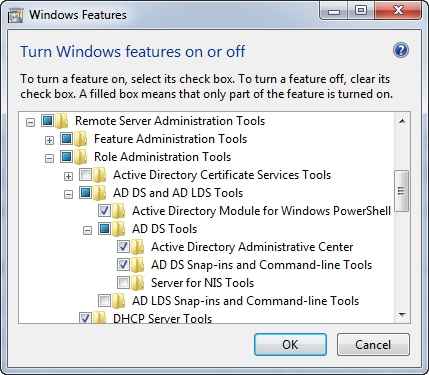
1. Сбросить пароль пользователя
2. Активировать и деактивировать учетные записи
3. Разблокировать учетную запись пользователя
4. Удалить учетную запись
5. Найти пустые группы
6. Добавить пользователей в группу
7. Вывести список членов группы
8. Найти устаревшие учетные записи компьютеров
9. Деактивировать учетную запись компьютера
10. Найти компьютеры по типу

#### Требования

Чтобы использовать PowerShell для управления AD, нужно соблюсти несколько требований. Я собираюсь продемонстрировать, как командлеты для AD работают на примере компьютера на Windows 7.

Чтобы использовать командлеты, контроллер домена у Вас должен быть уровня Windows Server 2008 R2, или же Вы можете скачать и установить Active Directory Management Gateway Service на наследуемых контроллерах домена (legacy DCs). Внимательно прочитайте документацию перед установкой; требуется перезагрузка КД.

На стороне клиента, скачайте и установите **Remote Server Administration Tools** (RSAT) либо для Windows 7, либо для Windows 8. В Windows 7, Вам необходимо будет открыть в *Панели управления (Control Panel)* раздел *Программы (Programs)* и выбрать *Включить или выключить функции Windows (Turn Windows Features On or Off)*. Найдите *Remote Server Administration Tools* и раскройте раздел *Role Administration Tools*. Выберите подходящие пункты для AD DS and AD LDS Tools, особенно обратите внимание на то, что должен быть выбран пункт *Active Directory Module for Windows PowerShell*, как показано на рисунке 1. (В Windows 8 все инструменты выбраны по умолчанию). Теперь мы готовы работать.

  
Рис.1 Включение AD DS и AD LDS Tools

Я вошел в систему под учетной записью с правами доменного администратора. Большинство командлетов, которые я буду показывать, позволят Вам уточнить альтернативные полномочия (credentials). В любом случае я рекомендую прочитать справку (**Get-Help**) и примеры, которые я буду демонстрировать ниже.

Начните сессию PowerShell и импортируйте модуль:

PS C:\> Import-Module ActiveDirectory

В результате импорта создается новый PSDrive, но мы не будем использовать его. Однако, Вы можете посмотреть, какие команды имеются в импортированном модуле.

PS C:\> get-command -module ActiveDirectory

Прелесть этих команд в том, что если я могу использовать команду для одного объекта AD, то ее можно использовать для 10, 100 и даже 1000. Посмотрим, как некоторые из этих командлетов работают.

#### Задача 1: Сброс пароля пользователя

Давайте начнем с типичной задачи: сброс пароля пользователя. Сделать это легко и просто можно через командлет **Set-ADAccountPassword**. Сложная часть заключается в том, что новый пароль должен быть уточнен как защищенная строка: фрагмент текста, который зашифрован и хранится в памяти на протяжении PowerShell сессии. Во-первых, создадим переменную с новым паролем:

PS C:\> $new=Read-Host "Enter the new password" -AsSecureString

Затем, введем новый пароль:

PS C:\>

Теперь мы можем извлечь учетную запись (использование **samAccountname** – лучший вариант) и задать новый пароль. Вот пример для пользователя Jack Frost:

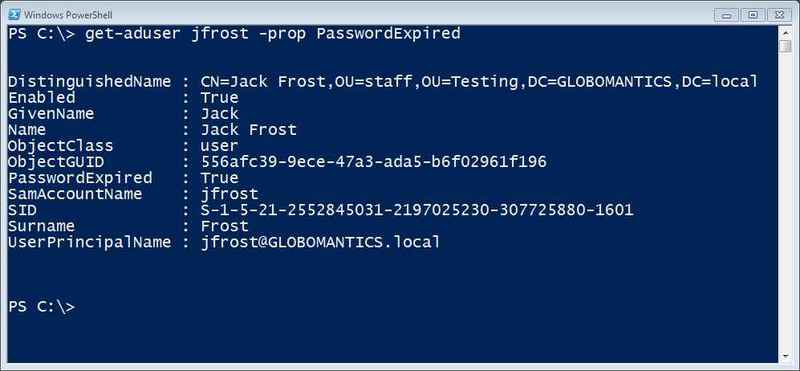
PS C:\> Set-ADAccountPassword jfrost -NewPassword $new

К сожалению, в случае с этим командлетом наблюдается баг: *-Passthru*, *-Whatif*, и *–Confirm* не работают. Если Вы предпочитаете короткий путь, попробуйте следующее:

PS C:\> Set-ADAccountPassword jfrost -NewPassword (ConvertTo-SecureString -AsPlainText -String "P@ssw0rd1z3" -force)

В итоге мне необходимо, чтобы Jack сменил пароль при следующем входе в систему, и я модифицирую учетную запись используя **Set-ADUser**.

PS C:\> Set-ADUser jfrost -ChangePasswordAtLogon $True

Результаты выполнения командлета не пишутся в консоль. Если это необходимо сделать, используйте **–True**. Но я могу узнать, успешно или нет прошла операция, произведя извлечения имени пользователя с помощью командлета **Get-ADUser** и уточнив свойство **PasswordExpired**, как показано на рисунке 2.  
   
  
Рис. 2. Результаты работы командлета Get-ADUser Cmdlet со свойством PasswordExpired  
   
Итог: сбросить пароль пользователя с помощью PowerShell совсем не сложно. Признаюсь, что сбросить пароль также просто через оснастку *Active Directory Users and Computers* консоли *Microsoft Management Console (MMC).* Но использование PowerShell подходит в том случае, если Вам необходимо делегировать задачу, Вы не хотите разворачивать вышеупомянутую оснастку или сбрасываете пароль в ходе большого автоматизированного ИТ-процесса.

#### Задача 2: Активировать и деактивировать учетные записи

А теперь давайте деактивируем учетную запись. Продолжим работать с Jack Frost. Этот код использует параметр **–Whatif**, который Вы можете встретить в других комадлетах, которые осуществляют изменения, чтобы проверить мою команду не запуская ее.

PS C:\> Disable-ADAccount jfrost -whatif What if: Performing operation "Set" on Target "CN=Jack Frost, OU=staff,OU=Testing,DC=GLOBOMANTICS,DC=local".

А теперь деактивируем по-настоящему:

PS C:\> Disable-ADAccount jfrost

А когда настанет время активировать учетную запись, какой командлет нам поможет?

PS C:\> Enable-ADAccount jfrost

Эти командлеты могут быть использованы в конвейерном выражении (pipelined expression), позволяя активировать или деактивировать столько учетных записей, сколько душе угодно. Например, этот код деактивирует все учетные записи в отделе продаж (Sales)

PS C:\> get-aduser -filter "department -eq 'sales'" | disable-adaccount

Конечно, писать фильтр для **Get-ADUser** довольно-таки сложно, но именно здесь использование параметра **–Whatif** вместе с командлетом **Disable-ADAccount** приходит на помощь.

#### Задача 3: Разблокировать учетную запись пользователя

Рассмотрим ситуацию, когда Jack заблокировал свою учетную запись, пытаясь ввести новый пароль. Вмест того, чтобы пытаться найти его учетную запись через GUI, процедуру разблокировки можно осуществить с помощью простой команды.

PS C:\> Unlock-ADAccount jfrost

Командлет также поддерживает параметры **-Whatif** и **-Confirm**.

#### Задача 4: Удалить учетную запись

Неважно, сколько пользователей Вы удаляете, — это просто осуществить с помощью командлета **Remove-ADUser**. Мне не хочется удалять Jack Frost, но если бы я захотел, то использовал бы такой код:

PS C:\> Remove-ADUser jfrost -whatif What if: Performing operation "Remove" on Target "CN=Jack Frost,OU=staff,OU=Testing,DC=GLOBOMANTICS,DC=local".

Или я могу ввести несколько пользователей и удалить их с помощью одной простой команды:

PS C:\> get-aduser -filter "enabled -eq 'false'" -property WhenChanged -SearchBase "OU=Employees, DC=Globomantics,DC=Local" | where {$\_.WhenChanged -le (Get-Date).AddDays(-180)} | Remove-ADuser -whatif

С помощью этой команды будут найдены и удалены все деактивованные учетные записи подразделения (OU) Employees, которые не менялись в течение 180 и более дней.

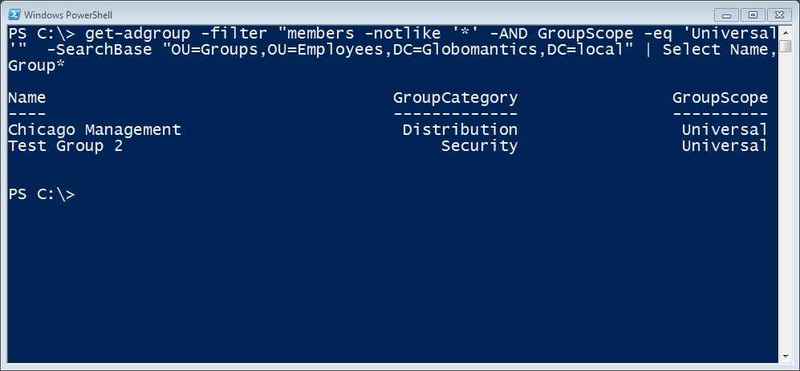
#### Задача 5: Поиск пустых групп

Управление группами – занятие бесконечное и неблагодарное. Существует множество способов найти пустые группы. Некоторые выражения могут работать лучше, чем другие, в зависимости от Вашей организации. Код, приведенный ниже, позволит найти все группы в домене, включая встроенные (built-in).

PS C:\> get-adgroup -filter \* | where {-Not ($\_ | get-adgroupmember)} | Select Name

Если у Вас есть группы с сотнями членов, тогда использование этой команды может занять много времени; **Get-ADGroupMember** проверяет каждую группу. Если Вы можете ограничить или настроить, это будет лучше.  
 Вот еще один подход:

PS C:\> get-adgroup -filter "members -notlike '\*' -AND GroupScope -eq 'Universal'" -SearchBase "OU=Groups,OU=Employees,DC=Globomantics, DC=local" | Select Name,Group\*

Эта команда находит все универсальные группы (Universal groups), которые не имеют членство в OU Groups и выводит некоторые из свойств. Результат приведен на рисунке 3.  
   
Рис. 3. Поиск и фильтрация универсальных групп

#### Задача 6: Добавление пользователей в группу

Давайте добавим Jack Frost в группу Chicago IT:

PS C:\> add-adgroupmember "chicago IT" -Members jfrost

Да, все так просто. Вы можете также легко добавлять сотни пользователей в группы, хотя, на мой взгляд, это слегка неудобно:

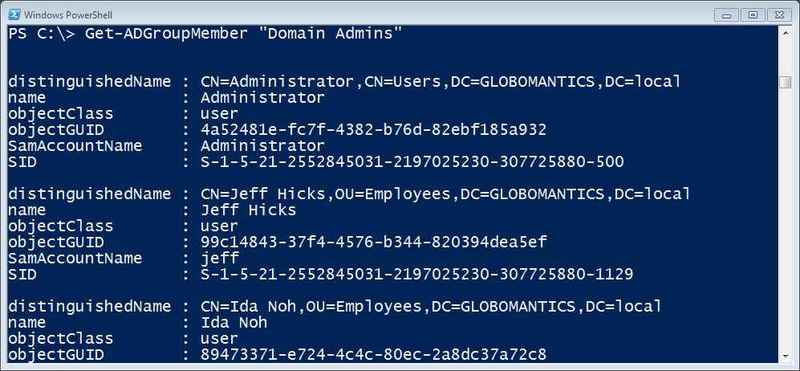
PS C:\> Add-ADGroupMember "Chicago Employees" -member (get-aduser -filter "city -eq 'Chicago'")

Я использовал вводное конвейерное выражение (parenthetical pipelined expression), чтобы найти всех пользователей, у которых имеется свойство City в Chicago. Код в скобках выполняется, и полученные объекты передаются в параметр –Member. Каждый пользовательский объект добавляется в группу Chicago Employees. Неважно, имеем ли мы дело с 5 или 5000 пользователей, обновление членства в группах занимает всего несколько секунд. Это выражение может также быть написано с использованием **ForEach-Object**, что может быть удобнее:

PS C:\> Get-ADUser -filter "city -eq 'Chicago'" | foreach {Add-ADGroupMember "Chicago Employees" -Member $\_}

Задача 7: Выводим список членов группы  
   
Вы возможно захотите узнать, кто находится в определенной группе. Например, Вы должны периодически узнавать, кто входит в группу доменных администраторов (Domain Admins):

PS C:\> Get-ADGroupMember "Domain Admins"

На рисунке 4 приведен результат.  
   
  
Рис. 4. Члены группы Domain Admins   
  
Командлет выводит объект AD для каждого члена группы. А что делать с вложенными группами? Моя группа Chicago All Users является коллекцией вложенных групп. Чтобы получить список всех учетных записей, я всего лишь должен использовать параметр **–Recursive**.

PS C:\> Get-ADGroupMember "Chicago All Users" -Recursive | Select DistinguishedName

Если Вы хотите пойти другим путем – найти, в каких группах пользователь состоит, — используйте свойство пользователя **MemberOf**:

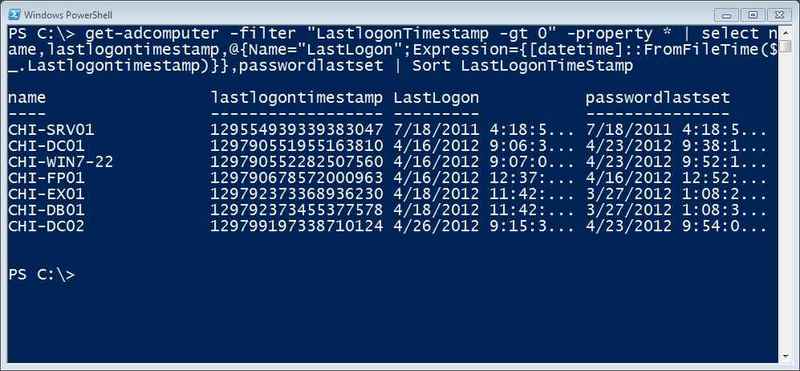
PS C:\> get-aduser jfrost -property Memberof | Select -ExpandProperty memberOf CN=NewTest,OU=Groups,OU=Employees, DC=GLOBOMANTICS,DC=local CN=Chicago Test,OU=Groups,OU=Employees, DC=GLOBOMANTICS,DC=local CN=Chicago IT,OU=Groups,OU=Employees, DC=GLOBOMANTICS,DC=local CN=Chicago Sales Users,OU=Groups,OU=Employees, DC=GLOBOMANTICS,DC=local

Я использовал параметр **-ExpandProperty**, чтобы вывести имена **MemberOf** как строки.

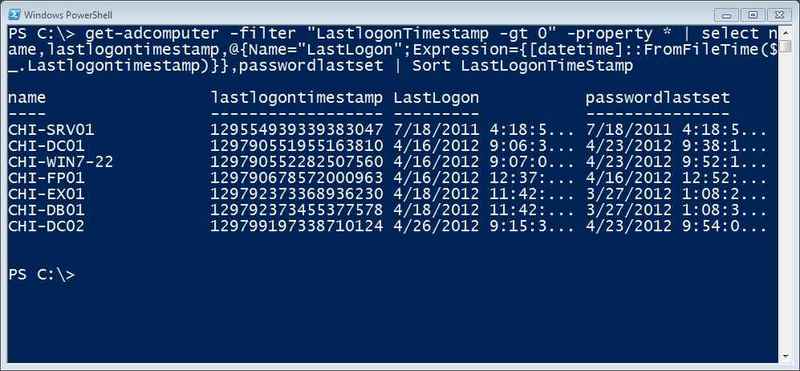
#### Задача 8: Найти устаревшие учетные записи компьютеров

Мне часто задают этот вопрос: “Как найти устаревшие учетные записи компьютеров?”. И я всегда отвечаю: “А что для вас является устаревшим?” Компании по-разному определяют то, когда учетная запись компьютера (или пользователя, неважно), признается устаревшей и не подлежит дальнейшему использованию. Что касается меня, то я обращаю внимание на те учетные записи, у которых пароли не менялись в течение определенного периода времени. Этот период для меня составляет 90 дней – если компьютер не сменил пароль вместе с доменом за этот период, скорее всего он находится оффлайн и является устаревшим. Используется командлет **Get-ADComputer**:

PS C:\> get-adcomputer -filter "Passwordlastset -lt '1/1/2012'" -properties \*| Select name,passwordlastset

Фильтр замечательно работает с жестким значением, но этот код будет обновляться для всех учетных записей компьютеров, которые не изменили своих паролей с 1 января 2012 года. Результаты приведены на рисунке 5.  
   
  
Рис. 5. Находим устаревшие учетные записи компьютеров   
  
Другой вариант: предположим, вы хотя бы на функциональном уровне домена Windows 2003. Поставьте фильтр по свойству **LastLogontimeStamp**. Это значение – число 100 наносекундных интервалов с 1 января, 1601 года, и храниться в GMT, поэтому работа с этим значением слегка сложно:

PS C:\> get-adcomputer -filter "LastlogonTimestamp -gt 0" -properties \* | select name,lastlogontimestamp, @{Name="LastLogon";Expression={[datetime]::FromFileTime ($\_.Lastlogontimestamp)}},passwordlastset | Sort LastLogonTimeStamp

Я взял на себя ответственность и добавил кастомное свойство, которое берет значение **LastLogontimeStamp** и конвертирует его в привычный формат. На рисунке 6 показан результат.  
   
  
Рис. 6. Конвертируем значение LastLogonTimeStamp в привычный формат   
  
Чтобы создать фильтр, мне необходимо конвертировать дату, например, 1 января 2012, в корректный формат. Конвертация осуществляется в FileTime:

PS C:\> $cutoff=(Get-Date "1/1/2012").ToFileTime() PS C:\> $cutoff 129698676000000000

Теперь я могу использовать эту переменную в фильтре для **Get-ADComputer**:

PS C:\> Get-ADComputer -Filter "(lastlogontimestamp -lt $cutoff) -or (lastlogontimestamp -notlike '\*')" -property \* | Select Name,LastlogonTimestamp,PasswordLastSet

Приведённый код находит те же самые компьютеры, что были показаны на рисунке 5.

#### Задача 9: Деактивировать учетную запись компьютера

Возможно, когда Вы найдете неактивные или устаревшие учетные записи, Вы захотите деактировать их. Сделать это довольно просто. Мы будем использовать тот же командлет, что использовали в работе с учетными записями пользователей. Вы можете уточнить его, использовав **samAccountname** учетной записи.

PS C:\> Disable-ADAccount -Identity "chi-srv01$" -whatif What if: Performing operation "Set" on Target "CN=CHI-SRV01, CN=Computers,DC=GLOBOMANTICS,DC=local".

Или же использовав конвейерное выражение:

PS C:\> get-adcomputer "chi-srv01" | Disable-ADAccount

Я также могу использовать мой код, чтобы найти устаревшие учетные записи и все их деактивировать:

PS C:\> get-adcomputer -filter "Passwordlastset -lt '1/1/2012'" -properties \*| Disable-ADAccount

#### Задача 10: Найти компьютеры по типу

Мне также часто задают вопрос, как найти учетные записи компьютеров по типу, например, серверы или рабочие станции. С вашей стороны это требует определенной креативности. В AD нет ничего такого, чтобы отличало сервер от клиента, разве что ОС. Если Ваш компьютер работает под Windows Server 2008, придется слегка сделать несколько дополнительных действий.  
 Для начала необходимо получить список ОС, а затем осуществляем фильтрацию учетных записей по имеющимся ОС.

PS C:\> Get-ADComputer -Filter \* -Properties OperatingSystem | Select OperatingSystem -unique | Sort OperatingSystem

Результаты показаны на рисунке 7.  
   
  
Рис. 7. Извлечение списка ОС  
   
Я хочу найти все компьютеры, на которых стоит серверная ОС:

PS C:\> Get-ADComputer -Filter "OperatingSystem -like '\*Server\*'" -properties OperatingSystem,OperatingSystem ServicePack | Select Name,Op\* | format-list

Результаты приведены на рисунке 8.  
   
  
  
Как и другими командлетами AD Get, Вы можете настроить поисковые параметры и ограничить запрос отдельными OU, если это необходимо. Все выражения, которые я показал, могут быть интегрированы в большие PowerShell выражения. Например, Вы можете сортировать, группировать, применять фильтры, экспортировать в CSV или создавать и отправлять на почту HTML отчеты – и все это из PowerShell! При этом Вам не придется писать ни единого скрипа.   
Вот Вам бонус: отчет о возрасте пароля пользователя (user password-age report), сохраненный в HTML файле:

PS C:\> Get-ADUser -Filter "Enabled -eq 'True' -AND PasswordNeverExpires -eq 'False'" -Properties PasswordLastSet,PasswordNeverExpires,PasswordExpired | Select DistinguishedName,Name,pass\*,@{Name="PasswordAge"; Expression={(Get-Date)-$\_.PasswordLastSet}} |sort PasswordAge -Descending | ConvertTo-Html -Title "Password Age Report" | Out-File c:\Work\pwage.htm

Хотя это выражение может выглядеть слегка пугающим, при минимальном знании PowerShell им легко воспользоваться. И остается лишь последний совет: как определить кастомное свойство под названием **PasswordAge**. Значение представляет собой промежуток между сегодняшним днем и свойством PasswordLastSet. Затем я сортирую результаты для моего нового свойства. На рисунке 9 показан выход для моего небольшого тестового домена.  
   
