****

**­**

**Министр науки и высшего образования Российской̆**

**Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

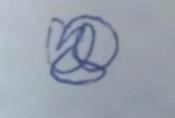
Рабочий протокол и отчет по лабораторной работе №4

Развертывание в середе Windows Server сетевых инфраструктурных сервисов на примере DHCP.

**Выполнил(а) студент группы** № **M33091**

Сидорцов Владимир Сергеевич

**Подпись:**



**Проверил:**

Санкт-Петербург

2022г.

1. **Цель работы.**

Получить представление и практические навыки работы по развертыванию и управлению сетевыми инфраструктурными сервисами на примере DHCP и IPAM в середе Windows Server, освоить основные понятия, связанные с работой DHCP сервера.

1. **Задачи, решаемые при выполнении работы.**

* .

1. **Объект исследования.**

* Windows Server 2012 r2
* Active Directory

1. **Исходные данные.**

●

1. **Окончательные результаты.**

Часть 2:

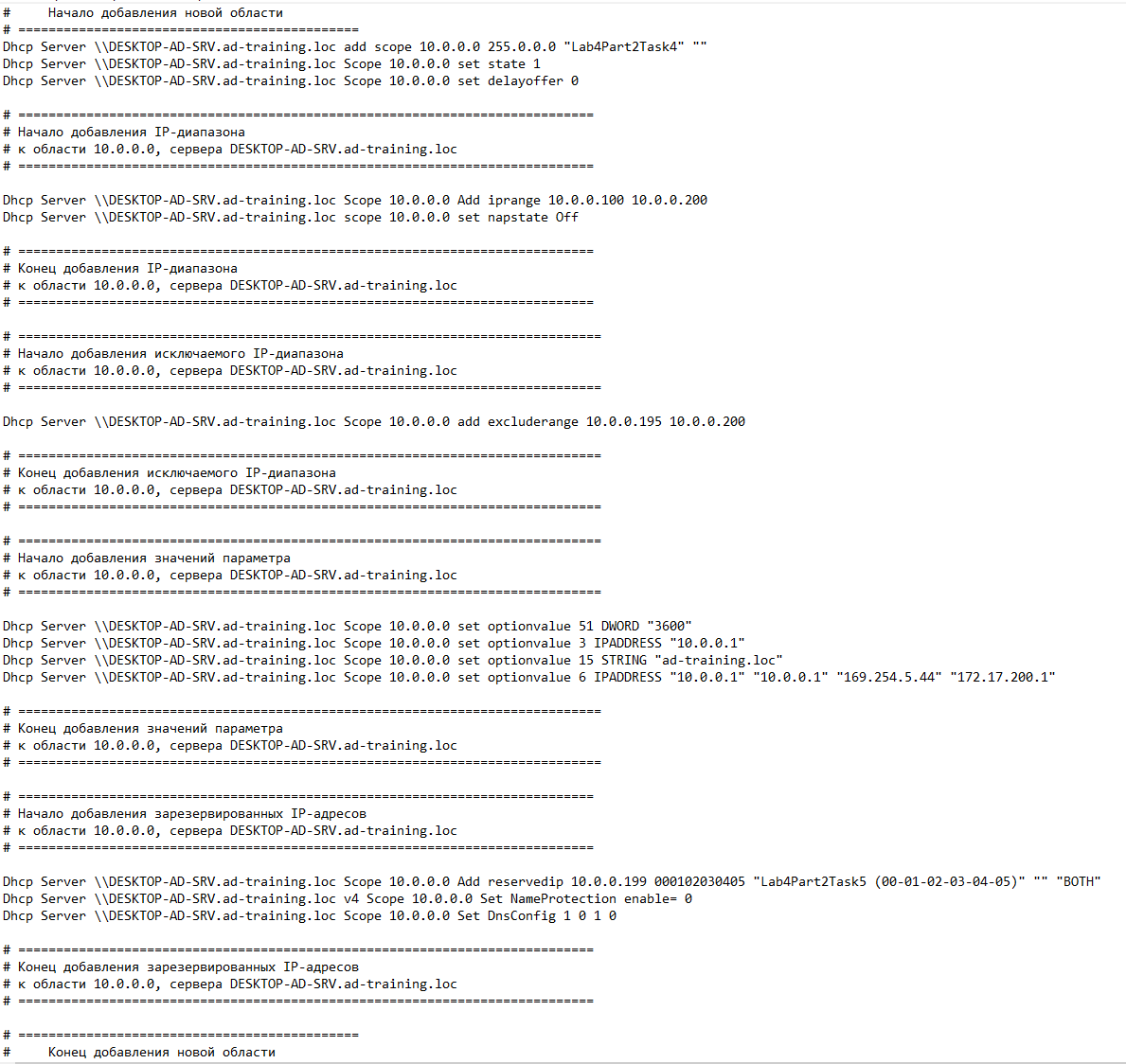
Мы настроили DHCP сервер и архивировали в файл, была сохранена полная копия конфигурации сервера. Это поможет нам в случае выхода сервера из строя или отката до предыдущих версий, при этом сохраняется все.

Теперь перед нами стоит задача сохранения контекстов сервера, тем самым мы сохраняем информацию обо всех областях и настройках, чтобы иметь возможность настроить другие DHCP сервера.

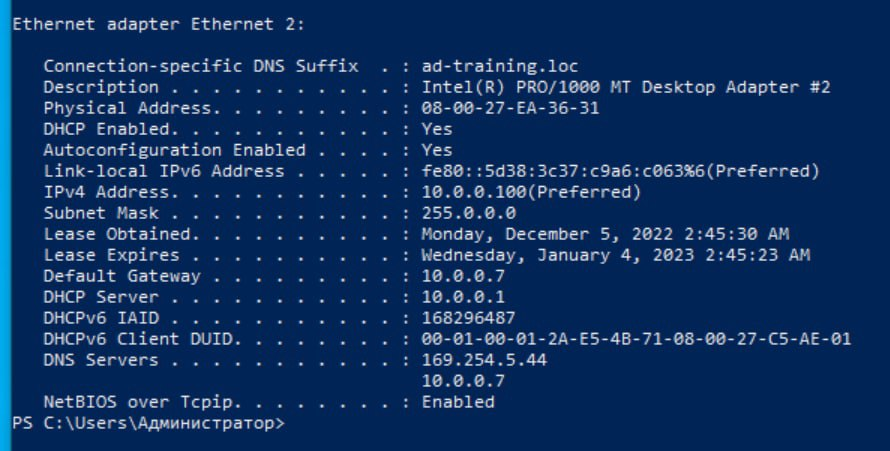
Выполним команду **netsh DHCP Server Dump** и поместим результаты ее выполнения в текстовый файл:



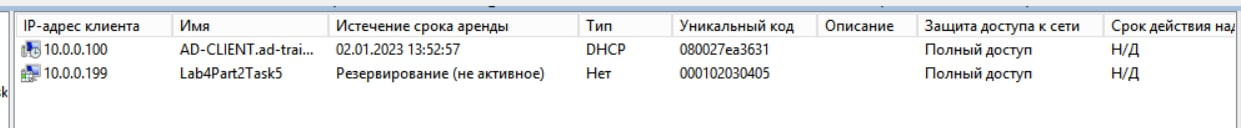
Секция добавления новой области из файла дампа конфигурации:



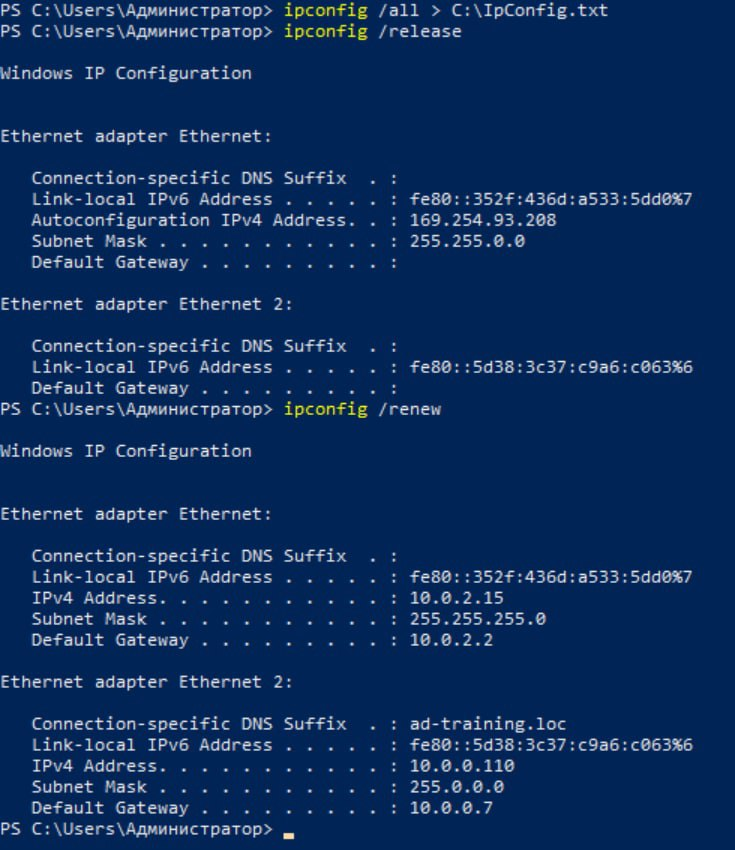
Запустим клиента и настроим получение IP адреса автоматически от нашего сервера, посмотрим что получилось:



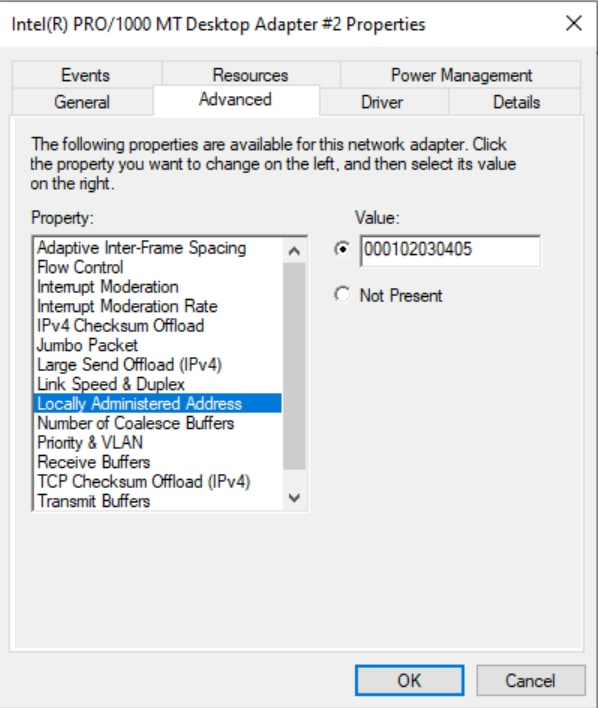
Как мы видим, работа DHCP сервера произошла успешно и он выдал клиенту адрес из указанного пула.  
Найдем это значение со стороны сервера:



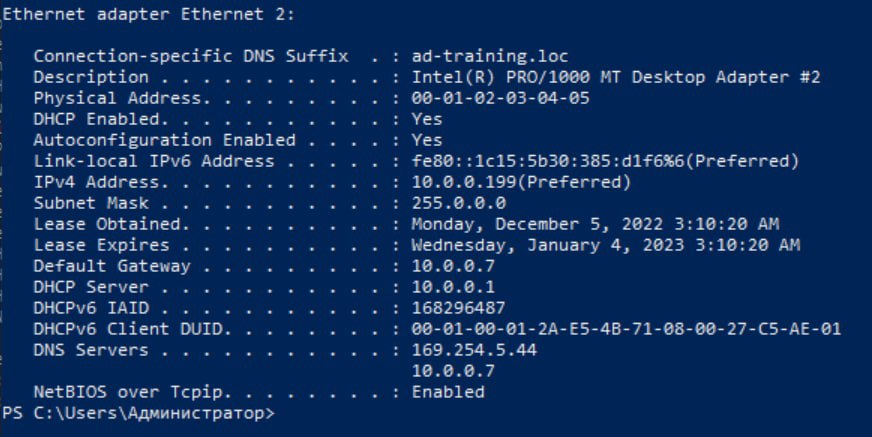
Вызовем команду **ipconfig /release** для того, чтобы сбросить значения сетевых интерфейсов, после чего с помощью команды **ipconfig /renew** получим новые данные интерфейсов. В случае нашего второго интерфейса, отвечающего за локальную сеть, мы пошлем на DHCP сервер запрос на выдачу нового адреса.



Теперь явно поменяем МАС адрес нашего клиента таким образом, чтобы сработал механизм резервирования:  
Через диспетчер устройств найдем наш сетевой адаптер, который отвечает за локальную сеть и в нем определим наш ранее зарезервированный адрес – 000102030405.

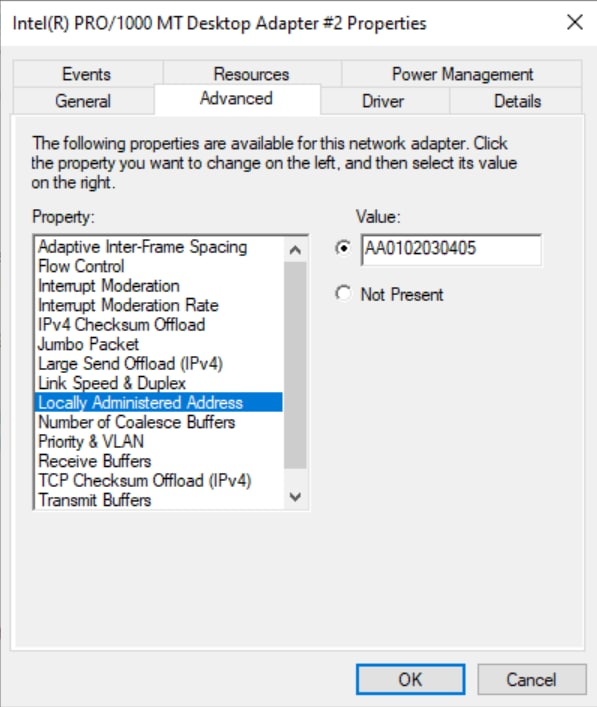


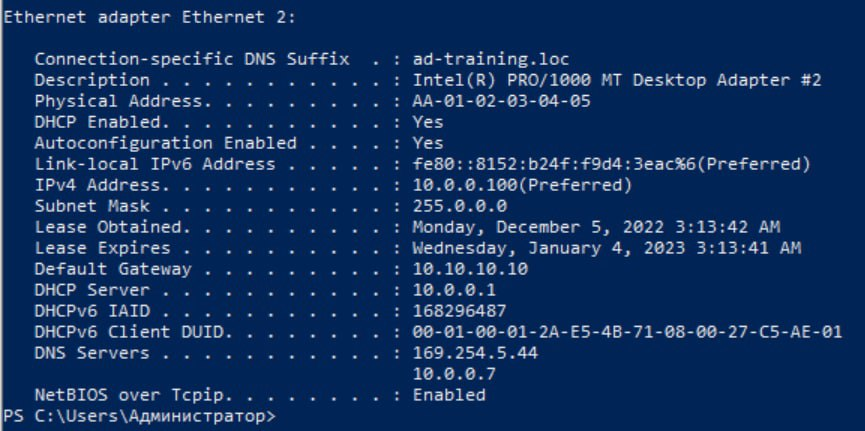
Теперь наш МАС адрес подходит для условия резервирования, убедимся, что резервация прошла успешно:



Наш адрес как раз 199, как мы и хотели.

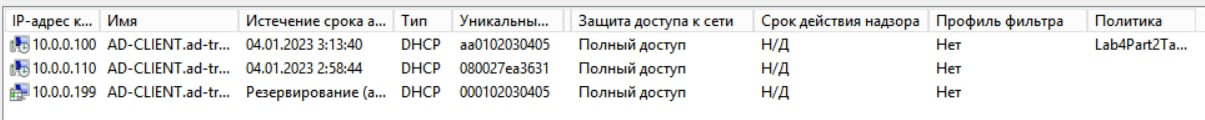
Затем изменим МАС адрес на АА0102030405, чтобы сработал механизм политики, которая заметит, что МАС адрес начинается на АА0102, как мы выставляли в условии, и изменит маршрутизатор на 10.10.10.10:



После того, как мы применяем новые параметры адреса, наш сетевой интерфейс автоматически перезагружается, так что информация обновляется даже в старой сессии powershell.  


Как мы помним, у клиента был IP адрес 10.0.0.110 после того, как мы обновили его значение командами ipconfig. Теперь, когда мы поменяли МАС адрес и наш интерфейс перезагрузился, клиент послал новый запрос, DHCP сервер выдал новый адрес 10.0.0.100. Однако, сработала также и политика на уровне области, ведь этот адрес входит в её пул, и назначил дополнительно значение маршрутизатора как 10.10.10.10.

Посмотрим на назначение адресов в консоли управления сервером DHCP:

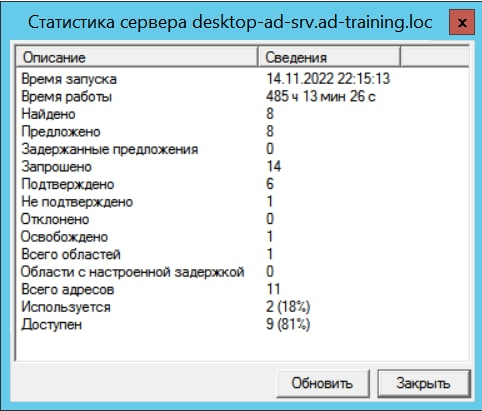


Как мы видим, у нас в данный момент активно 3 арендования адресов:

1. 10.0.0.100 – адрес, полученный после перезагрузки сетевого интерфейса при тестировании работы политики. Мы можем убедиться в работе политике, т.к. в соответствующем разделе информации в консоли указано ее название.
2. 10.0.0.110 – адрес, полученный после явного сброса и запроса нового адреса.
3. 10.0.0.199 – адрес, зарезервированный в области и полученный клиентом после перезагрузки сетевого интерфейса при тестировании работы резервирования. В разделе информации по истечению срока мы видим, что оно активно.

Все эти арендованные адреса активны потому, что время лицензии еще не вышло, а резервация активна всегда, до тех пор, пока ее не дезактивируют явно.

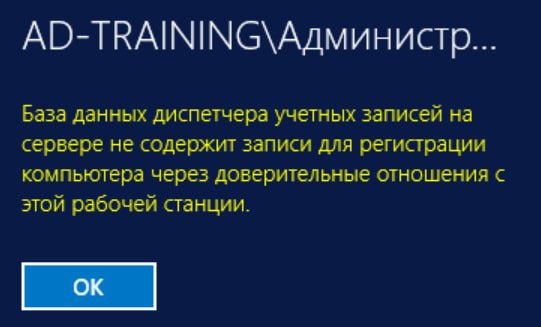
Посмотрим статистику работы протокола IPv4 DHCP сервера:



Как мы видим, сейчас используется 2 адреса: адреса, которые получил клиент в случаях простых запросов на выдачу адресов. Случай зарезервированного адреса здесь не учитывается, т.к. сейчас нет клиентов с необходимым для этого МАС адресом (000102030405). Политика назначала шлюз, но адрес запрашивался как обычно, поэтому и получил IP адрес в штатном режиме с предусмотренным сроком активности политикой области.

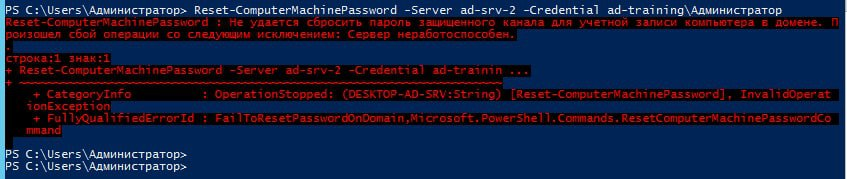
Часть 4:

После первоначальной настройки второго сервера в виде сетевого интерфейса и роли DHCP, я столкнулся с проблемой, что не мог поднять это роль.

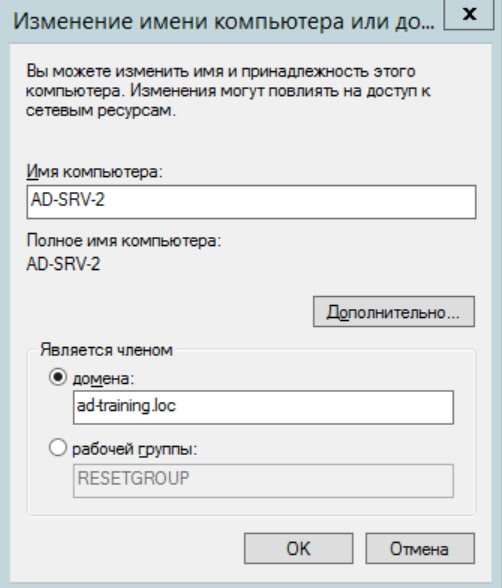
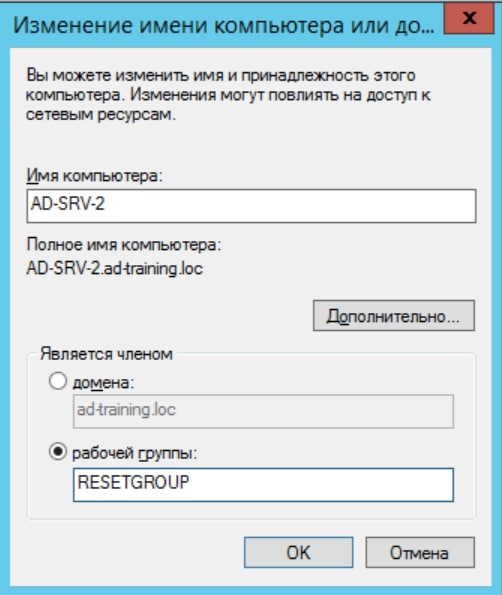


Как я выяснил, это частая проблема. Компьютер был восстановлен из старой точки восстановления или снапшота (если это виртуальная машина), созданной раньше, чем был изменен пароль компьютера в AD. Т.е. пароль в снапшоте отличается от пароля компьютера в AD. Если мы откатываем такой компьютер на предыдущее состояние, этот компьютер попытается аутентифицироваться на DC со старым паролем и они не совпадают.

Я попытался решить проблему через Powershell, путем сброса пароля второго сервера в системе, но это не увенчалось успехом:

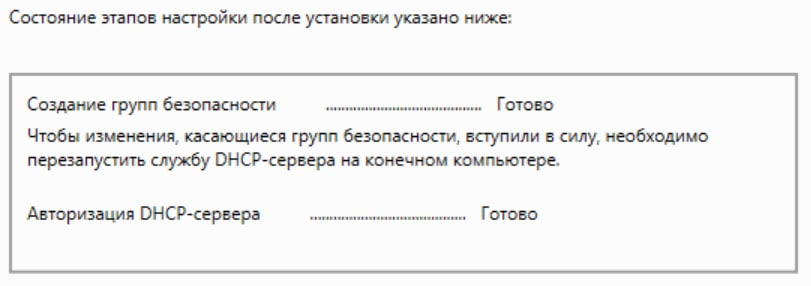


Тогда я прибегнул к классическому всемогущему методу: перезагрузке. Вывел второй сервер из домена и ввел его обратно:



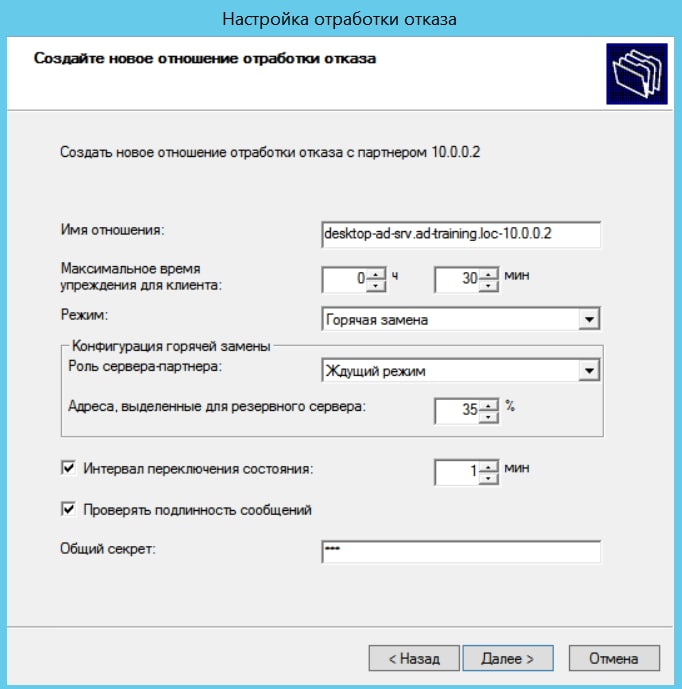
При вводе компьютера в домен в последнем появилась новая сгенерированная запись пароля.

После этого я смог авторизовать DHCP сервер на s2 без происшествий.



Произведем настройку обработки отказа нашей области, а точнее настроим сервер-партнер s2 так, чтобы:

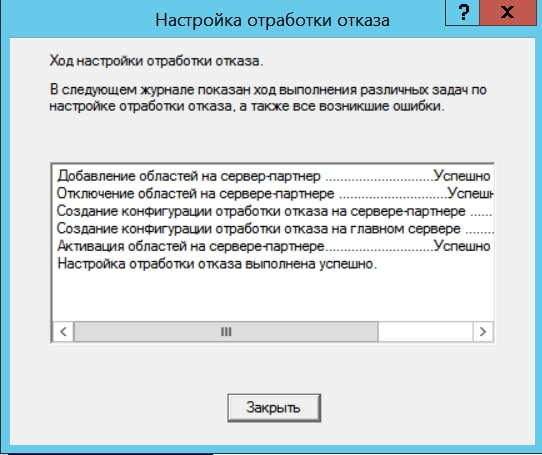
* Он работал в режиме горячей замены в ждущем режиме
* Имел 35% адресов пула для резерва
* Время упреждения клиента составляло 30 минут
* Интервал переключения был в 1 минуту
* Секретное слово для проверки – «123»



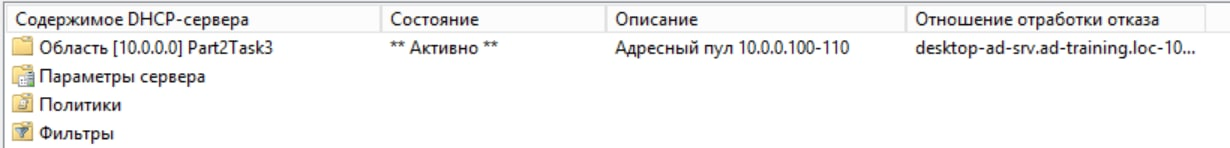
Параметры «Интервал переключения состояния» и «Максимальное время упреждения для клиента» определяют скорость срабатывания failover-а. Каждый из партнеров обслуживает свой диапазон адресов до того момента, пока один из серверов не перейдет в состояние PARTNER\_DOWN, и не пройдет время, указанное в параметре «Максимальное время упреждения для клиента». Только после этого ″оставшийся в живых″ сервер возьмет на себя контроль над всей областью.

* Интервал переключения состояния — интервал времени, по истечении которого партнер считается недоступным (PARTNER\_DOWN). Если не задавать этот параметр, то при падении партнера автоматического переключения не произойдет никогда и переключаться придется вручную.

• Максимальное время упреждения для клиента — параметр, определяющий срок продления аренды в случае падения» основного сервера. Когда клиент пытается продлить аренду, полученную на основном сервере, то резервный сервер продлевает ее не на срок аренды, указанный в свойствах области, а на время, указанное в данном параметре. И это будет происходить до тех пор, пока основной сервер не восстановит работу. Также этот параметр определяет, сколько времени сервер будет ждать возвращения партнера из состояния PARTNER\_DOWN прежде чем забрать контроль над всей областью. Кроме того, этот параметр определяет время перехода в нормальное состояние при возвращении партнера.

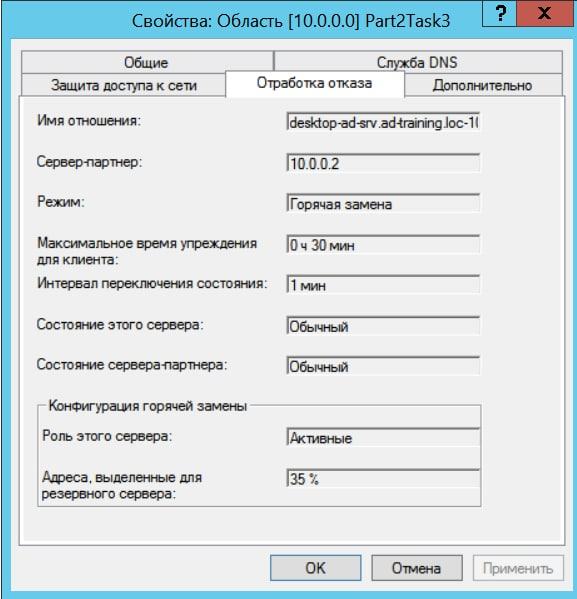


Теперь перейдем на второй сервер и убедимся в консоли DHCP, что наша область со всеми параметрами была реплицирована на него из основного сервера:



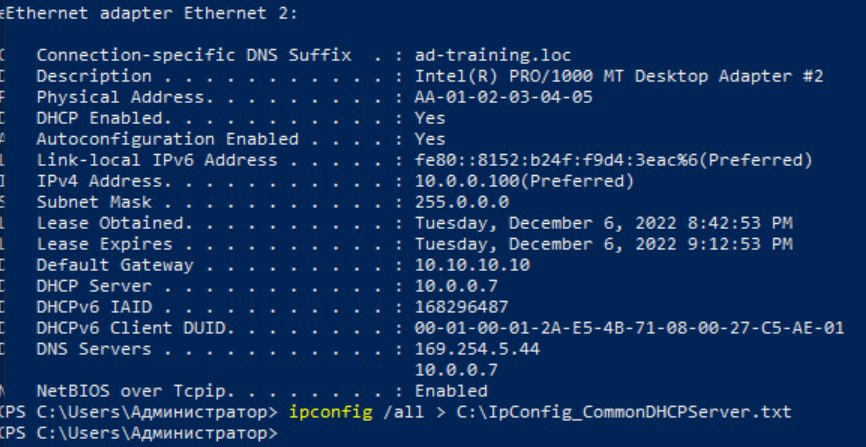
Здесь мы видим, что не только пул и остальные значения остались, но и тот факт что этот сервер имеет отношение к обработке отказов основного сервера и будет наподхвате.

Посмотрим закладку «Обработка отказа» в свойствах области:



То есть основной сервер видит обычный, параметры настроены как требовалось.

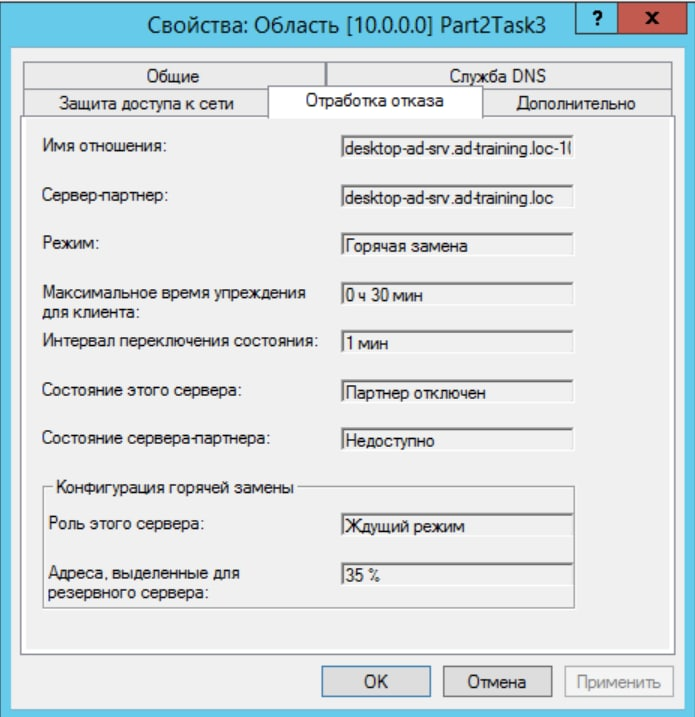
Теперь посмотрим на клиенте какой DHCP сервер выдал ему адрес:



Как мы видим, МАС адрес нашего клиента остался подходящим под условие политики у области DHCP, которая сработала корректно и назначила ему адрес маршрутизатора 10.10.10.10. Адрес DHCP сервера является адресом основного сервера, так что это штатное поведение.

Теперь, наша задача проверить отказоустойчивость DHCP, для это нам нужно устроить «отказ» основного сервера. Для этого мы просто отключим на нем сетевой интерфейс, отвечающий за локальную сеть, тем самым он пропадет из сети и потеряет со всеми связь, а, значит, не сможет функционировать и сработает механизм обработки отказа на втором сервере.

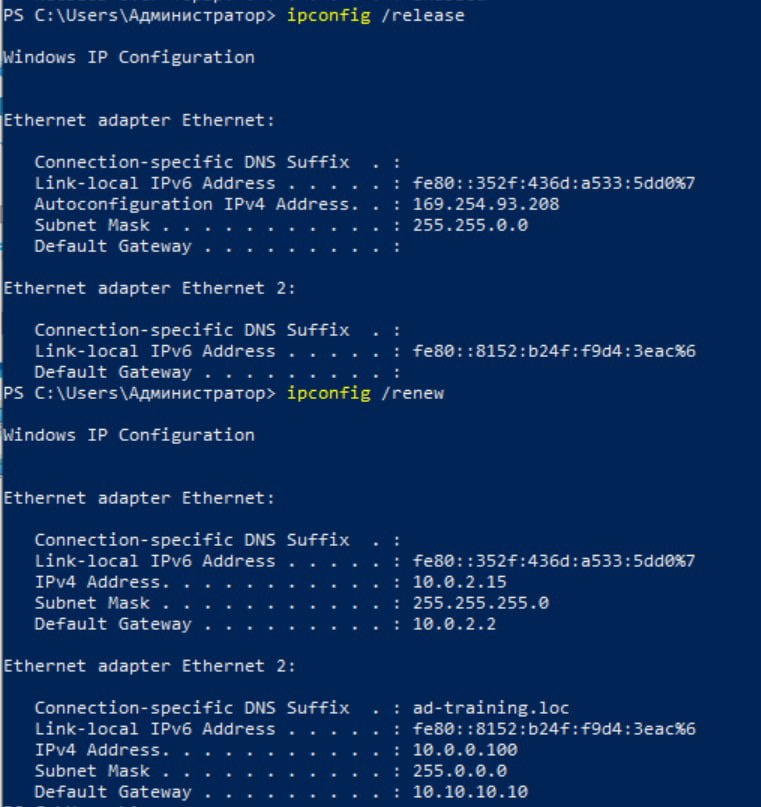
Посмотрим на произошедшее с точки зрения второго сервера:



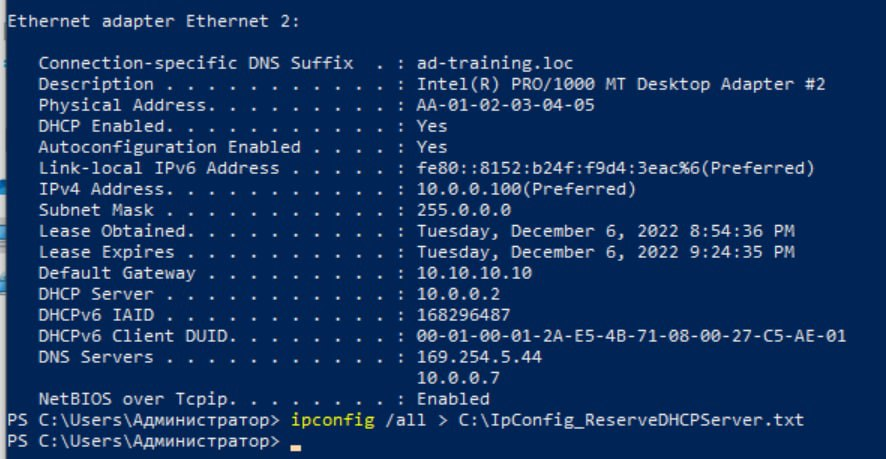
Как мы видим, второй сервер больше не видит состояние партнера и ничего о нем не знает.

Интервал переключения времени настроен как 1 минута, это значит, что спустя одну минуту без видимости партнера второй сервер посчитает, что основной сервер не восстановится и переведет его в состояние PARTNER\_DOWN. Далее, пока тот не вернется в нормальную работу, он будет выдавать IP адреса клиентам из пула в 35% от основного, каждый раз по полчаса.

Теперь перейдем к клиенту, который ничего не подозревает, и перезапросим адрес:



Далее посмотрим какие адреса получены и какой DHCP сервер их выдал:



МАС адрес клиента все еще не изменен и подходит под условие политики, а адрес маршрутизатора равен 10.10.10.10. Это значит, что политика успешно реплицировалась на второй сервер и корректно отработала.   
При это адрес DHCP сервера является адресом второго сервера, так что мы может убедиться, что сервер успешно обработал отказ основного сервера и выдал свободный IP адрес, и ровно на полчаса.

Включим сетевой интерфейс основного сервера обратно и видим, что все вернулось на круги своя.

Часть 5, текст скрипта для включения роли DHCP и создания области с необходимыми параметрами для основного сервера и для создания обработки отказа в режиме горячего ожидания для остальных серверов:

using namespace System.Collections.Generic

param (

[string]$Path2ServersData\_Json,

[string]$Path2ScopeData\_Json

)

function PrintOperationBegin {

param (

[string]$string

)

Write-Host $string -BackgroundColor DarkYellow -ForegroundColor Black

}

function PrintOperationSuccess {

param (

[string]$string

)

Write-Host $string -BackgroundColor DarkGreen -ForegroundColor Black

}

function PrintInfo {

param (

[string]$string

)

Write-Host $string -BackgroundColor Cyan -ForegroundColor White

}

function PrintCheck {

param (

[string]$string

)

Write-Host $string -BackgroundColor DarkMagenta -ForegroundColor Black

}

function PrintFatal {

param (

[string]$string

)

Write-Host $string -BackgroundColor Red -ForegroundColor Black

}

function PrintWarning {

param (

[string]$string

)

Write-Host $string -BackgroundColor Yellow -ForegroundColor Black

}

function PrintNewLine {

Write-Host ""

}

if (!(Test-Path $Path2ServersData\_Json -PathType Leaf)) {

PrintFatal "Can not find open the servers data file!"

break

}

if (!(Test-Path $Path2ScopeData\_Json -PathType Leaf)) {

PrintFatal "Can not find open the DHCP scope data file!"

break

}

Import-Module ActiveDirectory

$ServersData\_Json = Get-Content -Raw -Path $Path2ServersData\_Json | ConvertFrom-Json

$ScopeData\_Json = Get-Content -Raw -Path $Path2ScopeData\_Json | ConvertFrom-Json

# First way to get computer name:

# $(Get-ADComputer -Properties \* -Filter \* | Where-Object {$\_.IPv4Address -eq $ServersData\_Json[$key].Ipv4Address} | Select-Object Name).Name

# Second way to get computer name, but slower:

# $(Get-WmiObject -Class Win32\_ComputerSystem -ComputerName $ServersData\_Json.FirstServer.Ipv4Address).Name

foreach ($key in $ServersData\_Json.Keys.Clone())

{

$ServersData\_Json[$key].Add( `

"Name", `

$( `

Get-ADComputer -Properties \* -Filter \* | `

Where-Object {$\_.IPv4Address -eq $ServersData\_Json[$key].Ipv4Address} | `

Select-Object Name `

).Name `

)

$ServersData\_Json[$key].Add( `

"DomainName", `

$( `

Get-ADDomainController -Filter \* | `

Where-Object {$\_.IPv4Address -eq $ServersData\_Json[$key].Ipv4Address} | `

Select-Object HostName `

).HostName `

)

PrintNewLine

PrintCheck "Connecting to the computer $($ServersData\_Json[$key].Name)..."

if (!(Test-NetConnection -ComputerName $ServersData\_Json[$key].Name -InformationLevel Quiet))

{

PrintFatal "Connection failed: computer is ofline or data is incorrect."

break;

}

PrintNewLine

PrintCheck "Check for role DHCP..."

if ($(Get-WindowsFeature -ComputerName $ServersData\_Json[$key].Name -Name DHCP).InstallState -ne "Installed")

{

PrintNewLine

PrintWarning "Role DHCP isn't installed on $($ServersData\_Json[$key].Name)."

PrintNewLine

PrintOperationBegin "Adding DHCP role..."

Install-WindowsFeature DHCP -IncludeAllSubFeature -IncludeManagementTools -ComputerName $ServersData\_Json[$key].Name -Credential Ad-training\Администратор -Confirm

PrintNewLine

PrintOperationBegin "Autorization DHCP server..."

Add-DhcpServerInDC -DnsName $ServersData\_Json[$key].DomainName -IPAddress $ServersData\_Json[$key].Ipv4Address

PrintOperationSuccess "DHCP server has been autorized."

PrintNewLine

PrintOperationBegin "Creating server local security group..."

Add-DhcpServerSecurityGroup

PrintOperationSuccess "Server security group has been created."

PrintNewLine

PrintOperationBegin "Supressing server manager..."

# HKLM - HKey Lokal Machine

Set-ItemProperty -Path HKLM:\SOFTWARE\Microsoft\ServerManager\Roles\12 -Name ConfigurationState -Value 2

PrintOperationSuccess "Server manager has been supressed."

PrintNewLine

PrintOperationBegin "Setting additional setting for auto updates and cleaning..."

Set-DhcpServerv4DnsSetting -ComputerName $ServersData\_Json[$key].DomainName -DynamicUpdates Always -DeleteDnsRROnLeaseExpiry $True

PrintOperationSuccess "Automatical updates and cleaning has been setted."

PrintNewLine

PrintOperationBegin "Restarting DHCP service."

Restart-Service -Name DHCPServer -Force

PrintOperationSuccess "DHCP service has been restarted."

PrintNewLine

PrintOperationSuccess "DHCP role has been installed on $($ServersData\_Json[$key].Name)."

PrintNewLine

}

if ($ServersData\_Json[$key].Ipv4Address -eq $ServersData\_Json.FirstServer.Ipv4Address)

{

PrintNewLine

PrintInfo "Selected the main server.`nNew scope will be created and configured."

PrintNewLine

PrintCheck "Cheking for existence of the scope..."

$isScopeExist = $Null

try {

$isScopeExist = Get-DhcpServerv4Scope `

-ComputerName $ServersData\_Json.FirstServer.Name `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID

}

catch {

PrintNewLine

PrintInfo "The scope is already exist."

continue

}

if ($Null -eq $isScopeExist -or `

$isScopeExist.ScopeId.IPAddressToString -ne $ScopeData\_Json.CommonInfo.ScopeID)

{

PrintNewLine

PrintOperationBegin "Creating scope..."

Add-DhcpServerv4Scope `

-Name $ScopeData\_Json.CommonInfo.Name `

-ComputerName $ServersData\_Json.FirstServer.DomainName `

-StartRange $ScopeData\_Json.RangeInfo.StartRange `

-EndRange $ScopeData\_Json.RangeInfo.EndRange `

-SubnetMask $ScopeData\_Json.RangeInfo.SubnetMask `

-LeaseDuration $ScopeData\_Json.CommonInfo.LeaseDuration `

-Description $ScopeData\_Json.CommonInfo.Description `

-State $ScopeData\_Json.CommonInfo.State

PrintOperationSuccess "Base scope has been created."

PrintNewLine

PrintOperationBegin "Setting addresses..."

Set-DhcpServerv4OptionValue `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID `

-DnsDomain $( `

Get-ADDomainController -Filter \* | `

Where-Object {$\_.IPv4Address -eq $ServersData\_Json.FirstServer.Ipv4Address}|`

Select-Object Domain `

).Domain `

-DnsServer $ScopeData\_Json.ServersInfo.DnsAddress `

-Router $ScopeData\_Json.ServersInfo.RouterAddress

PrintOperationSuccess "Environment addresses has been configured."

PrintNewLine

PrintOperationBegin "Adding exclusions..."

Add-DhcpServerv4ExclusionRange `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID `

-StartRange $ScopeData\_Json.ExclusionInfo.StartRange `

-EndRange $ScopeData\_Json.ExclusionInfo.EndRange `

PrintOperationSuccess "Exclusion range has been added to scope."

PrintNewLine

PrintOperationBegin "Adding reservation for MAC address 00-01-02-03-04-05..."

Add-DhcpServerv4Reservation `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID `

-IPAddress $ScopeData\_Json.ReservationInfo.ReservedIPAdress `

-ClientId $ScopeData\_Json.ReservationInfo.MACAdress `

-Description $ScopeData\_Json.ReservationInfo.Description

PrintOperationSuccess "Reservation has been added."

PrintNewLine

PrintOperationBegin "Configuring policy..."

PrintOperationBegin "Creating policy and setting conditions..."

Add-DhcpServerv4Policy `

-Name $ScopeData\_Json.PolicyInfo.Name `

-ComputerName $ServersData\_Json.FirstServer.DomainName `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID `

-Description $ScopeData\_Json.PolicyInfo.Description `

-Condition $ScopeData\_Json.PolicyInfo.Condition `

-MacAddress $ScopeData\_Json.PolicyInfo.MacAdresses.AA\_01\_02\_any

PrintOperationSuccess "Scope level policy has been created."

PrintOperationSuccess "Policy conditions has been setted."

PrintNewLine

PrintOperationBegin "Setting IP range for policy..."

Add-DhcpServerv4PolicyIPRange `

-ComputerName $ServersData\_Json.FirstServer.DomainName `

-Name $ScopeData\_Json.PolicyInfo.Name `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID `

-StartRange $ScopeData\_Json.PolicyInfo.StartRange `

-EndRange $ScopeData\_Json.PolicyInfo.EndRange

PrintOperationSuccess "IP range for policy has been setted."

PrintNewLine

PrintOperationBegin "Setting router configuration for policy..."

Set-DhcpServerv4OptionValue `

-ComputerName $ServersData\_Json.FirstServer.DomainName `

-PolicyName $ScopeData\_Json.PolicyInfo.Name `

-ScopeId $ScopeData\_Json.PolicyInfo.CommonInfo.ScopeID `

-OptionId $ScopeData\_Json.PolicyInfo.FeatureInfo.Router.OptionID `

-Value $ScopeData\_Json.PolicyInfo.FeatureInfo.Router.Address

PrintOperationSuccess "Router configuration has been setted for policy."

PrintNewLine

PrintOperationBegin "Activating the scope..."

Set-DhcpServerv4Scope -ScopeId $ScopeData\_Json.CommonInfo.ScopeID -State Active

PrintOperationSuccess "Scope has been activated."

PrintNewLine

PrintOperationSuccess "The scope has been created and configured."

PrintNewLine

}

else

{

PrintNewLine

PrintWarning "The scope is already exist."

continue

}

}

else

{

PrintNewLine

PrintInfo "Selected reserve DHCP server.`nDHCP failover (active-passive) will be created and configured."

PrintNewLine

PrintOperationBegin "Creating failover ($($ServersData\_Json.FirstServer.Name)\_$($ServersData\_Json[$key].Name)) relationship with the main server..."

Add-DhcpServerv4Failover `

-Name $($ServersData\_Json.FirstServer.DomainName)\_$($ServersData\_Json[$key].DomainName) `

-ComputerName $ServersData\_Json.FirstServer.DomainName `

-PartnerServer $ServersData\_Json[$key].DomainName `

-ScopeId $ScopeData\_Json.CommonInfo.ScopeID `

-MaxClientLeadTime $ScopeData\_Json.FailoverInfo.MaxClientLeadTime `

-StateSwitchInterval $ScopeData\_Json.FailoverInfo.StateSwitchInterval `

-AutoStateTransition $ScopeData\_Json.FailoverInfo.AutoStateTransition `

-ReservePercent $ScopeData\_Json.FailoverInfo.ReservePercent `

-SharedSecret $ScopeData\_Json.FailoverInfo.SharedSecret `

-Force

PrintOperationSuccess "Failover $($ServersData\_Json.FirstServer.Name)\_$($ServersData\_Json[$key].Name) has been created."

PrintNewLine

}

}

1. **Выводы и анализ результатов работы.**

Научился импортировать и экспортировать данные в формат JSON.  
Научился работать со списками и хэштамблицами.

Научился проверять соединение с компьютерами в сети.  
Научился устанавливать роли DHCP.

Научился создавать и настраивать области DHCP.

Научился создавать и настраивать обработку отказа сервера DHCP.

1. **Дополнительные задания.**
2. **Замечания преподавателя (исправления, вызванные замечаниями преподавателя, также помещают в этот пункт)**