicesDependedOr
Disposed                  Event         System.EventHandler Disposed(System.
Close                     Method        void Close()
Continue                  Method        void Continue()
CreateObjRef              Method        System.Runtime.Remoting.ObjRef Creat.
Dispose                   Method        void Dispose(), void IDisposable.Dis.
Equals                    Method        bool Equals(System.Object obj)
ExecuteCommand            Method        void ExecuteCommand(int command)
GetHashCode               Method        int GetHashCode()
GetLifetimeService        Method        System.Object GetLifetimeService()
GetType                   Method        type GetType()
InitializeLifetimeService Method        System.Object InitializeLifetimeServ.
Pause                     Method        void Pause()
Refresh                   Method        void Refresh()
Start                     Method        void Start(), void Start(string[] arg
Stop                      Method        void Stop()
WaitForStatus             Method        void WaitForStatus(System.ServicePro.
```

get-member to największy przyjaciel programisty lub administratora w powershell, pozwala odkrywać co można zrobić z obiektami.

Komenda poprzedzająca get-member naprawdę **wykonuje się** następnie przesyła wynik do get-member który wyświetla informacje o typie zwracanych danych nie ma tu sytuacji jak w przypadku parametru WhatIf który odpowiadał na pytanie co by się stało gdybym uruchomił tą komendę.

Niektóre polecania zawracają mix obiektów różnych typów

np

```
Untitled1.ps1*(Recovered)
1  Get-ChildItem C:\Users

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        01.07.2024     18:28                admin
d-r---        26.06.2024     10:15                Public
-a----        03.07.2024     14:16          37006 service.txt
```

Katalog zawiera obiekty typu różnego typu pliki i katalogi

Aby sprawdzić, że pliki i katalogi to obiekty różnego typu wpisz jak poniżej

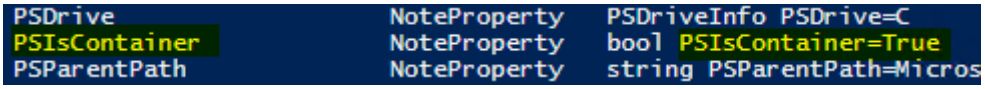
```
Untitled1.ps1*
1  Get-ChildItem C:\Users | Get-Member
```

Jak widać mamy informacje o typach:

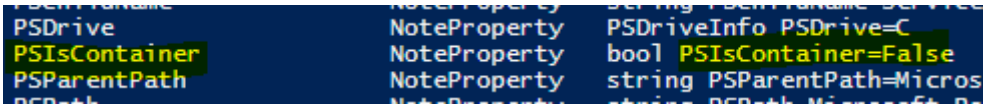
```
TypeName: System.IO.FileInfo
```
i
```
TypeName: System.IO.DirectoryInfo
```

Widać, że wyjście pojedynczej komendy może zwracać obiekty różnych typów, różniące się swoimi właściwościami i metodami.

Właściwością rozróżniającą plik od folderu jest PSIsContainer



- dla katalogów



- dla plików

Dowiedziałeś się, co to jest potok

Jak zapisać dane w pliku

Jak przeglądać dane w GridView

Jak działa potok

Do czego służy Gel-Member