# Overview

In this guided practice, you will create some common objects in Active Directory from the GUI and the command line.

# Objectives

Be able to create the common Active Directory objects from the GUI and command line including user accounts, computer accounts, security groups, distribution groups, and contact objects.

# Initial Conditions

* Guided Practice – Implementing an Active Directory Logical Design

# Instructions

## Creating Groups to Support Assigning Access to Resources

You will be configuring shared folders and printers for use by the employees within the departments and this organization. You will be using the ADGLP grouping strategy to implement security on these shared resources. In this step, you will create the groups necessary to implement the grouping strategy and secure these resources.

### Creating OUs to support assigning Permissions

To better track the groups that are created in AD that are used to assign permissions, you will often create an OU or OUs to contain these groups.

1. Logon to the **Server-01** virtual machine using the **kmwadmin** account.
2. **Create** an **OU** in the **root** of the domain named **Permissions.**

### Creating Global Groups for the Departments and Locations

Create a global group for the **Management** department in **Greenville** OU.

1. Open a **PowerShell** prompt with **administrative** privileges.
2. Type the following command:

New-ADGroup -name GV-Management -GroupScope Global -GroupCategory Security -path “OU=Greenville,DC=KMW,DC=local”

### Adding Users to Groups

1. Add users to the **GV-Management** group by typing the following in a PowerShell Administrative shell:

Add-ADGroupMember -Identity GV-Management -Members (Get-ADUser -filter \* -SearchBase “OU=Management,OU=Greenville,DC=KMW,DC=local”)

* 1. This command uses the **Get-ADUser** cmdlet to find all the users in the **Management** OU inside of **Greenville** and add them to the **GV-Management** group. The same can be done by browsing to the **Greenville** -> **Management** OU in **Active Directory Users and Computers** and selecting all the users and choosing **Add to group** from the context menu.

1. Create the groups listed in the table below and add the respective users to the group using the steps above. Use group nesting when available (e.g., adding **GV-Projects**, **GV-Administration**, and **GV-Management** to the **Greenville** group instead of adding all the users in **Greenville)**.

| Group Name | Group Scope | Group Category (Type) | Location | Members |
| --- | --- | --- | --- | --- |
| GV-Projects | Global | Security | Greenville OU | Personnel in the **Projects** Department at **Greenville** |
| GV-Administration | Global | Security | Greenville OU | Personnel in the **Administration** Department at **Greenville** |
| Greenville | Global | Security | Greenville OU | Personnel at the **Greenville** Location |
| CO-Projects | Global | Security | Columbia OU | Personnel in the **Projects** Department at **Columbia** |
| CO-Administration | Global | Security | Columbia OU | Personnel in the **Administration** Department at **Columbia** |
| CO-Management | Global | Security | Columbia OU | Personnel in the **Management** Department at **Columbia** |
| Columbia | Global | Security | Columbia OU | Personnel at the **Columbia** Location |
| Employees | Global | Security | Users folder | All **Greenville** & **Columbia** employees |
| Management | Global | Security | Users folder | **Management** employees from **Greenville** & **Columbia** |

**Note**: In the real world you would continue to make similar groups for Richmond and Greensboro. You will skip this for the sake of brevity.

## Create the Folder structure to support Group policy

Companies often require shared folders to implement group policy, user configuration and storage, and collaboration.

1. Log in as **KMWAdmin** on **Server-01.**
2. Open **PowerShell** *without* **Administrative** rights.
3. Create a folder in PowerShell, type the following:

New-Item -Path C:\Shares\Software -ItemType Directory

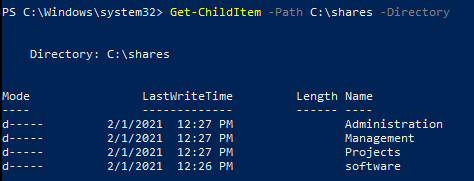
The example above creates the folder **Software** and the folder **Shares** if it does not exist.

1. Create the following additional folders to support your business requirements.

* C:\Shares\Administration
* C:\Shares\Management
* C:\Shares\Projects
  1. Type the following command to verify that all the folders were created:

Get-ChildItem -Path C:\Shares –Directory

* 1. Your output should look like the screenshot below.



## Creating Shared Folders

1. Open **PowerShell** with Administrative rights.
2. The following command will share the **C:\Shares** folder with the name **Shares** and the **Full Control** permissions granted to the **Everyone** group.

New-SmbShare -Name Shares -Path C:\Shares -FullAccess Everyone

1. Share each of the folders createdabove on **Server-01** with the same settings.
   1. **Note**: the shared folders can also be created with the following pipeline.

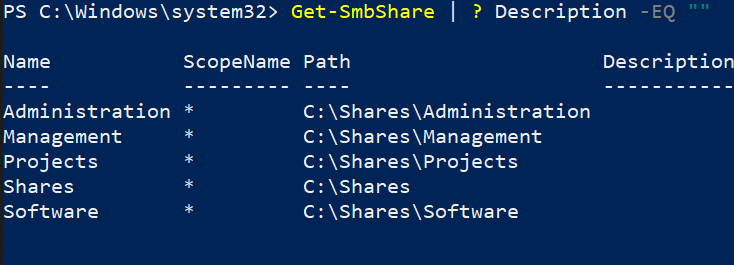
Get-ChildItem -Path C:\Shares -Directory | % { New-SmbShare -Name $\_.Name -Path $\_.FullName -FullAccess Everyone }

* 1. The command above lists the directories in the **C:\Shares** folder and sends the directory objects through the pipeline. The object is represented by the **$\_** variable the Name property and **Fullname** properties are used to provide values for the Name and Path parameters. The **%** symbol is an alias for the **ForEach-Object** cmdlet which is used to run commands on an object passing through a pipeline.

1. Verify the folders were shared with the following command:

Get-SmbShare | ? Description -EQ “”

1. You should see the output shown below.

******

## Securing the Folder structure

In the previous sections you have created global groups and added accounts to those groups (AG). You have also nested some global groups within other groups (GGU). You will now create domain local groups and place the global or universal groups within these and assign these groups permissions (DLP) to resources (folders) to implement your grouping strategy and secure the folder structure.

The KMW group uses the following naming standard for **domain local groups** used for assigning permissions:

* **Groups** will be in the **Permissions** OU
* **Groups** will be named as follows:
  + Server Name – DC-01 or DC-02
  + A “-“dash character
  + The name of the shared folder
  + A “-“dash character
  + The type of access the group has - R for Read and Execute, M for Modify, F for Full Control
  + Example**: DC-01-Software-F** would be the name of the group that has full control of the shared folder named Software folder on DC-01

The **Management** group in **Greenville** needs full control of the shares folder on **Server-01**.

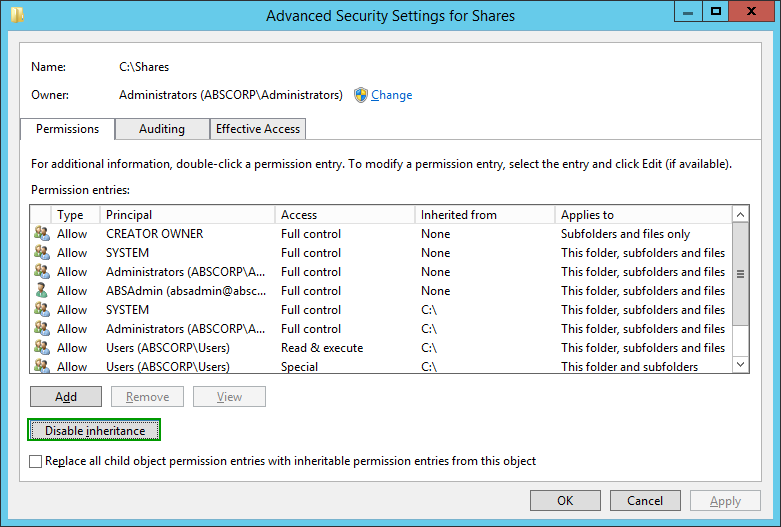
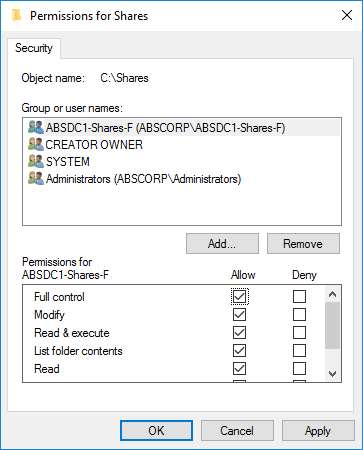
Use the following procedure to implement security on the shares folder on **Server-01**:

1. Create the domain local group using the following command:

New-ADGroup -Name DC-01-Shares-F -GroupScope DomainLocal -GroupCategory Security -Path “OU=Permissions,DC=KMW,DC=local”

1. Add the Management global group to the group using the following command:

Add-ADGroupMember -Identity DC-01-Shares-F -Members GV-Management

1. Disable inherited permissions on the **C:\Shares** folder as follows:
   1. Open the **properties C:\Shares** folder.
   2. On the **Security** **tab**, **click** the **Advanced** button
   3. In the **Advanced Security for Shares** dialog box, click the **Change Permissions button** and then **click** the **Disable inheritance** button as shown in the figure to the right.
   4. The **Block Inheritance** dialog box will open, select the **Convert inherited permissions to explicit permission on this object** option.
   5. In the **Advanced Security for Shares** dialog box, **check** the **Replace all child object permission entries with inheritable permission entries from this object** and then click the **OK** button.
   6. On the **Security** page of the **Shares** **properties** dialog box, click the **Edit** button and remove the **users** (**KMW\Users**) entry and then click **OK**.
2. Open the properties of the **C:\Shares** folder and configure security so that the **DC-01-Shares-F** group has **Full Control**. Make sure the **Creator Owner**, **System**, and **Administrators** group permissions are not changed.
3. **Optional** - Configure the description for the group above. This could have also been done when you created the group. Use the following command to set the description for the group above:

Set-AdGroup -Identity DC-01-Shares-F -Description “This group has full control permissions for the C:\Shares folder located on DC-01”

Configure the following security settings on the remaining folders using the **AGDLP** strategy shown above: (**Note**: You will have to create Domain local groups for each of the permissions assigned, for example: a **DL** group named **DC-01-Software-R** should be made for giving read access to the **Software** share on **DC-01**, you will also not have to create some of the groups because the **Management** group has **Full control** already because of permission inheritance.)

| Folder | Permissions |
| --- | --- |
| Server-01 (DC-01) | |
| C:\Shares\Software | **GV-Management** has full control, **Greenville** has read access |
| C:\Shares\Management | **GV-Management** has full control |
| C:\Shares\Projects | **GV-Management** has full control, **GV-Projects** has Modify |
| C:\Shares\Administration | **GV-Management** has full control, **GV-Administration** has Modify |
|  | |
| Server-02 (DC-02) (Practice – Optional) | |
| C:\Shares\Software | **CO-Management** has full control, **Columbia** has read access |
| C:\Shares\Management | **CO-Management** has full control |
| C:\Shares\Projects | **CO-Management** has full control, **CO-Projects** has Modify |
| C:\Shares\Administration | **CO-Management** has full control, **CO-Administration** has Modify |

## Submission Requirements

1. **Download** the **grading** **script** from the assignment page to the **C:\Scripts** folder.
2. Check your lab by running the following command:

Invoke-Pester -Path C:\Scripts\GP16-Creating\_AD\_Objects.test.ps1

**Note**: You will see a security warning when running the script. Enter **R** to run the script.

If you want to see more detail, add **-Output Detailed** to the command. This may assist you with troubleshooting

Invoke-Pester -Path C:\Scripts\GP16-Creating\_AD\_Objects.test.ps1  
-Output Detailed

1. You should not see any red in the output. Red in the PowerShell way of telling you that an error condition exists. Most of the time, the output will tell you what is wrong. If it is not obvious, contact your teacher and ask for assistance. You will be learning PowerShell during this term. **Correct** any **errors** you may have and run the script until all the output has no red. You should see the output like the images below

Text

Description automatically generated

1. Capture a snippet that shows the PowerShell Command and all its output. If you must use more than one snippet to capture the output, you must have at least **one line of overlap** in the snippets. The text in the snippets **must be legible** when pasted into the Word document. Paste the snippet(s) into a **new** **Word** **document**
2. **Fill** **in** the **information** in the following table. Copy the following table into the **Word** **document** and fill in the information about all the **new** commands used in this lab (the example provided is not a new command and should be deleted):

|  |  |  |
| --- | --- | --- |
| PowerShell Commands | | |
| Command | Example | Description |
| Get-Childitem | Get-Childitem -Path C:\ | Displays the files in the C:\ directory |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **Upload** the **document** in the submission area of the assignment.