Оглавление

[Введение 1](#_Toc516321715)

[1.Теоретическая глава 1](#_Toc516321716)

[1.1 Способы работы с Active Directory 2](#_Toc516321717)

[1.2 ADSI 3](#_Toc516321718)

[1.3 Провайдер AD 4](#_Toc516321719)

[1.4 Командлеты AD 6](#_Toc516321720)

[1.5 Управление Active Directory 7](#_Toc516321721)

[1.6 Работа с группами 11](#_Toc516321722)

[2. Аналитическая глава 14](#_Toc516321723)

[3. Проектная глава 20](#_Toc516321724)

[4. Технико-экономическое обоснование и расчет экономической эффективности предлагаемого проекта 26](#_Toc516321725)

[4.1 Расчет затрат на разработку и реализацию проекта 26](#_Toc516321726)

[4.2 Расчет текущих расходов в течение срока жизненного цикла проекта 28](#_Toc516321727)

[4.3 Расчет платежей в виде налогов и экономии налогов на прибыль за счет амортизации 30](#_Toc516321728)

[Заключение 32](#_Toc516321729)

[Список литературы 32](#_Toc516321730)

# Введение

Active Directory® - это технология, «внутри» которой многие администраторы проводят большую часть своего времени, выполняя повседневную административную работу – добавление новых пользователей, обновление объектов директории и .т.д. С внедрением в Windows Server® 2008 R2 командлетов, ориентированных на Active Directory, у администраторов появилась возможность экономить массу времени и сил, используя Windows PowerShell® для автоматизации выполнения многих повторяющихся задач, которые раньше отнимали львиную долю времени. Автоматизация также позволяет повысить безопасность и стабильность работы, так как влияние человеческого фактора, а значит, и количество ошибок сводится к минимуму.

В Windows Server 2008 R2 впервые появился модуль Active Directory для Windows PowerShell. Этот модуль установлен на всех контроллерах домена и является частью Remote Server Administration Toolkit (RSAT) для Windows® 7. Модуль содержит командлеты, которые позволяют осуществлять управление Active Directory из командной строки. Также эти командлеты обеспечивают поддержку технологии, которая отвечает за работу новой графической консоли Active Directory Administrative Center. Эти новые командлеты взаимодействуют с веб-сервисом, который является частью Active Directory в Windows Server 2008 R2. Этот же веб-сервис можно добавить в контроллеры доменов Windows Server 2008 и Windows Server 2003, загрузив его с официального сайта Microsoft. Найти его можно здесь:

http://go.microsoft.com/fwlink/?LinkId=193581

Нет необходимости устанавливать этот веб-сервис на все контроллеры домена. Вполне достаточно того, чтобы он был на контроллере домена того сайта, на котором находится административный центр. Точно так же, нет нужды устанавливать сами командлеты на ваши контроллеры домена. Командлеты нужны только на том клиентском компьютере, на котором вы хотите их использовать.

# 1.Теоретическая глава

## 1.1 Способы работы с Active Directory

Директория Active Directory является основой корпоративных сетей на базе Windows Server 2000, 2003 и 2008. Именно там хранятся все учетные записи пользователей, информация о группах, компьютерах сети, ящиках электронной почты и многом другом.

Всем этим богатством надо управлять, для чего предназначен соответствующий инструментарий, входящий в состав Windows Server, но именно PowerShell позволяет легко автоматизировать массовые действия, направленные на большое количество объектов.

Существует три основных способа работы с Active Directory в Windows PowerShell:

* с помощью интерфейса Active Directory Service Interfaces (ADSI) — этот способ является наиболее сложным, но работает в любой установке PowerShell и не требует дополнительных модулей. Он также наиболее близок к способу управления, который использовался в языке сценариев VBScript;
* с помощью провайдера Active Directory, входящего в расширения PowerShell, — этот способ позволяет подключить директорию в виде диска на вашем компьютере и перемещаться по ней с помощью соответствующих команд: dir, cd и т.д. Данный способ требует установки дополнительного модуля с сайта codeplex;
* с помощью командлетов управления Active Directory — это наиболее удобный способ манипулирования объектами директории, но он тоже требует дополнительной инсталляции соответствующих модулей.

## 1.2 ADSI

Active Directory Service Interfaces (ADSI) хорошо знаком всем, кто пытался писать сценарии на языке VBScript. В PowerShell этот интерфейс реализован с помощью так называемого адаптера. Указав в квадратных скобках название адаптера (ADSI) и путь к объекту в директории на языке LDAP-запроса (Lightweight Directory Access Protocol — протокол работы с директориями, который поддерживает и AD), мы получаем доступ к объекту из директории и можем дальше вызывать его методы.

Например, подсоединимся к одному из контейнеров директории и создадим в нем новую пользовательскую учетную запись.

$objOU=[ADSI]”LDAP://mydc:389/ou=CTO,dc=Employees,dc=testdomain,dc=local”

Итак, теперь у нас переменная $objOU содержит информацию о контейнере (имена переменных в PowerShell начинаются со значка доллара).

Вызовем метод *Create* и создадим в контейнере нового пользователя:

$objUser = $objOU.Create(“user”, “cn=Dmitry Sotnikov”)

Теперь мы можем устанавливать различные атрибуты:

$objUser.Put(«sAMAccountName”, «dsotnikov”)

И наконец, укажем директории, что эти изменения надо применить:

$objUser.SetInfo()

Преимуществами использования адаптера ADSI являются:

* его наличие в любой поставке PowerShell. Если у вас установлен PowerShell и есть директория, с которой вам надо работать, — вы имеете все, что вам надо;
* применение подхода, близкого к VBScript. Если у вас богатый опыт работы с директорией на языке сценариев VBScript или в приложениях .NET, вы сможете уверенно себя чувствовать, используя этот подход.

К сожалению, у метода есть и недостатки:

* сложность — это самый сложный способ работы с директорией. Писать путь к объекту в виде запроса LDAP нетривиально. Для любой работы с атрибутами требуется указание их внутренних имен, а значит, надо помнить, что атрибут, обозначающий город пользователя, называется не «City», а «l» и т.д.;
* громоздкость — как видно из примера, простейшая операция создания одной учетной записи занимает как минимум четыре строчки, включая служебные операции подсоединения к контейнеру и применения изменений. Таким образом, даже относительно простые операции становятся похожи на сложные сценарии.

## 1.3 Провайдер AD

PowerShell позволяет представлять различные системы в виде дополнительных дисков компьютера с помощью так называемых провайдеров. Например, в состав поставки PowerShell входит провайдер реестра и мы можем перемещаться по реестру с помощью знакомых и любимых всеми нами команд cd и dir (для любителей UNIX команда ls тоже поддерживается).

Провайдера Active Directory в составе PowerShell нет, но его можно установить, зайдя на сайт проекта расширений PowerShell — PowerShell Community Extensions: <http://www.codeplex.com/PowerShellCX>.

Это проект с открытым кодом, который добавляет большое количество команд в систему PowerShell, а кроме того, устанавливает провайдера AD.

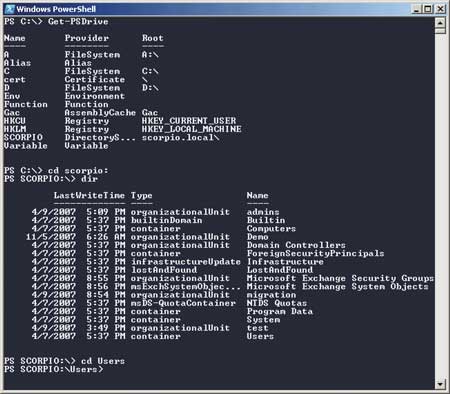
**

Рис. 1.1 Использование провайдера Active Directory

После установки расширений, набрав Get-PSDrive, мы видим, что к прежним дискам добавился диск текущей активной директории.

Теперь мы можем зайти в эту директорию, набрав cd и указав имя домена, а в любом контейнере использовать команду dir, чтобы увидеть его содержимое.

С объектами дальше можно работать так же, как это делалось с применением адаптера ADSI.

Кроме того, можно вызывать и другие привычные команды управления файлами (например, del).

К несомненным преимуществам использования провайдера можно отнести:

естественность представления структуры директории — директория AD по своей природе иерархична и похожа на файловую систему;

удобство нахождения объектов — применять cd и dir куда удобнее, чем составлять запрос на языке LDAP.

Из недостатков бросаются в глаза:

* сложность внесения изменений в объекты — провайдер помогает легко добраться до объекта, но чтобы что-либо поменять, нам опять приходится использовать все те же директорные объекты, что и в методе ADSI, а для этого надо оперировать на низком уровне служебных методов и атрибутов AD;
* необходимость дополнительной установки — провайдер не входит в состав PowerShell, и для его применения необходимо скачать и установить расширения PowerShell;
* третьестороннее происхождение — расширения PowerShell не являются продуктом компании Microsoft. Они созданы энтузиастами проекта. Вы вольны их использовать, но за технической поддержкой придется обращаться не в Microsoft, а на сайт проекта.

## 1.4 Командлеты AD

Кроме описанного выше провайдера, для работы с AD существует и набор командлетов (часто называемых также AD cmdlets или QAD cmdlets), доступный с сайта <http://www.quest.com/activeroles_server/arms.aspx>.

Командлеты состоят из стандартных глаголов операций (get-, set-, rename-, remove-, new-, move-, connect-) и существительных объектов с префиксом QAD (-QADUser, -QADGroup, -QADComputer, -QADObject).

Например, чтобы создать новую четную запись пользователя, понадобится выполнить такую команду:

New-QADUser -ParentContainer scorpio.local/Employees -Name ‘Dmitry Sotnikov’

Преимущества данного подхода таковы:

* простота — использование командлетов скрывает от вас сложность директории, ее схемы и внутренних атрибутов. Вы работаете с объектами директории на уровне понятных названий объектов (user, group, computer), их свойств (name, password, city, department) и действий над ними (get, set, remove, move, new);
* краткость и выразительность — как мы видели, большую часть действий с помощью командлетов можно выразить в виде простых и естественных однострочных операций.

Недостатками такого подхода можно считать:

* необходимость дополнительной установки — командлеты, как и провайдер, не входят в состав PowerShell, и для их использования необходимо скачать и установить соответствующую библиотеку;
* третьестороннее происхождение — командлеты для работы с AD не являются продуктом компании Microsoft. Они созданы партнером Microsoft — компанией Quest Software. Вы вольны их применять, но за технической поддержкой придется обращаться не в Microsoft, а на форумы по работе с Active Directory на сайте PowerGUI.org.

На наш взгляд, данные недостатки с лихвой компенсируются простотой и естественностью в использовании, так что практические примеры будут приведены с применением именно этого подхода.

## 1.5 Управление Active Directory

Давайте посмотрим, как PowerShell позволяет выполнять основные операции по работе с директорией AD:

* получение информации;
* изменение свойств;
* работа с группами;
* создание новых объектов;
* изменение структуры директории

Получение информации осуществляется в PowerShell с помощью командлетов с глаголом Get.

Например, чтобы получить список всех пользователей, наберем:

Get-QADUser

Для групп:

Get-QADGroup

Для записей компьютеров:

Get-QADComputer

Если вам нужны не все записи, а какие-то конкретные, вы можете выбрать именно их с помощью параметров команд.

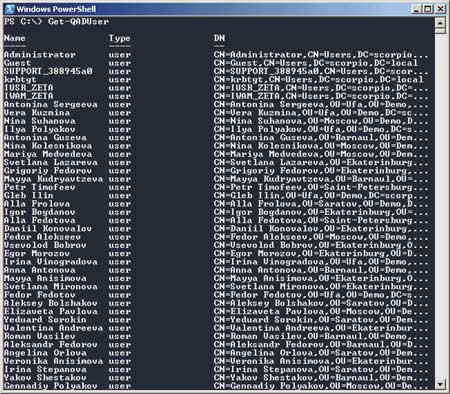
**

Рис. 1.2 Получение списка пользователей

Все группы из контейнера Users:

Get-QADGroup -SearchRoot scorpio.local/users

Все пользователи из отдела продаж московского офиса, чьи имена начинаются на букву A:

Get-QADUser -City Moscow -Department Sales -Name a\*

При этом вы можете сказать PowerShell’y, в каком виде вы хотите видеть получаемую информацию.

Таблица с именами, городами и подразделениями сотрудников:

Get-QADUser | Format-Table Name, City, Department

То же самое с сортировкой по городам:

Get-QADUser | Sort City | Format-Table DisplayName, City, Department

**

Рис. 1.3 Сортировка значений и выбор полей для вывода

Для списочного представления той же информации просто используем команду Format-List:

Get-QADUser | Format-List Name, City, Department

Экспортировать информацию в файл CSV (comma-separated values — значения через запятую):

Get-QADUser | Select Name, City, Department | Out-CSV users.csv

Создать отчет в формате HTML:

Get-QADUser | Select Name, City, Department | ConvertTo-HTML | Out-File users.html

Таким образом, одной строчкой простой команды PowerShell вы можете создавать сложные отчеты в удобном для вас формате.

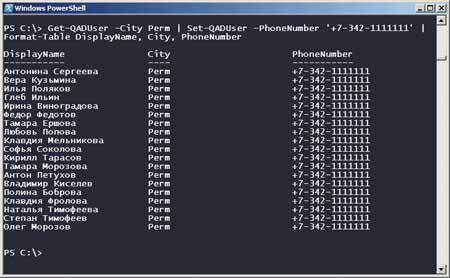
**

Рис. 1.4 PowerShell позволяет менять атрибуты множества   
записей одной командой

После того как мы освоились с получением информации из директории, пришла пора что-нибудь в ней поменять.

Свойствами объектов можно манипулировать с помощью команд Set-\*.

Например, поменяем мне телефон:

Set-QADUser ‘Dmitry Sotnikov’ -Phone ‘111-111-111’

Но, разумеется, куда более интересны массовые изменения. Для этого мы можем применять конвейер PowerShell, то есть получать список нужных нам объектов с помощью команд Get- и отправлять их в команду Set- для внесения изменений.

Например, наш пермский офис переехал в новое помещение. Возьмем всех пользователей Перми и присвоим им новый номер телефона:

Get-QADUser -City Perm | Set-QADUser -PhoneNumber ‘+7-342-1111111’

Для более сложных манипуляций можно использовать командлет ForEach-Object. Например, каждому пользователю присвоим описание, состоящее из его отдела и города:

Get-QADUser | ForEach-Object { Set-QADUser $\_ -Description (S\_.City + « « + $\_.Department) }

Переменная $\_ в данном примере обозначает текущий объект коллекции.

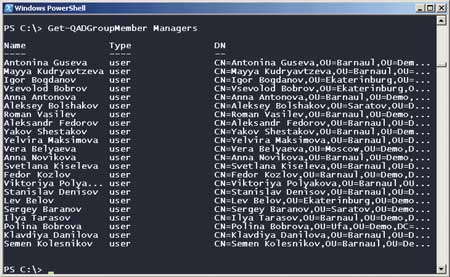
**

Рис1.5 PowerShell предоставляет возможности удобной работы   
с группами пользователей

## 1.6 Работа с группами

Работа с группами и членством в них — еще одна массовая операция, которую часто хочется автоматизировать. PowerShell предоставляет такую возможность.

Получение членов группы производится с помощью командлета Get-QADGroupMember:

Get-QADGroubMember Managers

Добавить объект в группу тоже несложно:

Add-QADGroupMember Scorpio\Managers -Member dsotnikov

Аналогично удаление из группы осуществляется с помощью командлеты Remove-QADGroupMember.

Но, разумеется, наиболее полезными являются массовые манипуляции. Добавим всех менеджеров в соответствующую группу:

Get-QADUser -Title Manager | Add-QADGroupMember Scorpio\Managers

Скопируем членство в группе:

Get-QADGroupMember Scorpio\Managers | Add-QADGroupMember Scorpio\Managers\_Copy

Используем фильтр, чтобы скопировать не всех членов группы, а только тех, кто отвечает определенному критерию (например, находится в нужном регионе):

Get-QADGroupMember Scorpio\Managers | where { $\_.City -eq ‘Ekaterinburg’} | Add-QADGroupMember Scorpio\Ekaterinburg\_Managers

Обратите внимание, как мы отфильтровали пользователей с помощью команды where и логического условия (логический оператор -eq — это оператор равенства в PowerShell, *от англ.*equals).

Создание объектов, как мы уже видели, осуществляется командами New:

New-QADUser -ParentContainer scorpio.local/Employees -Name ‘Dmitry Sotnikov’

New-QADGroup -ParentContainer scorpio.local/Employees -Name ‘Managers’ -Type Security -Scope Global

Вы можете установить и любые другие атрибуты в процессе создания записи:

New-QADUser -ParentContainer scorpio.local/Employees -Name ‘Dmitry Sotnikov’ -samAccountName dsotnikov -City ‘Saint-Petersburg’ -Password ‘P@ssword’

Чтобы активировать запись, просто отправьте ее по конвейеру в Enable-QADUser (не забудьте установить пароль — иначе операция не пройдет):

New-QADUser -ParentContainer scorpio.local/Employees -Name ‘Dmitry Sotnikov’ -Password ‘P@ssword’ | Enable-QADUser

Можно прочитать список пользователей и из файла. Например, если у нас есть файл, в котором через запятую перечислены атрибуты новых пользователей, то мы можем смело отправлять их на создание с помощью Import-CSV:

Import-CSV new\_users.csv | ForEach-Object { New-QADUser -ParentContainer scorpio.local/users -Name ($\_.Familia + ‘, ’ + $\_.Imya) -samAccountName ($\_.Imya[0] + $\_.Familia) -Department $\_.Department -Title $\_.Title}

Обратите внимание на то, что мы на лету составляем название учетной записи из фамилии и имени пользователя.

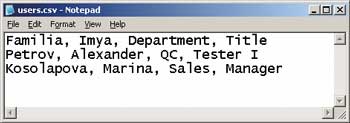
**

Рис.1.6 Пример использования файла импорта   
записей

И наконец, конечно же, можно управлять структурой директории.

Например, можно создавать новые контейнеры:

New-QADObject -type OrganizationUnit -ParentContainer scorpio.local -Name NewOU

и перемещать в них объекты по одному:

Move-QADObject MyServer -To scorpio.local/servers

или оптом:

Get-QADUser -Disabled | Move-QADObject -To scorpio.local/Disabled

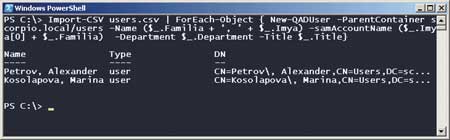
**

Рис. 1.7 Импортируем файл и создаем новые учетные записи

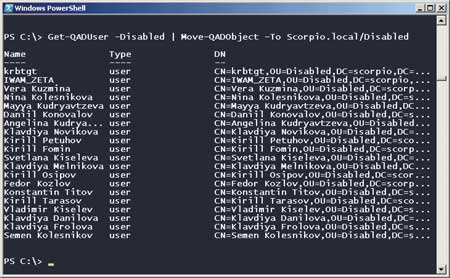
**

Рис. 1.8 Мы легко можем выбрать учетные записи, удовлетворяющие   
определенному критерию, и переместить их в другой контейнер

Мы рассмотрели только малую часть сценариев по управлению активной директорией. Чтобы получить полный перечень командлетов для AD, выполните команду:

Get-Command \*-QAD\*

Чтобы получить с правку по любой команде:

Get-Help Get-QADUser

Чтобы узнать, какие свойства есть у выдаваемого командой объекта:

Get-User | Get-Member

Возможности PowerShell практически безграничны, но при этом найти их достаточно легко.

# 2. Аналитическая глава

Для использования данных командлетов в вашей сессии PowerShell должен быть загружен специальный модуль взаимодействия с AD — **Active Directory Module for Windows PowerShell**. Данный модуль впервые был представлен в Windows Server 208 R2. В Windows Server 2012 и выше этот модуль включен по умолчанию. На клиентских компьютерах его можно установить и включить в качестве одного из компонентов RSAT. Проверить, загружен ли модуль можно так:

Get-Module -Listavailable

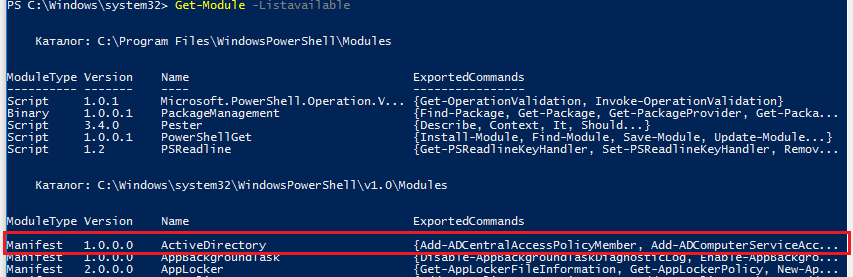


Рис.2.1 Проверка загрузки модуля ActiveDirectory

Как вы видите, модуль ActiveDirectory загружен. Если нет – импортируйте его командой:

Import-Module activedirectory

Полный список команд модуля можно получить так:

Get-Command -Module ActiveDirectory

В модуле всего доступно 147 командлетов, из которых с группами могут работать 11.

Get-Command -Module ActiveDirectory -Name "\*Group\*"

Вот их список:

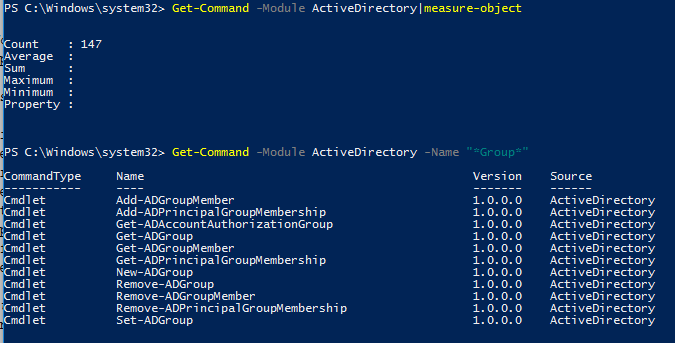


Рис.2.1 Вывод командлетов Group

* Add-ADGroupMember
* Add-ADPrincipalGroupMembership
* Get-ADAccountAuthorizationGroup
* Get-ADGroup
* Get-ADGroupMember
* Get-ADPrincipalGroupMembership
* New-ADGroup
* Remove-ADGroup
* Remove-ADGroupMember
* Remove-ADPrincipalGroupMembership
* Set-ADGroup

New-ADGroup – создаем новую группу AD

Создадим новую группу в указанном контейнере (OU) Active Directory с помощью команды **New-ADGroup**:

New-ADGroup "TestADGroup" -path 'OU=Groups,OU=Moscow,DC=corp,dc=winitpro,DC=ru' -GroupScope Global -PassThru –Verbose

С помощью атрибута **Description** можно задать описание группы, а с помощью **DisplayName** изменить отображаемое имя.

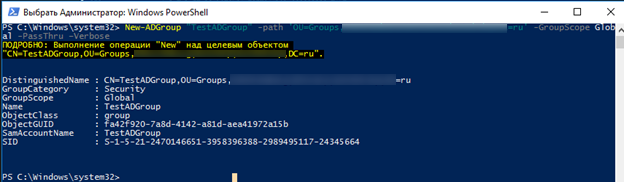


Рис.2.3 Проверка загрузки модуля ActiveDirectory

Параметром **GroupScope** можно задать один из следующих типов групп:

* 0 = DomainLocal
* 1 = Global
* 2 = Universal

Создать группу распространения можно так:

New-ADGroup "TestADGroup-Distr" -path 'OU=Groups,OU=Moscow,DC=corp,dc=winitpro,DC=ru' -GroupCategory Distribution -GroupScope Global -PassThru –Verbose

Add-AdGroupMember – добавить пользователей в группу AD

Добавить пользователей в группу Active Directory можно с помощью командлета Add-**AdGroupMember**. Добавим в новую группу двух пользователей:

Add-AdGroupMember -Identity TestADGroup -Members user1, user2

Если список пользователей, которых нужно добавить в группу довольно большой, можно сохранить список учетных записей в CSV файл, затем импортировать данный файл и добавить каждого пользователя в группу.

Формат CSV файла такой (список пользователей по одному в строке, имя столбца – users)

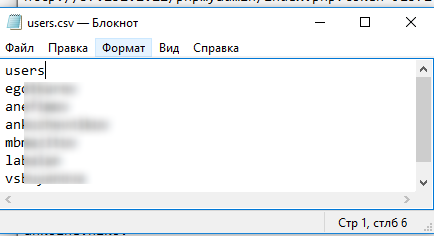


Рис.2.4 Файл users.csv

Import-CSV .\users.csv -Header users | ForEach-Object {Add-AdGroupMember -Identity ‘TestADGroup’ -members $\_.users}

Чтобы получить всех членов одной группы (groupA) и добавить их в другую группу (groupB), воспользуйтесь такой командой:

Get-ADGroupMember “GroupA” | Get-ADUser | ForEach-Object {Add-ADGroupMember -Identity “Group-B” -Members $\_}

В том случае, если нужно скопировать в новую группу и членов всех вложенных групп (рекурсивно), нужно воспользоваться такой командой:

Get-ADGroupMember -Identity “GroupA” -Recursive | Get-ADUser | ForEach-Object {Add-ADGroupMember -Identity “GroupB” -Members $\_}

Remove-ADGroupMember – удалить пользователей из группы

Для удаления пользователей из группы AD нужно использовать командует Remove-ADGroupMember. Удалим из группы двух пользователей:

Remove-ADGroupMember -Identity TestADGroup -Members user1, user2

Подтвердите удаление пользователей из группы:

Если нужно удалить из группы пользователей по списку из CSV файла, воспользуйтесь такой командой:

Import-CSV .\users.csv -Header users | ForEach-Object {Remove-ADGroupMember -Identity ‘TestADGroup’ -members $\_.users}

Get-ADGroup – получить информацию о группе AD

Получить информацию о группе поможет командлет **Get-ADGroup**:

Get-ADGroup 'TestADGroup'

Даная команда выводит информацию об основных атрибутах группы (DN, тип группы, имя, SID). Чтобы вывести значение всех атрибутов группы домена AD, выполните такую команду:

Get-ADGroup 'TestADGroup' -properties \*

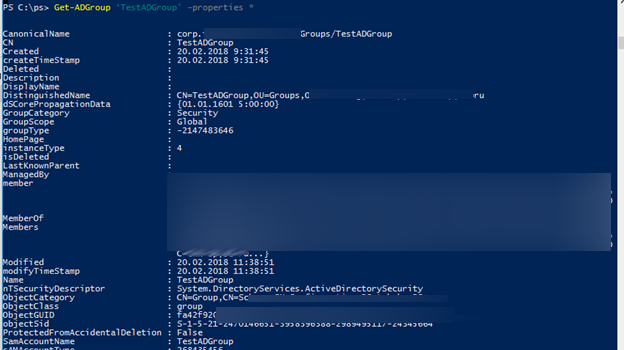


Рис.2.5 Вывод атрибутов группы

Как вы видите, теперь стали отображаться такие атрибуты, как время создания и модификации группы, описание и т.д.

С помощью командлета Get-ADGroup можно найти все интересующие вас группы по определенному шаблону. Например, нужно найти все группы AD, в имени которых содержится фраза ***admins*** :  
  
Get-ADGroup -LDAPFilter “(name=\*admins\*)” | Format-Table

Get-ADGroupMember – вывести список пользователей группы AD

Вывести на экран список пользователей группы:

Get-ADGroupMember 'TestADGroup'

Чтобы оставить в результатах только имена пользователей, выполните:

Get-ADGroupMember 'TestADGroup'| ft name

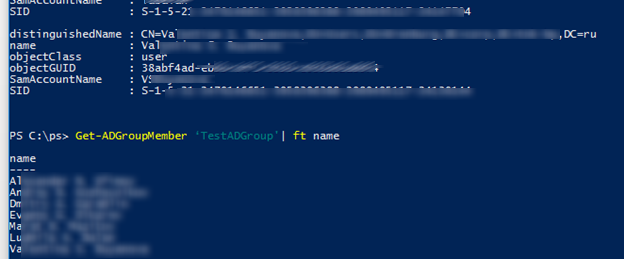


Рис.2.6 Вывод участников группы

Если в данную группу включены другие группы домена, чтобы вывести полный список членов, в том числе всех вложенных групп, воспользуйтесь параметром **Recursive**.

Get-ADGroupMember ‘server-admins' -recursive| ft name

Чтобы выгрузить список учетных записей, состоящих в определённой группе в CSV файл (для дальнейшего использования в Excel), выполните такую команду:

Get-ADGroupMember ‘server-admins' -recursive| ft samaccountname| Out-File c:\ps\admins.csv

Чтобы добавить в текстовый файл данные учетных записей пользователей в AD, воспользуемся командлетом [Get-ADUser](http://winitpro.ru/index.php/2015/05/21/powershell-get-aduser-poluchenie-dannih-o-polzovatelyax-active-directory/). Например, помимо учетной записи нужно вывести должность и телефон пользователя группы:

Get-ADGroupMember -Identity ’server-admins’ -recursive| foreach { Get-ADUser $\_ -properties title, OfficePhone|Select-Object title, OfficePhone }

Посчитать количество пользователей в группе можно так:

(Get-ADGroupMember -Identity 'domain admins').Count

Оказалось, что в группе «domain admins» у нас состоит 7 учетных записей администраторов.

Чтобы найти список пустых групп в определенном OU, воспользуйтесь такой командой:

Get-ADGroup -Filter \* -Properties Members -searchbase “OU=Moscow,DC=corp,dc=winitpro,DC=ru”  | where {-not $\_.members} | select Name

# 3. Проектная глава

Для успешной функциональности проектируемого скрипта сперва нужно установить sp1 (<https://www.microsoft.com/ru-ru/download/details.aspx?id=5842>) на сервер, если еще не установлен. Далее .net 4.5 (dotNetFX45\_Full\_setup). После чего нужно обновить powershell до версии 5.1 (в powershell запущенном от имени администратора выполняеем Set-ExecutionPolicy bypass, и запускаем скрипт установки из архива powershell51.zip). При запуске скрипта poweshell должен быть запущен от имени администратора При запуске без ключа, будут добаляться и обновляться пользователи. При запуске с ключом -del будут удалться пользователи. Разработанный скрипт приведен ниже:

import-module activedirectory

#переменные:

#путь до CSV файла

$pathToCSV=".\users.csv"

#пароль который будет задан новым пользователям

$defpass="As12345^"

#указываем наш домен

$domain="@matrix.ru"

#функция транслитерации

function global:Translit {

    param([string]$inString)

    $Translit = @{

    [char]'а' = "a"

    [char]'А' = "a"

    [char]'б' = "b"

    [char]'Б' = "b"

    [char]'в' = "v"

    [char]'В' = "v"

    [char]'г' = "g"

    [char]'Г' = "g"

    [char]'д' = "d"

    [char]'Д' = "d"

    [char]'е' = "e"

    [char]'Е' = "e"

    [char]'ё' = "yo"

    [char]'Ё' = "eo"

    [char]'ж' = "zh"

    [char]'Ж' = "zh"

    [char]'з' = "z"

    [char]'З' = "z"

    [char]'и' = "i"

    [char]'И' = "i"

    [char]'й' = "j"

    [char]'Й' = "j"

    [char]'к' = "k"

    [char]'К' = "k"

    [char]'л' = "l"

    [char]'Л' = "l"

    [char]'м' = "m"

    [char]'М' = "m"

    [char]'н' = "n"

    [char]'Н' = "n"

    [char]'о' = "o"

    [char]'О' = "o"

    [char]'п' = "p"

    [char]'П' = "p"

    [char]'р' = "r"

    [char]'Р' = "r"

    [char]'с' = "s"

    [char]'С' = "s"

    [char]'т' = "t"

    [char]'Т' = "t"

    [char]'у' = "u"

    [char]'У' = "u"

    [char]'ф' = "f"

    [char]'Ф' = "f"

    [char]'х' = "h"

    [char]'Х' = "h"

    [char]'ц' = "ts"

    [char]'Ц' = "ts"

    [char]'ч' = "ch"

    [char]'Ч' = "ch"

    [char]'ш' = "sh"

    [char]'Ш' = "sh"

    [char]'щ' = "sch"

    [char]'Щ' = "sch"

    [char]'ъ' = ""

    [char]'Ъ' = ""

    [char]'ы' = "y"

    [char]'Ы' = "y"

    [char]'ь' = ""

    [char]'Ь' = ""

    [char]'э' = "e"

    [char]'Э' = "e"

    [char]'ю' = "yu"

    [char]'Ю' = "yu"

    [char]'я' = "ya"

    [char]'Я' = "ya"

    }

    $outCHR=""

    foreach ($CHR in $inCHR = $inString.ToCharArray())

        {

        if ($Translit[$CHR] -cne $Null )

            {$outCHR += $Translit[$CHR]}

        else

            {$outCHR += $CHR}

        }

    Write-Output $outCHR}

#импортируем csv файл в переменную

$csv=import-Csv $pathToCSV -Encoding OEM -Delimiter ';'

#разбираем переменную

foreach ($user in $csv)

    {

        #заносим в переменные значения из csv файла

        $fio="$($user.ФИО)"

        $surname=$fio.split(' ')[0]

        $name=$fio.split(' ')[1]

        $sname=$fio.split(' ')[2]

        $dolzhnost="$($user.должность)"

        $depart="$($user.отдел)"

        $room="$($user.'номер комнаты')"

        $phone="$($user.'номер телефона')"

        $mail="$($user.'электронная почта')"

        $id=$($user.'идентификатор')

        #переводим в транслит имя и фамилию

        $transName=Translit($name)

        $transSurname=Translit($surname)

        #отчищаем первые буквы имени

        $shortName=""

        #добаляем буквы к переменной shortname (переменная для создания логина)

        for ($i=1; $i -lt $transName.length; $i++)

        {

            #в зависимости от числа проходов цикла, добавляем i букв

            $shortName=$transName.substring(0,$i)

            #добавляем буквы имени к фамилии

            $userName=$shortName+$transSurname

            try

            {

                #проеряем, есть ли пользователь

                $user=Get-ADUser "$userName"

            }

            catch

            {

                $user=$false

            }

            #если пользователь существует

            if ($user)

            {

                #получаем id из AD

                $IDinAD=Get-ADUser $userName -Properties comment | select comment | ft -HideTableHeaders | out-string

                #если номер страхового из AD совпал с номером из csv

                if ($IDinAD -match $id)

                {

                    #если запутили скрипт без аргументов

                    if ($args[0] -eq "" -or !$args[0] )

                    {

                        #обновляем данные пользователя

                        Set-ADUser -Identity "$userName" -Surname "$surname" -DisplayName "$surname $name $sname" `

                        -OfficePhone "$phone" -EmailAddress "$mail" -Department "$depart" -Title "$dolzhnost" `

                        -UserPrincipalName "$userName$domain" -GivenName "$name" -Office "$room" -enabled $true -SamAccountName "$userName"

                        #прерываем цикл

                        break

                    }

                    #если запустили скрипт с аргументом -del

                    if ($args[0] -eq "-del")

                    {

                        #удаляем пользователя

                         Remove-ADUser -Identity $userName -Confirm:$false

                    }

                }

                #если id не совпадают, и найдено имя пользователя, идем к следующему шагу цикла

                else

                {

                }

            }

            #если пользователя не существует

            else

            {

                #и запустили без аргументов

                if ($args[0] -eq "" -or !$args[0])

                {

                    try

                    {

                        $users=get-aduser -Filter "\*" -Properties comment | select comment, name

                    }

                    catch

                    {

                        $users=$false

                    }

                    if ($users)

                    {

                        foreach ($user in $users)

                        {

                            #если у какого то пользователя есть id из csv, обновляем его

                            if ($user.comment -match $id)

                            {

                                $uname=$user.name.toString()

                                $distName=Get-ADObject -Filter 'name -eq $uname'

                                Set-ADUser -Identity "$uname" -Surname "$surname" -DisplayName "$surname $name $sname" `

                                -OfficePhone "$phone" -EmailAddress "$mail" -Department "$depart" -Title "$dolzhnost" `

                                -UserPrincipalName "$userName$domain" -GivenName "$name" -Office "$room" -enabled $true `

                                -SamAccountName "$userName"

                                Rename-ADObject $distName.DistinguishedName -NewName $userName

                            }

                        }

                    }

                        try

                        {

                        #добавляем пользователя и прерываем цикл

                        New-ADUser -Name "$userName" -Surname "$surname" -DisplayName "$surname $name $sname" `

                        -OfficePhone "$phone" -EmailAddress "$mail" -Department "$depart" -Title "$dolzhnost" `

                        -UserPrincipalName "$userName$domain" -GivenName "$name" -Office "$room" -OtherAttributes @{comment="$id"} `

                        -AccountPassword (ConvertTo-SecureString -AsPlainText "$defpass" -force) -enabled $true `

                        -ChangePasswordAtLogon $true -SamAccountName "$userName" -erroraction 'silentlycontinue'

                        }

                        catch

                        {

                        }

                        break

                }

            }

        }

    }

# 4**. Технико-экономическое обоснование и расчет экономической эффективности предлагаемого проекта**

Совокупная стоимость владения (ССВ) является суммой затрат на инвестиционную часть проекта внедрения организационно-технических средств защиты информации, текущих расходов в течение срока жизненного цикла проекта и платежей в виде налогов и других отчислений, связанных с реализацией проекта.

ССВ = P АПК + P ТЕК + P Пл (1)

где P АПК - совокупная стоимость затрат на разработку и реализацию организационно-технической системы защиты информации; PТЕК - совокупная стоимость текущих расходов в течение срока жизненного цикла проекта;PПл - платежи в виде налогов и других отчислений, связанных с реализацией проекта.

## 4.1 Расчет затрат на разработку и реализацию проекта

Одним из этапов оценки экономической эффективности проекта является оценка затрат на приобретение оборудования, разработку или приобретение программ, средств информационного обеспечения, подготовку помещений, движение и обучение персонала.

Затраты по заработной плате персонала Рзп при реализации комплексной системы защиты информации рассчитываются по следующей формуле (2)

 (2)

Где:

Зl - среднемесячная основная заработная плата работника l, руб.;

Tl - среднемесячные трудовые затраты работника на выполнение исследуемой операции, чел./час;

 - ставка страховых взносов (=34%);

 - дополнительная заработная плата (премия);

Q - среднее количество рабочих часов в месяце, час.

Перечень сотрудников ОАО «Кузнецкий металлургический комбинат», занятых в обеспечении информационной безопасности представлен в таблице 3.1.

Таблица 3.1 Сотрудники, обеспечивающие разработку и внедрение

|  |  |  |
| --- | --- | --- |
| Название должности | Количество штатных единиц | Заработная плата, руб |
| Разработчик | 1 | 40000 |
| Системный аминистратор | 1 | 40000 |

В нашем случае затраты по заработной плате персонала, работающего с составлением различных положений и регламентов исходя из трудоемкости ведения данных работ составят:

Рзп= (40000\*58+40000\*116)\*1,34/176 = 49032 руб.

Затраты на настройку программных средств защиты и затраты на проведение инженерно-технических работ по установке средств защиты исходя из трудоемкости ведения данных работ составят:

РИТ-персонала = (10000\*60\*1,34)/176= 4569 руб.

Руст = = 46410 руб.

Обучение персонала работе с предложенной комплексной системой защиты информации не требуется.

Инвестиционный план проекта сведен в таблицу 5.2.

Таблица 3.2 Инвестиционный план проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Статьи расходов | Денежные затраты в тыс. руб. | | | |
|  | 0 | 1 | 2 | 3 |
| Расходы, связанные с организационно-правовым направлением защиты | | | | |
| Затраты на разработку положений и регламентов | 49 |  |  |  |
| Расходы, связанные с программно-аппаратным направлением защиты | | | | |
| Затраты на разработку или/и приобретение программных средств | 99,4 | 0,9 | 0,9 | 0,9 |
| Затраты, связанные с обслуживанием и настройкой программных средств защиты | 4,6 |  |  |  |
| Расходы, связанные с инженерно-техническим направлением защиты | | | | |
| Затраты на приобретение средств инженерно-технической защиты | 51,5 |  |  |  |
| Затраты на установку и настройку инженерно-технических средств защиты | 46,4 |  |  |  |
| Итого за проект | 250,9 | 251,8 | 252,7 | 253,6 |

## 4.2 Расчет текущих расходов в течение срока жизненного цикла проекта

Данный этап экономического обоснования проектных решений в области информационной безопасности связан с определением состава и объёма затрат в периоды жизненного цикла ИТ, соответствующие процессам эксплуатации спроектированного комплекса - текущих затрат Ртек за каждый из периодов m.

В рамках данной работы предусматриваются следующие текущие затраты:

* Затраты на оплату труда управленческого персонала;
* Затраты на связь, литературу, канцтовары, связанные с поддержкой процессов информационной безопасности;
* Затраты, связанные с обслуживанием и настройкой программных средств защиты;
* Затраты на оплату расхода электроэнергии на инженерно-технический комплекс;
* Затраты на проведение инженерно-технических работ по установлению средств защиты.

Затраты, связанные с обслуживанием и настройкой программных средств защиты рассчитываются по формуле (3):

 (3)

Где  - расходы по оплате труда ИТ-персонала, занятого настройкой и обслуживанием программных средств защиты, которые рассчитываются по формуле (2), а  - материальные затраты, связанные с обслуживанием и восстановлением системы исходя из трудоемкости ведения данных работ.

В нашем случае  = (10000\*60\*1,34)/176= 4569 руб.

Затраты на оплату расхода электроэнергии на комплекс приобретенных средств рассчитываются по формуле (4):

 (4)

Где Мпотр - максимальная потребляемая мощность оборудования g, Кисп - коэффициент использования оборудования g по мощности, Сэл - стоимость 1 кВт/час электроэнергии, Тg - время работы оборудования g.

Максимальная потребляемая мощность оборудования, задействованного в программно-аппаратной и инженерно-технической системах защиты информации представлена в таблице 3.3.

Таблица 3.3 Установленная мощность оборудования, задействованного в программно-аппаратной системе защиты информации

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Установленная мощность, Вт. | Количество, шт. | Итого, Вт. |
| Время работы - 24 часа, 365 дней в году | | | |
| Cisco 1921 Integrated Services Router | 13 | 1 | 13 |
| ППК «Сигнал-20» | 18 | 1 | 18 |
| Магнитоконтактный охранный извещатель ИО-102-6 (СМК-6) | 10 | 6 | 60 |
| Извещатель охранный звуковой GBD-2 | 12 | 6 | 72 |
| Источник бесперебойного электропитания «Скат-1200» | 34 | 2 | 68 |
| Оповещатель совмещенный «Свирель-2» | 14 | 1 | 14 |
| Монитор Ai-ML247N | 42 | 1 | 42 |
| Миниатюрная видеокамера KT&C KPC-190SP4 | 18 | 5 | 90 |
| Видеорегистратор «PVDR-1654» | 3 | 1 | 3 |
| Итого: |  |  | 380 |

Затраты на оплату расхода электроэнергии на инженерно-технический комплекс в год составят

Эн = 0,380\*1\*5,2\*24\*365 = 17310 руб.

Текущие затраты на обеспечение процессов ИБ сведены в таблицу 3.4.

## 4.3 Расчет платежей в виде налогов и экономии налогов на прибыль за счет амортизации

К налогам и другим отчислениям относится налог на имущество организации. Сумма платежей за период B рассчитывается по формуле (5):

 (5)

где SостBk - остаточная стоимость k -ого вида имущественной номенклатуры на начало периода, L - ставка налога на имущество (L=2,2%), N - количество имущественных номенклатур; Ln- ставка налога на прибыль, А- амортизационные отчисления. Остаточная стоимость имущественной номенклатуры на начало периода i рассчитывается по формуле (7):

 (6)

где На - норма амортизации.

Норма амортизации определяется по формуле (8):

 (7)

где: Sлик - ликвидационная стоимость имущественной номенклатуры; Тэ срок эксплуатации.

Норма амортизации для имущества ОАО «Кузнецкий металлургический комбинат» представлена в таблице 3.5.

Пользуясь формулой (5) была определена сумма платежей за три года. Результаты приведены в таблице 3.7.

Таблица 3.4 Сумма платежей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование имущества | Сумма отчислений, тыс. руб. | | | |
|  | 1 | 2 | 3 |  |
| Все виды | 1,25 | 1,06 | 0,94 | 3,25 |
| Итого | 3,25 | | | |

Используя формулу (1) рассчитаем совокупную стоимость владения

ССВ=253,6+3\*(22,6+3,4+17,3)+3,25=386,85 тыс. руб.

# Заключение

Windows PowerShell — новая командная строка и язык сценариев от компании Microsoft. PowerShell является компонентом Windows Server 2008 (надо только выбрать его в Server Manager) и доступна для загрузки со странички www.microsoft.com/powershell для Windows XP, Windows Server 2003 и Windows Vista.

Итак, команды PowerShell называются командлетами (cmdlet) и состоят из глагола (например, get, set, new, remove, move, connect) и существительного в единственном числе, описывающего объект действия. Между ними ставится дефис. Получается что-то вроде: get-process, stop-service и т.п.

Команды, как правило, связываются конвейером, обозначаемым вертикальной чертой (|). Этот знак означает, что вся коллекция объектов из предыдущей команды передается на вход следующей.

Такая объектная ориентированность очень удобна, поскольку позволяет легко оперировать объектами и связывать команды вместе. В этой статье мы расскажем, как подобный подход облегчает управление корпоративной директорией на базе Active Directory.

# Список литературы

1. Payette B. Windows PowerShell in Action. — Manning Publications Co, 2007.

2. Holmes L. Windows PowerShell Cookbook. — O'Reilly, 2007.

3. Wilson E. Windows PowerShell Step-by-Step. — Microsoft Press, 2007.

4. Watt A. Professional Windows PowerShell (Programmer to Programmer). —

Wrox Press, 2007.

5. Kopczynski T. Windows PowerShell Unleashed. — SAMS Publishing, 2007.

6. Jones D., Hicks J. Windows PowerShell: TFM. — Sapien, 2006.

7. Cookey-Gam J., Keane B., Rosen J., Runyon J., Stidley J. Professional Windows

PowerShell for Exchange Server 2007 Service Pack 1 (Programmer to

Programmer). — Wrox Press, 2008.

8. Koch F. Windows PowerShell. — (Электронная версия книги доступна для

бесплатного скачивания по адресу http://blogs.technet.com/chitprode/archive/2007/05/10/english-version-of-windows-powershell-coursebook-available-for-download.aspx).

9. Koch F. Administrative tasks using PowerShell. — (Электронная версия книги

доступна для бесплатного скачивания по адресу http://blogs.technet.com/

chitpro-de/archive/2008/02/28/fre