IT-PRO-BERLIN.DE

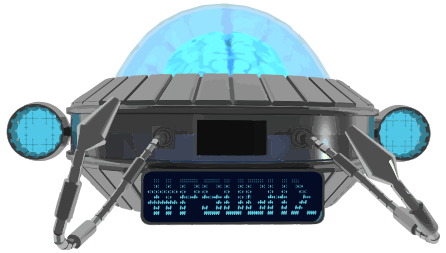
Arbeiten mit Active Directory

jenseits von `Get-ADUser -Filter *`

Netz-Weise
Lernen von den Besten.

au2mator

ScriptRunner®
Your #1 for PowerShell Automation



Microsoft® MVP Most Valuable Professional

Get-SpeakerInfo -Brief

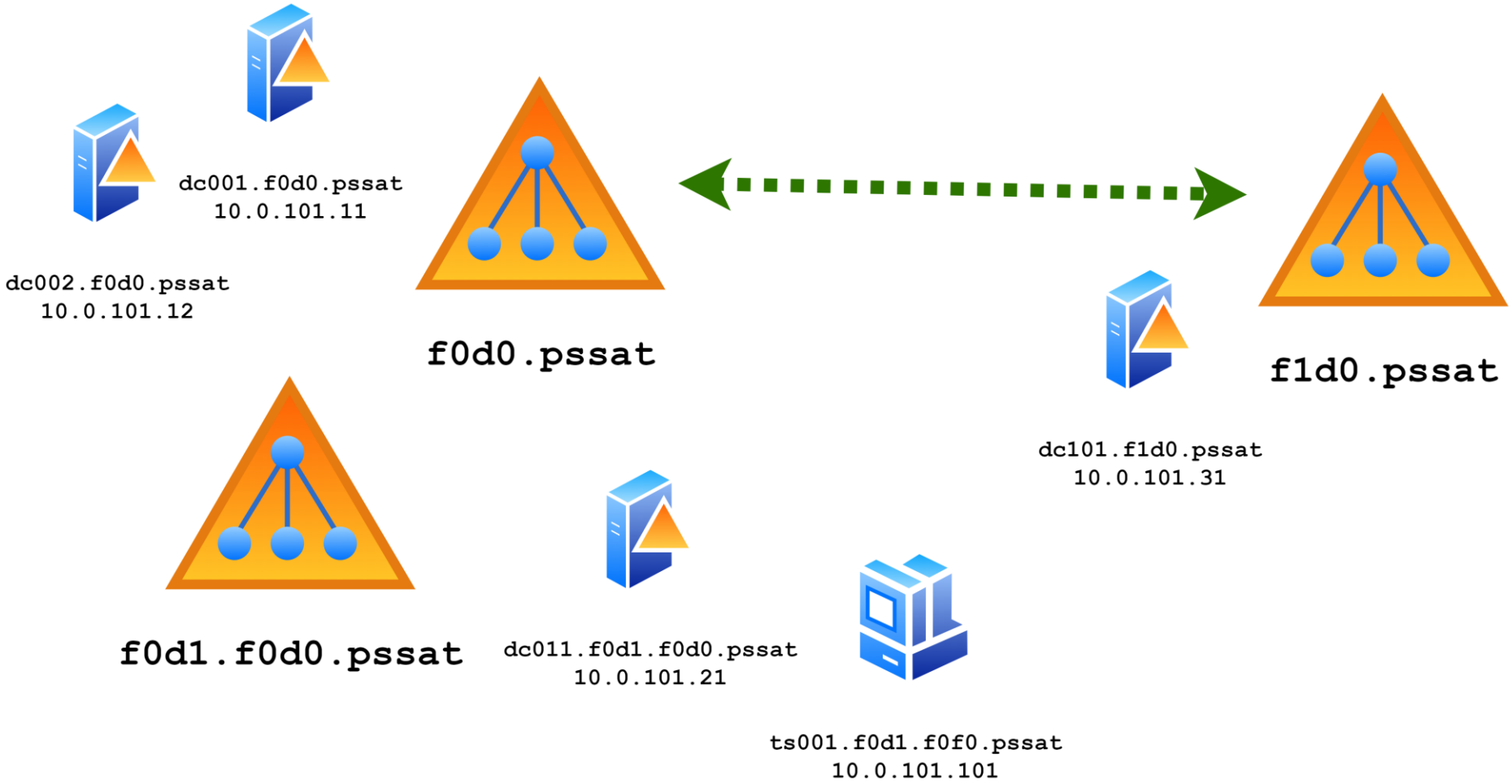
- **Name:** Evgenij Smirnov
- **YearOfBirth:** 1972
- **JobTitle:** { Senior Solutions Architect }
- **TwitterID:** @cj_berlin
- **EMailAddress:** es@it-pro-berlin.de
- **Employer:** Semperis
- **MVP:** { CDM 2020-2024 }
- **Certifications:** { MCSE, MCSA, VCP, VCAP, VCIX, CCA, QCIC }
- **UserGroups:** { WSUG-B, EXUSG, PSUGB}

(unverbindliche) Workshop-Ziele

- **Skripte für AD produzieren, die**
  - **schnell**,
  - **robust** und
  - **portabel** sind
- **Spaß mit AD und PowerShell haben**
- **Voneinander lernen**

Workshop = Talk Shop

Zum Mitmachen…



Get-Technique | Sort-Object -Property Nerdiness

- (*Quest ActiveRoles PowerShell-Modul*)
- ActiveDirectory PowerShell-Modul
- System.DirectoryServices.ActiveDirectory
- System.DirectoryServices
- System.DirectoryServices.Protocols

Was ist falsch am PowerShell-Modul?

- Benötigt ADWS auf DCs
- Nutzt Port, der separat vom AD-Betrieb ist (ADWS = 9389)
- Muss installiert werden
- Initialisiert einen PSProvider beim Laden
- Generell langsam
- Kann gegen einen Timeout laufen *
- Single Result-Quirk
- Kann nur mit AD verwendet werden

- Basic AuthN kann verwendet werden, obwohl in LDAP abgeschaltet ☺
- Gut dokumentiert, einfach zu nutzen
- Findet das AD automatisch auf
- Verfolgt (normalerweise) die Referrals automatisch

Weniger bekannte AD module cmdlets

- (*über AuthN Policies, Central Access und andere AD-Features, die nicht so oft benutzt werden, sprechen wir heute mal nicht → wer Bock auf einen AD-Workshop hat, spreche mich an*)
- **Move-ADObject**
- **Sync-ADObject**
- **Get-ADReplicationAttributeMetadata**
  Get-ADReplicationPartnerMetadata
  Get-ADReplicationQueueOperation
  Get-ADreplicationConnection
  Get-ADreplicationFailure

ActiveDirectory Namespace

- **Reichhaltiges Objektmodell, das die Infrasturktur von AD / ADLDS komfortable abbilden kann:**
  - Schema
  - Forests, Domänen, Vertrauensstellungen
  - Domain Controller, Globale Kataloge
  - Sites, Subnets, Replikation
- **Nutzt LDAP, nicht ADWS**
- **Ist nicht zum Hantieren mit Objekten in einzelnen Partitionen geeignet!**
  - Dafür gibt es eine Brücke zum DS-Namespace: .GetDirectoryEntry()

DirectoryServices Namespace

- **Hat mit Objekten im Verzeichnis zu tun**
  - Streng genommen, nicht nur ActiveDirectory – muss aber AuthN annehmen ☺
- **Wichtigste Objektklassen, um die sich alles dreht:**
  - DirectoryEntry
  - DirectorySearcher
  - ActiveDirectorySecurity
- **Das DirectoryEntry-Objekt erhält man auch mit**
  - [ADSI]"LDAP://<DistinguishedName>"

Protocols Namespace

- **Low-level Kommunikations-API für LDAP**
  - Kann alle Eigenarten von LDAP abbilden → nicht-AD-Verzeichnisse, paginierte Suchen!
  - Ist extrem aufwendig, extrem mächtig und extrem schnell!
  - Kann im AD beispielsweise nach Eigenschaften des Security Descriptors mit dem SecurityDescriptorFlagControl filtern
  - Kann einen NetworkCredential-Parameter beim Verbindungsaufbau benutzen → PSCredential muss nicht im Code ausgepackt werden!
- **SDSP hat ein paar interessante Eigenarten → dazu später mehr ☺**
- **Wird in Windows PowerShell nicht automatisch geladen!**

```
[System.Reflection.Assembly]::LoadWithPartialName("System.DirectoryServices.Protocols")
```

Wer bin ich? Wo bin ich?

Wer oder was bin ich?

- **$env:**
  - USERNAME
  - USERDOMAIN
  - USERDNSDOMAIN
- **[Environment]::**
  - UserName
  - UserDomainName
- **Security namespaces:**
  - [System.Security.Principal.WindowsIdentity]::GetCurrent()
- **DirectoryServices namespaces:**
  - [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

Wo bin ich?

- **$env:**
  - COMPUTERNAME
- **[Environment]::**
  - MachineName
- **WMI:**
  - (Get-WMIObject Win32_ComputerSystem).Domain
- **DirectoryServices namespaces:**
  - [System.DirectoryServices.ActiveDirectory.Domain]::GetComputerDomain()

Portabilität sicherstellen

Heute hier, morgen dort…

Die Portabilität von AD-Skripten hat drei hauptsächliche Anwendungsfälle:

- **Skript wird mit einem Benutzer des zu untersuchenden Forests gestartet**
- **Skript wird mit einem Benutzer aus einem vertrauten Forest gestartet, der Benutzer hat aber die erforderlichen Rechte im Ziel-Forest**
- **Skript wird ohne Vertrauen gestartet (Workgroup / ganz fremder Forest)**

- **Daher:**
  - Nicht zur Eingabe der Zieldomäne zwingen (#1), diese jedoch vorsehen (#2 + #3)
  - Nicht zur Eingabe der Credentials zwingen (#1 + #2), diese jedoch vorsehen (#3)
  - Möglichst viele Informationen direkt aus der Umgebung besorgen

Berechtigungen

Mit AD-Berechtigungen operieren

- **Welchen Weg man auch immer nimmt, am Ende landen wir bei**
  - System.DirectoryServices.ActiveDirectorySecurity
- **Bei umfangreichen Untersuchungen (z.B.: ACL von jedem einzelnen Objekt) können Replikations-Metadaten schneller sein als direkte Abfrage.**

Cross-Domain-Abenteuer

Cross-Forest-Abenteuer

Krasse Spezialfälle

Anonymer Verzeichniszugriff

- **Mit AD sehr schwer bis unmöglich nachzustellen**
  - DAS IST KEIN NACHTEIL VON AD!
- **Es gibt aber Verzeichnisse, die einen „echten Anonymen“ Zugriff erlauben**
  - Z.B. HCL Notes LDAP
- **System.DirectoryServices.AuthenticationType**
  - Wert 0 → Simple Bind
  - Anonym gibt es nicht!
- **Mit Protocols kein Problem ☺**
  - System.DirectoryServices.Protocols.AuthType = 0 → Anonymous

Was ist mit Cross Platform?

System.DirectoryServices unter Linux/Mac

- **Wohl dem, der sich mit SDP auseinandergesetzt hat…**
- **…denn nur dieser Namespace ist derzeit unter Nicht-Windows verfügbar…**
- **…und kann dennoch Kerberos und NTLM ☺**

Vielen Dank!

Und jetzt – ab (nach Hause|in den Biergarten)!