GROUP POLICY

ON LINUX

David Mulder

Group Policy on Linux

David Mulder

Group Policy on Linux

1 Preface

This book introduces the user to opensource tools for managing Linux clients via Samba’s Group Policy.

Samba is a popular opensource tool that allows Linux systems to integrate with Windows environments, particularly when it comes to file and printer sharing. One of the key features of Samba is its ability to apply Group Policy objects (GPOs) to Linux clients.

Group Policy is a feature of the Microsoft Windows operating system that allows administrators to centrally manage system and user settings. With Samba, Linux users can take advantage of this powerful tool to centrally manage and configure their systems.

In this book, we will introduce the reader to the basics of Group Policy and show how to use Samba to apply GPOs to Linux clients. We will cover topics such as configuring Samba’s Group Policy Server Side Extensions (SSE), troubleshooting common issues with Client Side Extensions (CSEs), and how to create and apply your own Group Policy. By the end of this book, the reader should have a good understanding of how to use Group Policy with Linux systems and be able to confidently manage their Linux clients using this powerful tool.

2 About the Author

David Mulder is a developer known for his work on integrating Group Policy support into Samba, which has allowed Linux users to take advantage of this powerful feature to centrally manage their systems. Mulder's work on Samba's Group Policy support began in 2016, when he began reviewing code from Luke Morrison, an intern who had submitted his implementation of Group Policy to the Samba project. Mulder previously contributed to the Vintela Group Policy project beginning in 2012, and brought that expertise to the Samba team.

Some of the text in this book, as well as the images, were generated using OpenAI's GPT-3 model, which is a state-of-the-art language processing system. OpenAI's GPT-3 technology is an example of the incredible advances that have been made in the field of AI and natural language processing. This technology has the potential to revolutionize many areas of research and industry, and its use in generating text and images for this book is a testament to its capabilities.

3 Introduction

Starting with version 4.14, Samba has included support for applying Group Policy objects (GPOs) to Linux clients, making it possible to use Group Policy to centrally manage and configure Linux systems in a Windows environment.

Samba’s Group Policy support is designed to be similar to what is offered by proprietary tools, such as Vintela’s and Centrify’s Group Policy solutions. This allows Linux users to take advantage of the same powerful Group Policy features that are available to Windows users, without having to rely on proprietary tools.

Overall, Samba’s Group Policy support makes it possible for Linux users to manage and configure their systems using the same powerful Group Policy features that are available to Windows users. This allows Linux users to easily integrate their systems with Windows environments and take advantage of Group Policy’s central management capabilities.

3.1 What’s the difference between Group Policy and a Group Policy Object?

The key difference between Group Policy and a Group Policy Object (GPO) is that Group Policy is the overall concept and framework for managing and configuring settings on computers in an environment, while a GPO is a specific collection of settings that are applied to a group of machines or users.

Group Policy allows administrators to define and manage the settings that are applied to computers and users in a domain. This includes settings for various

aspects of the operating system, such as security policies, user accounts, and network settings. Group Policy also includes the infrastructure and tools for distributing and applying these settings to the appropriate computers and users. You can think of *Group Policy* like a template for a work order.

A GPO, on the other hand, is a specific set of settings that are defined by an administrator and applied to a group of computers or users. A GPO can be thought of as a filled out copy of a work order that specifies the settings that should be applied to the members of the group. These settings are stored in the GPO and distributed to the appropriate computers and users by the Group Policy infrastructure.

Server-side extensions (SSEs) are responsible for processing and managing GPOs on the domain controller, while client-side extensions (CSEs) are responsible for applying the settings in a GPO to the local system. Together, these components work to manage and apply GPOs in an environment.

3.2 Server Side Extensions

The purpose of a Server Side Extension (SSE) is to process and manage Group Policy objects (GPOs) on the domain controller (to fill out a *work order*). In a Windows environment, this generally refers to some component of the Group Policy Management Editor.

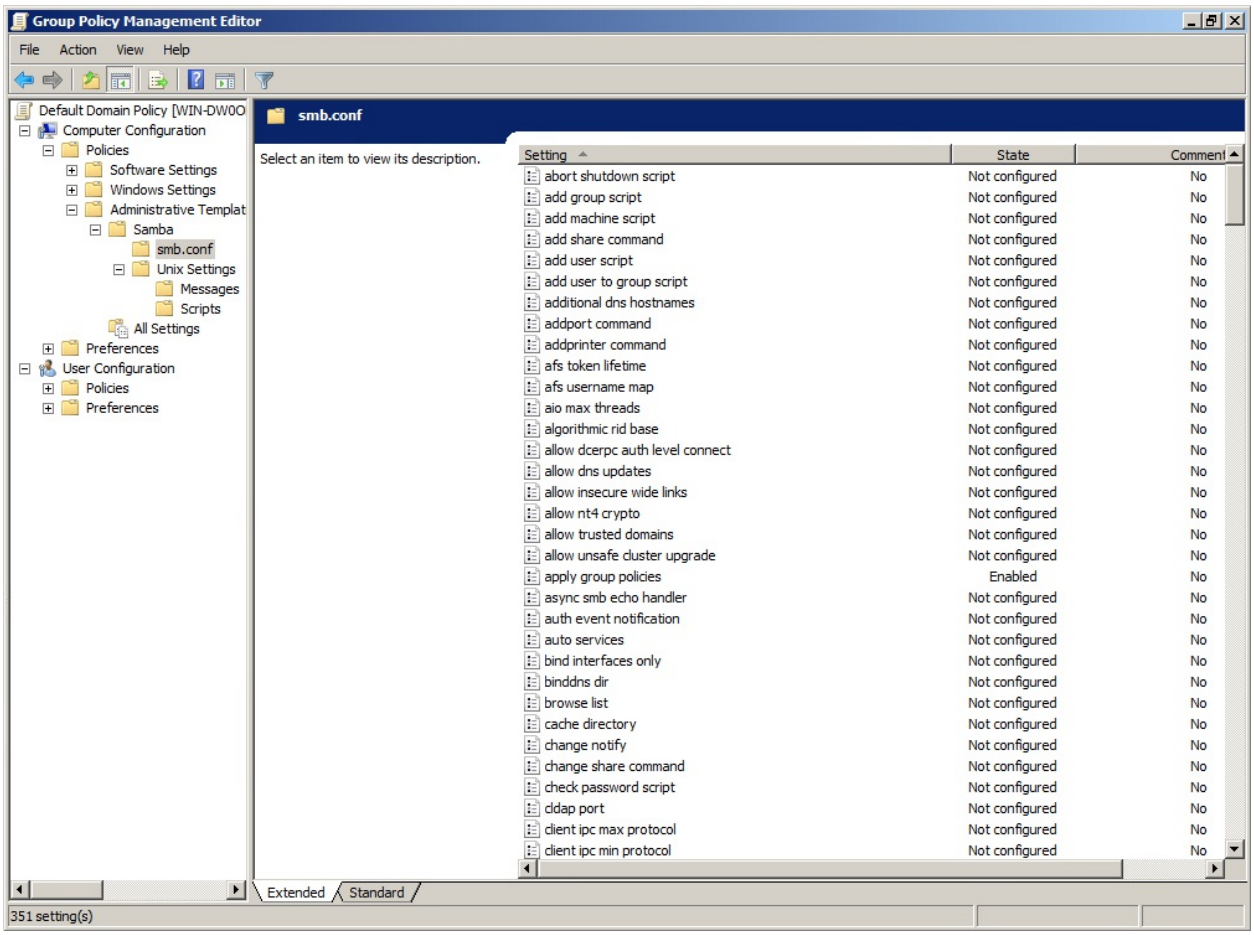


Figure 3.1: Group Policy Management Editor

In the case of Samba, SSEs also include the `samba-tool gpo` command, which allows administrators to manage GPOs from the command line. This command allows administrators to create, link, and modify GPOs.

```
> samba-tool gpo
Usage: samba-tool gpo <subcommand>

Group Policy Object (GPO) management.


Options:
  -h, --help  show this help message and exit


Available subcommands:
  aclcheck        - Check all GPOs have matching LDAP and DS
ACLs.
```

```
  admxload        - Loads samba admx files to sysvol
  backup          - Backup a GPO.
  create          - Create an empty GPO.
  del             - Delete a GPO.
  dellink         - Delete GPO link from a container.
  fetch           - Download a GPO.
  getinheritance  - Get inheritance flag for a container.
  getlink         - List GPO Links for a container.
  list            - List GPOs for an account.
  listall         - List all GPOs.
  listcontainers  - List all linked containers for a GPO.
  manage          - Manage Group Policy Objects
  restore         - Restore a GPO to a new container.
  setinheritance  - Set inheritance flag on a container.
  setlink         - Add or update a GPO link to a container.
  show            - Show information for a GPO.
> samba-tool gpo manage
Usage: samba-tool gpo manage <subcommand>

Manage Group Policy Objects


Options:
  -h, --help  show this help message and exit


Available subcommands:
  access    - Manage Host Access Group Policy Objects
  files     - Manage Files Group Policy Objects
  issue     - Manage Issue Group Policy Objects
  motd      - Manage Message of the Day Group Policy Objects
  openssh   - Manage OpenSSH Group Policy Objects
  scripts   - Manage Scripts Group Policy Objects
  security  - Manage Security Group Policy Objects
  smb_conf  - Manage smb.conf Group Policy Objects
  sudoers   - Manage Sudoers Group Policy Objects
  symlink   - Manage symlink Group Policy Objects
```

When working with Linux clients, using `samba-tool gpo manage` to fill out your GPO is generally the preferred method.

Overall, the purpose of an SSE is to manage and process GPOs on the domain controller, enabling administrators to define and apply settings to the appropriate computers and users in the domain. These extensions work behind the scenes to ensure that GPOs are processed and managed correctly

on the domain controller.

3.2.1 Enabling Group Policy Server Side Extensions on the Server

In order to use the Samba Administrative Templates in the Group Policy Management Console, you'll need to install them first, using the command `sudo samba-tool gpo admxload -UAdministrator`. See chapter 22 for specifics on how to do this.

3.3 Client Side Extensions

In Group Policy vernacular, a Client Side Extension (CSE) is an encapsulated module on a client machine intended to enforce a specific policy. The responsibility of the CSE is to install that policy on the client and ensure it is enforced. It's also the responsibility of the CSE to remove the policy when disabled by the server.

In Samba, a CSE looks like a single python file. Within that file, the CSE inherits from one of several base classes provided by Samba which provide settings from the server. This is discussed in more detail in chapter 20.2.

3.3.1 Enabling Group Policy Client Side Extensions on the Linux Client

To enable Group Policy in Winbind, set the apply group policies global smb.conf option to Yes. You can even deploy this setting from Group Policy smb.conf options, then running the apply command manually the first time with `sudo samba-gpupate --force`.

Policies are enforced at a random interval between 90 and 120 seconds.

Policies can be manually enforced at any time on a Linux domain member using the `samba-gpupdate --force` command.

Winbind will enforce both machine policy (as of Samba 4.14) and user policy

(as of Samba 4.18).

If the Linux client is joined using SSSD, you can instead enforce the policy using oddjob-gpupdate. The samba-gpupdate command from Samba must also be installed.

Further details on how to configure Automatic Policy Refresh via Winbind (23.2) and SSSD (23.3) can be found in chapter 23.

3.3.2 Resultant Set of Policy

A Resultant Set of Policy (RSoP) is a summary of the policies that will be applied to a computer or user, or that have already been applied. The RSoP is generated by the client-side extensions (CSEs) that are responsible for applying Group Policy objects (GPOs) to the local system.

Each CSE is configured to return an RSoP for the policies that it manages.

The RSoP is generated by the CSEs before the GPOs are applied, allowing administrators to see what policies will be applied to the system before they are actually applied. This can be useful for troubleshooting or for verifying that the correct policies are being applied. Once the GPOs are applied, the CSEs will also generate an RSoP for the policies that have been applied.

To see the RSoP, run the command `sudo samba-gpupdate --rsop` for Machine policy, or `sudo samba-gpupdate --rsop --target=User -U<username>` to see User policy.

```
linux-h7xz:~ # samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_sec_ext
  -----------------------------
  -----------------------------
  CSE: gp_sec_ext
  -----------------------------
  -----------------------------
  CSE: gp_scripts_ext
```

```
  ------------------------------
  ------------------------------
  CSE: gp_sudoers_ext
  ------------------------------
    Policy Type: Sudo Rights
    ------------------------------
    [ tux ALL=(ALL) NOPASSWD: ALL ]
    ------------------------------
  ------------------------------
  CSE: gp_smb_conf_ext
  ------------------------------
    Policy Type: smb.conf
    ------------------------------
    [ apply group policies ] = 1
    [ client max protocol ] = SMB2_02
    ------------------------------
  ------------------------------
  CSE: gp_msgs_ext
  ------------------------------
    Policy Type: /etc/motd
    ------------------------------
This message is distributed by Samba!
    ------------------------------
    Policy Type: /etc/issue
    ------------------------------
Samba Group Policy \s \r \l
    ------------------------------
  ------------------------------
============================================================
```

3.4 Policies Introduced in this Book

In the upcoming chapters of this book, we will be discussing a variety of Group Policies that can be used to manage and configure systems in a Linux environment. These policies cover a wide range of settings, including security policies, user account settings, preferences, and many others.

This section will provide a brief overview of those policies. By the end of this book, the reader should have a good understanding of how to use these policies to manage and configure Linux systems.

3.4.1 smb.conf Policies

These policies distribute smb.conf global options to the client. These are found in the Group Policy Management Editor (GPME) under `Computer Configuration > Policies > Administrative Templates > Samba > smb.conf`. This policy is unable to apply idmap policies.

3.4.2 Password and Kerberos Policies

Password and Kerberos policies are found in the GPME under `Computer Configuration > Policies > OS Settings > Security Settings > Account Policy`. These policies are only applicable to Samba Active Directory Domain Controllers.

3.4.3 Script Policies

Script policies create cron jobs on client machines which execute the specified commands. These are found in the GPME under `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Scripts`.

3.4.3.1 Centrify Crontab Entries

Samba provides an extension which adds compatibility with Centrify's Crontab Entries Group Policy. If you are currently using Centrify Group Policy to distribute Crontab entry policies, these will automatically be applied by samba-gpupdate.

3.4.4 Startup Script Policies

Startup script policies allow you to upload the script that will be executed to the SYSVOL, as well as scheduling the command to run at startup. These scripts can be set using the `samba-tool gpo manage scripts startup` command.

3.4.5 Files Policy

The Files policy deploys files to client machines. These files are uploaded to the SYSVOL via the `samba-tool gpo manage files` command.

3.4.6 Symlink Policies

The symlink policy creates symbolic links on client machines. This policy is set via the `samba-tool gpo manage symlink` command. This policy is compatible with Vintela's Symlink Group Policy.

3.4.7 Sudoers Policies

Sudoers policies add sudo rules to client machines. These policies can be managed in the GPME under `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Sudo Rights`.

3.4.7.1 VGP Sudoers Policies

Another Sudoers extension is available for compatibility with Vintela's Sudoers Group Policy. The policy for this extension can be modified using the `samba-tool gpo manage sudo` command.

3.4.7.2 Centrify Sudoers Policies

A third Sudoers extension is available to provide compatibility with Centrify's Sudoers Group Policy. If you are currently using Centrify Group Policy to distribute Sudoers policies, these will automatically be applied by samba-gpupdate.

Samba Sudoers, VGP Sudoers, and Centrify Sudoers policies can be safely used in conjunction with one another, since these policies are non-overlapping.

3.4.8 Message Policies

Message policies set the contents of the /etc/motd and /etc/issue files on client machines. These policies can be managed in the GPME under `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Messages`.

3.4.8.1 VGP Message Policies

Other VGP Message extensions are available for compatibility with Vintela's MOTD and Issue Group Policies. The policies for these extensions can be modified using the `samba-tool gpo manage motd` and `samba-tool gpo manage issue` commands.

Beware that applying both the Samba and VGP message policies will cause unpredictable behavior, since both policies will apply and will overwrite one another.

3.4.9 PAM Access Policies

PAM Access policies set access rules within `/etc/security/access.d`. These policies are set using the `samba-tool gpo manage access` command. This policy is compatible with Vintela's Access Group Policy.

3.4.10 Certificate Auto Enrollment

Certificate Auto Enrollment allows devices to enroll for certificates from Active Directory Certificate Services. Certificate Auto Enrollment is available in Samba 4.16 and above.

3.4.11 Firefox Policy

Firefox policies can be administered using the Mozilla policy templates provided by Mozilla.

3.4.12 Chromium/Chrome Policy

Chromium and Google Chrome policies can be administered using the Chrome policy templates provided by Google.

3.4.13 GNOME Settings

GNOME Settings policies are found in the GPME under `Computer Configuration > Policies > Administrative Templates > Samba >`

GNOME. These policies manage some GNOME user settings, as described in the GNOME system admin guide, such as the compose key, screen dimming, online account management, extensions, and the ability to disable printing, file saving, command line access, fingerprint logon, logout, user switching, and repartitioning. There is also a general method for disabling any specific GNOME lockdown value.

3.4.14 OpenSSH Policy

OpenSSH policy applies settings to `/etc/ssh/sshd_config.d`. These policies can be set using the `samba-tool gpo manage openssh` command. These policies are compatible with Vintela's OpenSSH Group Policy.

3.4.15 Firewalld Policy

Firewalld policy applies firewall rules using the firewall-cmd command. These policies can be found in the GPME under `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Firewalld`.

4 Managing Group Policies

Before diving into specific Group Policies, let's review the basics of managing Group Policy.

A Windows Active Directory Administrator should know that Group Policies are managed via the Group Policy Management Console (GPMC). Unfortunately there isn't a similar tool for Samba Active Directory Administrators. The Samba project does provide the comprehensive `samba-tool gpo` command, which supplants much of the GPMC.

There are instances where Samba's Group Policy can only be managed via the GPMC, with no `samba-tool gpo` alternative. These cases will be highlighted in the text.

4.1 Opening a Group Policy Object in the Group Policy Management Console

To open the Default Domain Policy (for example) in the Group Policy Management Console:

1. Open the Group Policy Management Console by going to Start > Administrative Tools > Group Policy Management.

2. In the Group Policy Management Console, expand the Forest node, then expand the Domains node.

3. Select the domain that contains the Default Domain Policy.

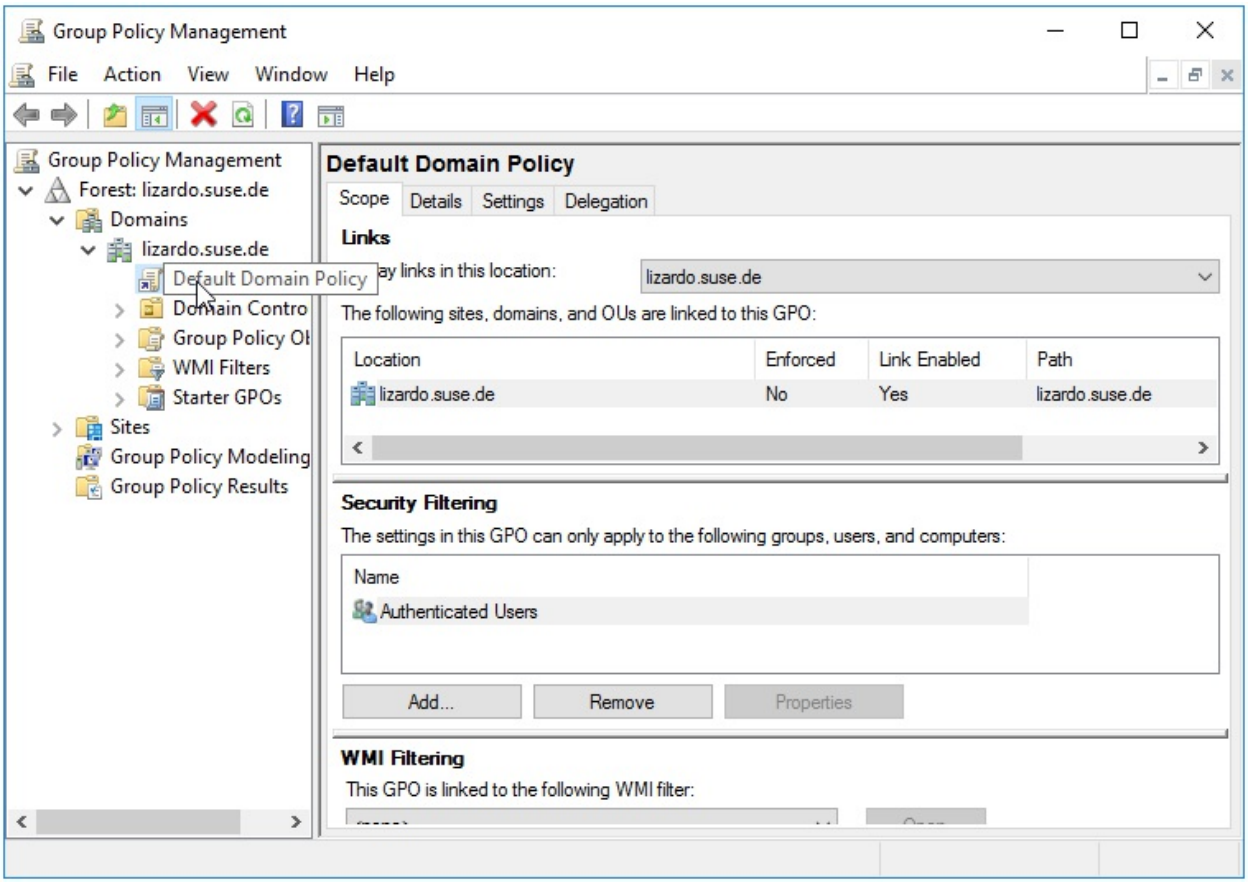


Figure 4.1: Group Policy Management Console

4. In the right pane, right-click on the Default Domain Policy and select “Edit” from the context menu.

5. The Group Policy Management Editor window will open, allowing you to view and edit the Default Domain Policy.

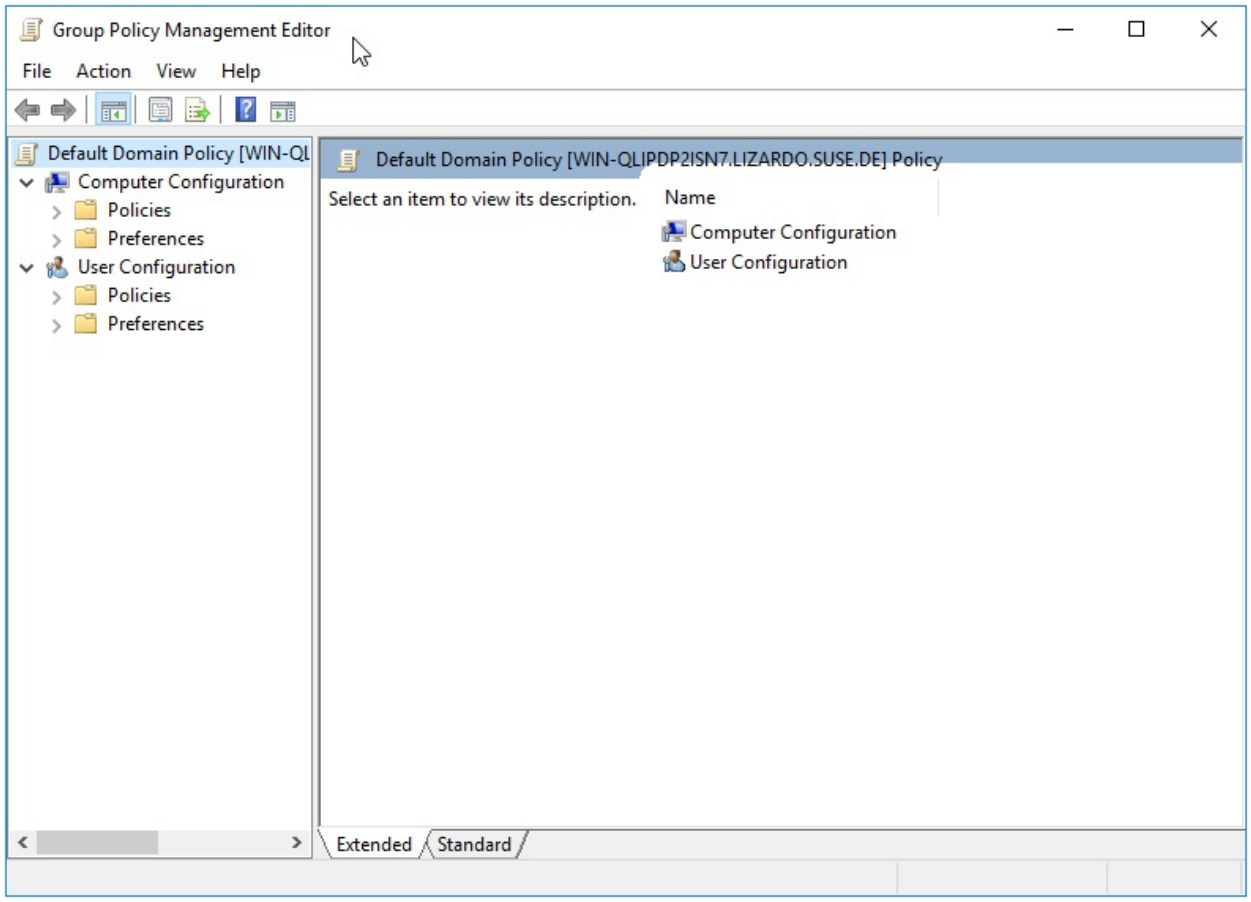


Figure 4.2: Default Domain Policy

Note: You will need to have the appropriate permissions to edit the Default Domain Policy. If you do not have the necessary permissions, you will not be able to edit the policy.

4.2 Creating a Group Policy Object

4.2.1 samba-tool

The `samba-tool gpo create` command is used to create a Group Policy Object (GPO). To create a GPO using the `samba-tool gpo create` command, you would use the following syntax:

```
samba-tool gpo create GPO_NAME
```

Where `GPO_NAME` is the name of the GPO that you want to create. This name

should be unique within the domain, as it will be used to identify the GPO when it is linked to a domain or organizational unit (OU).

Once the GPO has been created, you can use the `samba-tool gpo setlink` command to add or update a GPO link to a container. The syntax for this command is:

```
samba-tool gpo setlink CONTAINER_DN GPO_NAME
```

Where `CONTAINER_DN` is the distinguished name of the container from which you want to create the GPO link, and `GPO_NAME` is the name of the GPO you want to link.

4.2.2 GPMC

To create a Group Policy object (GPO) using the Group Policy Management Console (GPMC):

1. Open the GPMC by going to `Start > Administrative Tools > Group Policy Management`.

2. In the GPMC, expand the Domains node in the tree, and then expand the domain where you want to create the GPO.

3. Right-click on the domain, or the container where you'd like the GPO created and linked, and select `Create a GPO in this domain, and Link it here...`.

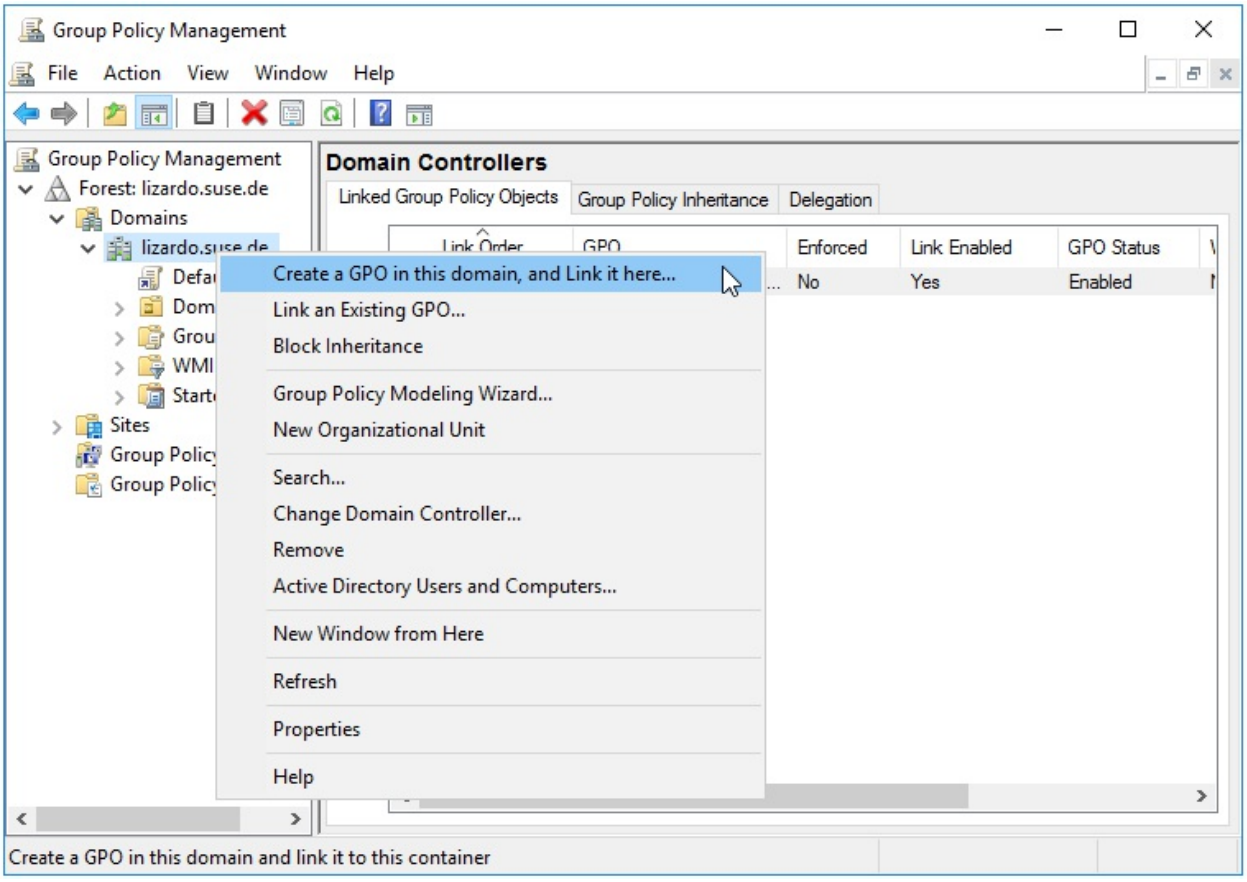


Figure 4.3: Create a GPO

4. In the `New GPO` dialog box, enter a name for the GPO and click on the `OK` button.

5. The new GPO will be created and will appear in the list of GPOs under the `Group Policy Objects` node.

4.3 Deleting a Group Policy Object

4.3.1 samba-tool

To delete a Group Policy Object (GPO), you would use the `samba-tool gpo del` command. The syntax for this command is:

```
samba-tool gpo del GPO_NAME
```

Where `GPO_NAME` is the name of the GPO you want to delete.

This command will delete the GPO from the server. Keep in mind that this operation cannot be undone, so make sure you really want to delete the GPO before running this command.

To delete a Group Policy Object (GPO) link from a container, you would use the `samba-tool gpo dellink` command. The syntax for this command is:

```
samba-tool gpo dellink CONTAINER_DN GPO_NAME
```

Where `CONTAINER_DN` is the distinguished name of the container from which you want to delete the GPO link, and `GPO_NAME` is the name of the GPO you want to unlink.

4.3.2 GPMC

To delete a Group Policy object (GPO) using the Group Policy Management Console (GPMC):

1. In the left pane of the GPMC, expand the forest and domain that contain the GPO you want to delete.

2. In the left pane, select the Group Policy Objects container. This will display a list of GPOs in the right pane.

3. In the right pane, right-click the GPO you want to delete and select “Delete.”

4. A warning message will appear, asking you to confirm that you want to delete the GPO. Click “Yes” to delete the GPO.

4.4 Listing a Group Policy

To list the contents of a Group Policy Object (GPO), you would use the `samba-tool gpo list` command. The syntax for this command is:

```
samba-tool gpo list GPO_NAME
```

Where `GPO_NAME` is the name of the GPO you want to list the contents of.

This command will list all of the settings and policies contained in the specified GPO.

4.5 Modifying a Group Policy

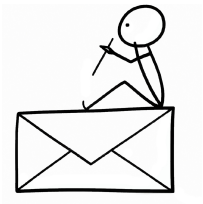
The `samba-tool gpo manage` is used for modifying settings on a Group Policy Object. See the individual chapters for each subcommand explanation.

Command	Chapter
samba-tool gpo manage smb_conf	4
samba-tool gpo manage security	5
samba-tool gpo manage scripts startup	7
samba-tool gpo manage files	8
samba-tool gpo manage symlink	9
samba-tool gpo manage sudoers	10
samba-tool gpo manage issue	11
samba-tool gpo manage motd	11
samba-tool gpo manage access	12
samba-tool gpo manage openssh	17

Each of these subcommands has its own set of options and parameters that can be used to specify the details of the operation. For more information on a specific subcommand, you can consult the documentation for that subcommand.

Additionally, the `samba-tool gpo load` and `samba-tool gpo remove` commands (see section 21.1) may be used to modify any policy which is loaded to the SYSVOL in a `Registry.pol` file. This applies to smb.conf policies, Script Policies, Sudoers Policies, Message Policies, Certificate Auto Enrollment Policy (advanced configuration only), Firefox Policy, Chromium/Chrome Policy, GNOME Settings Policy, and Firewalld Policy.

5 smb.conf Policies



The purpose of the smb.conf policies is to be able to distribute smb.conf settings to Linux clients. This policy only supports a physical smb.conf file, and currently does not support smb.conf registry settings.

These policies are physically stored on the SYSVOL in the **MACHINE/Registry.pol** file in the subdirectory of the Group Policy Object. They are stored in registry format, and are difficult to modify manually. See chapter 21 for details on how to manually modify this file.

5.1 Server Side Extension

The Server Side Extension for smb.conf policies is distributed using Administrative Templates (ADMX). Refer to chapter 20.1 in section 20.1.1 for details about Administrative Templates.

Setting up the ADMX templates for this policy is described in chapter 22 section 22.1.

5.1.1 Managing smb.conf Policies via the GPME

After successfully installing the ADMX templates, open the Group Policy Management Editor (GPME). For instructions on accessing the GPME, see chapter 4 section 4.1. For this example, we're going to enable the `apply group policies` setting.

1. In the left pane of the GPME, navigate to `Computer Configuration > Policies > Administrative Templates > Samba > smb.conf`.

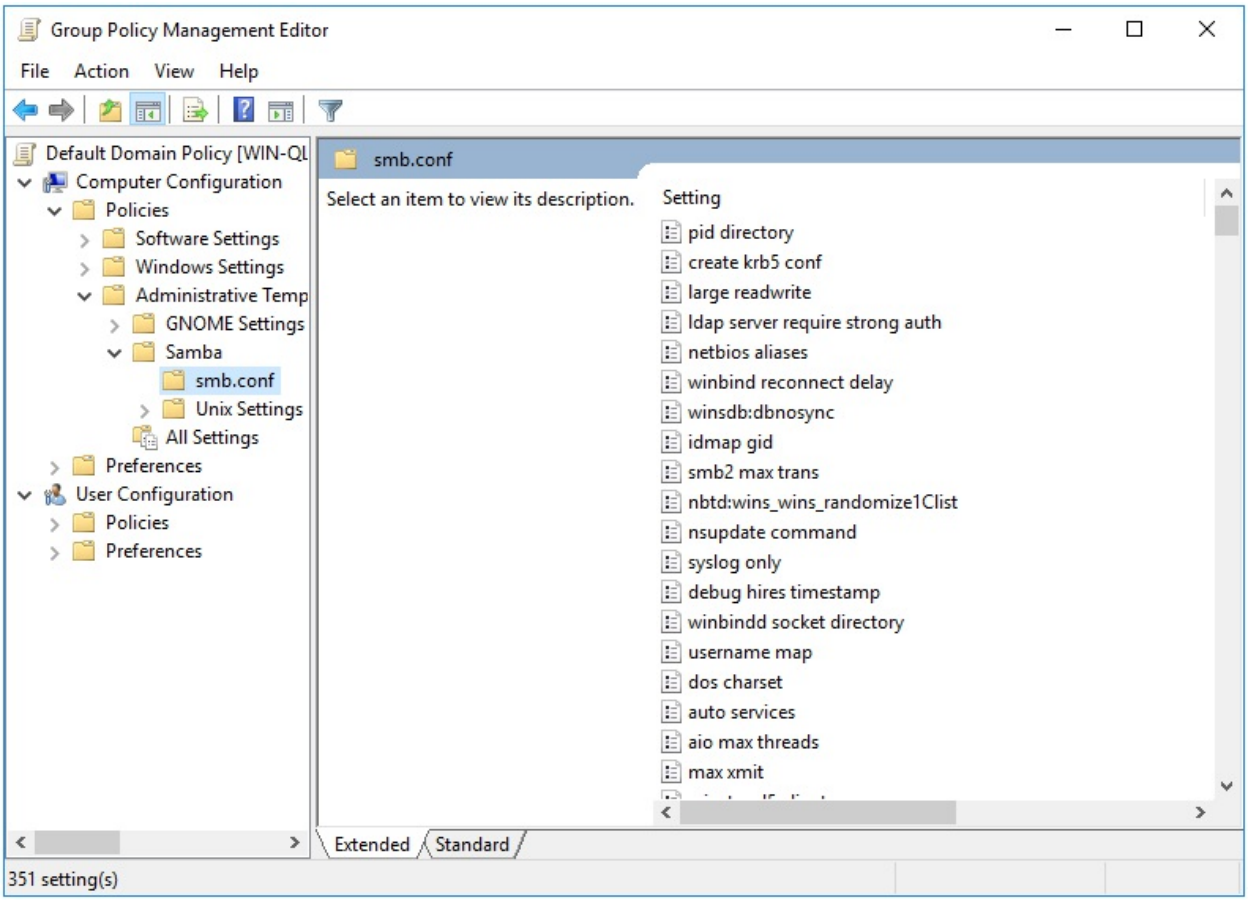


Figure 5.1: smb.conf Server Side Extension (ADMX)

2. In the right pane, double-click the “apply group policies” policy.
3. In the “apply group policies” dialog box, click the Enabled option.
4. Check the box next to “apply group policies”.
5. Click OK to close the “Apply group policies” dialog box.

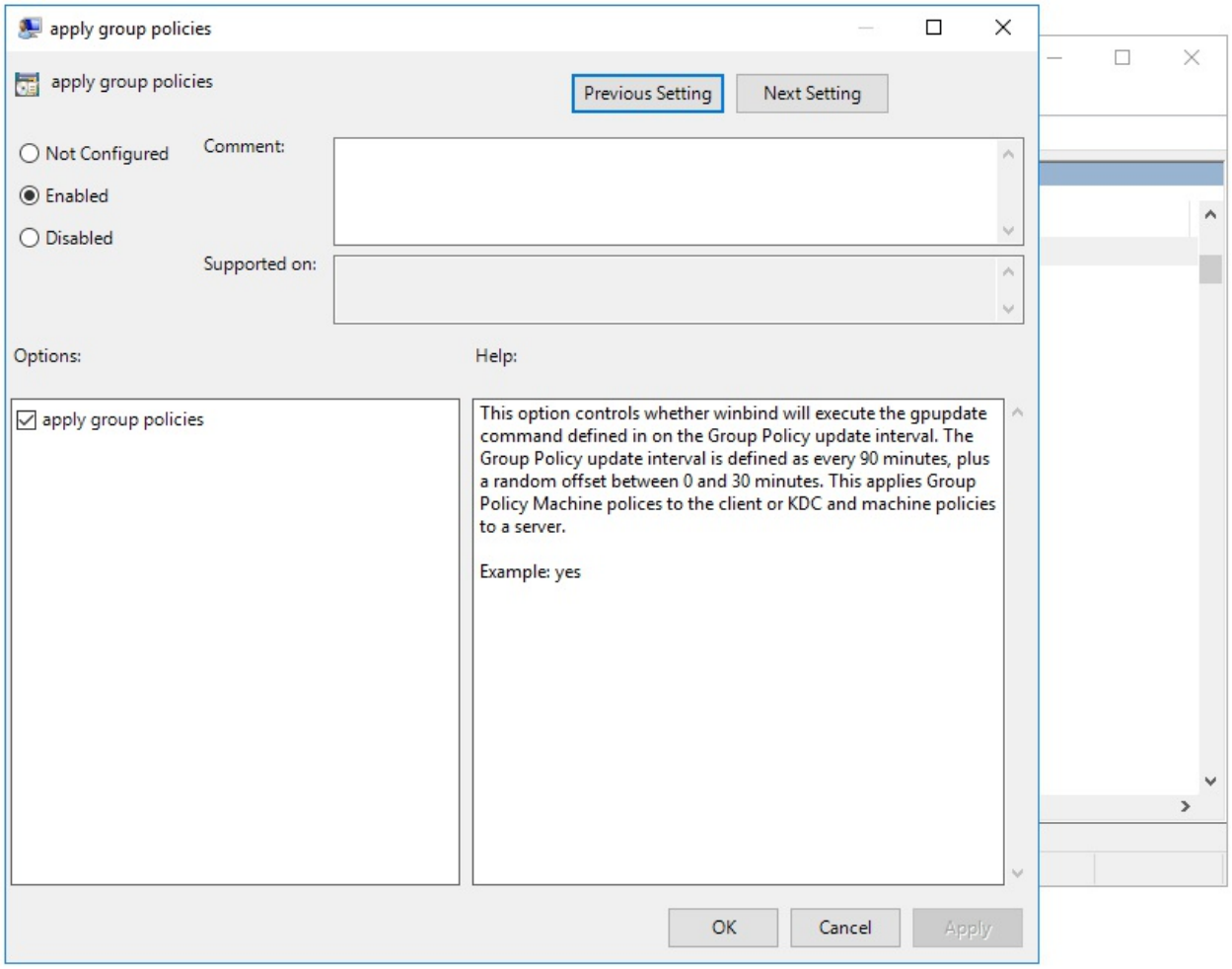


Figure 5.2: apply group policies Setting

Note: The `apply group policies` setting instructs Winbind to execute the `samba-gpupdate` command on the Group Policy interval (every 90 to 120 minutes). This allows you to apply Group Policy updates to Samba clients without having to log off and log back on.

There are many other settings available here, but notice that idmap policies are not. That's because idmap policies modify the setting name (not just the value), so these couldn't be included.

5.1.2 Managing smb.conf Policies via samba-tool

Setting an smb.conf Group Policy via `samba-tool gpo manage smb_conf` is arguably much simpler.

Use the `samba-tool gpo manage smb_conf set` command, providing the following arguments:

1. `<gpo>`: The name of the GPO that you want to modify.
2. `<setting>`: The name of the smb.conf setting that you want to set.
3. `<value>`: The value that you want to set for the specified setting.

For example, to set the `apply gpo policies` setting to `yes` in the GPO named `{31B2F340-016D-11D2-945F-00C04FB984F9}`, you would use the following command:

```
samba-tool gpo manage smb_conf set \
    {31B2F340-016D-11D2-945F-00C04FB984F9} 'apply gpo policies' y
```

If you want to unset a policy, you can omit the `<value>` argument. For example, to unset the `apply gpo policies` setting in the GPO named `{31B2F340-016D-11D2-945F-00C04FB984F9}`, you would use the following command:

```
samba-tool gpo manage smb_conf set \
    {31B2F340-016D-11D2-945F-00C04FB984F9} 'apply gpo policies'
```

This command does not require the ADMX templates to be installed, and also does not suffer from the same limitation as the GPME for idmap policies.

5.2 Client Side Extension

The smb.conf Client Side Extension (CSE) directly modifies the default smb.conf file. Any custom formatting or comments in the smb.conf file may be overwritten. The CSE will open your existing smb.conf file, read in the current settings, set the settings provided by the GPO, then write the file back to disk. This CSE will only write `global` smb.conf options.

In the previous section, we enabled the `apply group policies` smb.conf option. If we now go to our Linux client, and check the Resultant Set of Policy, we see this:

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_smb_conf_ext
  ------------------------------------------------------------
    Policy Type: smb.conf
    ----------------------------------------------------------
    [ apply group policies ] = 1
    ----------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

If we now force the policy, we'll see our setting gets applied to the default smb.conf:

```
> sudo /usr/sbin/samba-gpupdate --force
> diff -u /etc/samba/smb.conf.BAK /etc/samba/smb.conf
--- /etc/samba/smb.conf.BAK
+++ /etc/samba/smb.conf
@@ -1,5 +1,6 @@
 # Global parameters
 [global]
+   apply group policies = Yes
    kerberos method = secrets and keytab
    logon drive = P:
    logon home = \\%L\%U\.9xprofile
```

If for whatever reason the policy did not apply, it is sometimes helpful to look at the Group Policy Cache, which keeps track of applied policies.

```
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
| sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
| xmllint --xpath "//gp_ext[@name='smb.conf']" - \
| xmllint --format -
<?xml version="1.0"?>
<gp_ext name="smb.conf">
  <attribute name="apply group policies">yes</attribute>
</gp_ext>
```

Where **TESTSYSDM$** is the system name. You can see in our case Samba has recorded applying the Group Policy Object, and that it set `apply group policies = yes` in our smb.conf.

6 Password and Kerberos Policies

The purpose of these policies is to enforce password complexity and kerberos rules on a Samba Active Directory Domain Controller (ADDC). When a Linux client is not an ADDC, these policies are disabled and ignored automatically.

These policies are physically stored on the SYSVOL in the **MACHINE/Microsoft** */Windows NT/SecEdit/GptTmpl.inf** file in the subdirectory of the Group Policy Object. They are stored in ini format, and are easily modified manually using a text editor.

6.1 Server Side Extension

The Group Policy Management Editor (GPME) contains a built in Server Side Extension for Password and Kerberos Policies. There is also a `samba-tool` command to modify these policies.

6.1.1 Managing Password and Kerberos Policies via the GPME

Open the GPME and navigate to `Computer Configuration > Policies > Windows Settings > Security Settings > Account Policy`.

6.1.1.1 Password Policies

The following password policies are applicable to a Samba ADDC:

- Maximum password age
- Minimum password age
- Minimum password length
- Password must meet complexity requirements

To set password policy settings using the Group Policy Management Editor (GPME), follow these steps:

1. Open the Group Policy Management Editor (GPME). For instructions on accessing the GPME, see chapter 4 section 4.1.

2. In the left pane of the GPME, navigate to Computer Configuration > Policies > Windows Settings > Security Settings > Account Policy > Password Policy.

3. In the right pane, double-click the “Maximum password age” policy.

4. In the “Maximum password age” dialog box, click the Enabled option and enter 42 in the “Value” field.

5. Click OK to close the “Maximum password age” dialog box.

6. In the right pane, double-click the “Minimum password age” policy.

7. In the “Minimum password age” dialog box, click the Enabled option and enter 1 in the “Value” field.

8. Click OK to close the “Minimum password age” dialog box.

9. In the right pane, double-click the “Minimum password length” policy.

10. In the “Minimum password length” dialog box, click the Enabled option and enter 7 in the “Value” field.

11. Click OK to close the “Minimum password length” dialog box.

12. In the right pane, double-click the “Password must meet complexity requirements” policy.

13. In the “Password must meet complexity requirements” dialog box, click the Enabled option.

14. Click OK to close the “Password must meet complexity requirements” dialog box.

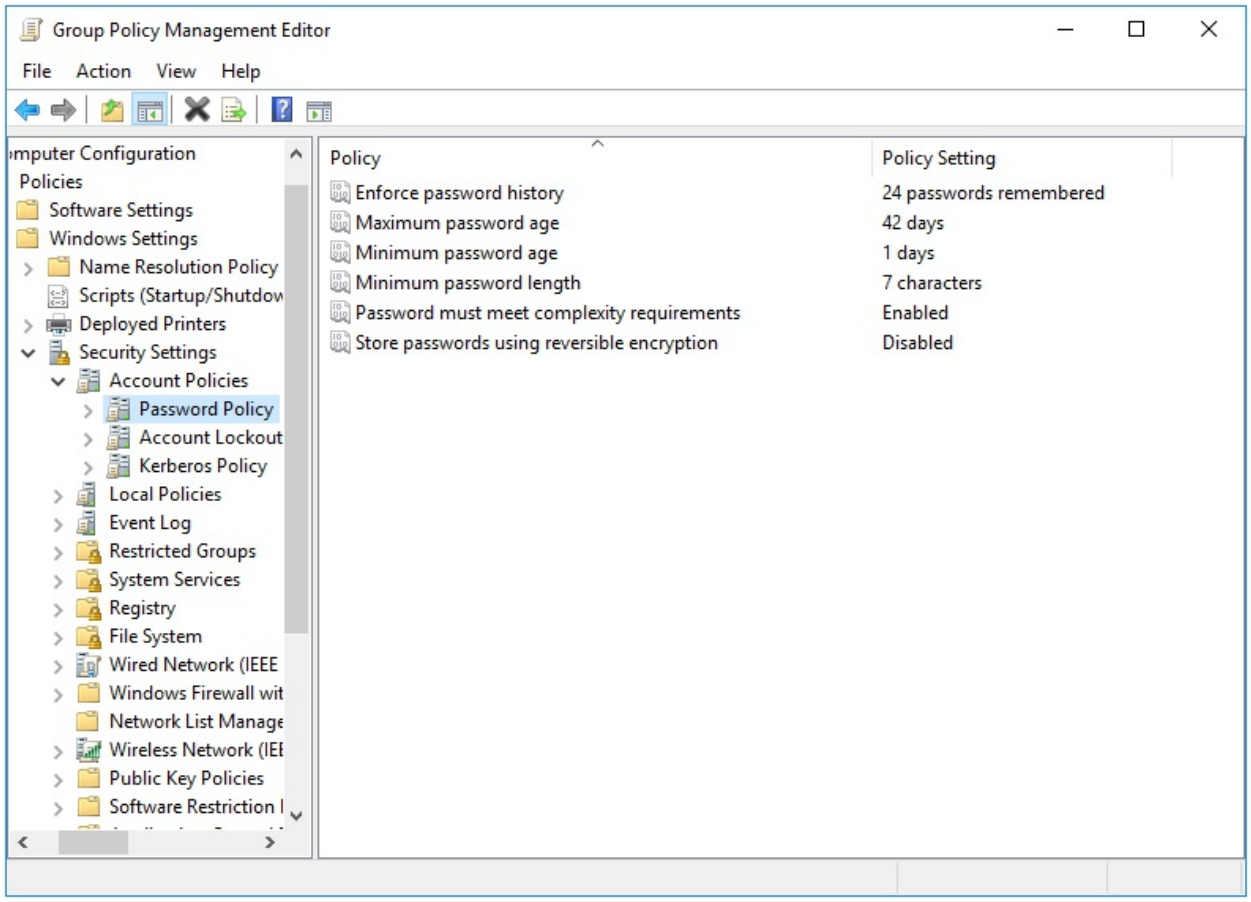


Figure 6.1: Password Policies

6.1.1.2 Kerberos Policies

The following Kerberos policies are applicable to a Samba ADDC:

- Maximum lifetime for service ticket
- Maximum lifetime for user ticket
- Maximum lifetime for user ticket renewal

To set Kerberos policy settings, follow these steps:

1. In the left pane of the GPME, navigate to `Computer Configuration > Policies > Windows Settings > Security Settings > Account Policy > Kerberos Policy`.

2. In the right pane, double-click the “Maximum lifetime for service ticket” policy.

3. In the “Maximum lifetime for service ticket” dialog box, click the Enabled option and enter 600 in the “Value” field.

4. Click OK to close the “Maximum lifetime for service ticket” dialog box.

5. In the right pane, double-click the “Maximum lifetime for user ticket” policy.

6. In the “Maximum lifetime for user ticket” dialog box, click the Enabled option and enter 10 in the “Value” field.

7. Click OK to close the “Maximum lifetime for user ticket” dialog box.

8. In the right pane, double-click the “Maximum lifetime for user ticket renewal” policy.

9. In the “Maximum lifetime for user ticket renewal” dialog box, click the Enabled option and enter 7 in the “Value” field.

10. Click OK to close the “Maximum lifetime for user ticket renewal” dialog box.

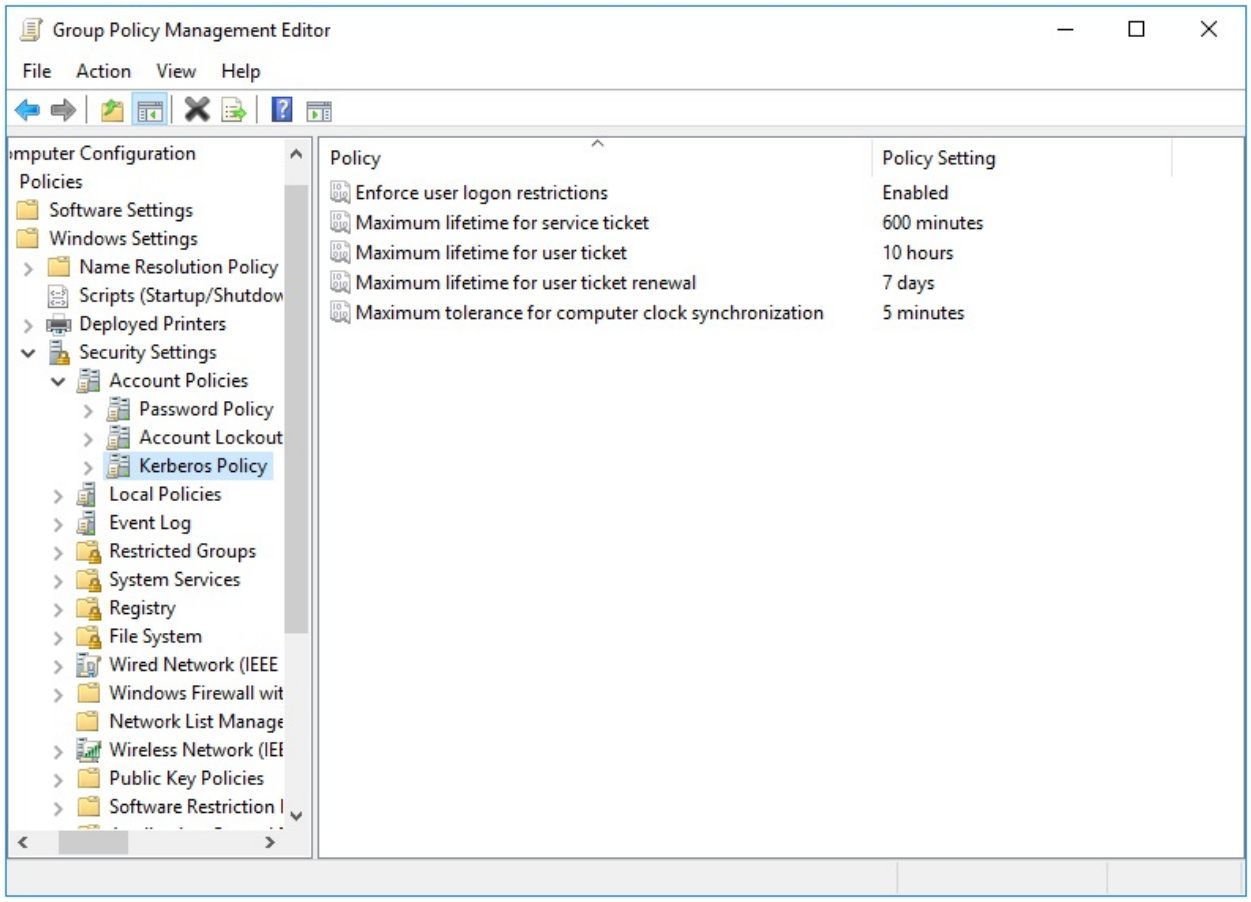


Figure 6.2: Kerberos Policies

6.1.2 Managing Password and Kerberos Policies via samba-tool

The Password and Kerberos policies can also be set via `samba-tool gpo manage security set <gpo> <setting> <value>`.

The command accepts the following parameters:

1. `<gpo>`: The name of the GPO that you want to modify.
2. `<setting>`: The name of the smb.conf setting that you want to set.
3. `<value>`: The value that you want to set for the specified setting.

The `setting` parameter must be one of the following:

Setting	Description
MaxTicketAge	Maximum lifetime for user ticket
MaxServiceAge	Maximum lifetime for service ticket
MaxRenewAge	Maximum lifetime for user ticket renewal
MinimumPasswordAge	Minimum password age
MaximumPasswordAge	Maximum password age
MinimumPasswordLength	Minimum password length
PasswordComplexity	Password must meet complexity requirements

6.2 Client Side Extension

The Password and Kerberos policies are separated into two Client Side Extensions (CSE), `gp_access_ext` and `gp_krb_ext`. The Password policies (internally known as *System Access*) apply password complexity rules directly to the ADDC SamDB in the applicable LDAP attributes. The Kerberos policies are stored in our Group Policy Cache, and are fetched directly by the `samba` daemon when it loads.

In the previous section we saw that our Password and Kerberos policies were already initialized to some defaults. If we now go to our Linux client, and check the Resultant Set of Policy, we see this:

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_access_ext
  ------------------------------------------------------------
    Policy Type: System Access
    ----------------------------------------------------------
    [ MinimumPasswordAge ] = 1
    [ MaximumPasswordAge ] = 42
    [ MinimumPasswordLength ] = 7
    [ PasswordComplexity ] = 1
    ----------------------------------------------------------
  ------------------------------------------------------------
  CSE: gp_krb_ext
```

```
  ------------------------------------------------------------
    Policy Type: Kerberos Policy
    ------------------------------------------------------------
    [ MaxTicketAge ] = 10
    [ MaxRenewAge ] = 7
    [ MaxServiceAge ] = 600
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

Remember that these policies will only be listed if your Linux client is a Samba ADDC.

Let's now force a policy apply, and check that the settings have changed.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='System Access' or
                             @name='Kerberos Policy']"  -
<gp_ext name="System Access">
  <attribute name="minPwdAge"/>
  <attribute name="maxPwdAge"/>
  <attribute name="minPwdLength"/>
  <attribute name="pwdProperties"/>
</gp_ext>
<gp_ext name="Kerberos Policy">
  <attribute name="kdc:user_ticket_lifetime"/>
  <attribute name="kdc:renewal_lifetime"/>
  <attribute name="kdc:service_ticket_lifetime"/>
</gp_ext>
> sudo tdbdump /var/lib/samba/gpo.tdb \
 -k 'kdc:user_ticket_lifetime'; echo
10
> sudo tdbdump /var/lib/samba/gpo.tdb \
 -k 'kdc:service_ticket_lifetime'; echo
10
> sudo tdbdump /var/lib/samba/gpo.tdb \
 -k 'kdc:renewal_lifetime'; echo
168
```

You can see the Kerberos policies are all stored in their own keys in the Group Policy Cache (`/var/lib/samba/gpo.tdb`), and they are all stored in hours. On the SYSVOL, they are actually stored in hours, minutes, and days, respectively. The reason these are all stored in hours on the system is that the

samba daemon expects these attributes in hours.

If we check the contents of the GptTmpl.inf and do some conversion, we can confirm these are correct.

```
[Kerberos Policy]
MaxTicketAge = 10
MaxServiceAge = 600
MaxRenewAge = 7
```

```
>>> from samba.gp.gp_sec_ext import mins_to_hours, days_to_hours
>>> MaxTicketAge = 10
>>> MaxServiceAge = 600
>>> MaxRenewAge = 7
>>> mins_to_hours(MaxServiceAge)
'10'
>>> days_to_hours(MaxRenewAge)
'168'
```

It also helps to know how these Kerberos policies map to the samba daemon settings.

Kerberos.Policy	Samba.Setting	Conversion
MaxTicketAge	kdc:user_ticket_lifetime	None
MaxServiceAge	kdc:service_ticket_lifetime	Minutes to Hours
MaxRenewAge	kdc:renewal_lifetime	Days to Hours

While the Kerberos policies have been stored to the Group Policy Log as expected, let's next verify that the Password policies have been applied using the following ldapsearch.

```
> ldapsearch -H ldap://lizardo.suse.de -x -W \
 -D "Administrator@lizardo.suse.de" \
 -b DC=lizardo,DC=suse,DC=de \
 -s base minPwdAge maxPwdAge minPwdLength pwdProperties
# lizardo.suse.de
dn: DC=lizardo,DC=suse,DC=de
maxPwdAge: -36288000000000
minPwdAge: -864000000000
minPwdLength: 7
pwdProperties: 1
```

We can confirm these were set correctly by checking the contents of the GptTmpl.inf, plus doing some type conversion.

```
[System Access]
MinimumPasswordAge = 1
MaximumPasswordAge = 42
MinimumPasswordLength = 7
PasswordComplexity = 1
```

```
>>> from samba.gp.gp_sec_ext import days2rel_nttime
>>> MinimumPasswordAge = 1
>>> MaximumPasswordAge = 42
>>> MinimumPasswordLength = 7
>>> PasswordComplexity = 1
>>> days2rel_nttime(MaximumPasswordAge)
'-36288000000000'
>>> days2rel_nttime(MinimumPasswordAge)
'-864000000000'
```

Finally, here is how these policies map.

Password.Policy	LDAP.Attribute	Conversion
MinimumPasswordAge	minPwdAge	Days to NTTIME
MaximumPasswordAge	maxPwdAge	Days to NTTIME
MinimumPasswordLength	minPwdLength	None
PasswordComplexity	pwdProperties	None

7 Script Policies

The purpose of this policy is to schedule cron jobs on a Linux client. Both Machine and User policy is supported. This policy does not upload a script for execution, it only schedules an existing script to run. To first load a script onto the client, see the Files Policy in chapter 9.

This policy is physically stored on the SYSVOL in the **MACHINE/Registry.pol** and **USER/Registry.pol** files within the subdirectory of the Group Policy Object. It is stored in registry format. See chapter 21 for details on how to manually modify this file.

7.1 Server Side Extension

The Server Side Extension for smb.conf policies is distributed using Administrative Templates (ADMX). Refer to chapter 20.1 in section 20.1.1 for details about Administrative Templates.

Setting up the ADMX templates for this policy is described in chapter 22 section 22.1.

7.1.1 Managing Machine Scripts Policies via the GPME

As an example, let's create a simple cron job which echo's "hello world" once every day.

1. Open the Group Policy Management Editor (GPME). For instructions on accessing the GPME, see chapter 4 section 4.1.

2. In the left pane of the GPME, navigate to `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Scripts`.

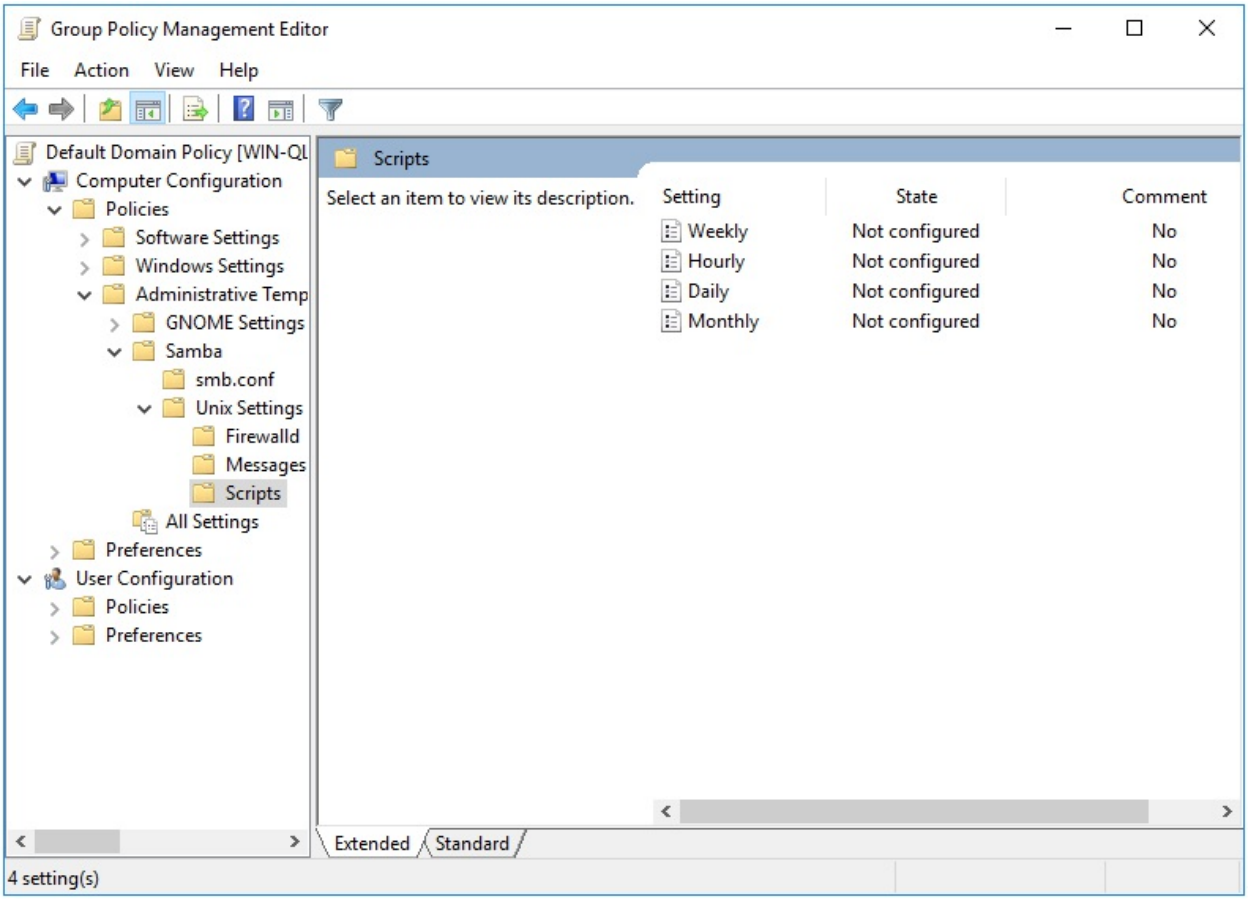


Figure 7.1: Scripts Server Side Extension (ADMX)

3. In the right pane, double-click the “Daily” policy.

4. In the “Daily” dialog box, click the Enabled option and then click the Show button.

5. In the “Show Contents” dialog box, type the following script in the “Value” field:

```
echo "hello world"
```

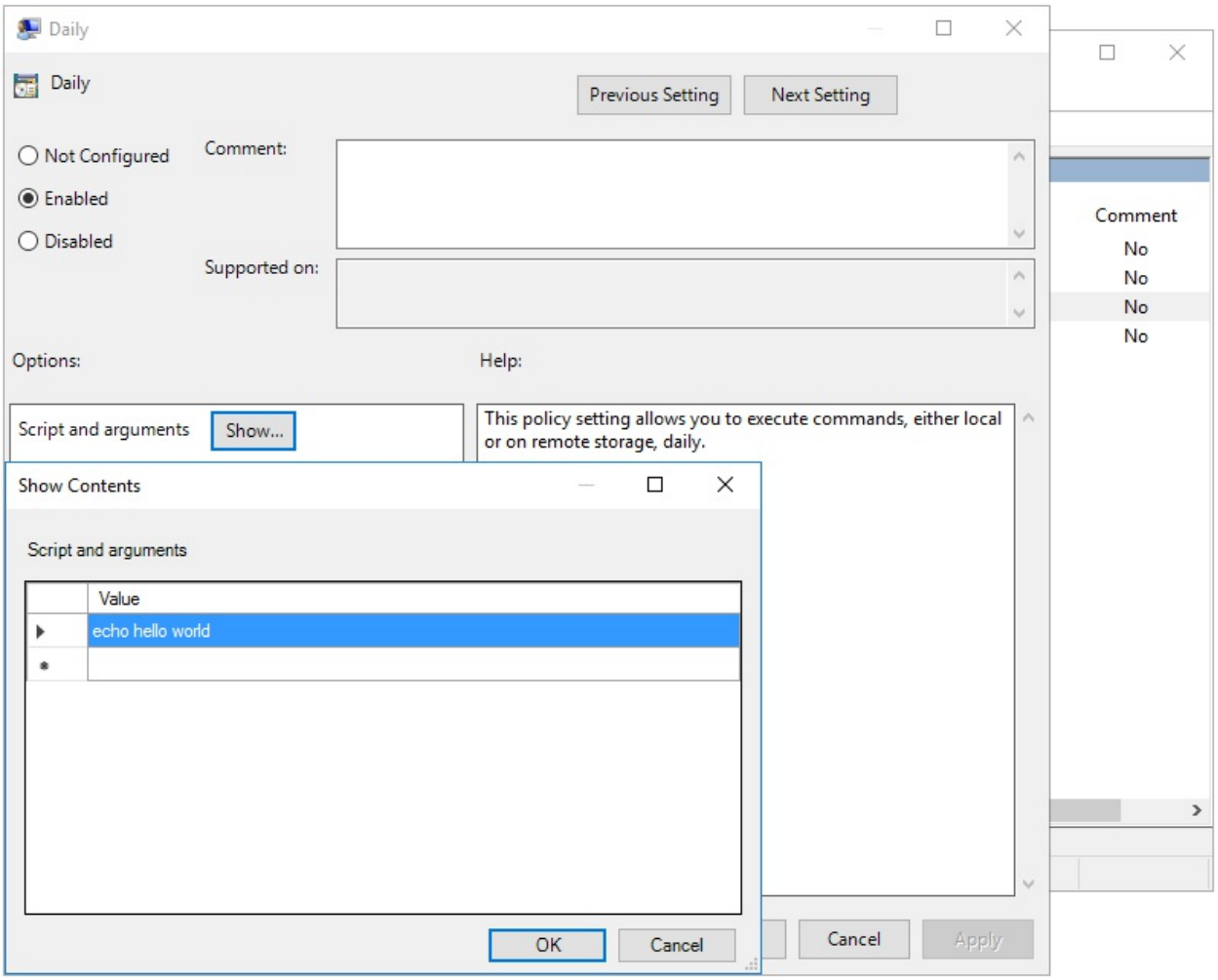


Figure 7.2: Script Example

6. Click OK to close the “Show Contents” dialog box, and then click OK again to close the “Daily” dialog box.

7.1.2 Managing User Scripts Policies via the GPME

Next we’ll create a user script that echo’s the text “Don’t do that Dave” every hour.

1. In the left pane of the GPME, navigate to `User Configuration > Policies > Administrative Templates > Samba > Unix Settings > Scripts`.

2. In the right pane, double-click the “Hourly” policy.

3. In the “Hourly” dialog box, click the Enabled option and then click the Show button.

4. In the “Show Contents” dialog box, type the following script in the “Value” field:

```
echo "Don't do that Dave"
```

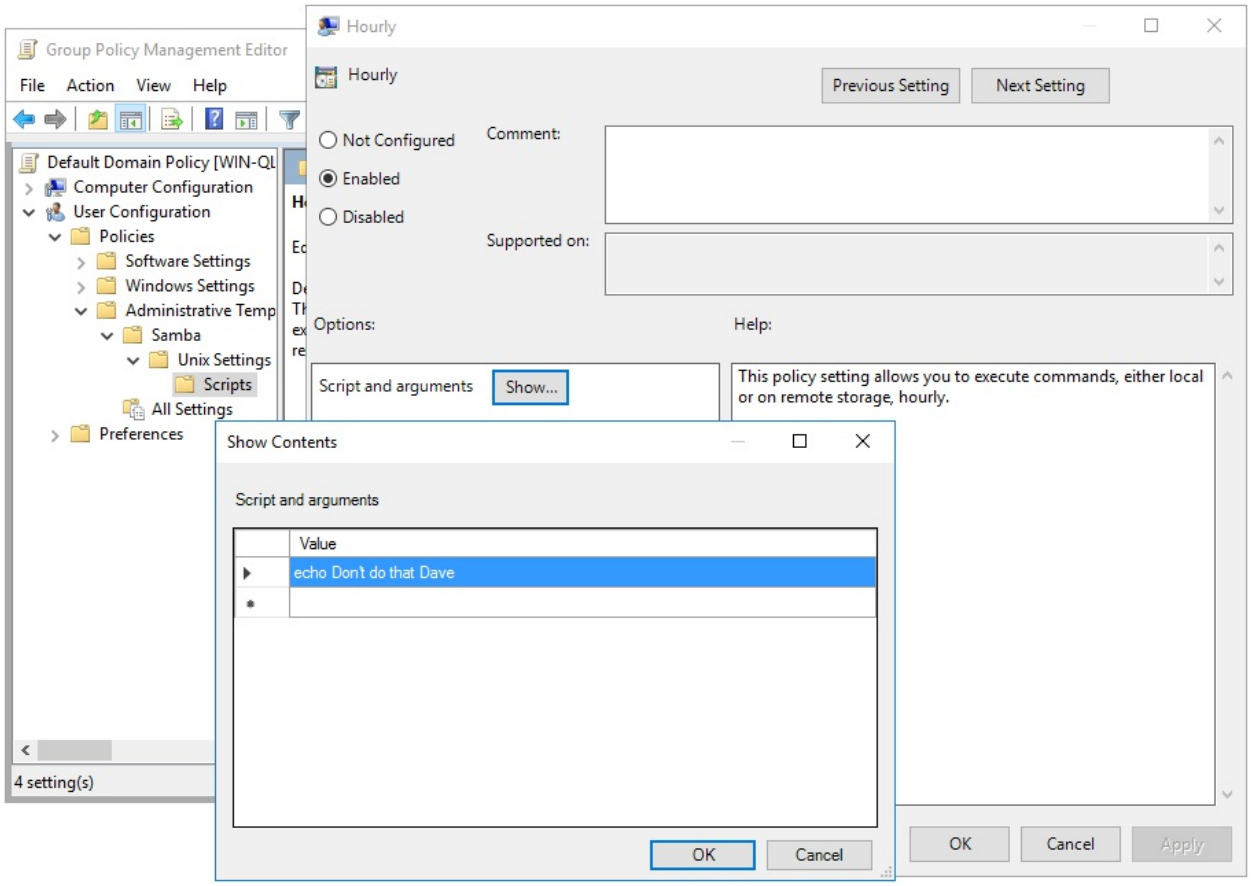


Figure 7.3: User Script Example

5. Click OK to close the “Show Contents” dialog box, and then click OK again to close the “Hourly” dialog box.

The quote “Don’t do that Dave” is a line spoken by the character HAL 9000 in the science fiction film “2001: A Space Odyssey.” HAL is a sentient computer that controls the systems of a spacecraft, and the quote is spoken in a scene where HAL is attempting to prevent one of the astronauts from disconnecting its memory.

7.2 Client Side Extension

The Scripts Client Side Extension (CSE) creates cron jobs on the Linux client. For Machine policy, these jobs are installed in a file within the `/etc/cron.daily`, `/etc/cron.monthly`, `/etc/cron.weekly` and `/etc/cron.hourly` directories. For User policy, the user's crontab file is directly modified.

In the previous section we created two test Script policies. If we now go to our Linux client, and check the Resultant Set of Policy, we see this:

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_scripts_ext
  ------------------------------------------------------------
    Policy Type: Daily Scripts
    ----------------------------------------------------------
    [ echo hello world ]
    ----------------------------------------------------------
  ------------------------------------------------------------
  CSE: gp_centrify_crontab_ext
  ------------------------------------------------------------
    Policy Type: Centrify/CrontabEntries
    ----------------------------------------------------------
    [ @daily echo hello world from Centrify ]
    ----------------------------------------------------------
  ------------------------------------------------------------
============================================================
> sudo /usr/sbin/samba-gpupdate --target=User -U tux --rsop
Resultant Set of Policy
User Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_user_scripts_ext
  ------------------------------------------------------------
    Policy Type: Hourly Scripts
    ----------------------------------------------------------
    [ echo Don't do that Dave ]
    ----------------------------------------------------------
```

```
  ------------------------------------------------------------
  CSE: gp_user_centrify_crontab_ext
  ------------------------------------------------------------
    Policy Type: Centrify/CrontabEntries
    ------------------------------------------------------------
    [ @hourly echo Don't do that Dave from Centrify ]
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

In addition to the expected scripts that we added previously, you'll notice there are 2 additional entries. The `gp_centrify_crontab_ext` and `gp_user_centrify_crontab_ext` CSEs parse policies provided by a Centrify Server Side Extension. These weren't introduced previously in the chapter because they are a proprietary solution not provided by Samba. Samba provides a CSE to apply these for compatability reasons, but does not provide a SSE to set them. These CSEs are provided to assist in migration from proprietary technologies. We won't discuss these any further.

Let's now force an apply, and verify that the cron jobs are scheduled.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Unix Settings/Scripts']" - \
 | xmllint --format -
<?xml version="1.0"?>
<gp_ext name="Unix Settings/Scripts">
  <attribute name="Software\Policies\Samba\Unix
               Settings\Daily Scripts:ZWNobyBoZWxsbyB3b3JsZA==">
    /etc/cron.daily/gp_m94kdru9
  </attribute>
</gp_ext>
> sudo /usr/sbin/samba-gpupdate --target=User -U tux --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "LIZARDO\\tux" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Unix Settings/Scripts']" - \
 | xmllint --format -
<?xml version="1.0"?>
<gp_ext name="Unix Settings/Scripts">
  <attribute name="Software\Policies\Samba\Unix
                  Settings\Hourly Scripts:94d6...e415">
    @hourly echo Don't do that Dave
  </attribute>
</gp_ext>
```

First we see that the machine policy created the script `/etc/cron.daily/gp_m94kdru9`. Let's take a look at the contents.

```
> sudo cat /etc/cron.daily/gp_m94kdru9
#!/bin/sh

### autogenerated by samba
#
# This file is generated by the gp_scripts_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

echo hello world
```

Next we notice that the user policy created the entry `@hourly echo Don't do that Dave`. If we inspect the crontab of the user `tux`, we see the entry.

```
> sudo crontab -l -u LIZARDO\\tux
### autogenerated by samba
#
# This file is generated by the gp_scripts_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

@hourly echo Don't do that Dave

### autogenerated by samba ###
```

8 Startup Script Policies

Startup scripts earn themselves a chapter independant of the Script Polices chapter because they are so thoroughly different. The Startup Scripts Policy started as a compatability layer for Vintela's Startup Scripts, but became the defacto standard for Samba also. This policy has no Server Side Extension in the Group Policy Management Editor (GPME), but only provides a `samba-tool` command for setting the policy. A nice advantage to using this policy is that when using the `samba-tool gpo manage` command to set the policy, it automatically uploads the script file you specify to the SYSVOL.

This policy is physically stored on the SYSVOL in the **MACHINE/VGP /VTLA/Unix/Scripts/Startup/manifest.xml** file in the subdirectory of the Group Policy Object. They are stored in an xml format, and are easily modified manually using a text editor.

8.1 Server Side Extension

Startup Script Policies have no GPME Server Side Extension (SSE), so this policy may only be administered using `samba-tool gpo manage scripts startup`. This is because this SSE is stored on the SYSVOL as an xml file, not in the Registry.pol from an ADMX template.

8.1.1 Managing Startup Script Policies via samba-tool

The Startup Scripts `samba-tool` command has 3 subcommands; add, list, and remove.

```
> samba-tool gpo manage scripts startup --help
Usage: samba-tool gpo manage scripts startup <subcommand>
```

```
Manage Startup Scripts Group Policy Objects


Options:
  -h, --help  show this help message and exit


Available subcommands:
  add     - Adds VGP Startup Script Group Policy to the sysvol
  list    - List VGP Startup Script Group Policy from the sysvol
  remove  - Removes VGP Startup Script Group Policy from the
sysvol
```

To add a new Startup Script policy to the SYSVOL, call the `samba-tool gpo manage scripts startup add` command.

```
samba-tool gpo manage scripts startup add <gpo> <script> [args]
 [run_as]
```

The `samba-tool gpo manage scripts startup add` command is used to add a startup script policy to the SYSVOL. The command takes a Group Policy Object (GPO) identifier and a script file as arguments, as well as optional arguments for script arguments, and the user to run the script as.

When adding a script, you pass the relative path to an existing script file. This script will be uploaded to the SYSVOL and made available to clients for execution. You can also provide an optional set of arguments that will be passed to the script when it is executed. These arguments are parsed as a single argument to the command, so they must be wrapped in quotes and all dashes (‘-’) must be escaped. There is also an optional argument `run_as` to instruct the client to run the script as a specific user. The optional `--run-once` parameter can instruct the script to execute only once, on the next startup, and not again.

Let’s add a simple test script now which echos a message, and takes no arguments. By default this command will run as root.

```
> cat test_script.sh
#!/bin/sh

echo Something is happening here at startup
> samba-tool gpo manage scripts startup add \
```

```
 {31B2F340-016D-11D2-945F-00C04FB984F9} test_script.sh \
 -UAdministrator
> samba-tool gpo manage scripts startup list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
@reboot root \\lizardo.suse.de\Policies\
 {31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\VGP\VTLA\Unix\
 Scripts\Startup\test_script.sh
```

Notice that the path to the script is now pointing to an uploaded copy on the SYSVOL.

If we mount the SYSVOL, we can take a look at the xml file created by the policy.

```
> sudo mount.cifs \\\\lizardo.suse.de\\SYSVOL /mnt/ \
 -ouser=Administrator
> cat /mnt/lizardo.suse.de/Policies/
 {31B2F340-016D-11D2-945F-00C04FB984F9}/MACHINE/VGP/VTLA/Unix/
 Scripts/Startup/manifest.xml | xmllint --format -
<?xml version="1.0" encoding="UTF-8"?>
<vgppolicy>
  <policysetting>
    <version>1</version>
    <name>Unix Scripts</name>
    <description>
      Represents Unix scripts to run on Group Policy clients
    </description>
    <data>
      <listelement>
        <script>test_script.sh</script>
        <hash>3F1F0449B3070AD113B2878C751C4887</hash>
      </listelement>
    </data>
  </policysetting>
</vgppolicy>
```

If you wanted to remove this policy later, we would issue the `samba-tool gpo manage scripts startup remove` command.

```
> samba-tool gpo manage scripts startup remove \
 {31B2F340-016D-11D2-945F-00C04FB984F9} \
 -UAdministrator test_script.sh
```

8.2 Client Side Extension

The Startup Scripts Client Side Extension (CSE) creates `@reboot` cron jobs on the Linux client. Startup Scripts only apply for Machine policy.

We created a test script in the previous section. If we now list the Resultant Set of Policy on the client, we see this:

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: vgp_startup_scripts_ext
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/Startup Scripts
    ----------------------------------------------------------
    [ @reboot root /var/lib/samba/gpo_cache/LIZARDO.SUSE.DE/
                   POLICIES/{31B2F340-016D-11D2-945F-
00C04FB984F9}/
           MACHINE/VGP/VTLA/UNIX/SCRIPTS/STARTUP/
           TEST_SCRIPT.SH  ]
    ----------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

If we now force the policy to apply, we'll see our script is scheduled to execute using a cron job.

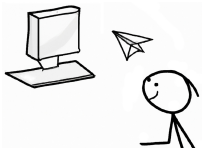
```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
> | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" | \
 xmllint --xpath "//gp_ext[@name='VGP/Unix Settings/
                                  Startup Scripts']" - | \
 xmllint --format -
<?xml version="1.0"?>
<gp_ext name="VGP/Unix Settings/Startup Scripts">
  <attribute
   name="test_script.sh:3F1F0449B3070AD113B2878C751C4887:">
    /etc/cron.d/gp_vzldfcii
  </attribute>
</gp_ext>
> sudo cat /etc/cron.d/gp_vzldfcii

### autogenerated by samba
#
```

```
# This file is generated by the vgp_startup_scripts_ext Group
# Policy Client Side Extension. To modify the contents of this
# file, modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

@reboot root /var/lib/samba/gpo_cache/LIZARDO.SUSE.DE/POLICIES/
 {31B2F340-016D-11D2-945F-00C04FB984F9}/MACHINE/VGP/VTLA/UNIX/
 SCRIPTS/STARTUP/TEST_SCRIPT.SH
```

You can see we found the location of our cron job by outputting the applied policy from our Group Policy Cache. The cron job instructs the script to execute as root on every reboot. Let’s also verify our script has been deployed on the client.

```
> sudo cat /var/lib/samba/gpo_cache/LIZARDO.SUSE.DE/POLICIES/
 {31B2F340-016D-11D2-945F-00C04FB984F9}/MACHINE/VGP/VTLA/UNIX/
 SCRIPTS/STARTUP/TEST_SCRIPT.SH
#!/bin/sh

echo Something is happening here at startup
```

9 Files Policy



The Files Policy is useful to use in conjunction with the Scripts Policy, since it can be used to copy scripts to your client machine. This policy is also useful for deploying custom config files, etc. Like Startup Scripts, this policy began as a compatability layer for Vintela's Files policy, but has become a Samba standard also. There is no Server Side Extension for the Group Policy Management Editor (GPME), but must be modified using the `samba-tool gpo manage`.

This policy is physically stored on the SYSVOL in the **MACHINE/VGP/VTLA/Unix/Files/manifest.xml** file in the subdirectory of the Group Policy Object. It is stored in an xml format, and is easily modified manually using a text editor.

9.1 Server Side Extension

The Files Policy has no GPME Server Side Extension (SSE), so this policy may only be administered using `samba-tool gpo manage files`. This is because this SSE is stored on the SYSVOL as an xml file, not in the Registry.pol from an ADMX template.

9.1.1 Managing the Files Policy via samba-tool

The Files `samba-tool` command has 3 subcommands; add, list, and remove.

```
> samba-tool gpo manage files --help
Usage: samba-tool gpo manage files <subcommand>

Manage Files Group Policy Objects
```

```
Options:
  -h, --help  show this help message and exit


Available subcommands:
  add     - Add VGP Files Group Policy to the sysvol
  list    - List VGP Files Group Policy from the sysvol
  remove  - Remove VGP Files Group Policy from the sysvol
```

To add a new File policy to the SYSVOL, call the `samba-tool gpo manage files add` command.

```
samba-tool gpo manage files add <gpo> <source> <target> <user>
 <group> <mode>
```

For example:

```
samba-tool gpo manage files add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} ./source.txt \
 /usr/share/doc/target.txt root root 600
```

The `source` argument refers to the source file on the host you are running the command from. This source file will be uploaded to the SYSVOL. The `target` argument is where the file should be deployed to on the client by the CSE. The `user`, `group` and `mode` refer to the attributes which will be assigned to the file when it is deployed to the client.

If, for example, we were to create a daily script (as described in chapter 7), we could use this policy to deploy that script to the Linux client. Let's now create a policy for testing that deployment.

```
> cat test_script.sh
#!/bin/sh

echo Something is happening daily
> samba-tool gpo manage files add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} ./test_script.sh \
 /usr/bin/test_script.sh root root 555 -UAdministrator
> samba-tool gpo manage files list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
-r-xr-xr-x  root    root    /usr/bin/test_script.sh ->
                                test_script.sh
```

The output of the `samba-tool gpo manage files list` command now shows that we have a policy set which will deploy a link to our test_script.sh on the host. If we check the contents of the SYSVOL, we can see that our file has been uploaded successfully.

```
> sudo mount.cifs \\\\lizardo.suse.de\\SYSVOL /mnt/ \
 -ouser=Administrator
> l /mnt/lizardo.suse.de/Policies/
    {31B2F340-016D-11D2-945F-00C04FB984F9}/MACHINE/VGP/VTLA/
    Unix/Files/
total 2
drwxr-xr-x 2 root root   0 Nov 15 09:55 ./
drwxr-xr-x 2 root root   0 Nov 15 09:55 ../
-rwxr-xr-x 1 root root 532 Nov 15 09:55 manifest.xml*
-rwxr-xr-x 1 root root  45 Nov 15 09:55 test_script.sh*
```

Let’s take a look at the contents of the `manifest.xml` which stores our policy.

```
> cat /mnt/lizardo.suse.de/Policies/
      {31B2F340-016D-11D2-945F-00C04FB984F9}/MACHINE/VGP/VTLA/
      Unix/Files/manifest.xml | xmllint --format -
<?xml version="1.0" encoding="UTF-8"?>
<vgppolicy>
  <policysetting>
    <version>1</version>
    <name>Files</name>
    <description>
      Represents file data to set/copy on clients
    </description>
    <data>
      <file_properties>
        <source>test_script.sh</source>
        <target>/usr/bin/test_script.sh</target>
        <user>root</user>
        <group>root</group>
        <permissions type="user">
          <read/>
          <execute/>
        </permissions>
        <permissions type="group">
          <read/>
          <execute/>
        </permissions>
        <permissions type="other">
          <read/>
          <execute/>
```

```
        </permissions>
      </file_properties>
    </data>
  </policysetting>
</vgppolicy>
```

Our `source` now refers to the file uploaded to the same directory as the manifest.

If you wanted to remove this policy later, we would issue the `samba-tool gpo manage files remove` command.

```
> samba-tool gpo manage files remove \
 {31B2F340-016D-11D2-945F-00C04FB984F9} /usr/bin/test_script.sh \
 -UAdministrator
```

Afterward the files list should be empty.

9.2 Client Side Extension

The Files Client Side Extension (CSE) copies the file from the SYSVOL to the location specified in the `target` variable earlier. The Files Policy only applies to Machine policy.

Let’s now list the Resultant Set of Policy on the Linux client to see the test file we created previously.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: vgp_files_ext
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/Files
    ------------------------------------------------------------
    [ -r-xr-xr-x root root /usr/bin/test_script.sh ->
                           test_script.sh ]
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

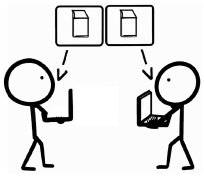
Note that while the output appears to suggest we will be creating a symlink, it is actually a hard copy of the file that is created. The syntax of the output is simply for illustrative purposes.

If we now force the policy to apply, we'll see our file is physically copied to the requested location, along with the requested permissions.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='VGP/Unix Settings/Files']" -
\
 | xmllint --format -
<gp_ext name="VGP/Unix Settings/Files">
  <attribute name="/usr/bin/test_script.sh">
    268d...9b39:root:root:365
  </attribute>
</gp_ext>
> l /usr/bin/test_script.sh
-r-xr-xr-x 1 root root 45 Nov 15 12:07 /usr/bin/test_script.sh*
> cat /usr/bin/test_script.sh
#!/bin/sh

echo Something is happening daily
```

The script is present where we requested. You can now refer to chapter 7 section 7.1.1 for details on how to schedule a job to execute this script via the Scripts Policy.

10 Symlink Policies



The purpose of this policy is to create a symbolic link on a Linux client. Only Machine policy is supported. This policy could be useful in conjunction with the Files Policy found in chapter 9.

This policy is physically stored on the SYSVOL in the **MACHINE/VGP /VTLA/Unix/Symlink/manifest.xml** file in the subdirectory of the Group Policy Object. It is stored in an xml format, and is easily modified manually using a text editor.

10.1 Server Side Extension

The Symlink Policy has no GPME Server Side Extension (SSE), so this policy may only be administered using `samba-tool gpo manage symlink`. This is because this SSE is stored on the SYSVOL as an xml file, not in the Registry.pol from an ADMX template.

10.1.1 Managing the Symlink Policy via samba-tool

The Symlink `samba-tool` command has 3 subcommands; add, list, and remove.

```
> samba-tool gpo manage symlink --help
Usage: samba-tool gpo manage symlink <subcommand>

Manage symlink Group Policy Objects


Options:
  -h, --help  show this help message and exit
```

```
Available subcommands:
  add     - Adds a VGP Symbolic Link Group Policy to the sysvol
  list    - List VGP Symbolic Link Group Policy from the sysvol
  remove  - Removes a VGP Symbolic Link Group Policy from the
            sysvol
```

To add a new Symlink policy to the SYSVOL, call the `samba-tool gpo manage symlink add` command.

```
samba-tool gpo manage symlink add <gpo> <source> <target>
```

This command will add a policy instructing the client to create a symbolic link pointing to `source` named `target`.

Let's add a simple policy, which uploads a configuration file using the Files Policy (see chapter 9), then symlinks that configuration file to somewhere useful on the system.

```
> cat servlist.conf
N=Libera.Chat
L=1
E=UTF-8 (Unicode)
F=23
D=0
S=irc.libera.chat/6697
J=#samba-technical
> samba-tool gpo manage files add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} servlist.conf \
 /usr/share/servlist.conf 'LIZARDO\tux' \
 'LIZARDO\domain users' 600 -UAdministrator
> samba-tool gpo manage files list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
-rw------- LIZARDO\tux LIZARDO\domain users
 /usr/share/servlist.conf -> servlist.conf
> samba-tool gpo manage symlink add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} /usr/share/servlist.conf
\
 /home/LIZARDO/tux/.config/hexchat/servlist.conf -UAdministrator
> samba-tool gpo manage symlink list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
ln -s /usr/share/servlist.conf
 /home/LIZARDO/tux/.config/hexchat/servlist.conf
```

Here we are uploading a configuration file for hexchat, then symlinking it to

a user's profile. The `samba-tool gpo manage symlink list` command displays the link operations that will be performed on the client.

Later when we choose to remove this policy, we will do so with the `samba-tool gpo manage symlink remove` command.

```
> samba-tool gpo manage symlink remove \
 {31B2F340-016D-11D2-945F-00C04FB984F9} /usr/share/servlist.conf
\
 /home/LIZARDO/tux/.config/hexchat/servlist.conf -UAdministrator
```

10.2 Client Side Extension

The Symlink Client Side Extension (CSE) creates a symlink between the `source` and `target`. Startup Scripts only apply for Machine policy.

Let's list the Resultant Set of Policy to view the symbolic link policy we created in the previous section.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: vgp_symlink_ext
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/Symbolic Links
    ------------------------------------------------------------
    [ ln -s /usr/share/servlist.conf
      /home/LIZARDO/tux/.config/hexchat/servlist.conf ]
    ------------------------------------------------------------
  ------------------------------------------------------------
  CSE: vgp_files_ext
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/Files
    ------------------------------------------------------------
    [ -rw------- LIZARDO\tux LIZARDO\domain users
      /usr/share/servlist.conf -> servlist.conf ]
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

In addition to our Symlink policy, we also see the Files policy which we added in conjunction with this.

Let’s now force our policy to apply and see how the CSE behaves.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
> | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='VGP/Unix Settings/Files' or
                              @name='VGP/Unix Settings/Symbolic
                                     Links']" - \
 | xmllint --format -
<gp_ext name="VGP/Unix Settings/Files">
  <attribute name="/usr/share/servlist.conf">
    d5b5...820c:LIZARDO\5Ctux:LIZARDO\5Cdomain users:384
  </attribute>
</gp_ext>
<gp_ext name="VGP/Unix Settings/Symbolic Links">
  <attribute name="/usr/share/servlist.conf:
             /home/LIZARDO/tux/.config/hexchat/servlist.conf">
    /home/LIZARDO/tux/.config/hexchat/servlist.conf
  </attribute>
</gp_ext>
> l /usr/share/servlist.conf
-rw------- 1 LIZARDO\tux LIZARDO\domain users 87 Nov 15 13:51
 /usr/share/servlist.conf
> sudo l /home/LIZARDO/tux/.config/hexchat/servlist.conf
lrwxrwxrwx 1 root root 24 Nov 15 13:51 /home/LIZARDO/tux/.config/
 hexchat/servlist.conf -> /usr/share/servlist.conf
```

Our Group Policy Cache at `/var/lib/samba/gpo.tdb` shows the two policies have been applied. Listing the `target`, we also see that the symlink now exists. If we output the contents of our symlink, we can see that it is indeed pointing to our configuration file that we uploaded to the SYSVOL earlier.

```
> sudo cat /home/LIZARDO/tux/.config/hexchat/servlist.conf
N=Libera.Chat
L=1
E=UTF-8 (Unicode)
F=23
D=0
S=irc.libera.chat/6697
J=#samba-technical
```

11 Sudoers Policies

The purpose of the Sudoers Polices are to deploy sudo rules to a Linux client. Naturally, only Machine policy is supported.

This policy is physically stored in two different locations on the SYSVOL, in **MACHINE/Registry.pol** and in **MACHINE/VGP/VTLA/Sudo/SudoersConfiguration/manifest.xml**. The `manifest.xml` is in xml format, and is easily modified manually using a text editor. The `Registry.pol` is in registry format. See chapter 21 for details on how to manually modify this file.

11.1 Server Side Extension

The Server Side Extensions (SSE) for Sudoers policies are distributed via either Administrative Templates (see 20.1 in section 20.1.1) or via the command `samba-tool gpo manage sudoers`. Rules added via the GPME can be modified from samba-tool (as of Samba 4.18), but rules added via `samba-tool gpo manage sudoers` will not be visible in the GPME.

11.1.1 Managing Sudoers Policy via the GPME

Setting up the ADMX templates for this policy is described in chapter 22 section 22.1.

To add a new sudo rule using the Group Policy Management Editor (GPME):

1. Open the Group Policy Management Editor. For instructions on accessing the GPME, see chapter 4 section 4.1.

2. In the Group Policy Management Editor window, navigate to `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings`.

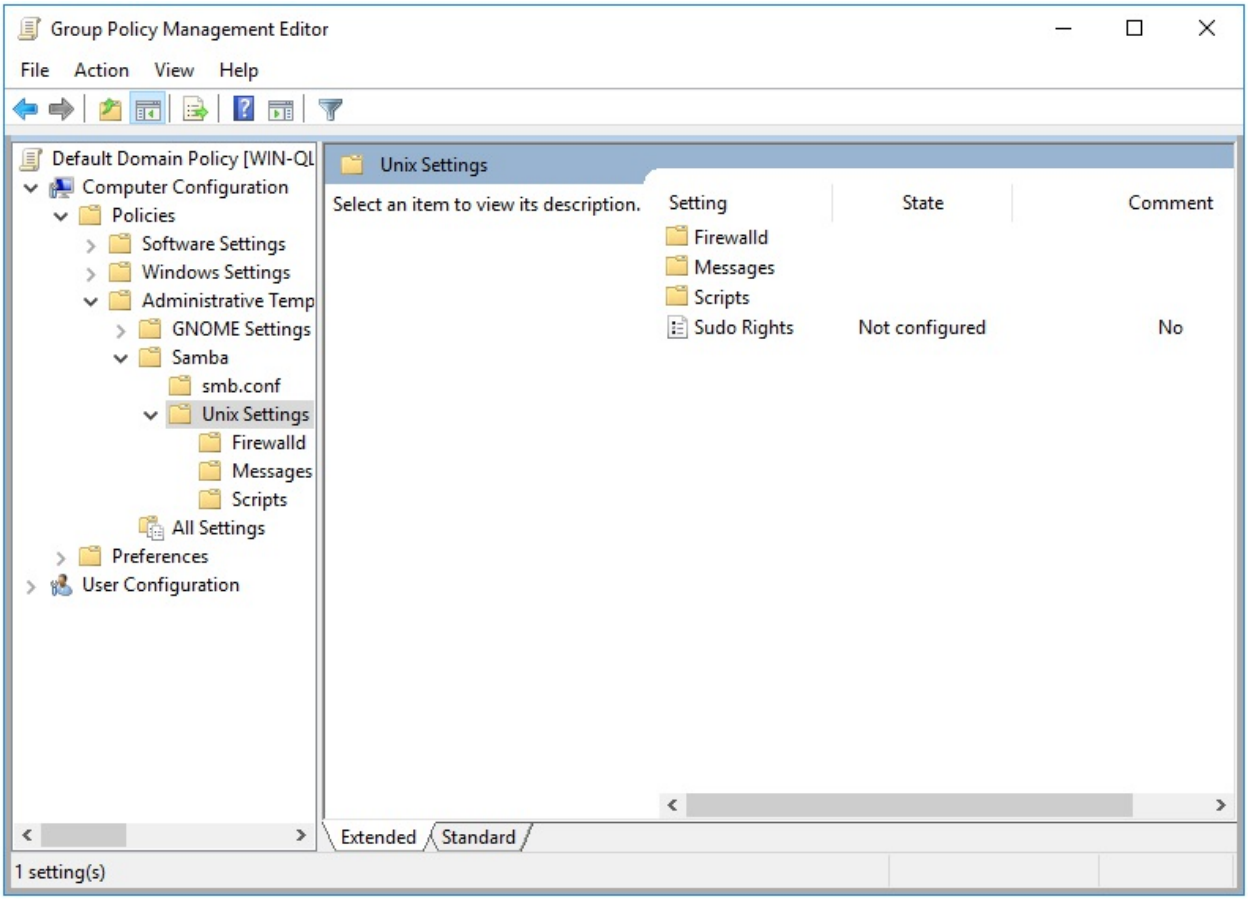


Figure 11.1: Group Policy Management Editor

3. In the right pane, double-click on the “Sudo Rights” setting.

4. In the “Sudo Rights” dialog box, click on the “Show” button next to “Sudoers commands”.

5. In the “Show Contents” dialog box, enter a new sudo rule.

6. Click “OK” to save the new sudo rule and close the dialog box.

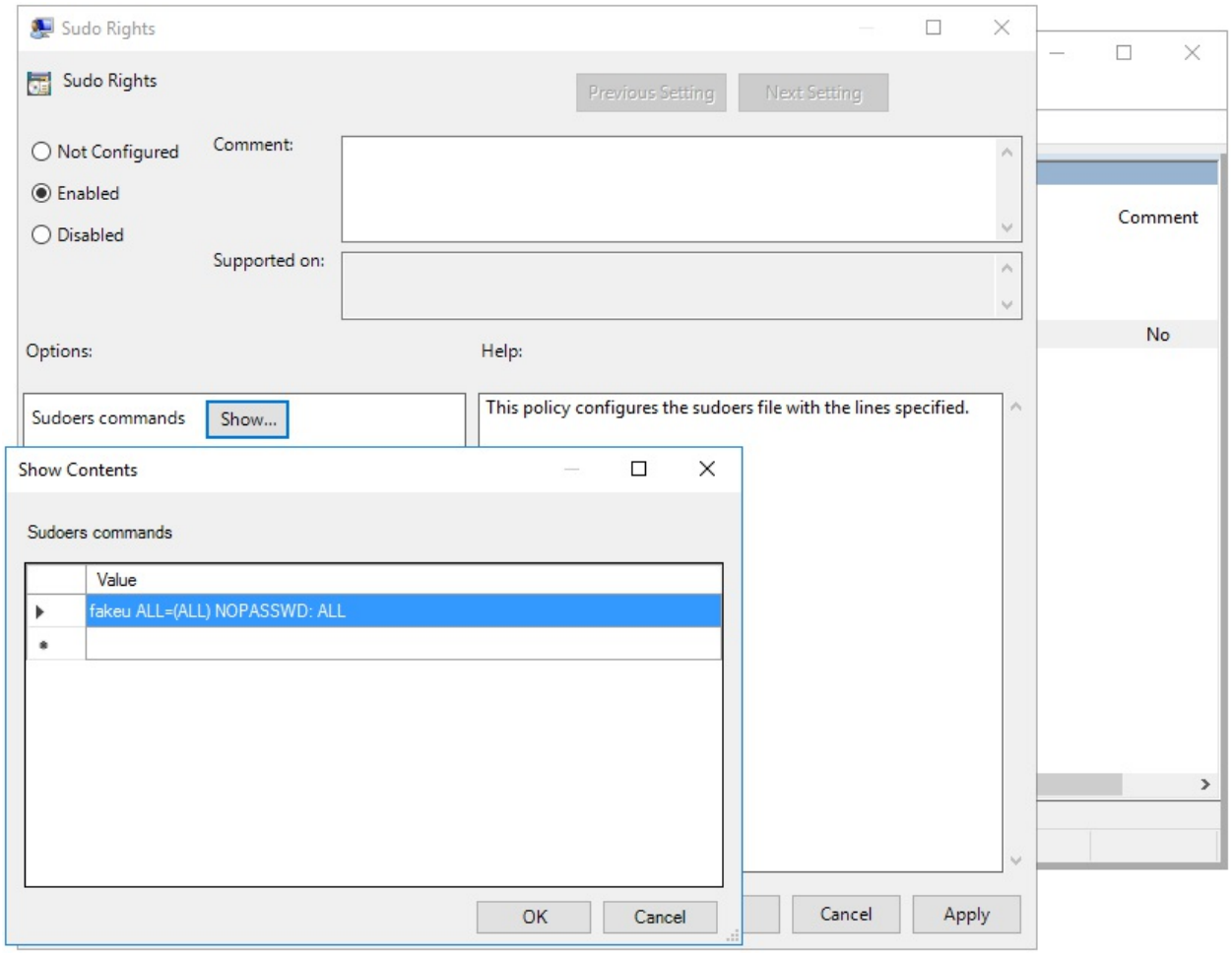


Figure 11.2: Adding a Sudo Rule

After applying this rule, you can list the rule using `samba-tool`.

```
> samba-tool gpo manage sudoers list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
fakeu ALL=(ALL) NOPASSWD: ALL
```

11.1.2 Managing Sudoers Policy via samba-tool

The Sudoers Policy can also be set via the `samba-tool gpo manage sudoers` command, which has 3 subcommands; add, list, and remove.

```
> samba-tool gpo manage sudoers --help
Usage: samba-tool gpo manage sudoers <subcommand>

Manage Sudoers Group Policy Objects
```

```
Options:
  -h, --help  show this help message and exit


Available subcommands:
  add     - Adds a Samba Sudoers Group Policy to the sysvol
  list    - List Samba Sudoers Group Policy from the sysvol
  remove  - Removes a Samba Sudoers Group Policy from the sysvol
```

To add a new Sudoers rule to the SYSVOL, call the `samba-tool gpo manage sudoers add` command.

```
samba-tool gpo manage sudoers add <gpo> <command> <user> <users>
```

Let's add a simple rule for testing.

```
> samba-tool gpo manage sudoers add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} ALL ALL fakeu fakeg \
 -UAdministrator
> samba-tool gpo manage sudoers list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
fakeu,fakeg% ALL=(ALL) NOPASSWD: ALL
fakeu ALL=(ALL) NOPASSWD: ALL
```

You'll notice that listing the entries now includes the entry we just created, as well as the one we added earlier using the GPME.

11.2 Client Side Extension

Samba actually supplies 3 different Client Side Extensions (CSE) for the Sudoers policy. This is to support 3 different SSE providers. The first is Samba's original policy, which is modified using the GPME as explained in section 11.1.1. The second is the Vintela compatible policy, which can be set using `samba-tool` as explained in section 11.1.2. The final CSE is the Centrify compatible policy, which is only provided as a convenience for migration. All these CSEs behave the same in how they apply policy (and share some of the same code). The preceding instructions refer to all three CSEs.

The Sudoer CSEs create a file within `/etc/sudoers.d` for each rule specified on the SYSVOL. Each rule is validated before installing, to ensure the system isn't left in a broken state.

Let's list the Resultant Set of Policy to view the Sudoer rules we created in the previous sections.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_sudoers_ext
  ------------------------------------------------------------
    Policy Type: Sudo Rights
    ------------------------------------------------------------
    [ fakeu ALL=(ALL) NOPASSWD: ALL ]
    ------------------------------------------------------------
  ------------------------------------------------------------
  CSE: vgp_sudoers_ext
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/Sudo Rights
    ------------------------------------------------------------
    [ fakeu,fakeg% ALL=(ALL) NOPASSWD: ALL ]
    ------------------------------------------------------------
  ------------------------------------------------------------
  CSE: gp_centrify_sudoers_ext
  ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

Notice that both the rules we created earlier are listed, but under different CSEs. These policies will be applied by different CSEs because they are stored on the SYSVOL differently. The results will be similar though. Currently we don't have any Centrify combatible policy in our environment, so this CSE remains empty.

Let's now force our policy to apply and see how the CSEs behave.

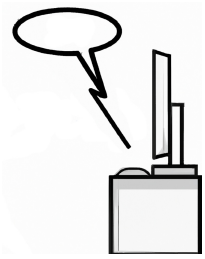
```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Unix Settings/Sudo Rights' or
```

```
                    @name='VGP/Unix Settings/Sudo Rights']" - \
 | xmllint --format -
<gp_ext name="Unix Settings/Sudo Rights">
  <attribute name="ZmFrZXUgQUxMPShBTEwpIE5PUEFTU1dEOiBBTEw=">
    /etc/sudoers.d/gp_mzarfh6k
  </attribute>
</gp_ext>
<gp_ext name="VGP/Unix Settings/Sudo Rights">
  <attribute
name="ZmFrZXUsZmFrZWclIEFMTD0oQUxMKSBOT1BBU1NXRDogQUxM">
    /etc/sudoers.d/gp_qy2eo07y
  </attribute>
</gp_ext>
```

We can see the Sudoer rules have been applied to `/etc/sudoers.d/gp_mzarfh6k` and `/etc/sudoers.d/gp_qy2eo07y`.

```
> sudo cat /etc/sudoers.d/gp_mzarfh6k

### autogenerated by samba
#
# This file is generated by the gp_sudoers_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

fakeu ALL=(ALL) NOPASSWD: ALL
> sudo cat /etc/sudoers.d/gp_qy2eo07y

### autogenerated by samba
#
# This file is generated by the gp_sudoers_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

fakeu,fakeg% ALL=(ALL) NOPASSWD: ALL
```

12 Message Policies



The purpose of the Message policy is to set the contents of the `/etc/motd` and `/etc/issue` files. These are Machine only policies.

This policy is physically stored in three different locations on the SYSVOL, in **MACHINE/Registry.pol**, **MACHINE/VGP/VTLA/Unix/Issue/ manifest.xml**, and **MACHINE/VGP/VTLA/Unix/MOTD/ manifest.xml**. The `manifest.xml` files are in xml format, and are easily modified manually using a text editor. The `Registry.pol` is in registry format. See chapter 21 for details on how to manually modify this file.

12.1 Server Side Extension

The Server Side Extensions (SSE) for Message policies are distributed via either Administrative Templates (see chapter 20.1 in section 20.1.1) or via the commands `samba-tool gpo manage motd` and `samba-tool gpo manage issue`. Rules added via GPME will not be visible to the respective `samba-tool` commands, and vice versa. This is because the `samba-tool` commands are intended to manage Vintela Group Policy compatability. These two SSEs should not be used in conjunction to one another, as it will cause unpredictable results on the client.

12.1.1 Managing Message Policy via the GPME

Setting up the ADMX templates for this policy is described in chapter 22 section 22.1.

To edit the “Message of the day” and “Login Prompt Message” settings using the Group Policy Management Editor (GPME):

1. Open the Group Policy Management Editor. For instructions on accessing the GPME, see chapter 4 section 4.1.

2. In the Group Policy Management Editor window, navigate to `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Messages`.

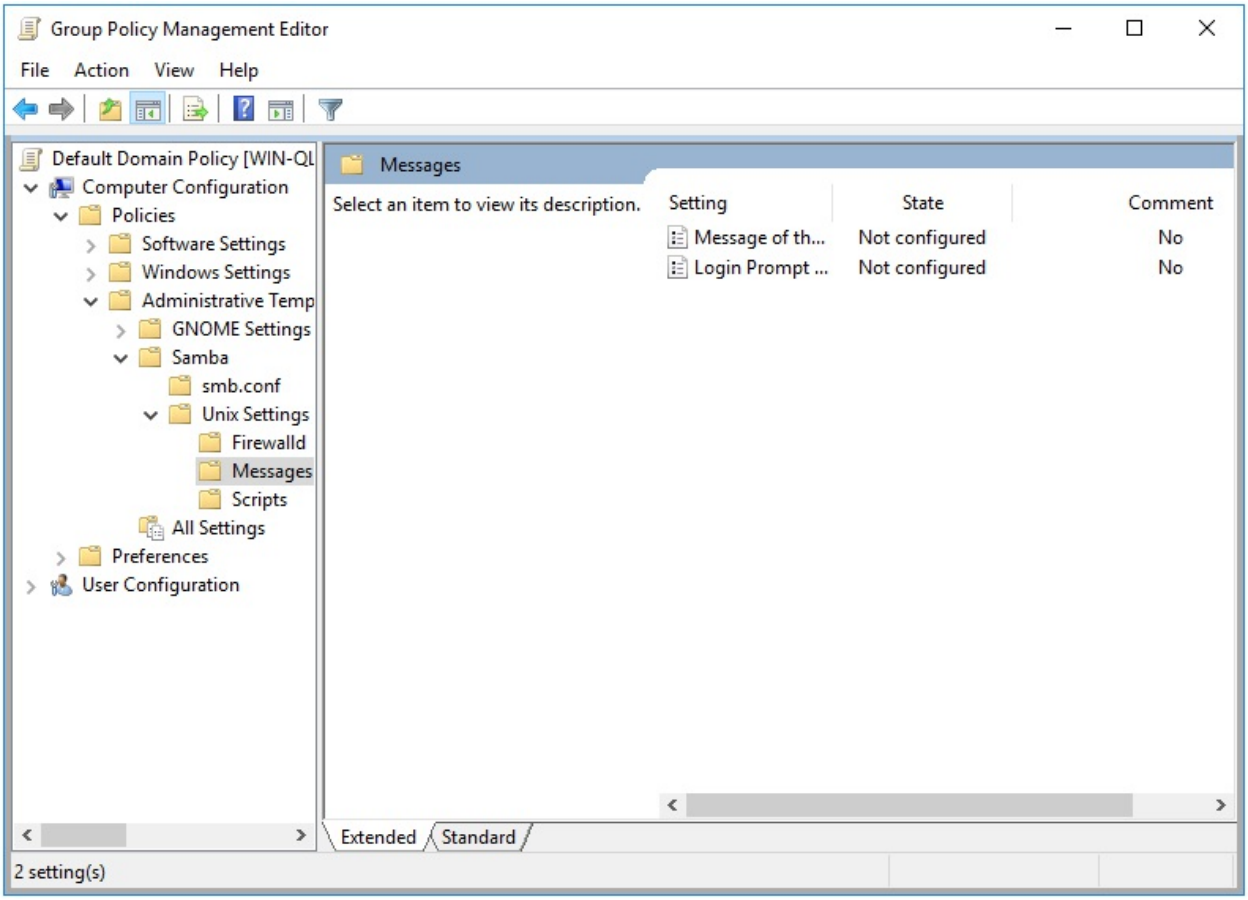


Figure 12.1: Group Policy Management Editor

3. In the right pane, double-click on the “Message of the day” setting.

4. In the “Message of the day” dialog box, enter the desired message in the text field.

5. Click “OK” to save the changes and close the dialog box.

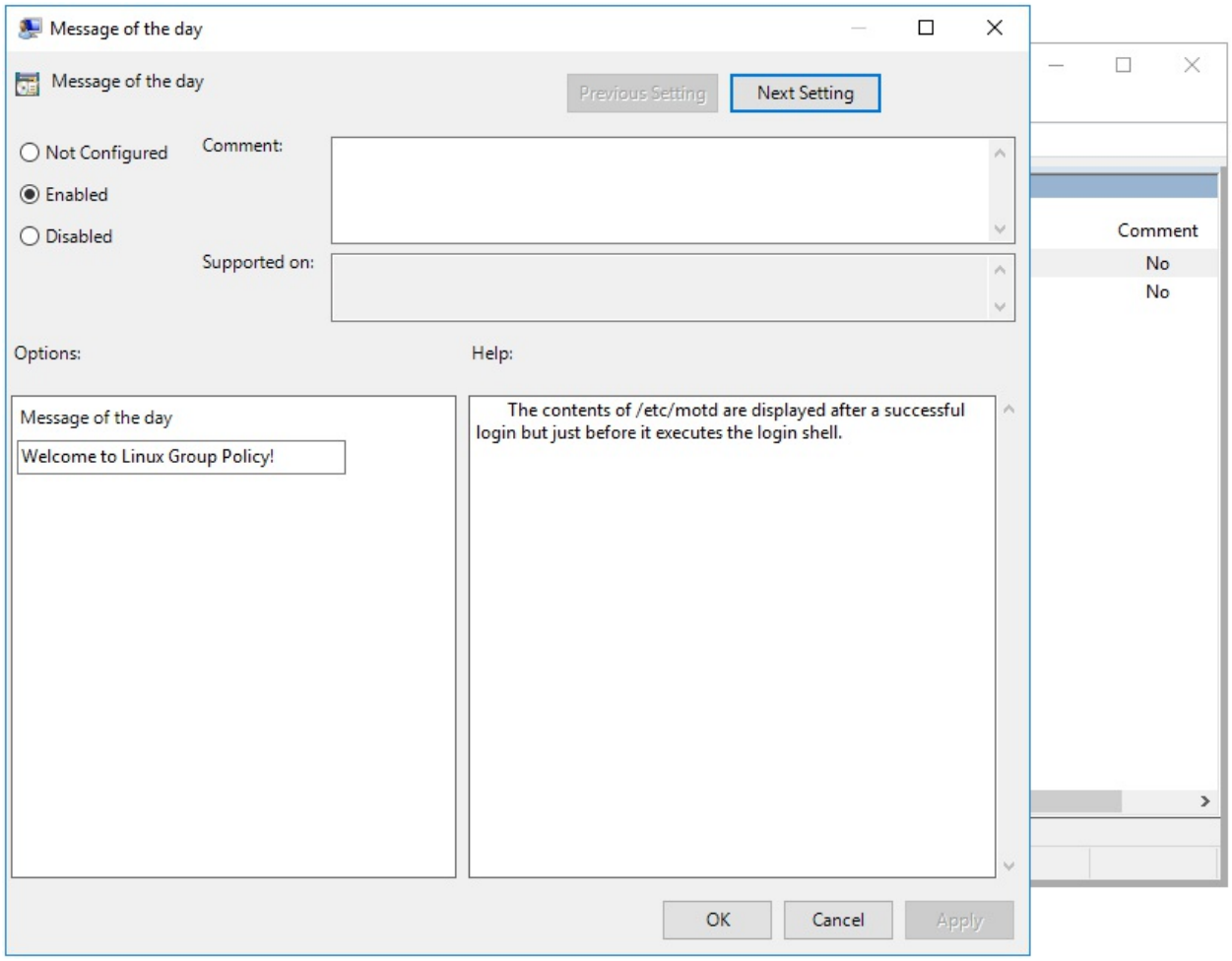


Figure 12.2: Message of the day

6. In the right pane, double-click on the “Login Prompt Message” setting.

7. In the “Login Prompt Message” dialog box, enter the desired message in the text field. The “Help” field on the right explains the various options for this message.

8. Click “OK” to save the changes and close the dialog box.

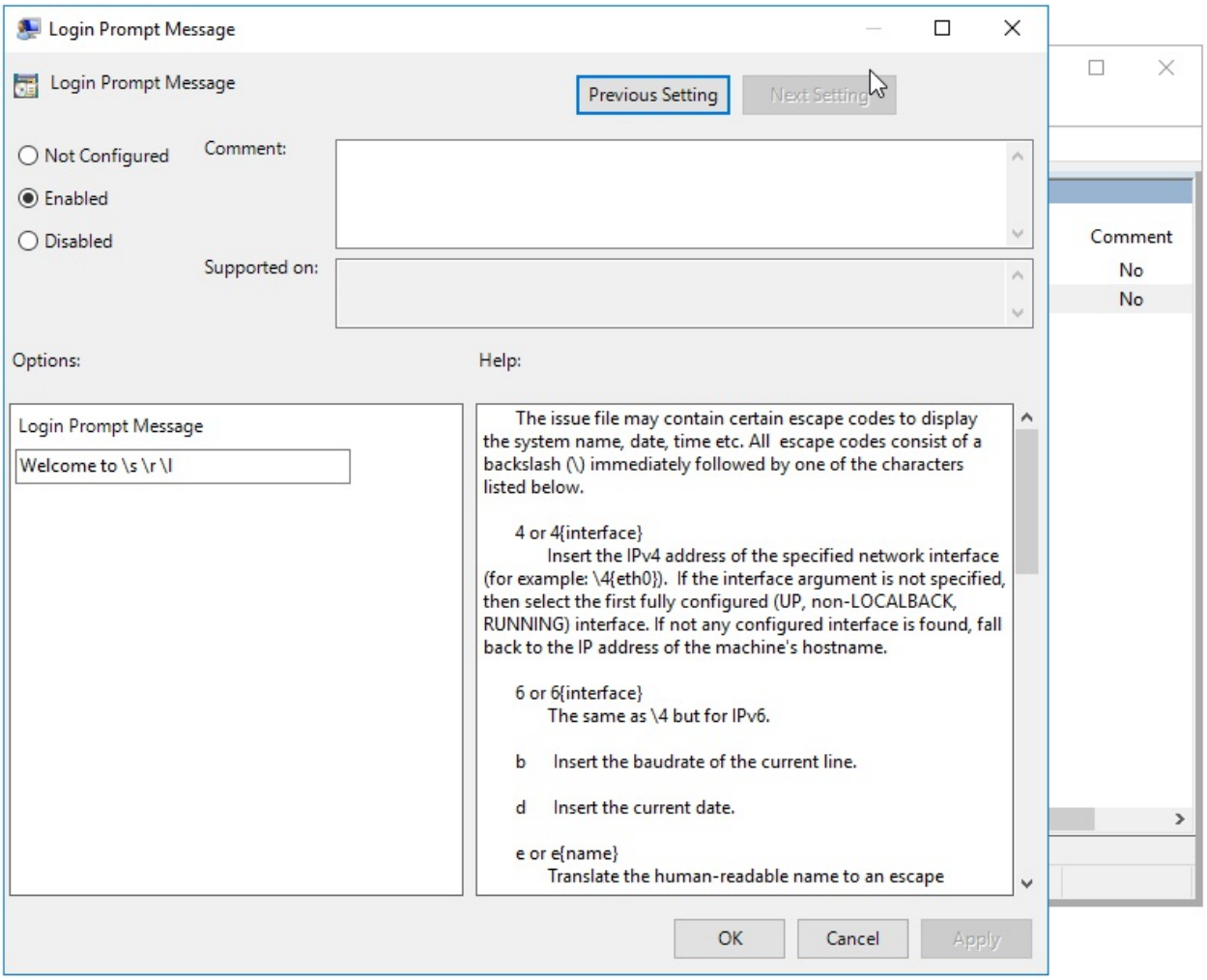


Figure 12.3: Login Prompt Message

12.1.2 Managing Message Policy via samba-tool

The `samba-tool gpo manage motd` and `samba-tool gpo manage issue` commands each have 2 subcommands; set and list.

```
> samba-tool gpo manage motd
Usage: samba-tool gpo manage motd <subcommand>

Manage Message of the Day Group Policy Objects


Options:
  -h, --help  show this help message and exit
```

```
Available subcommands:
  list  - List VGP MOTD Group Policy from the sysvol
  set   - Sets a VGP MOTD Group Policy to the sysvol
```

The syntax is the same for both `motd` and `issue`. The `list` command simply lists the current value of the setting, while the `set` command will set the contents of the setting.

To use the `samba-tool gpo manage motd list` command, you need to provide the name of the GPO as an argument. For example:

```
samba-tool gpo manage motd list \
  {31B2F340-016D-11D2-945F-00C04FB984F9}
```

To use the `samba-tool gpo manage motd set` command, you need to provide the name of the GPO as an argument, followed by the message you want to set as the MOTD. For example:

```
samba-tool gpo manage motd set \
 {31B2F340-016D-11D2-945F-00C04FB984F9} "Welcome to the server!
 Please make sure to read the guidelines before proceeding."
```

If no value is provided for the message, the MOTD will be unset and will not be displayed to users when they log in.

Let’s set some messages for testing later.

```
> samba-tool gpo manage motd set \
 {31B2F340-016D-11D2-945F-00C04FB984F9} \
 "motd set from samba-tool" -UAdministrator
> samba-tool gpo manage issue set \
 {31B2F340-016D-11D2-945F-00C04FB984F9} \
 "issue set from samba-tool" -UAdministrator
```

Doing a list for good measure, we see that the policy is set on the SYSVOL.

```
> samba-tool gpo manage motd list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator; echo
motd set from samba-tool
> samba-tool gpo manage issue list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator; echo
issue set from samba-tool
```

12.2 Client Side Extension

Samba provides 3 different Client Side Extensions (CSE) for the Messages policy. The Samba policy distributed via the GPME discussed in 12.1.1 called `gp_msgs_ext`, and the Vintela compatible policy split in 2 parts discussed in 12.1.2 called `vgp_motd_ext` and `vgp_issue_ext`.

These CSEs set the contents of `/etc/motd` and `/etc/issue`. If both Samba and Vintela compatible policies are set, they will conflict.

Let's list the Resultant Set of Policy to view what will be applied by `samba-gpupdate`.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_msgs_ext
  ------------------------------------------------------------
    Policy Type: /etc/motd
    ------------------------------------------------------------
Welcome to Linux Group Policy!
    ------------------------------------------------------------
    Policy Type: /etc/issue
    ------------------------------------------------------------
Welcome to \s \r \l
    ------------------------------------------------------------
  ------------------------------------------------------------
  CSE: vgp_motd_ext
  ------------------------------------------------------------
    Policy Type: /etc/motd
    ------------------------------------------------------------
motd set from samba-tool
    ------------------------------------------------------------
  ------------------------------------------------------------
  CSE: vgp_issue_ext
  ------------------------------------------------------------
    Policy Type: /etc/issue
    ------------------------------------------------------------
issue set from samba-tool
```

```
    ------------------------------------------------------------
  ----------------------------------------------------------------
================================================================
```

Because both these policies are set, we can't predict which one will actually be applied. Let's do a force and see what the result is.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Unix Settings/Messages' or
                             @name='Unix Settings/Message
                of the Day' or
                             @name='Unix Settings/Issue']" - \
 | xmllint --format -
<gp_ext name="Unix Settings/Messages">
  <attribute name="motd"/>
  <attribute name="issue">
    Welcome to openSUSE Tumbleweed
  </attribute>
</gp_ext>
<gp_ext name="Unix Settings/Message of the Day">
  <attribute name="motd">
    Welcome to Linux Group Policy!
  </attribute>
</gp_ext>
<gp_ext name="Unix Settings/Issue">
  <attribute name="issue">
    Welcome to \5Cs \5Cr \5Cl
  </attribute>
</gp_ext>
```

Note that Messages policy stores the previous value of the message content in the Group Policy Cache. So we can discern from the message content that `vgp_motd_ext` and `vgp_issue_ext` applied last (since the messages from `gp_msgs_ext` show up in the log). We can confirm this by checking the contents of `/etc/motd` and `/etc/issue`.

```
> cat /etc/motd; echo
motd set from samba-tool
> cat /etc/issue; echo
issue set from samba-tool
```

13 PAM Access Policies

PAM Access Policy allows you to set host access rules for client machines. Specifically, it specifies rules in `/etc/security/access.d` to allow or deny access to the host.

This policy is physically stored on the SYSVOL in two files, MACHINE/VGP/VTLA/VAS/HostAccessControl/Allow/manifest.xml and MACHINE/VGP/VTLA/VAS/HostAccessControl/Deny/manifest.xml. The manifest.xml files are in xml format, and are easily modified manually using a text editor.

13.1 Server Side Extension

The Server Side Extensions (SSE) for PAM Access Policies is administered using the `samba-tool gpo manage access` command. This SSE cannot be modified using the GPME.

13.1.1 Managing PAM Access Policies via samba-tool

The `samba-tool gpo manage access` command has 3 subcommands; add, list, and remove.

```
> samba-tool gpo manage access
Usage: samba-tool gpo manage access <subcommand>

Manage Host Access Group Policy Objects


Options:
  -h, --help  show this help message and exit
```

```
Available subcommands:
  add     - Adds a VGP Host Access Group Policy to the sysvol
  list    - List VGP Host Access Group Policy from the sysvol
  remove  - Remove a VGP Host Access Group Policy from the sysvol
```

To use the `samba-tool gpo manage access add` command, you need to provide the name of the GPO as the first argument, followed by the access setting (either “allow” or “deny”), the common name (cn) of the host or user you want to allow or deny access to, and the domain of the host or user. For example:

```
samba-tool gpo manage access add <gpo> <allow/deny> <cn> <domain>
```

Any time an allow entry is detected by the client, an implicit deny ALL will be assumed.

Let’s add a few rules that restricts access to a couple of specific users.

```
> samba-tool gpo manage access add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} allow Administrator \
 lizardo.suse.de -UAdministrator
> samba-tool gpo manage access add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} allow tux \
 lizardo.suse.de -UAdministrator
```

These grant access to the users `tux` and `Administrator` on the host. If we list our access policies, we can see they are ready for delivery to the client.

```
> samba-tool gpo manage access list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
+:lizardo.suse.de\Administrator:ALL
+:lizardo.suse.de\tux:ALL
```

13.2 Client Side Extension

The PAM Access Client Side Extension (CSE) will create a new file in the `/etc/security/access.d` directory for each host access rule.

The PAM module `pam_access` must be configured or this CSE will do nothing (see `man pam_access`). This can be configured using the command `pam-config --add --access`. It may be beneficial to ensure this is enabled

by enforcing a Script policy which executes `pam-config --add --access` (see chapter 7 on how to schedule a script policy).

Let's list the Resultant Set of Policy to view the policies we've created for our host access control.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_scripts_ext
  ------------------------------------------------------------
    Policy Type: Hourly Scripts
    ----------------------------------------------------------
    [ pam-config --add --access ]
    ----------------------------------------------------------
  ------------------------------------------------------------
  CSE: vgp_access_ext
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/Host Access
    ----------------------------------------------------------
    [ +:Administrator\lizardo.suse.de:ALL ]
    [ +:tux\lizardo.suse.de:ALL ]
    ----------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

Our PAM Access policy and our `pam-config` check are both listed.

Let's now force an apply.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Unix Settings/Scripts' or
                              @name='VGP/Unix Settings/Host
                         Access']" - \
 | xmllint --format -
<gp_ext name="Unix Settings/Scripts">
  <attribute name="Software\\Policies\\Samba\\Unix Settings\\
                   Daily Scripts:ZWNobyBoZWxsbyB3b3JsZA==">
    /etc/cron.daily/gp_pawtjsiq
  </attribute>
```

```
</gp_ext>
<gp_ext name="VGP/Unix Settings/Host Access">
  <attribute name="6bf0...fb9c">
    /etc/security/access.d/9000000001_gp_DENY_ALL.conf:
    /etc/security/access.d/0000000001_gp.conf
  </attribute>
</gp_ext>
```

Notice that the PAM Access policy generated 2 different files, `/etc/security/access.d/0000000001_gp.conf` and `/etc/security/access.d/9000000001_gp_DENY_ALL.conf`. Let's check the contents of these files to see what was generated.

```
> cat /etc/security/access.d/0000000001_gp.conf; echo

### autogenerated by samba
#
# This file is generated by the vgp_access_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

+:lizardo.suse.de\Administrator:ALL
+:lizardo.suse.de\tux:ALL
> cat /etc/security/access.d/9000000001_gp_DENY_ALL.conf; echo

### autogenerated by samba
#
# This file is generated by the vgp_access_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

-:ALL:ALL
```

The PAM Access CSE automatically added a deny all entry. The pam_access pam module reads host access rules in numerical order from the `/etc/security/access.d` directory. This automatic deny all entry was intentionally placed numerically after our allow entries (9000000001 > 0000000001), to ensure the allow entries are processed first.

You should notice at this point that it would be senseless to intermix allow

and deny rules. Any time an allow rule is applied to a client, *deny all else* is implicitly assumed. It doesn't hurt to have extra deny rules, but they would be pointless. When only deny rules are set in the policy, then *allow all else* is implicitly assumed (which is the pam default, and no extra rules are added).

You can safely stack group policies which contain different allow rules, since the *deny all else* entries will always be placed in a range above the host access rules. You will see multiple *deny all else* entries generated, and this is intentional. This ensures there is always a *deny all else* entry associated with any allow rules which are applied.

14 Certificate Auto Enrollment Policy

Certificate Auto Enrollment allows devices to enroll for certificates from Active Directory Certificate Services. Samba's Certificate Auto Enrollment uses the certmonger service to keep track of certificates. It also uses the cepces plugin to certmonger.

14.1 Server Side Extension

The Server Side Extension (SSE) for Certificate Auto Enrollment is part of the Group Policy Management Editor (GPME). Currently, no `samba-tool` command is available for managing this policy. The policy requires access to a Windows certificate server. On the certificate server, the roles Certification Authority, Certificate Enrollment Policy Web Service, and Certificate Enrollment Web Service all must be installed and configured. Optionally the role Network Device Enrollment Service can be installed to simplify the fetching of the root certificate chain by the client. Configuring the certificate server is beyond the scope of this book.

Optionally, the Network Device Enrollment role can be configured on the server, which will allow the client to fetch the root certificate chain. Without this role, the Client Side Extension will report the following warnings:

```
[W26775]| Failed to fetch the root certificate chain. | {}
[W05621]| The Network Device Enrollment Service is either not
          installed or not configured. | {}
[W11946]| Installing the server certificate only. | {}
```

14.1.1 Managing Certificate Auto Enrollment via the GPME

To enable Certificate Auto Enrollment using the Group Policy Management Editor (GPME):

1. Open the Group Policy Management Editor. For instructions on accessing the GPME, see chapter 4 section 4.1.

2. In the Group Policy Management Editor window, navigate to `Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies`.

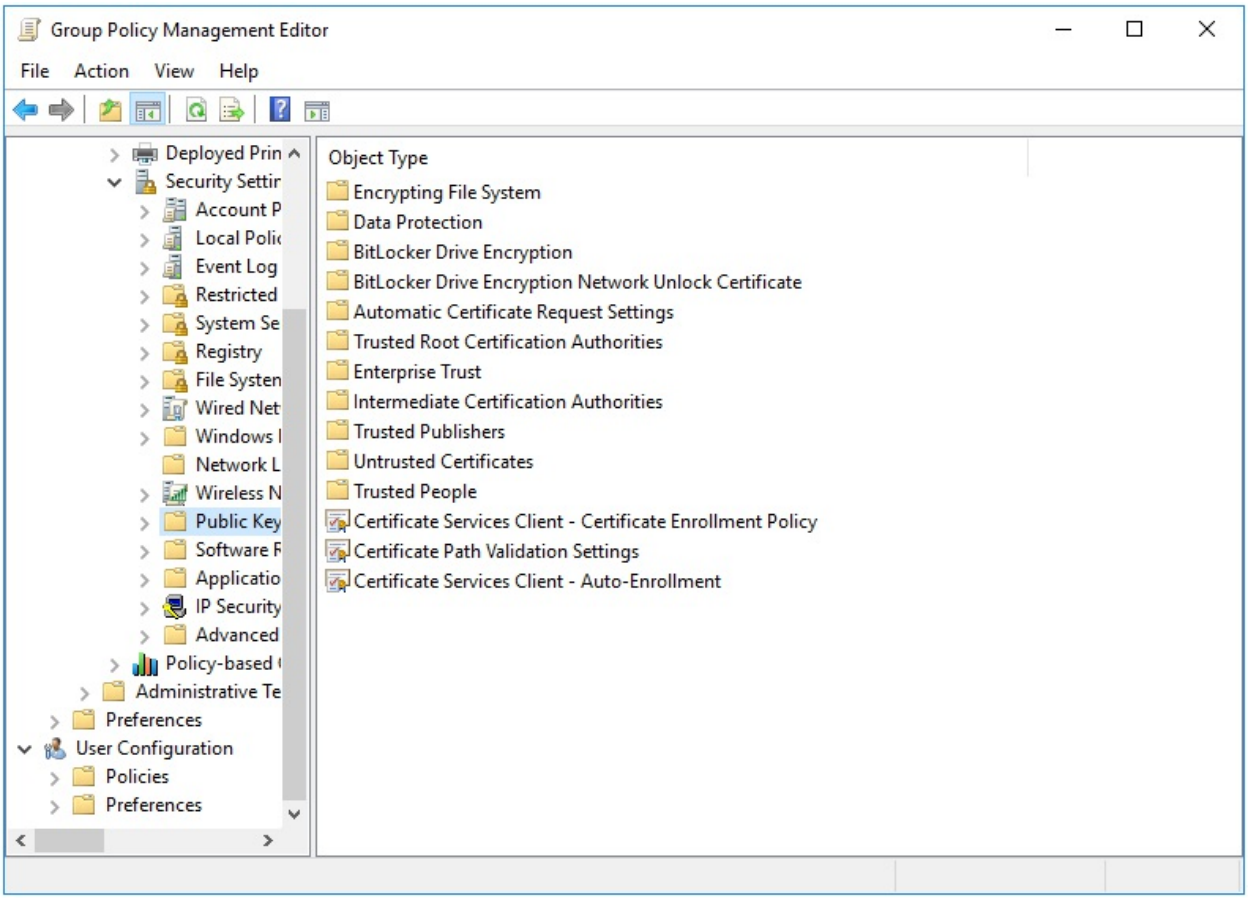


Figure 14.1: Public Key Policies

3. Double click on and open the `Certificate Services Client - Auto-Enrollment` properties.

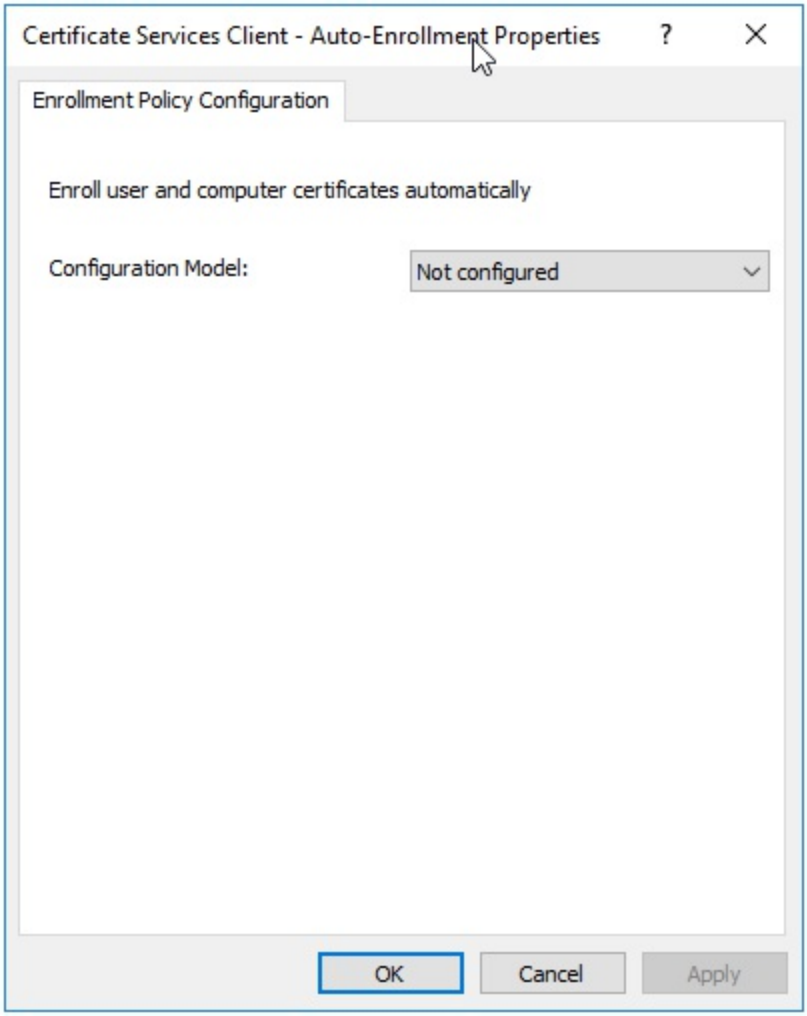


Figure 14.2: Certificate Services Client - Auto-Enrollment

4. In the properties dialog, set the `Configuration Model` to “Enabled” and check the boxes to enable `Renew expired certificates, update pending certificates, and remove revoked certificates` and `Update certificates that use certificate templates`.

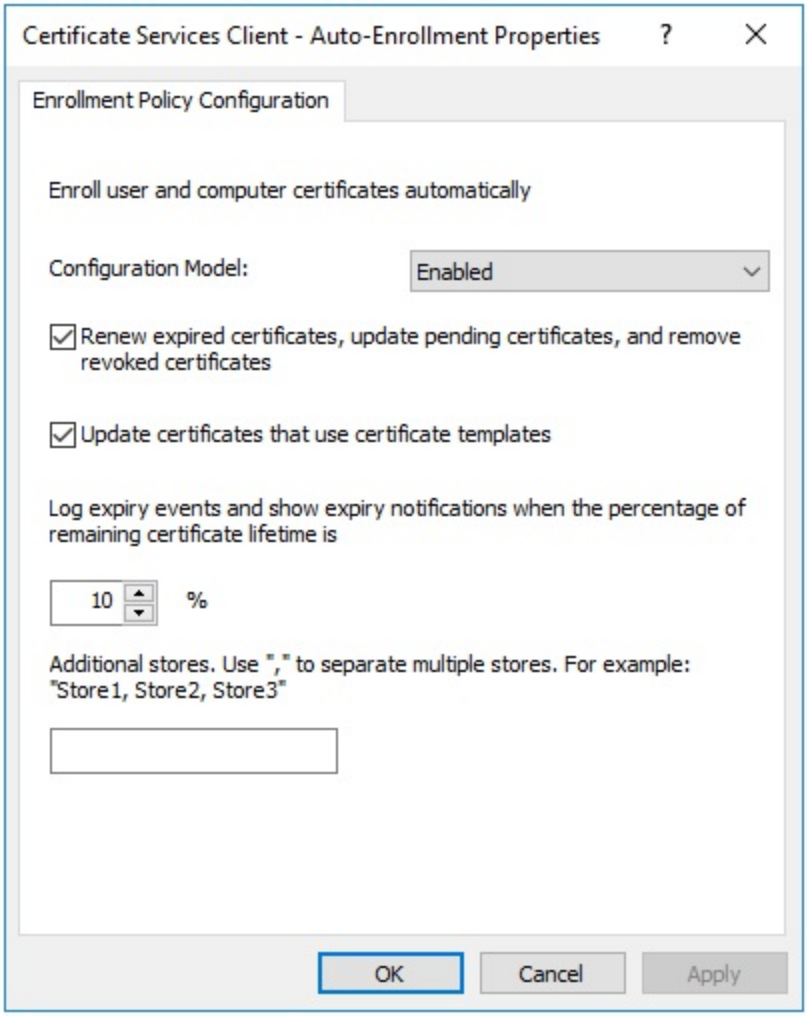


Figure 14.3: Enable Certificate Auto-Enrollment

5. Click “Apply” to apply the changes and “OK” to close the properties dialog.

This has enabled simple Certificate Auto Enrollment. Next lets configure advanced Certificate Auto Enrollment.

Advanced Certificate Auto Enrollment allows you to add multiple certificate servers to your configuration. This is useful in a more complex environment with multiple certificate servers, but is not necessary in a simple environment with a single certificate server. The advanced configuration stores the policy directly on the SYSVOL in the Registry.pol file, whereas a simple configuration is stored in LDAP.

To configure Advanced Certificate Auto Enrollment:

1. In the Group Policy Management Editor window, navigate to `Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies`.

2. Double-click on “Certificate Services Client - Certificate Enrollment Policy” to open its properties.

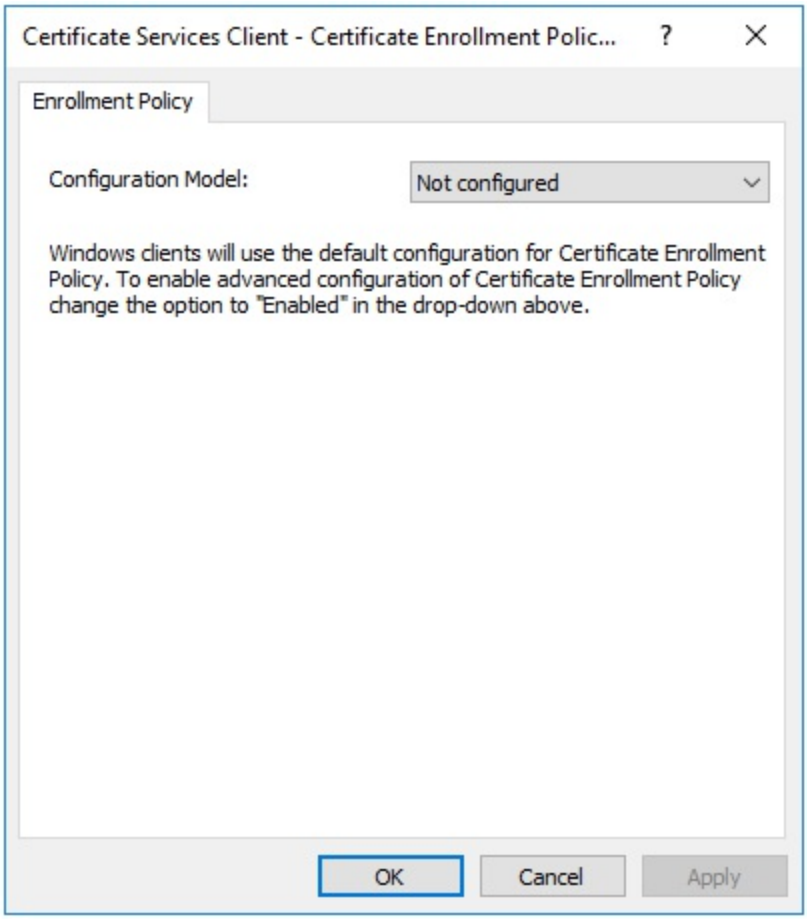


Figure 14.4: Certificate Services Client - Certificate Enrollment Policy

3. Set the `Configuration Model` to Enabled. The default policy (configured previously in the “Certificate Services Client - Auto-Enrollment” properties) should already be listed under “Certificate enrollment policy list.” You can disable this policy if desired by unchecking the box next to the item in the list.

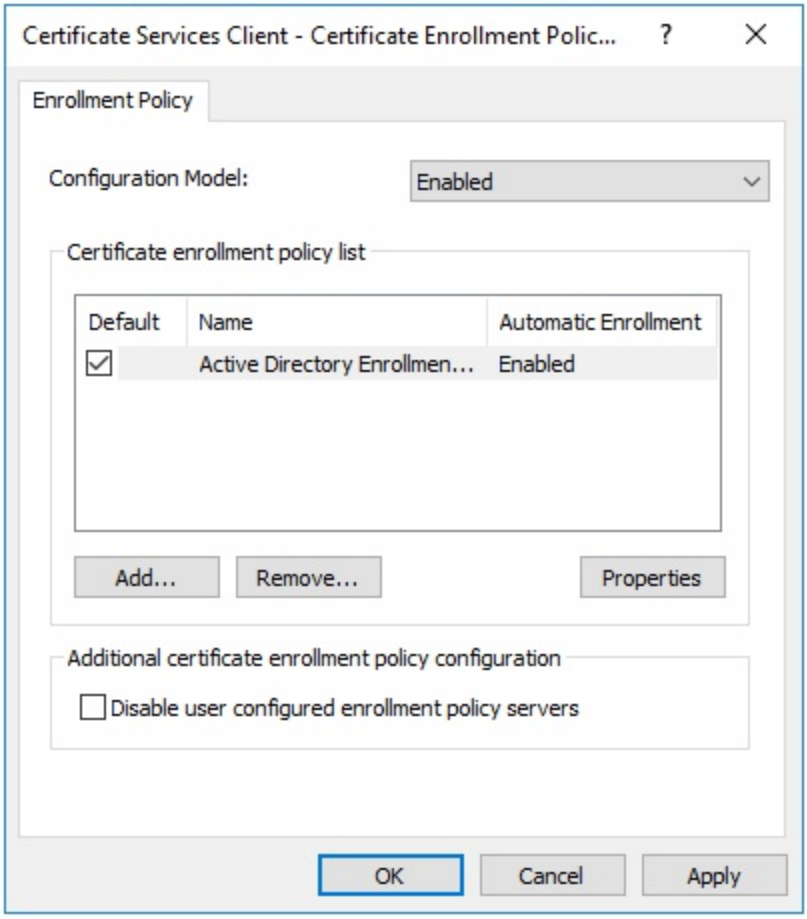


Figure 14.5: Certificate Services Client - Certificate Enrollment Policy: Enabled

4. To add additional certificate servers for Certificate Auto Enrollment, click “Add…”

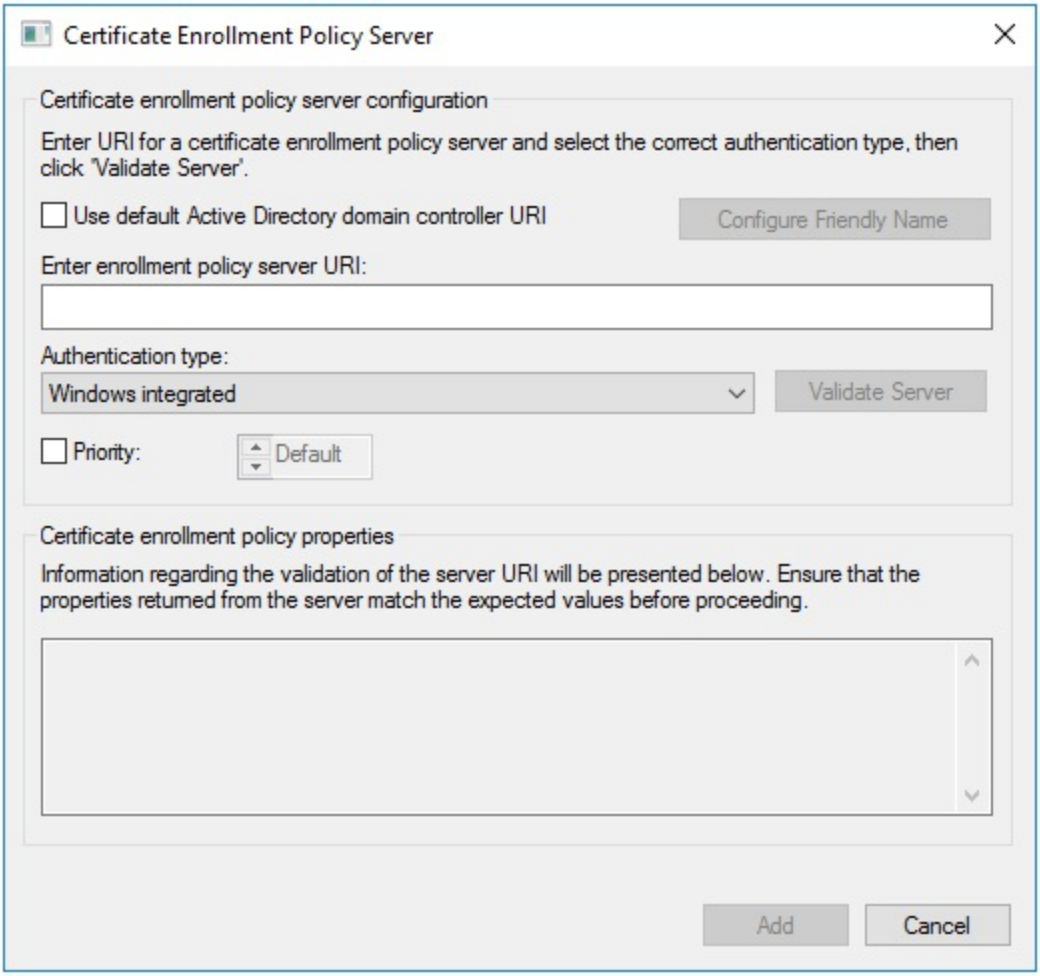


Figure 14.6: Add a Certificate Enrollment Policy Server

5. In the `Certificate Enrollment Policy Server` dialog, enter the enrollment policy server URI and set the authentication type. “Windows integrated” authentication refers to Kerberos authentication.

6. You can set a priority for the policy, which determines the order in which the policy is applied to a client machine.

7. When finished, click “Add” and then “Apply” in the properties dialog. Click “OK” to close the dialog.

14.1.2 Certificate Templates

Certificate Templates instruct the client how to generate a certificate request

for the Certificate Authority. These templates are configured using the `Certification Authority` utility included with the Remote Server Administration Tools (RSAT) found in Windows.

To create a certificate template:

1. Open the `Certification Authority` utility, and click on `Certificate Templates` in the tree.

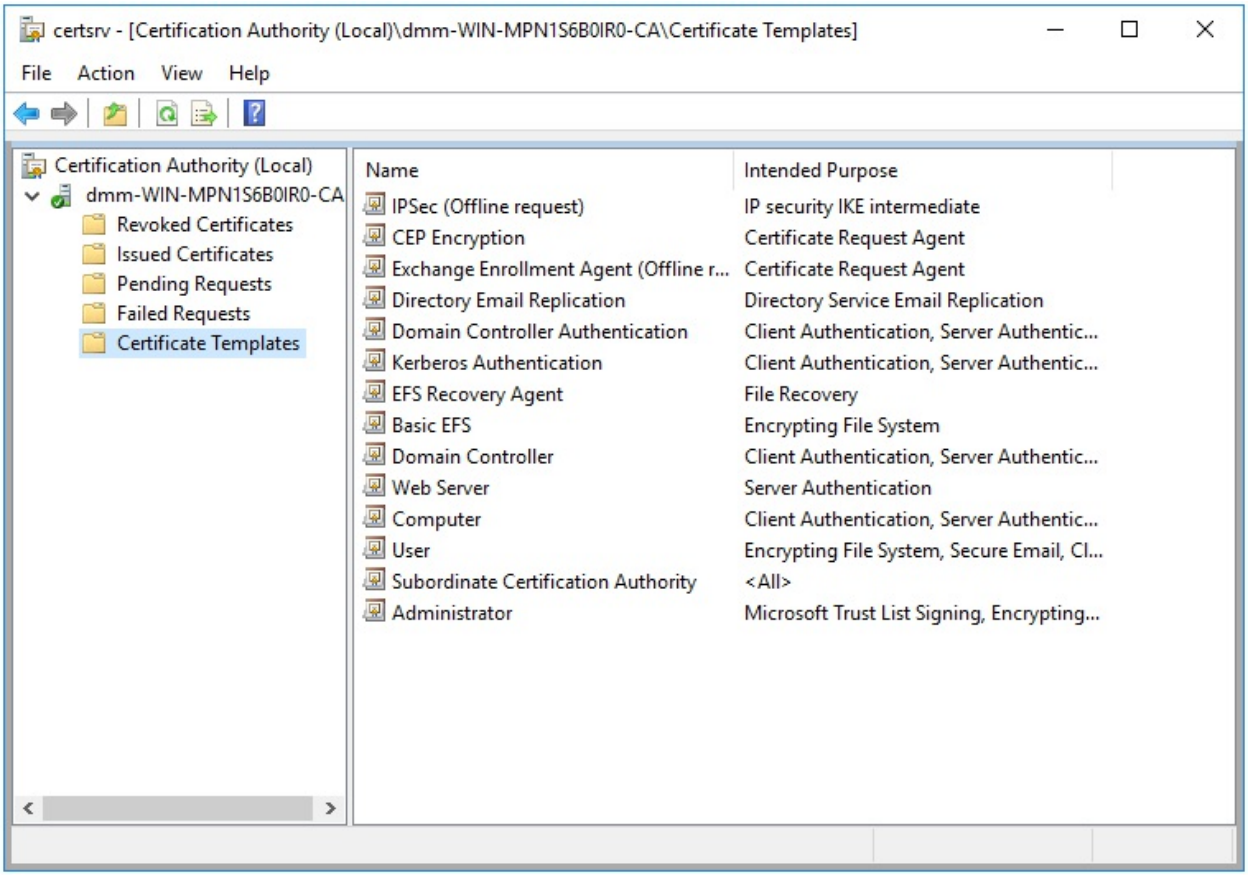


Figure 14.7: Certification Authority utility

2. Right click on `Certificates Templates` in the tree and select `Manage`.

3. You'll notice that there are a number of existing templates to choose from. Right click on one of the templates in the list, and click `Duplicate Template`.

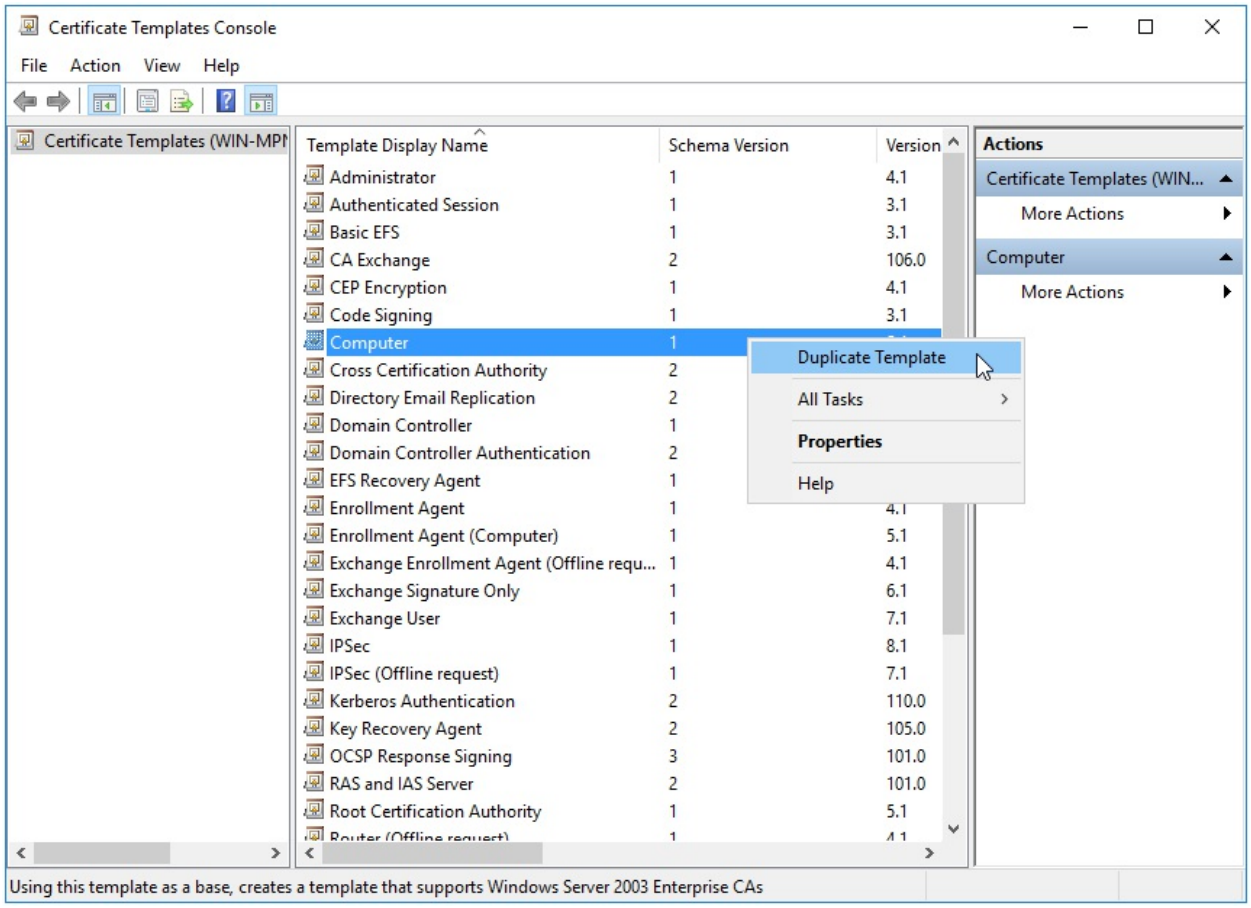


Figure 14.8: Certificate Templates Console

4. Switch to the `General` tab, and set a name for your new template.

5. Set the number of years the certificate will be valid and the renewal period.

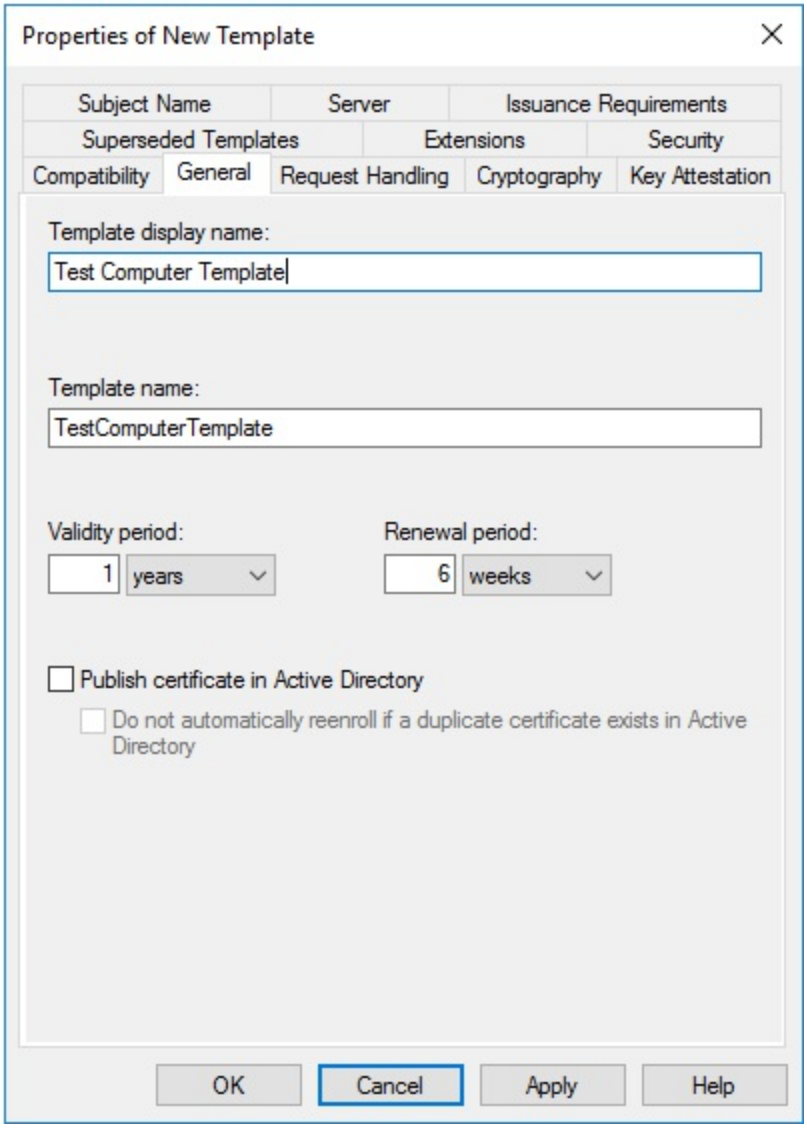


Figure 14.9: Template Properties

6. It may be useful to also review the `Cryptography` tab and ensure the cryptographic requirements for the certificate will meet the needs of your organization.

7. Apply and close the properties by clicking OK. Close the Certificate Templates Console

8. Next, to enable the new template, right click on `Certificate Templates` in the tree within the Certification Authority utility, and click `New > Certificate Template to Issue`.

9. In the `Enable Certificate Templates` dialog, select the template we just created, and click OK.

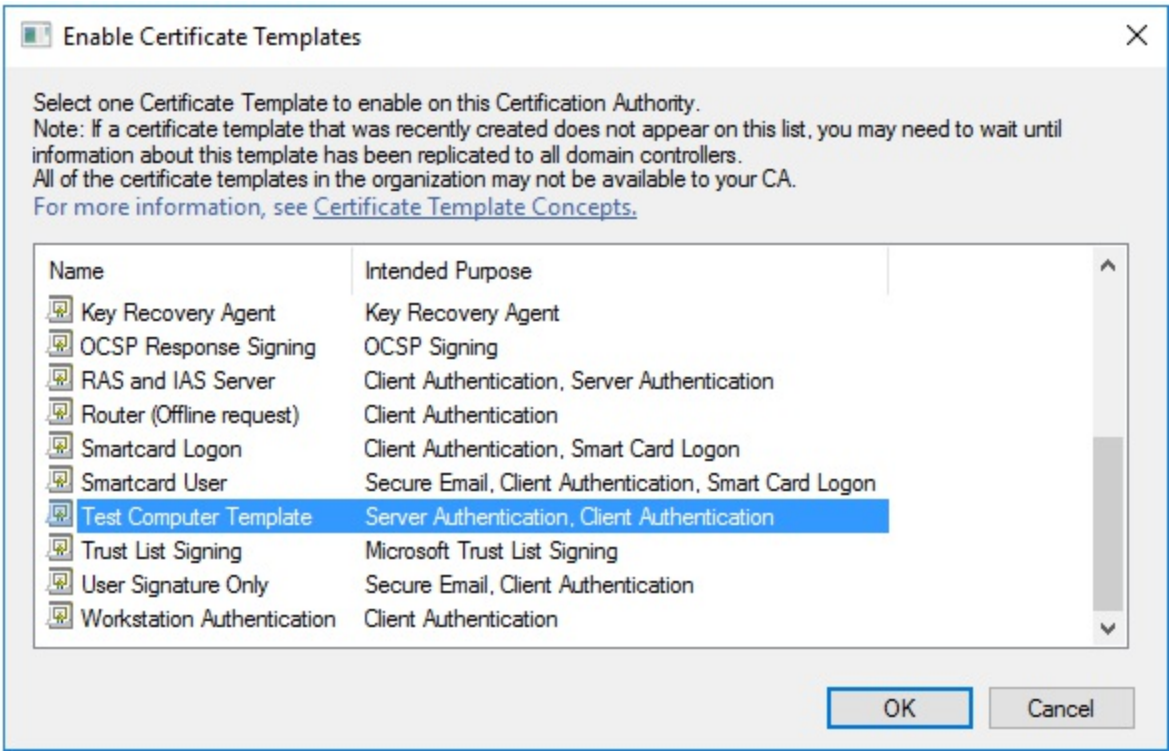


Figure 14.10: Enable Certificate Templates

Our new template is now enabled, and should be in the list of enabled templates in the Certification Authority utility.

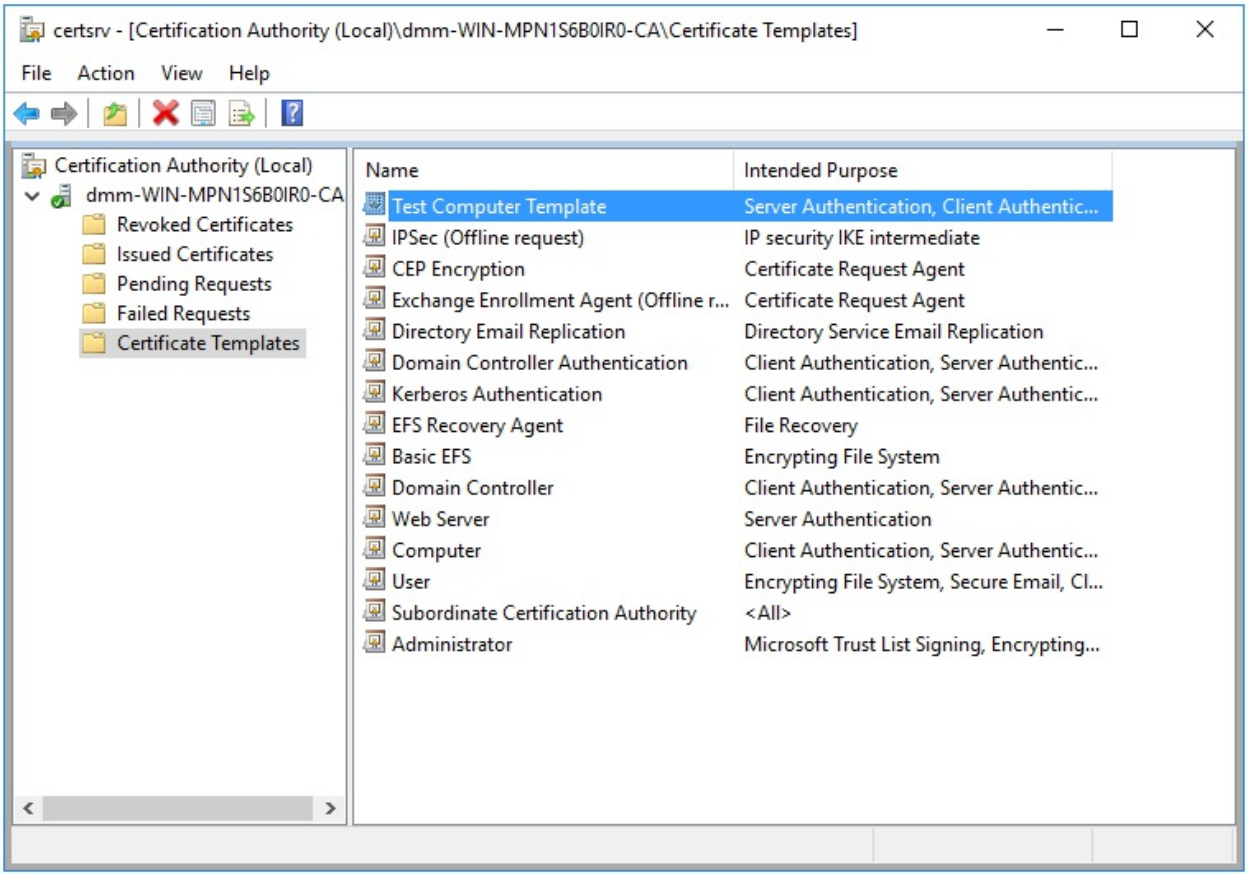


Figure 14.11: Template Enabled

14.2 Client Side Extension

The Certificate Auto Enrollment Client Side Extension (CSE) will add new Certificate Authorities to certmonger, and automatically request to track new certificates based on the assigned Certificate Templates. This CSE requires that the `certmonger` and `cepces` packages be installed.

Certificates will be installed in `/var/lib/samba/certs` and private keys in `/var/lib/samba/private/certs` by default. It may be necessary to configure a symlink policy, as explained in chapter 10, in order to link to a more appropriate location for these files.

Let’s list the Resultant Set of Policy to view the Certificate Auto Enrollment policy we created in the previous sections.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_cert_auto_enroll_ext
  ------------------------------------------------------------
    Policy Type: Auto Enrollment Policy
    ----------------------------------------------------------
    [ lizardo-WIN-QLIPDP2ISN7-CA ] =
      [ CA Certificate ] =
-----BEGIN CERTIFICATE-----
<REDACTED>
-----END CERTIFICATE-----
      [ Auto Enrollment Server ] = WIN-
QLIPDP2ISN7.lizardo.suse.de
      [ Templates ] =
        [ Machine ]
        [ TestComputerTemplate ]
    ----------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

Notice that the new template we created in the previous section is listed under the templates. The `Machine` template is also listed. This is a default template enabled for joined computers (this template was named `Computer` in the list of enabled templates we saw previously).

Let’s now force an apply and observe the results.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Cryptography\\
                                    AutoEnrollment']" - \
 | xmllint --format -
<gp_ext name="Cryptography\\AutoEnrollment">
  <attribute name="bGl6YXJkby1XSU4tUUxJUERQMklTTjctQ0E=">
    {"files": [
      "/var/lib/samba/certs/lizardo-WIN-QLIPDP2ISN7-CA.crt",
      "/etc/pki/trust/anchors/lizardo-WIN-QLIPDP2ISN7-CA.crt",
      "/var/lib/samba/private/certs/
        lizardo-WIN-QLIPDP2ISN7-CA.Machine.key",
      "/var/lib/samba/certs/
        lizardo-WIN-QLIPDP2ISN7-CA.Machine.crt",
```

```
      "/var/lib/samba/private/certs/
        lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate.key",
      "/var/lib/samba/certs/
        lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate.crt"
      ],
    "templates": [
      "lizardo-WIN-QLIPDP2ISN7-CA.Machine",
      "lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate"
      ]
    }
  </attribute>
</gp_ext>
```

The log details show that a number of files were created. Let's take a look at these files.

```
> sudo ls /var/lib/samba/certs/
lizardo-WIN-QLIPDP2ISN7-CA.crt
lizardo-WIN-QLIPDP2ISN7-CA.Machine.crt
lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate.crt
> sudo ls /var/lib/samba/private/certs/
lizardo-WIN-QLIPDP2ISN7-CA.Machine.key
lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate.key
> sudo ls /etc/pki/trust/anchors/
lizardo-WIN-QLIPDP2ISN7-CA.crt ->
  /var/lib/samba/certs/lizardo-WIN-QLIPDP2ISN7-CA.crt
```

The CSE download the root certificate from our CA (`lizardo-WIN-QLIPDP2ISN7-CA.crt`), and then linked it inside `/etc/pki/trust/anchors`. This is the system trust store. In addition to linking our CA root certificate to the trust store, the CSE also called the `update-ca-certificates` command to refresh the certificate store. Take a look at `man update-ca-certificates` for more details on how this works.

Now that we see our certificates have been installed, lets take a look at certmonger to see our CA and make sure our certificates are being auto-renewed.

```
> sudo getcert list-cas
CA 'lizardo-WIN-QLIPDP2ISN7-CA':
    is-default: no
    ca-type: EXTERNAL
    helper-location: /usr/libexec/certmonger/cepces-submit \
      --server=WIN-QLIPDP2ISN7.lizardo.suse.de \
```

```
      --auth=Kerberos
> sudo getcert list
Number of certificates and requests being tracked: 2.
Request ID 'lizardo-WIN-QLIPDP2ISN7-CA.Machine':
    status: MONITORING
    stuck: no
    key pair storage: type=FILE,
      location='/var/lib/samba/private/certs/
                lizardo-WIN-QLIPDP2ISN7-CA.Machine.key'
    certificate: type=FILE,
      location='/var/lib/samba/certs/
                lizardo-WIN-QLIPDP2ISN7-CA.Machine.crt'
    CA: lizardo-WIN-QLIPDP2ISN7-CA
    issuer: CN=lizardo-WIN-QLIPDP2ISN7-CA,
            DC=lizardo,DC=suse,DC=de
    subject: CN=testsysdm.lizardo.suse.de
    issued: 2022-12-08 12:57:35 MST
    expires: 2023-12-08 12:57:35 MST
    dns: testsysdm.lizardo.suse.de
    key usage: digitalSignature,keyEncipherment
    eku: id-kp-clientAuth,id-kp-serverAuth
    certificate template/profile: Machine
    profile: Machine
    pre-save command:
    post-save command:
    track: yes
    auto-renew: yes
Request ID 'lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate':
    status: MONITORING
    stuck: no
    key pair storage: type=FILE,
      location='/var/lib/samba/private/certs/
        lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate.key'
    certificate: type=FILE,
      location='/var/lib/samba/certs/
        lizardo-WIN-QLIPDP2ISN7-CA.TestComputerTemplate.crt'
    CA: lizardo-WIN-QLIPDP2ISN7-CA
    issuer: CN=lizardo-WIN-QLIPDP2ISN7-CA,
            DC=lizardo,DC=suse,DC=de
    subject:
    issued: 2022-12-08 12:57:35 MST
    expires: 2023-12-08 12:57:35 MST
    dns: testsysdm.lizardo.suse.de
    key usage: digitalSignature,keyEncipherment
    eku: id-kp-serverAuth,id-kp-clientAuth
    profile: TestComputerTemplate
    pre-save command:
    post-save command:
```

```
    track: yes
    auto-renew: yes
```

We see that certmonger is aware of our CA, and is tracking the template we created, and also the default Machine template. The status on your certificates should say `MONITORING`, as it does here. If not, then you'll need to do some investigation to see why.

14.2.1 Trouble Shooting Certificates

If your certificates are not listed with a `MONITORING` status in `getcert list`, then you can start trouble shooting by rerunning `sudo /usr/sbin/samba-gpupdate --force`. Look for any errors in the output that could indicate what caused the problem. For example, a common error you'll encounter is `Failed to fetch the list of supported templates`. This error comes with a detailed backtrace. Look for the `Caused by` message.

14.2.1.1 Certificate doesn't match

This particular error happens when Internet Information Services (IIS) has the wrong root certificate selected.

```
(Caused by
  SSLError(
    CertificateError(
      "hostname \'win-qlipdp2isn7.lizardo.suse.de\' doesn\'t
      match \'lizardo-WIN-QLIPDP2ISN7-CA\'"
    )
  )
)
```

To resolve this error, open the IIS Manager, select the Default Web Site in the tree, then click `Bindings` in the right side `Actions` pane.

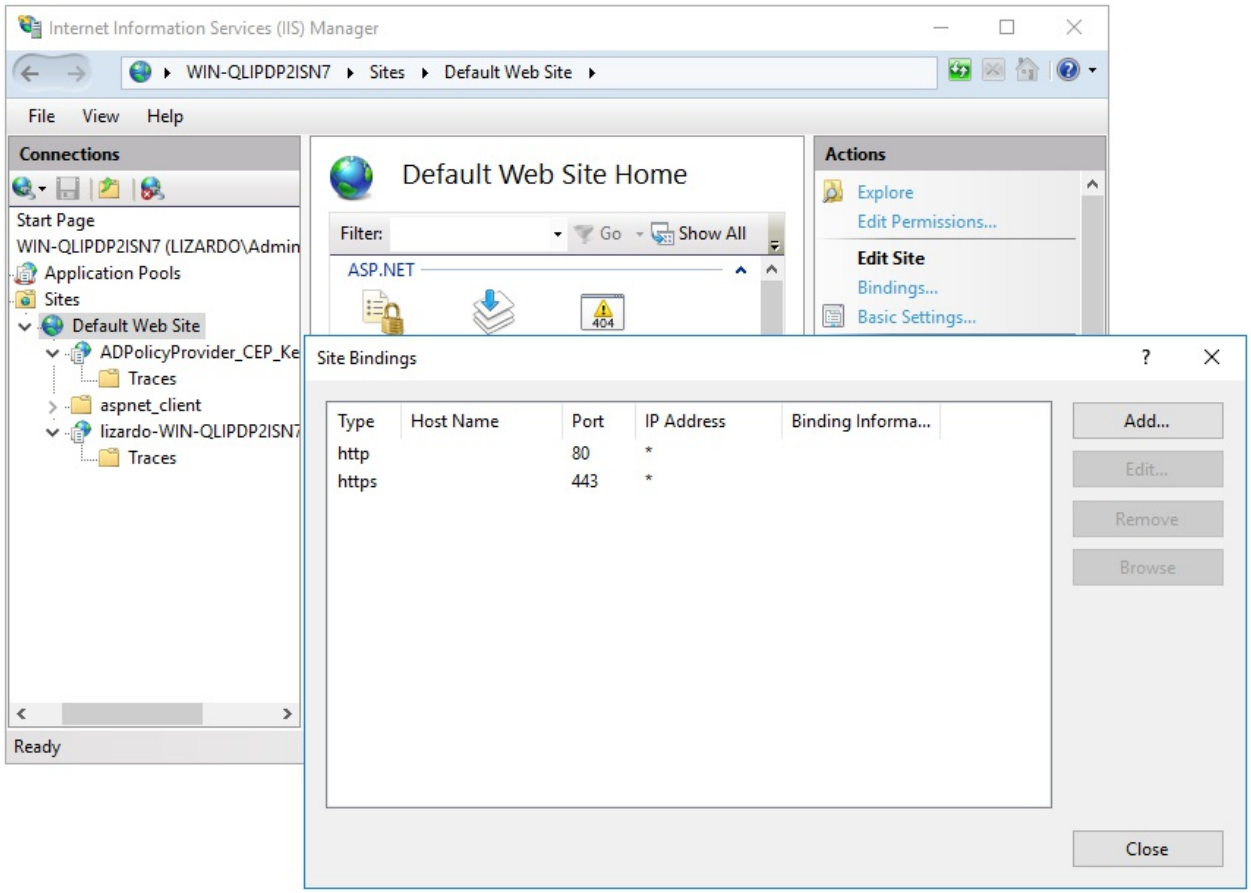


Figure 14.12: IIS Manager

Highlight the https binding, then select `Edit...`. Select the correct SSL certificate, then click OK to save.

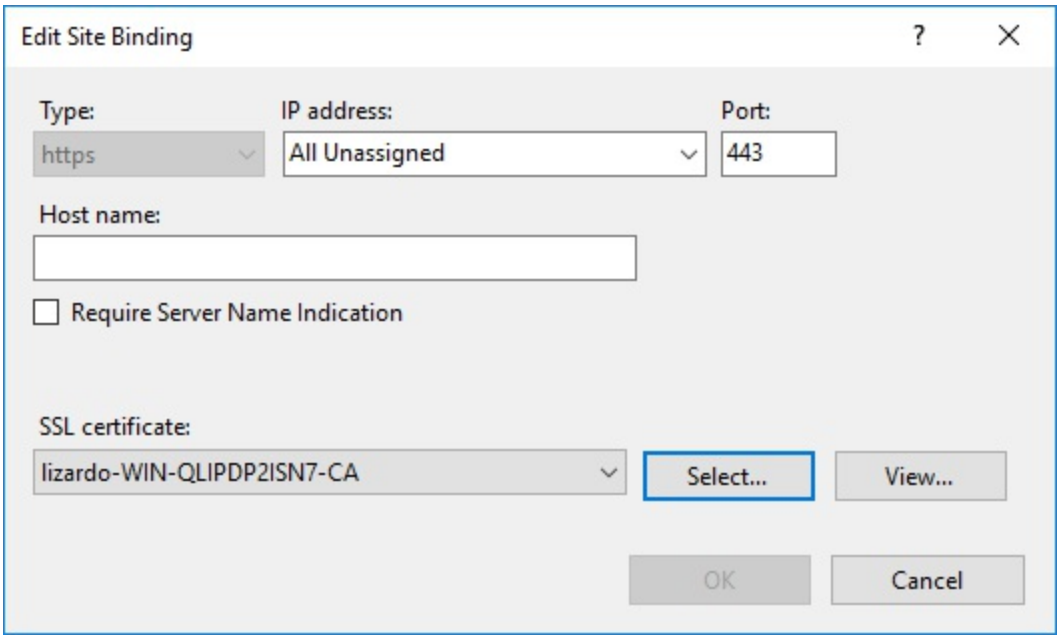


Figure 14.13: Edit Site Binding

14.2.1.2 python-requests security level

Another error you may encounter is actually specific to the *python-requests* module.

```
(Caused by
  SSLError(
    SSLError(1,
      '[SSL: DH_KEY_TOO_SMALL] dh key too small (_ssl.c:852)'
    ),
  )
)
```

This `DH_KEY_TOO_SMALL` error is caused by a default security level change in *python-requests*. You can work around this error by setting `openssl_seclevel=1` in the **global** section of your `cepces.conf`.

14.2.1.3 Checking request failures

If the CA failed to issue a certificate when the template request was sent, we can check the Certification Authority utility to see why a request was rejected. If you click in the tree, there are lists of `Failed Requests`, `Pending`

`Requests`, `Issued Certificates`, and `Revoked Certificates`. If your request failed for some reason, it may show up under `Failed Requests`, for example.

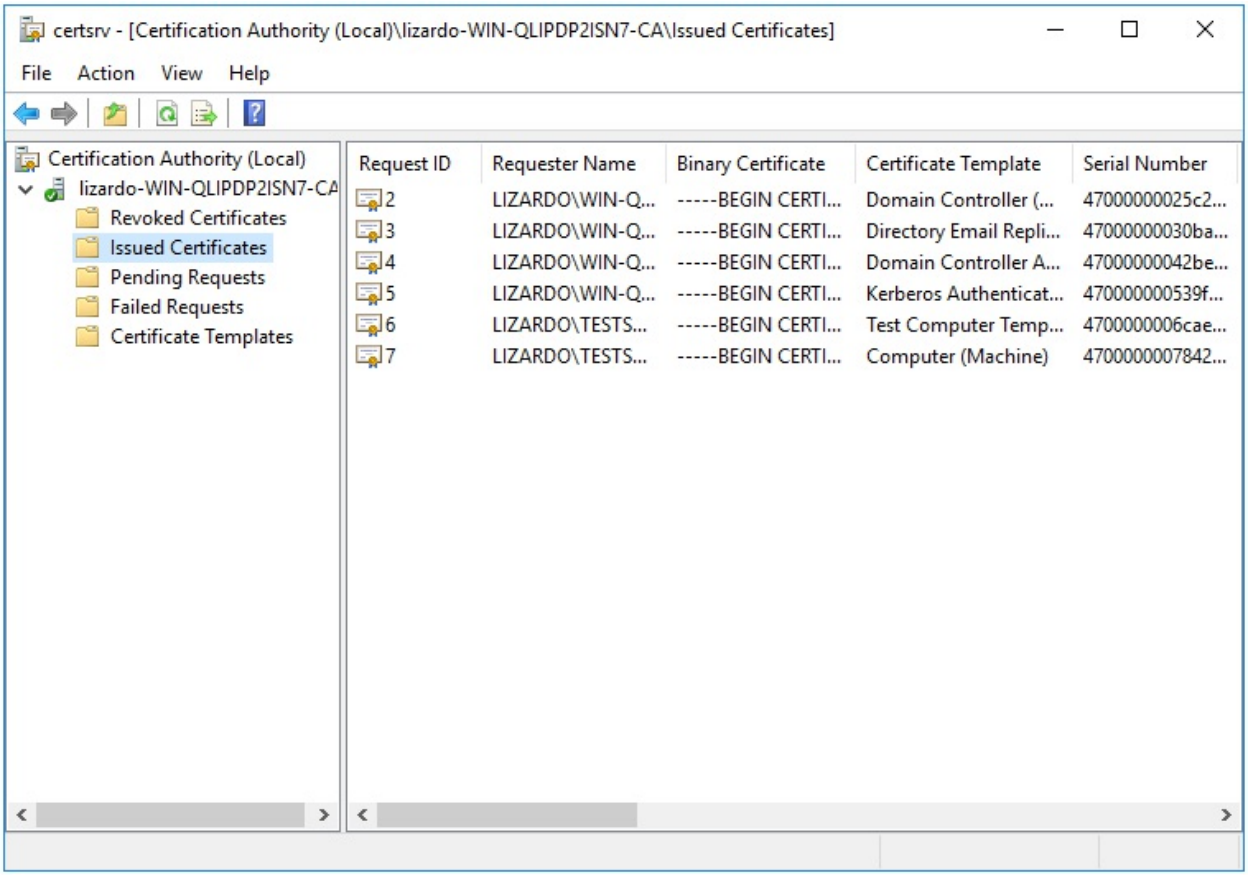


Figure 14.14: Issued Certificates

You can see, for example, in the `Issued Certificates` list, that the certificates we are monitoring on our test machine are listed here (the two most recent requests).

14.2.1.4 Examining logs

Check the logs to track down other issues. The *cepces* plugin logs to `/var/log/cepces/cepces.log` by default. Check the **handler_fileHandler** section of `/etc/cepces/logging.conf` for the exact location of the log. Additionally, some errors may end up in the *certmonger* logs, instead of *cepces*. You can check `journalctl -xe -t certmonger` for *certmonger*

messages.

15 Firefox Policy

Firefox Policy deploys a `policies.json` file to client machines to customize how Firefox looks and operates.

This policy is physically stored on the SYSVOL in **MACHINE/Registry.pol**. It is stored in registry format. See chapter 21 for details on how to manually modify this file.

15.1 Server Side Extension

The Server Side Extension (SSE) for Firefox Policy is distributed via Administrative Templates (see chapter 20.1 in section 20.1.1). This SSE uses the admx templates provided by Mozilla.

Setting up the ADMX templates for this policy is described in chapter 22 section 22.2.

15.2 Managing Firefox Policy via the GPME

Open the GPME and navigate to `Computer Configuration > Administrative Templates > Mozilla > Firefox`.

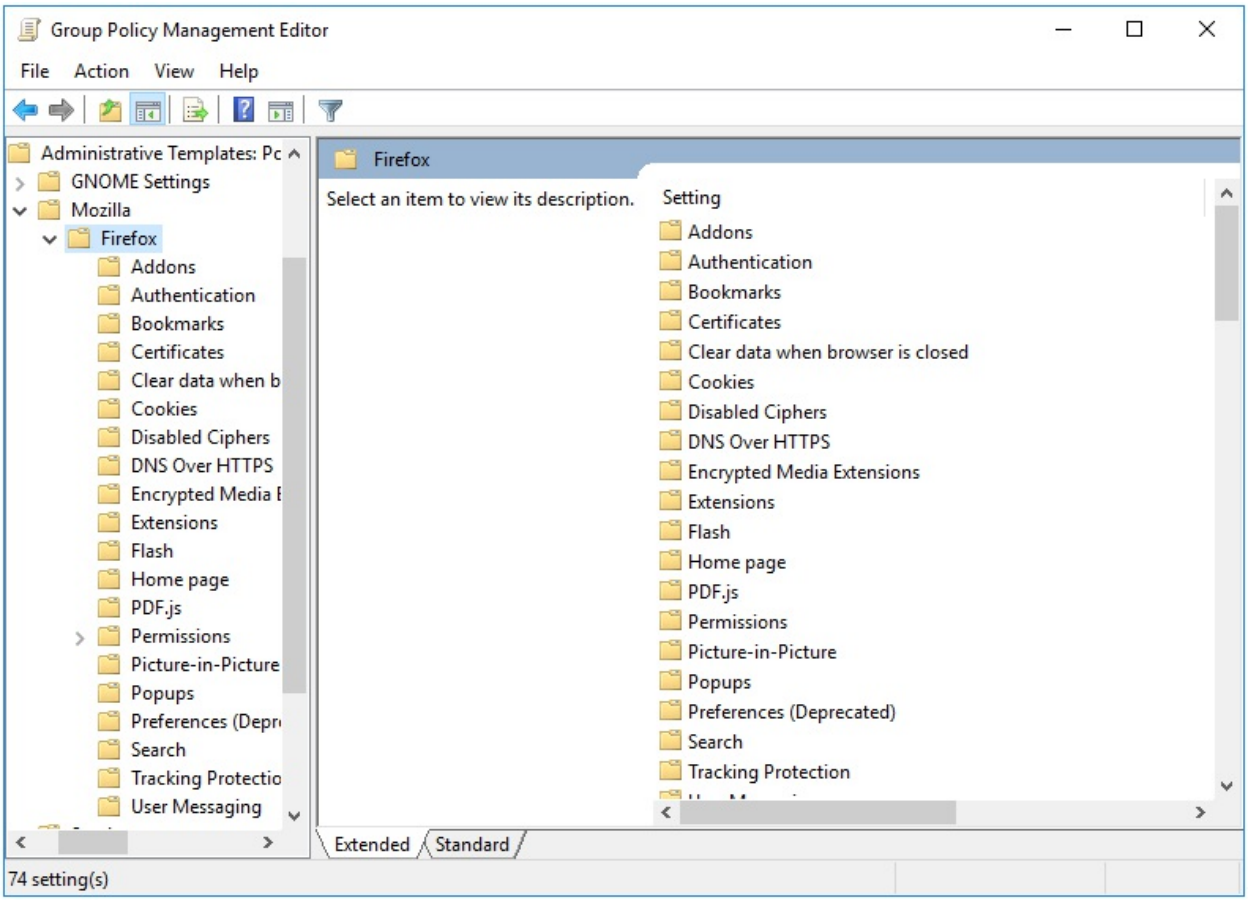


Figure 15.1: Mozilla Administrative Templates

There are many options to choose from, but for this example we'll just test setting a homepage. Click on `Home page` in the tree, and we're going to modify both the `Start Page` and the `URL for Home page`.

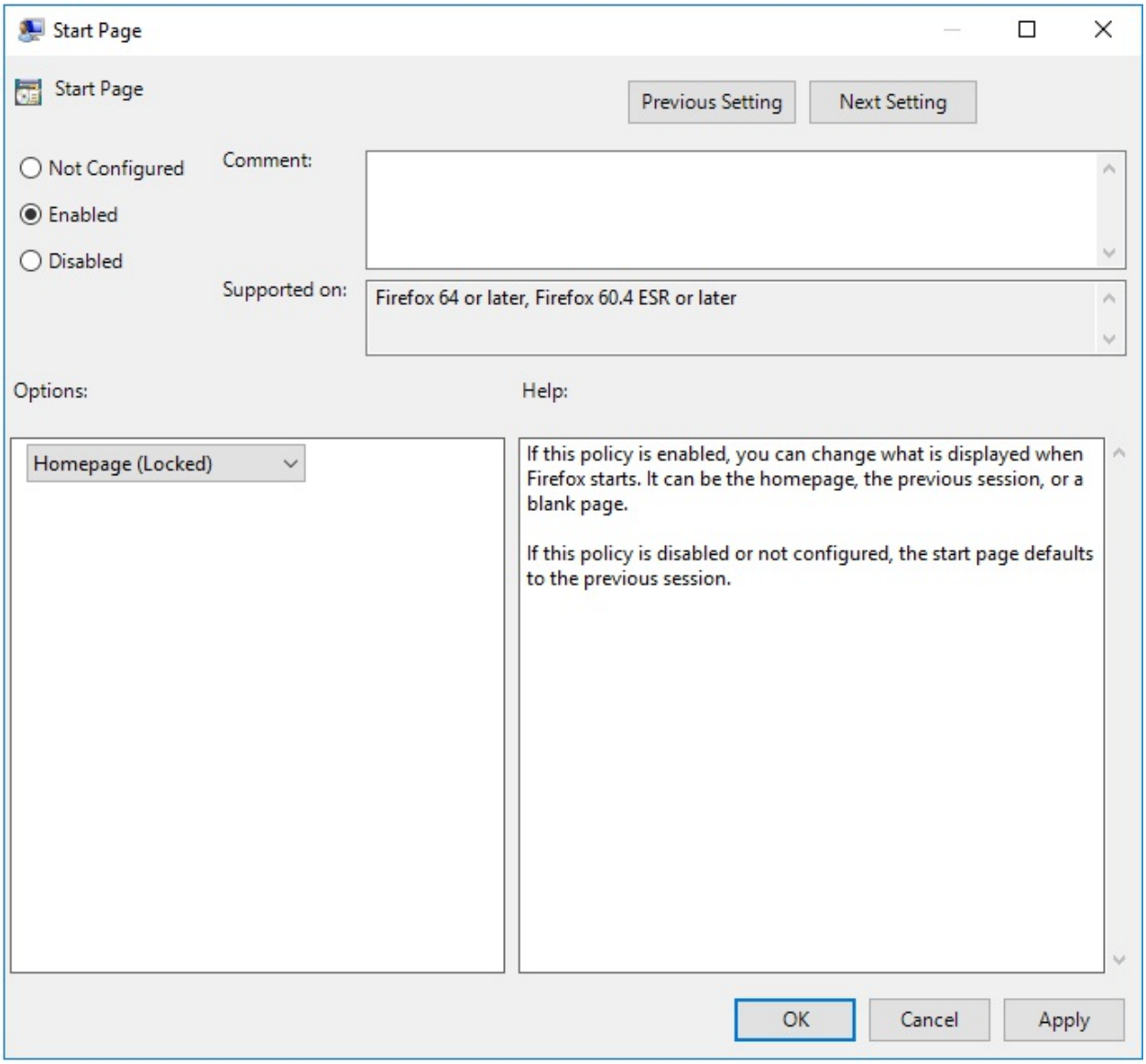


Figure 15.2: Start Page

First we'll set the `Start Page` to `Homepage (Locked)`. This requires the start page to always open the homepage, and the user will be unable to change it to point anywhere else.

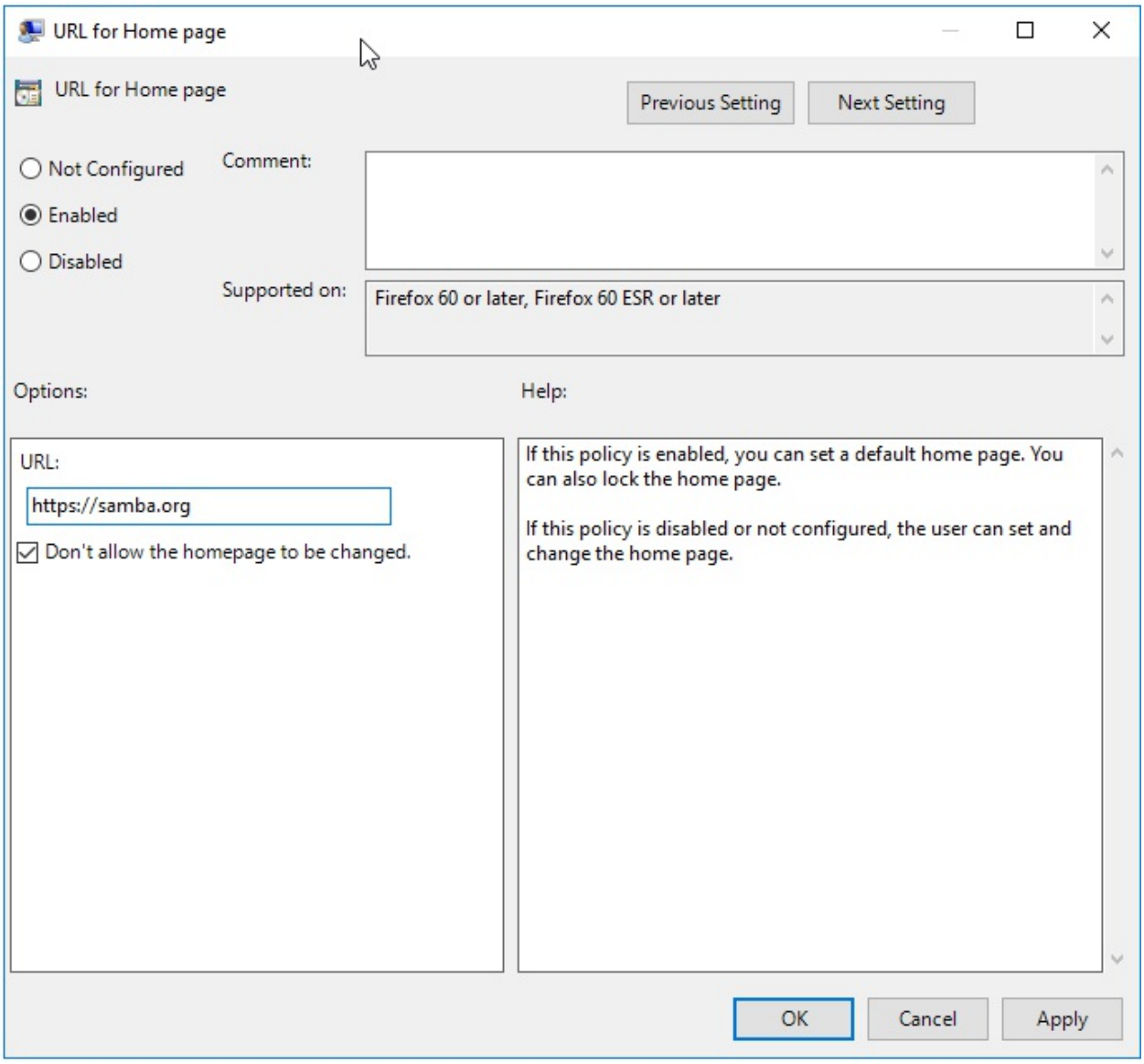


Figure 15.3: Home Page

Next we'll set the homepage URL, and check the box `Don't allow the homepage to be changed`. This will prevent the user from changing the homepage.

Note that when setting a home page, you *MUST* include the http or https prefix, or Firefox will ignore the policy.

15.3 Client Side Extension

The Firefox Client Side Extension (CSE) creates 2 policy files. One at `/usr/lib64/firefox/distribution/policies.json` and the other at `/etc/firefox/policies/policies.json`. These files will contain exactly the same information. The only reason there is a duplicate, is because different versions of Firefox require the policy file in different locations.

Let's list the Resultant Set of Policy to view policies we set in the previous section.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_firefox_ext
  ------------------------------------------------------------
    Policy Type: Software\Policies\Mozilla\Firefox\
                 Homepage\StartPage
    ------------------------------------------------------------
      homepage-locked
    ------------------------------------------------------------
    Policy Type: Software\Policies\Mozilla\Firefox\
                 Homepage\URL
    ------------------------------------------------------------
      https://samba.org
    ------------------------------------------------------------
    Policy Type: Software\Policies\Mozilla\Firefox\
                 Homepage\Locked
    ------------------------------------------------------------
      1
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

We see that the policy we set is listed. Next let's force an apply of the policy, and see what is logged in the Group Policy Cache.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Mozilla/Firefox']" - \
 | xmllint --format -
<gp_ext name="Mozilla/Firefox">
  <attribute name="policies.json">
```

```
    {"policies": {}}
  </attribute>
</gp_ext>
```

There isn't much interesting in the cache, since it just tells us that we had no policies set previously. Let's look inside our policy file and see what was applied.

```
> npx prettier /etc/firefox/policies/policies.json
{
  "policies": {
    "Homepage": {
      "StartPage": "homepage-locked",
      "URL": "https://samba.org",
      "Locked": true
    }
  }
}
```

It appears that our policies are applied. Opening Firefox, we see that the policy is being enforced.

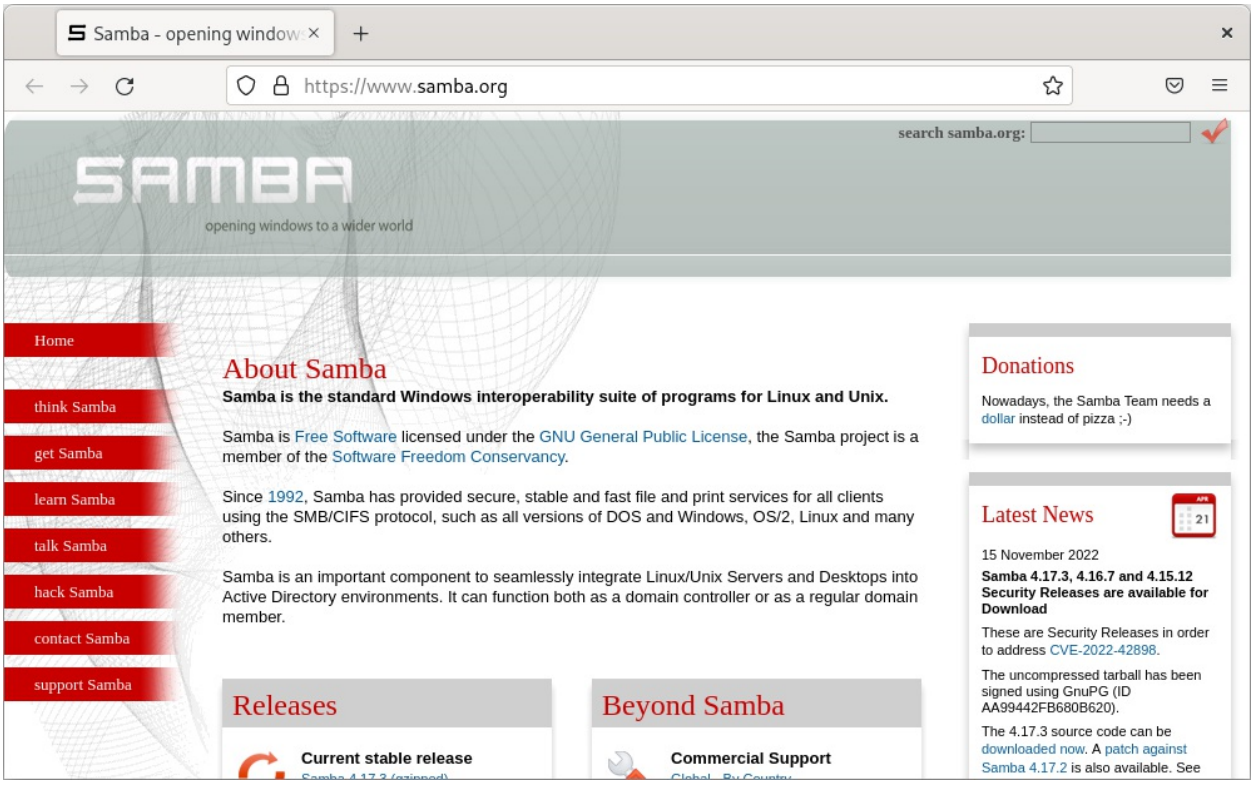


Figure 15.4: Firefox

If you open your browser settings, you'll notice a warning message indicating that your organization is managing the browser. If Firefox encountered any problems enforcing the policy, a warning will be listed here.

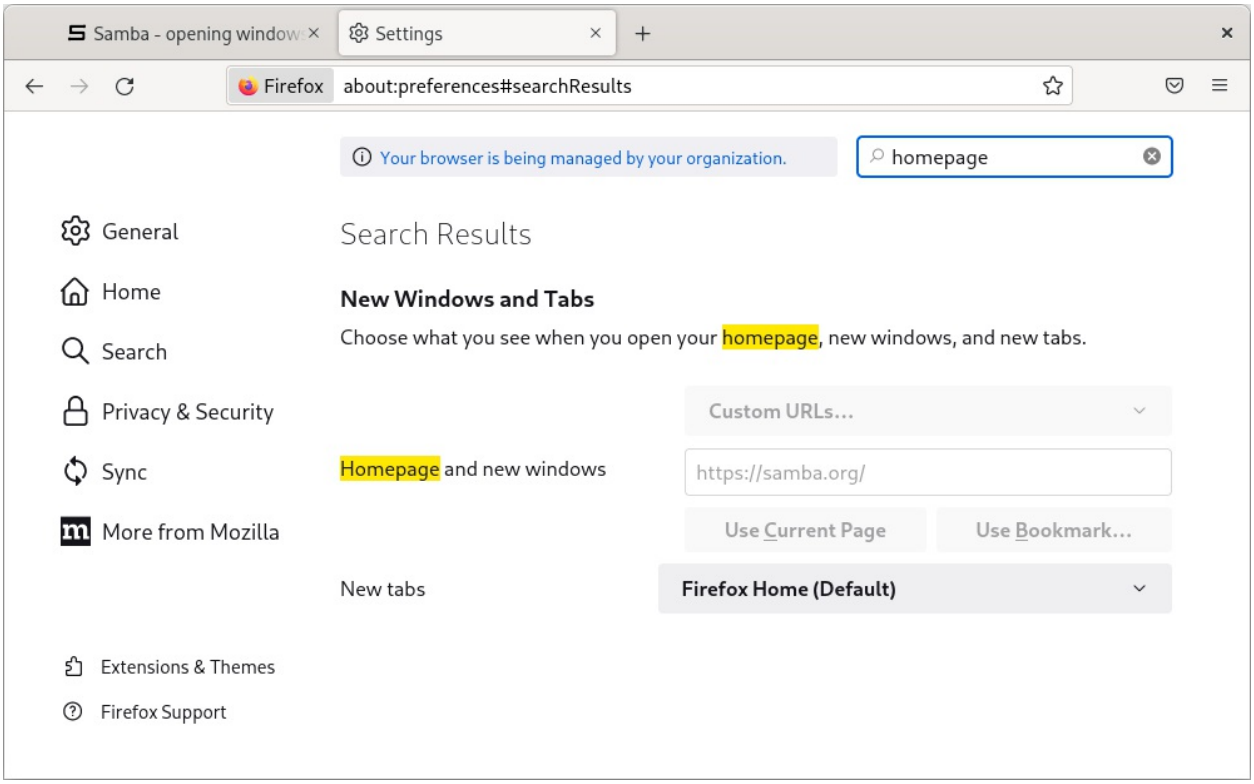


Figure 15.5: Managed Firefox

Notice that the homepage settings are now grayed out in the settings dialog, and can't be modified by the user.

16 Chromium/Chrome Policy

Chromium Policy deploys a json file to client machines to customize how the browser looks and operates.

This policy is physically stored on the SYSVOL in **MACHINE/Registry.pol**. It is stored in registry format. See chapter 21 for details on how to manually modify this file.

16.1 Server Side Extension

The Server Side Extension (SSE) for Chromium Policy is distributed via Administrative Templates (see chapter 20.1 in section 20.1.1). This SSE uses the admx templates provided by Google. See https://support.google.com/chrome/a/answer/187202 for the latest release of Google's templates.

Setting up the ADMX templates for this policy is described in chapter 22 section 22.3.

16.1.1 Managing Chromium Policy via the GPME

Open the GPME and navigate to `Computer Configuration > Administrative Templates > Google`.

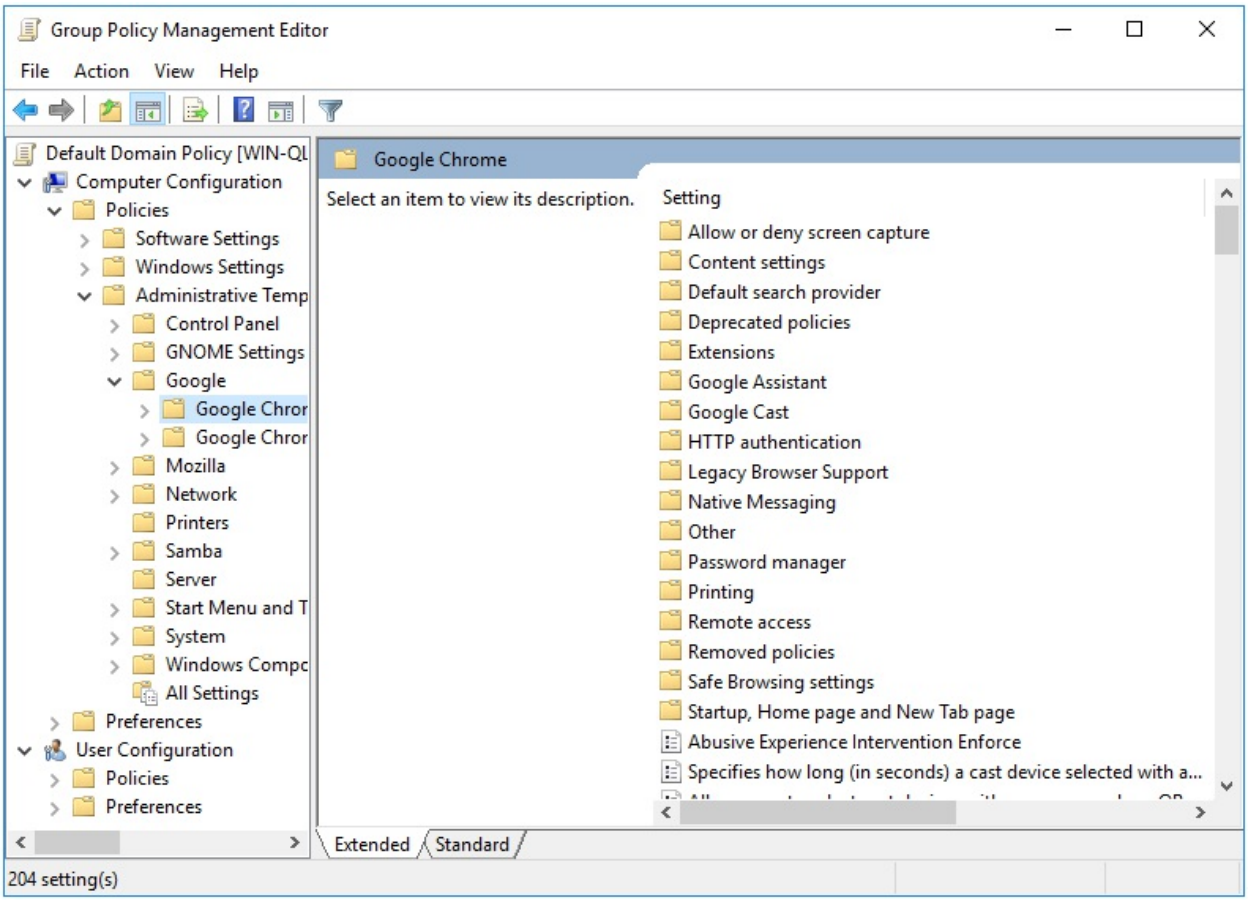


Figure 16.1: Google Administrative Templates

You'll notice that there are two sections for Chrome, titled `Google Chrome` and `Google Chrome - Default Settings (users can override)`. The settings in `Google Chrome` will always be enforced. The settings in `Default Settings` will be distributed, but will not be enforced (users can modify these).

For this example, let's enforce a homepage. Select `Google Chrome > Startup, Home page and New Tab page`. First, let's set the `New Tab Page as homepage` option, forcing our homepage choice to load when we first open a Chromium tab.

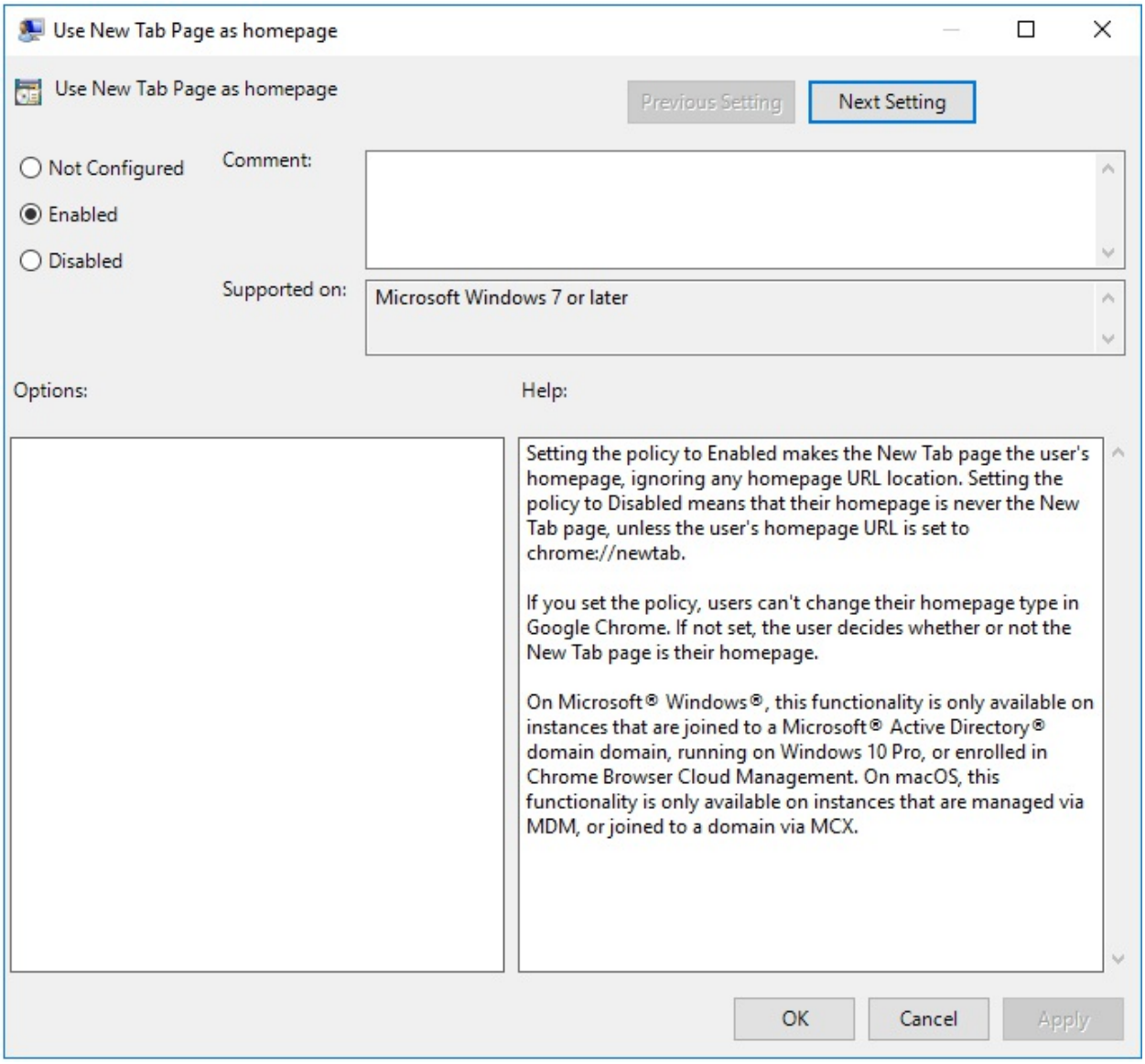


Figure 16.2: Use New Tab Page as homepage

Next, let’s set the new tab page URL.

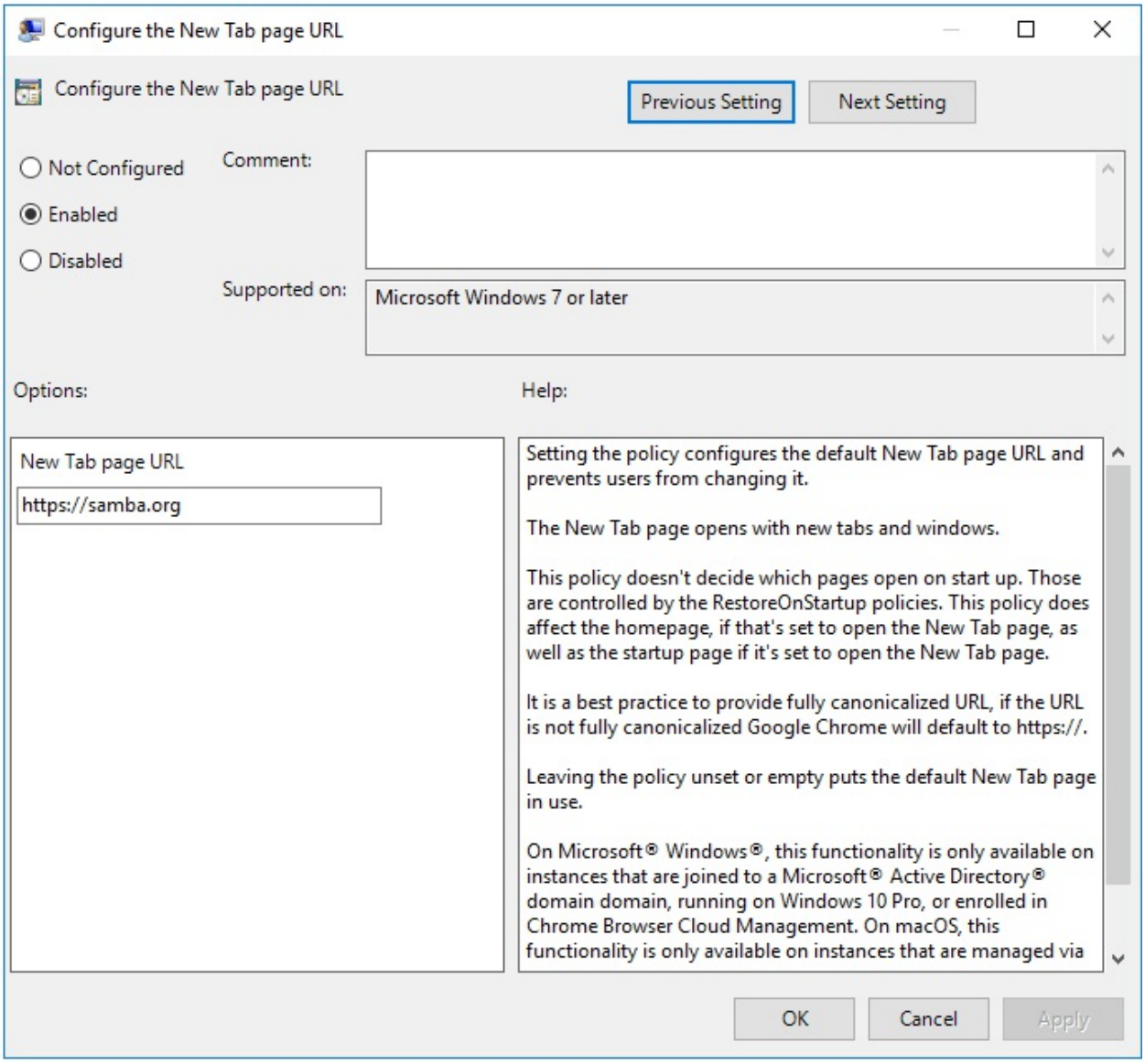


Figure 16.3: Configure the New Tab page URL

Finally, let’s tell Chromium to default to the New Tab Page when first opening.

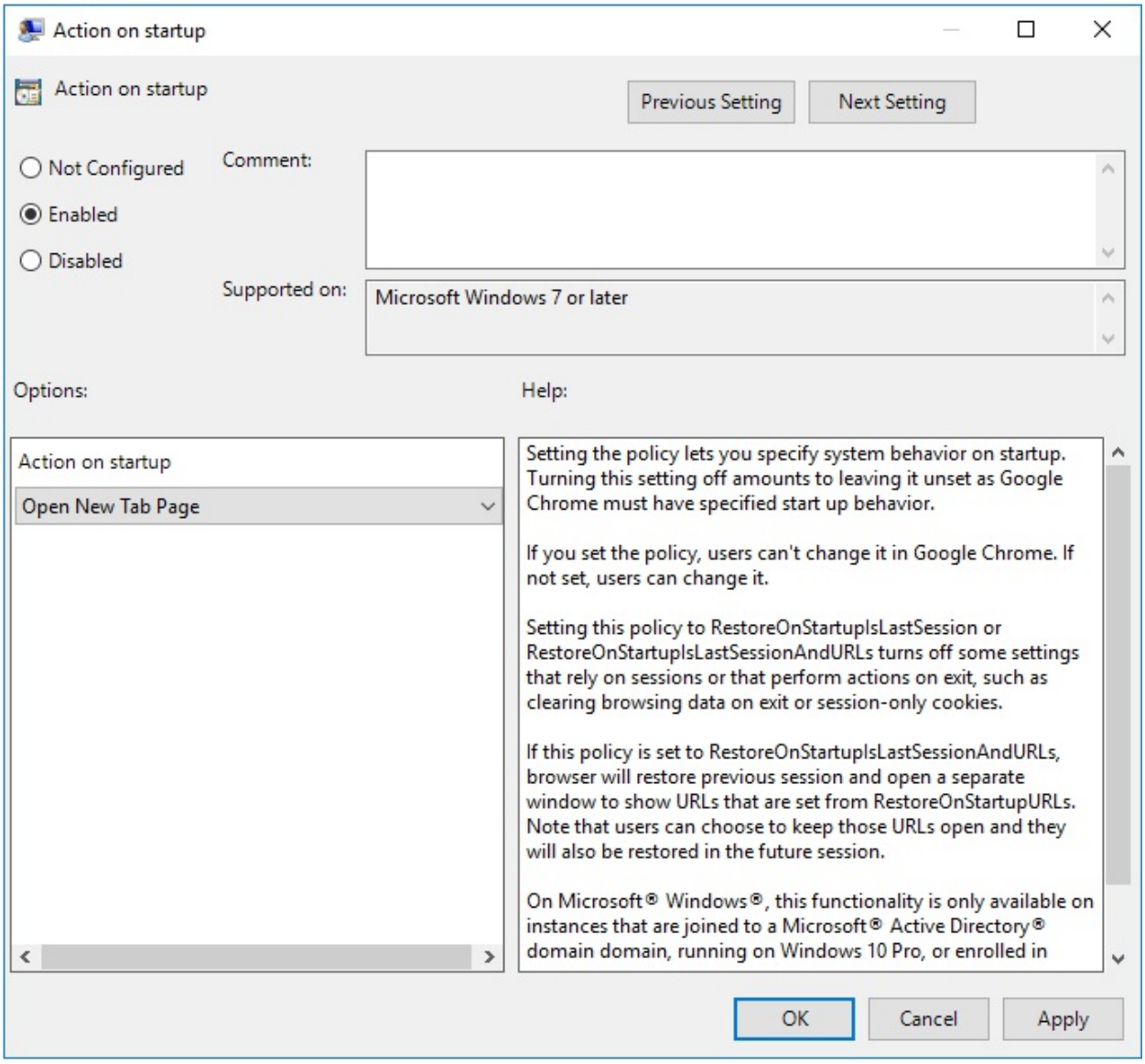


Figure 16.4: Action on startup

16.2 Client Side Extension

Chromium policy comes with 2 Client Side Extensions (CSEs), which generate a total of 4 policy files. Each CSE generates a `managed` policy file (these policies are enforced), and a `recommended` policy file (these policies are set but not enforced). The generated json files can be found in `/etc/chromium/policies/managed`, `/etc/chromium/policies/recommended`,

`/etc/opt/chrome/policies/managed`, and `/etc/opt/chrome/policies/recommended`.

Let’s list the Resultant Set of Policy to view policies we set in the previous section.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_chromium_ext
  ------------------------------------------------------------
    Policy Type: Software\Policies\Google\Chrome\
                 HomepageIsNewTabPage
    ------------------------------------------------------------
      1
    ------------------------------------------------------------
    Policy Type: Software\Policies\Google\Chrome\
                 NewTabPageLocation
    ------------------------------------------------------------
      https://samba.org
    ------------------------------------------------------------
    Policy Type: Software\Policies\Google\Chrome\
                 RestoreOnStartup
    ------------------------------------------------------------
      5
    ------------------------------------------------------------
  ------------------------------------------------------------
  CSE: gp_chrome_ext
  ------------------------------------------------------------
    Policy Type: Software\Policies\Google\Chrome\
                 HomepageIsNewTabPage
    ------------------------------------------------------------
      1
    ------------------------------------------------------------
    Policy Type: Software\Policies\Google\Chrome\
                 NewTabPageLocation
    ------------------------------------------------------------
      https://samba.org
    ------------------------------------------------------------
    Policy Type: Software\Policies\Google\Chrome\
                 RestoreOnStartup
    ------------------------------------------------------------
      5
```

```
    ------------------------------------------------------------
  ----------------------------------------------------------
==============================================================
```

Notice that the policy which will be applied for Chrome and Chromium are identical. This is because each CSE is reading from the same policy. Let’s now force a policy apply, and see what is logged in the Group Policy Cache.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Google/Chromium' or
                             @name='Google/Chrome']" - \
 | xmllint --format -
<gp_ext name="Google/Chromium">
  <attribute name="recommended">
    9327...d13f:
      /etc/chromium/policies/recommended/gp_g4_82tuo.json
  </attribute>
  <attribute name="managed">
    d452...1935:
      /etc/chromium/policies/managed/gp_l_k9uvxk.json
  </attribute>
</gp_ext>
<gp_ext name="Google/Chrome">
  <attribute name="recommended">
    9327...d13f:
      /etc/opt/chrome/policies/recommended/gp_7p6q0bxf.json
  </attribute>
  <attribute name="managed">
    d452...1935:
      /etc/opt/chrome/policies/managed/gp_fcthg4bc.json
  </attribute>
</gp_ext>
```

Our cache shows that 4 json policies were created in the directories where we expected them. Let’s look at the contents.

```
> npx prettier
/etc/chromium/policies/recommended/gp_g4_82tuo.json
{}
> npx prettier /etc/chromium/policies/managed/gp_l_k9uvxk.json
{
  "HomepageIsNewTabPage": true,
  "NewTabPageLocation": "https://samba.org",
  "RestoreOnStartup": 5
```

```
}
> npx prettier \
  /etc/opt/chrome/policies/recommended/gp_7p6q0bxf.json
{}
> npx prettier /etc/opt/chrome/policies/managed/gp_fcthg4bc.json
{
  "HomepageIsNewTabPage": true,
  "NewTabPageLocation": "https://samba.org",
  "RestoreOnStartup": 5
}
```

As expected, the `recommended` policy files are empty, since we only set `managed` policy. The `managed` is what we expected. Let's now open Chrome, and check that the policy was applied.

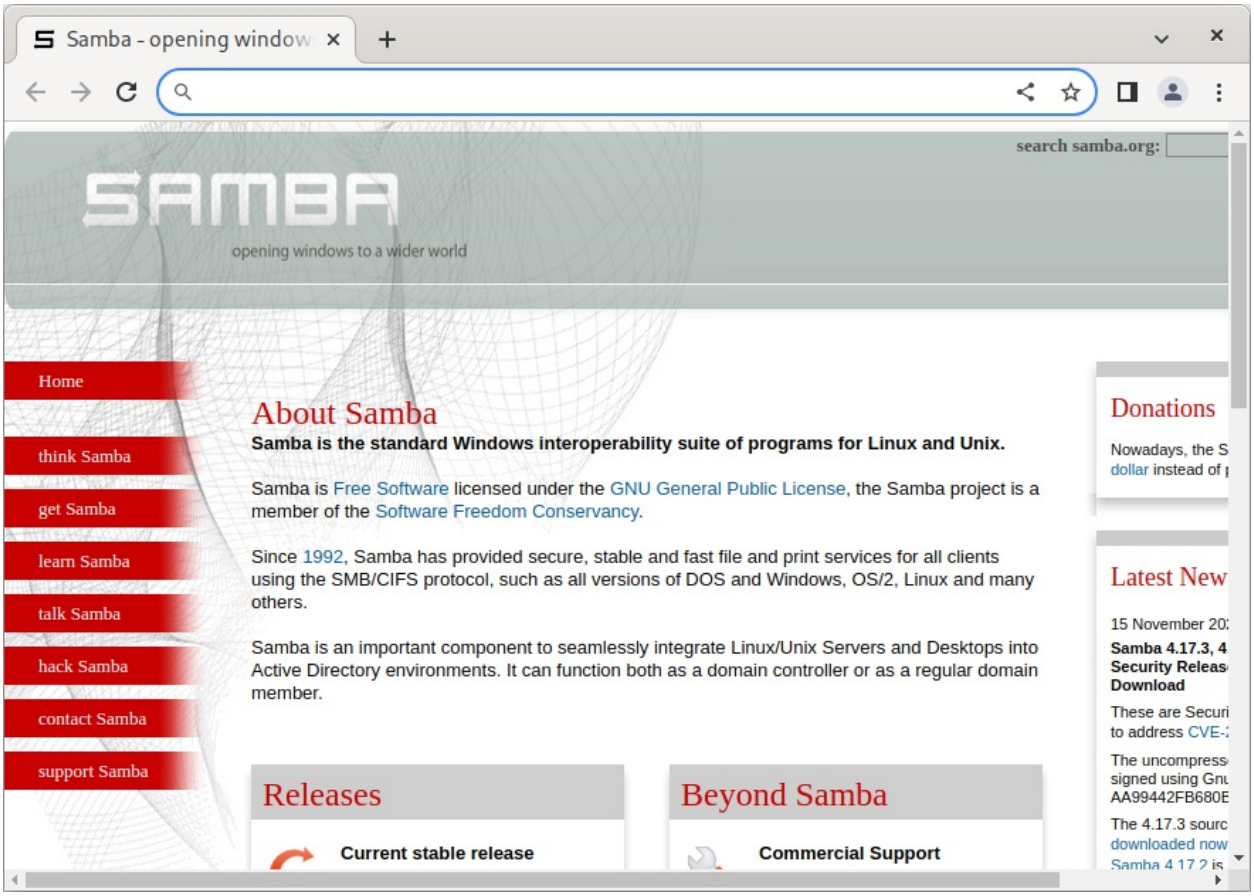


Figure 16.5: Chrome with default homepage

If you now open the browser settings, you'll see a warning indicating that your browser is being managed by your organization.

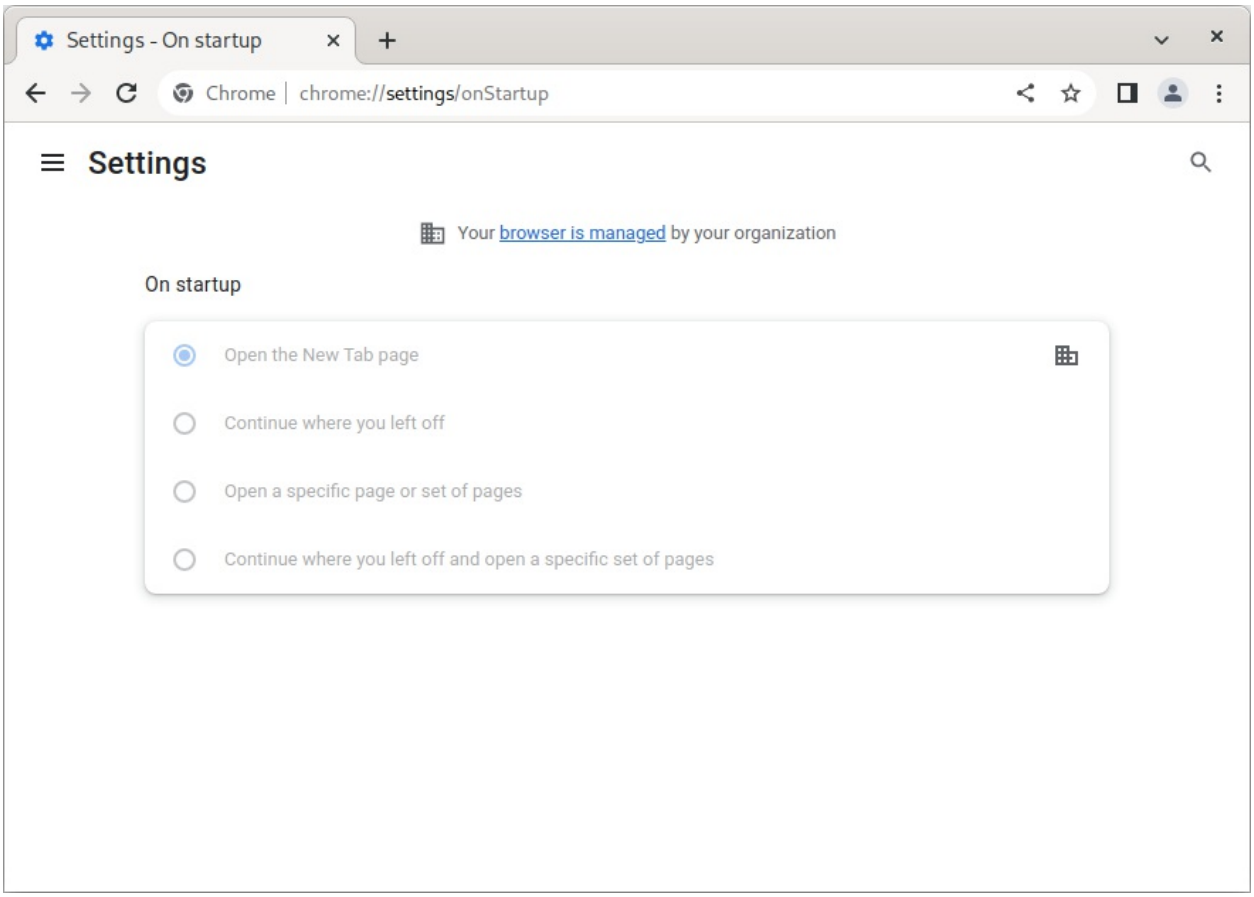


Figure 16.6: Chrome settings

Notice that the `On startup` options have been grayed out. Had we not set the `Action on startup`, the user would still be able to modify the startup options, and avoid the default homepage we just set.

17 GNOME Settings Policy

The GNOME Settings Policy deploys dconf settings that control the desktop experience. These settings correspond to those defined in the GNOME System Administration Guide found at https://help.gnome.org/admin/system-admin-guide/stable/user-settings.html.en.

This policy is physically stored on the SYSVOL in **MACHINE/Registry.pol**. It is stored in registry format. See chapter 21 for details on how to manually modify this file.

17.1 Server Side Extension

The Server Side Extension (SSE) for Chromium Policy is distributed via Administrative Templates (see chapter 20.1 in section 20.1.1). Setting up the ADMX templates for this policy is described in chapter 22 section 22.1.

17.1.1 Managing GNOME Settings Policy via the GPME

Open the GPME and navigate to `Computer Configuration > Administrative Templates > GNOME Settings`.

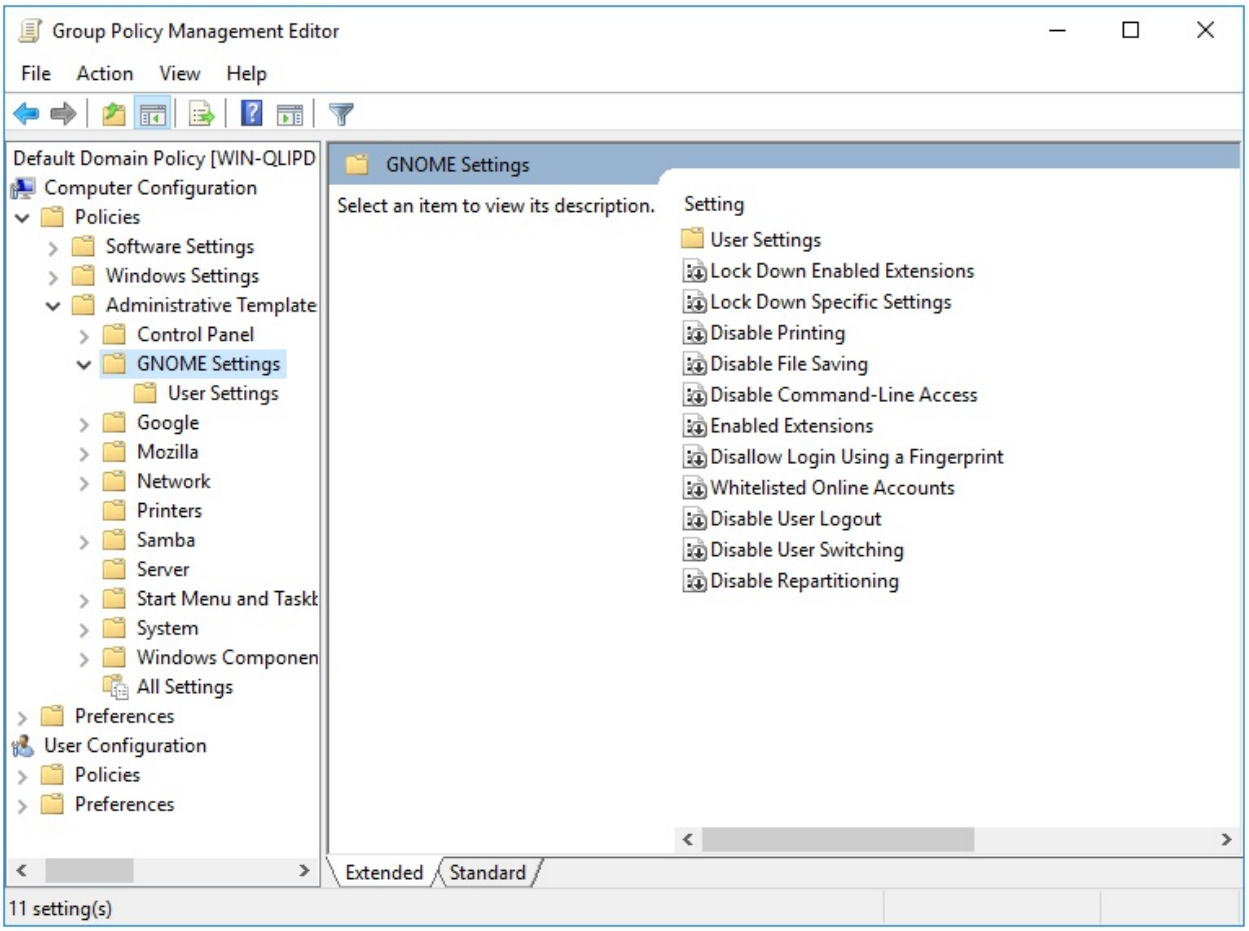


Figure 17.1: GNOME Settings Administrative Templates

Let’s set the allowed online accounts, which will effect the `Online Accounts` list in the GNOME Settings Dialog.

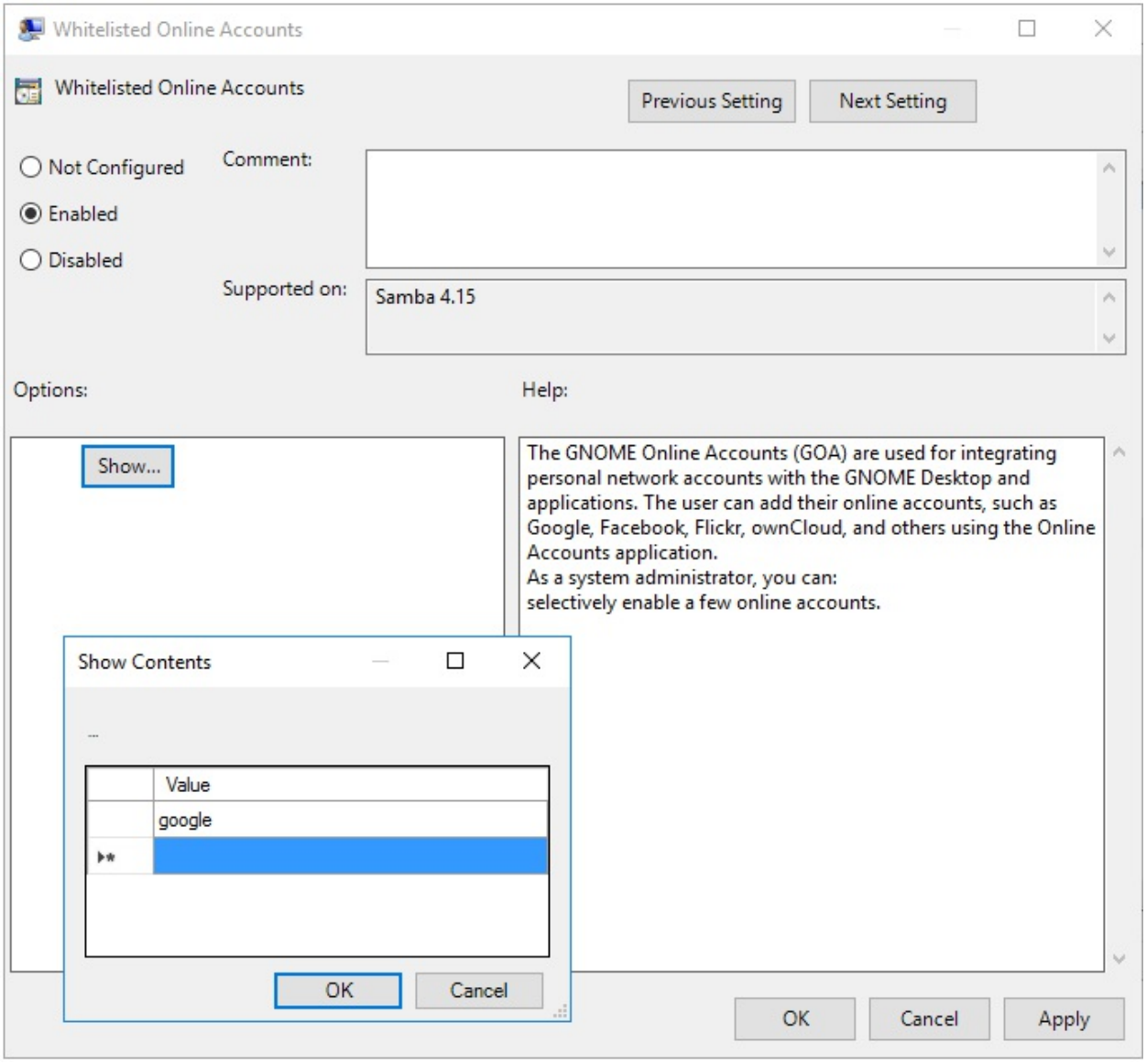


Figure 17.2: Whitelisted Online Accounts

17.2 Client Side Extension

The GNOME Settings Client Side Extension (CSE) creates various `dconf` files to deploy the policy. After creating these policy files, the CSE executes `dconf update` to apply the new policy. Users will need to log out and back in again in order for these policies to take effect.

Let’s list the Resultant Set of Policy and view the lockdown settings we’ve set.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_gnome_settings_ext
  ------------------------------------------------------------
    Policy Type: Whitelisted Online Accounts
    ----------------------------------------------------------
    [ google ]
    ----------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

Next let's force an apply of the policy, and see what is logged in the Group Policy Cache.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='GNOME Settings/
                                    Lock Down Settings']" - \
 | xmllint --format -
<gp_ext name="GNOME Settings/Lock Down Settings">
  <attribute name="Whitelisted Online Accounts">
    /etc/dconf/db/local.d/0000000001-goa;
    /etc/dconf/db/local.d/locks/0000000001-goa
  </attribute>
</gp_ext>
```

Checking the contents of the files, we see that our whitelisted provider is set, and that the policy is locked (which prevents user modification).

```
> cat /etc/dconf/db/local.d/0000000001-goa
[org/gnome/online-accounts]
whitelisted-providers = ['google']
> cat /etc/dconf/db/local.d/locks/0000000001-goa; echo
/org/gnome/online-accounts/whitelisted-providers
```

We can also check the dconf output to see that the setting is applied.

```
> dconf dump / \
 | grep -E "(whitelisted-providers|org/gnome/online-accounts)"
[org/gnome/online-accounts]
```

```
whitelisted-providers=['google']
```

Recall that it’s necessary to log the user out and back in again. After which, we can check the GNOME Settings dialog and see that `Online Accounts` are restricted as requested.

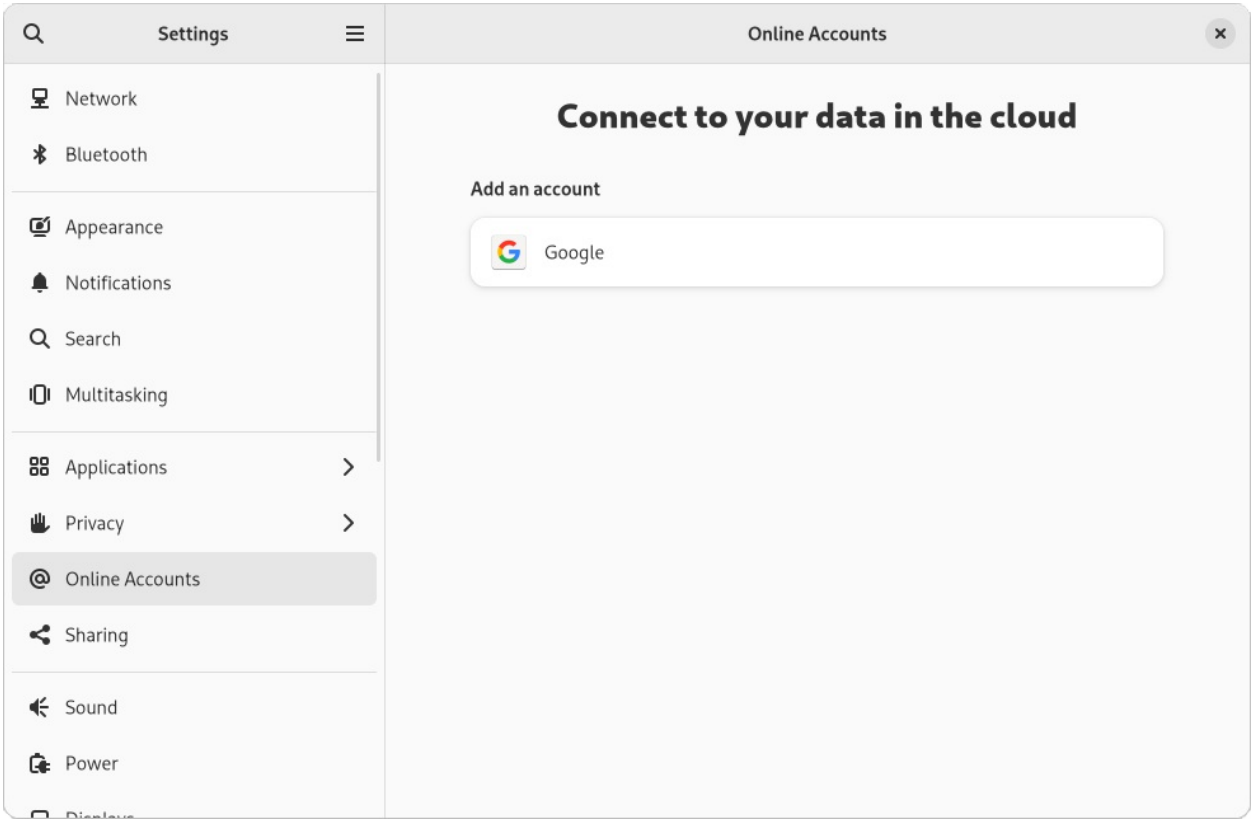
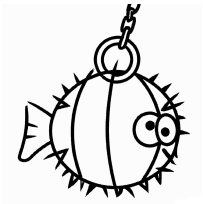


Figure 17.3: Online Accounts

18 OpenSSH Policy



The OpenSSH Policy allows you to deploy OpenSSH settings to client machines. The policies are applied to a file in the `/etc/ssh/sshd_config.d` directory.

This policy is physically stored on the SYSVOL in **MACHINE/VGP/VTLA /SshCfg/SshD/manifest.xml**. The manifest.xml file is in xml format, and is easily modified manually using a text editor.

18.1 Server Side Extension

The Server Side Extensions (SSE) for OpenSSH Policy is administered using the `samba-tool gpo manage openssh` command. This SSE cannot be modified using the GPME.

18.1.1 Managing OpenSSH Policy via samba-tool

The `samba-tool gpo manage openssh` command has 2 subcommands; set and list.

```
> samba-tool gpo manage openssh
Usage: samba-tool gpo manage openssh <subcommand>

Manage OpenSSH Group Policy Objects


Options:
  -h, --help  show this help message and exit
```

```
Available subcommands:
  list  - List VGP OpenSSH Group Policy from the sysvol
  set   - Sets a VGP OpenSSH Group Policy to the sysvol
```

To set a new OpenSSH rule, call the `samba-tool gpo manage openssh set` command, providing the following arguments:

1. `gpo`: the Group Policy Object (GPO) that you want to modify. This **MUST** be the GUID of the GPO.
2. `setting`: the OpenSSH setting that you want to modify. See the man page for sshd_config (`man sshd_config`) for a list of possible settings.
3. `value`: the value that you want to set for the specified setting. If you do not provide a value, the policy will be unset.

Here is an example of how you might use this command to set the `KerberosAuthentication` to `Yes`:

```
samba-tool gpo manage openssh set \
 {31B2F340-016D-11D2-945F-00C04FB984F9} \
 KerberosAuthentication Yes -UAdministrator
```

Then let's list the policy to see what has been set on the SYSVOL.

```
> samba-tool gpo manage openssh list \
 {31B2F340-016D-11D2-945F-00C04FB984F9} -UAdministrator
KerberosAuthentication Yes
```

18.2 Client Side Extension

The OpenSSH Client Side Extension (CSE) will create a new file in the `/etc/ssh/sshd_config.d` directory.

Let's list the Resultant Set of Policy to view the policies we've created.

```
> sudo /usr/sbin/samba-gpupdate --rsop
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: vgp_openssh_ext
```

```
  ------------------------------------------------------------
    Policy Type: VGP/Unix Settings/OpenSSH
    ------------------------------------------------------------
    [ KerberosAuthentication ] =         Yes
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

The `KerberosAuthentication` setting we set is listed as expected.

Let's now force an apply.

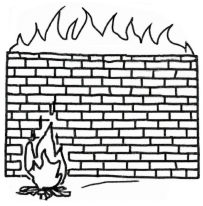
```
sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='VGP/Unix
                                    Settings/OpenSSH']" - \
 | xmllint --format -
<gp_ext name="VGP/Unix Settings/OpenSSH">
  <attribute name="ezMx...Zy5k">
    /etc/ssh/sshd_config.d/gp_c7hytho4
  </attribute>
</gp_ext>
```

Notice that our new policy has been stored in `/etc/ssh/sshd_config.d/gp_c7hytho4`. Let's check the contents of this file to see what was generated.

```
> sudo cat /etc/ssh/sshd_config.d/gp_c7hytho4

### autogenerated by samba
#
# This file is generated by the vgp_openssh_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#

KerberosAuthentication Yes
```

Our policy was successfully applied.

19 Firewalld Policy



The purpose of the Firewalld Policy is to apply firewalld rules to the client machine.

This policy is physically stored on the SYSVOL in the **MACHINE/Registry.pol** file within the subdirectory of the Group Policy Object. It is stored in registry format. See chapter 21 for details on how to manually modify this file.

19.1 Server Side Extension

The Server Side Extension (SSE) for the Firewalld Policy is distributed using Administrative Templates (ADMX). Refer to chapter 20.1 in section 20.1.1 for details about Administrative Templates.

Setting up the ADMX templates for this policy is described in chapter 22 section 22.1.

19.1.1 Managing Firewalld Policy via the GPME

Open the Group Policy Management Editor (GPME) and navigate to `Computer Configuration > Policies > Administrative Templates > Samba > Unix Settings > Firewalld`.

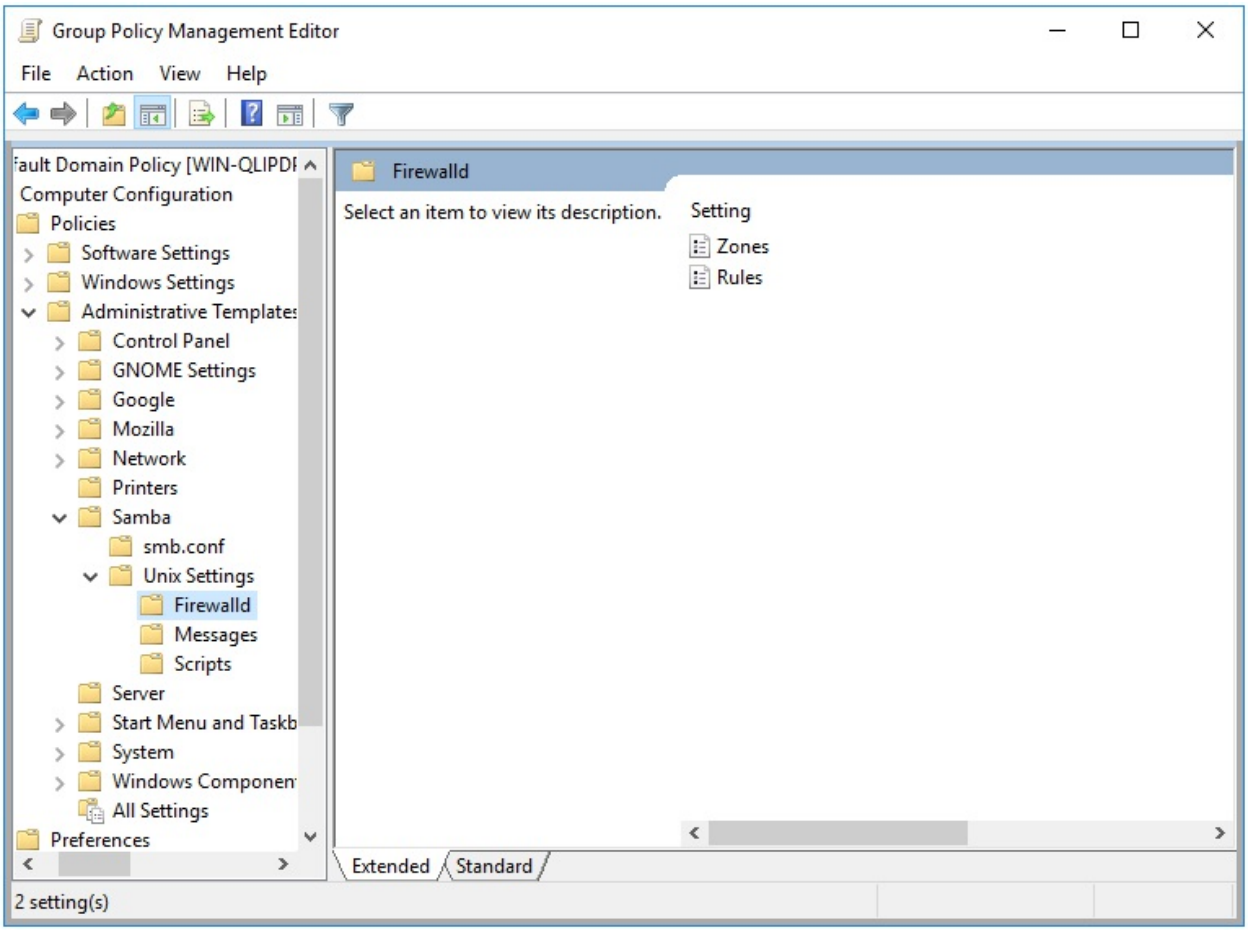


Figure 19.1: Group Policy Management Editor

You can see we have the options to create firewall rules, and firewall zones. For this example, we're going to create a single rule in a new zone named "work".

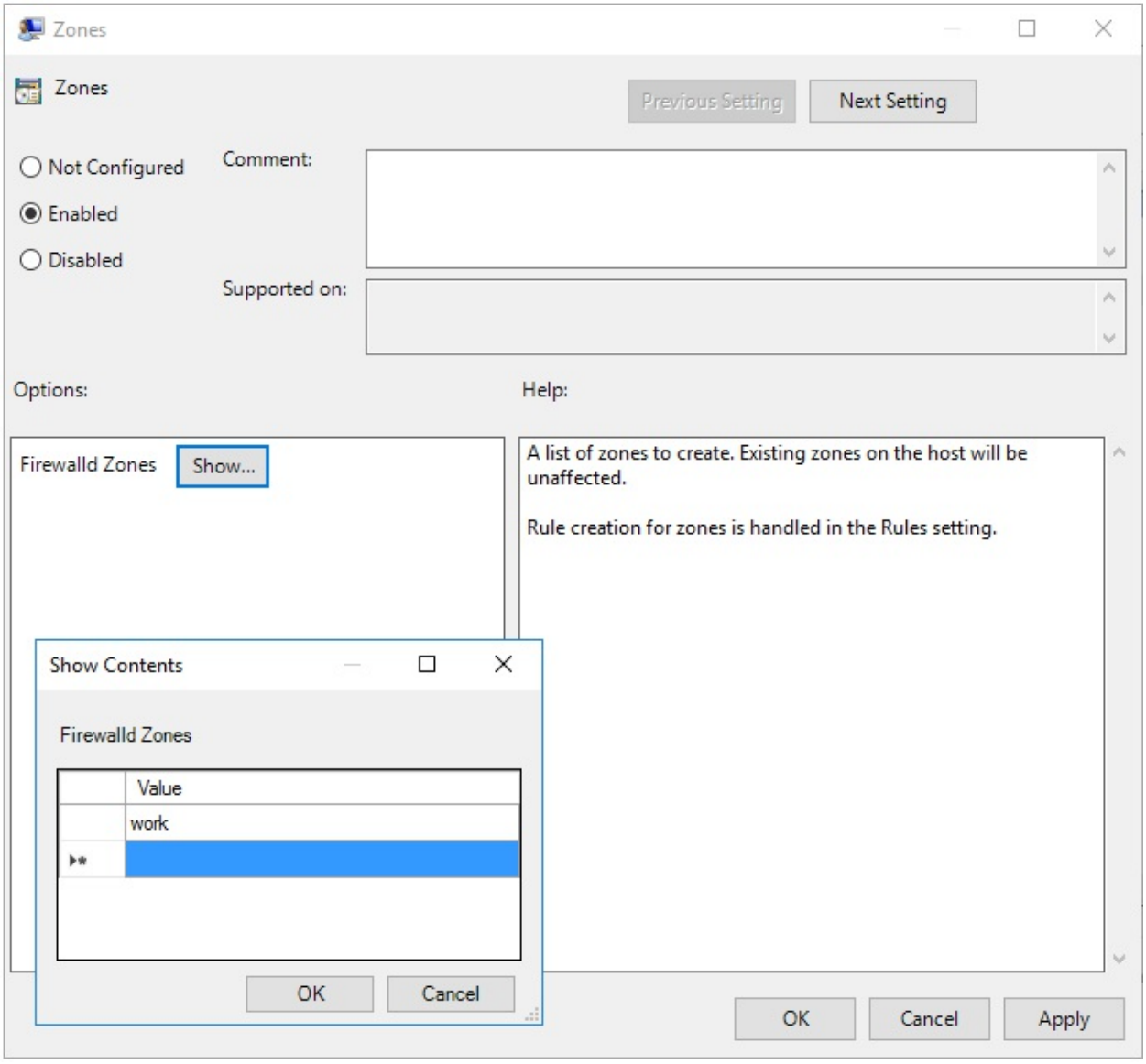


Figure 19.2: New Zone

We'll define our new rule as follows.

```
{ "work":
  [
    {
      "rule": { "family": "ipv4"},
      "source address": "172.25.1.7",
      "service name": "ftp",
      "reject": {}
    }
```

```
  ]
}
```

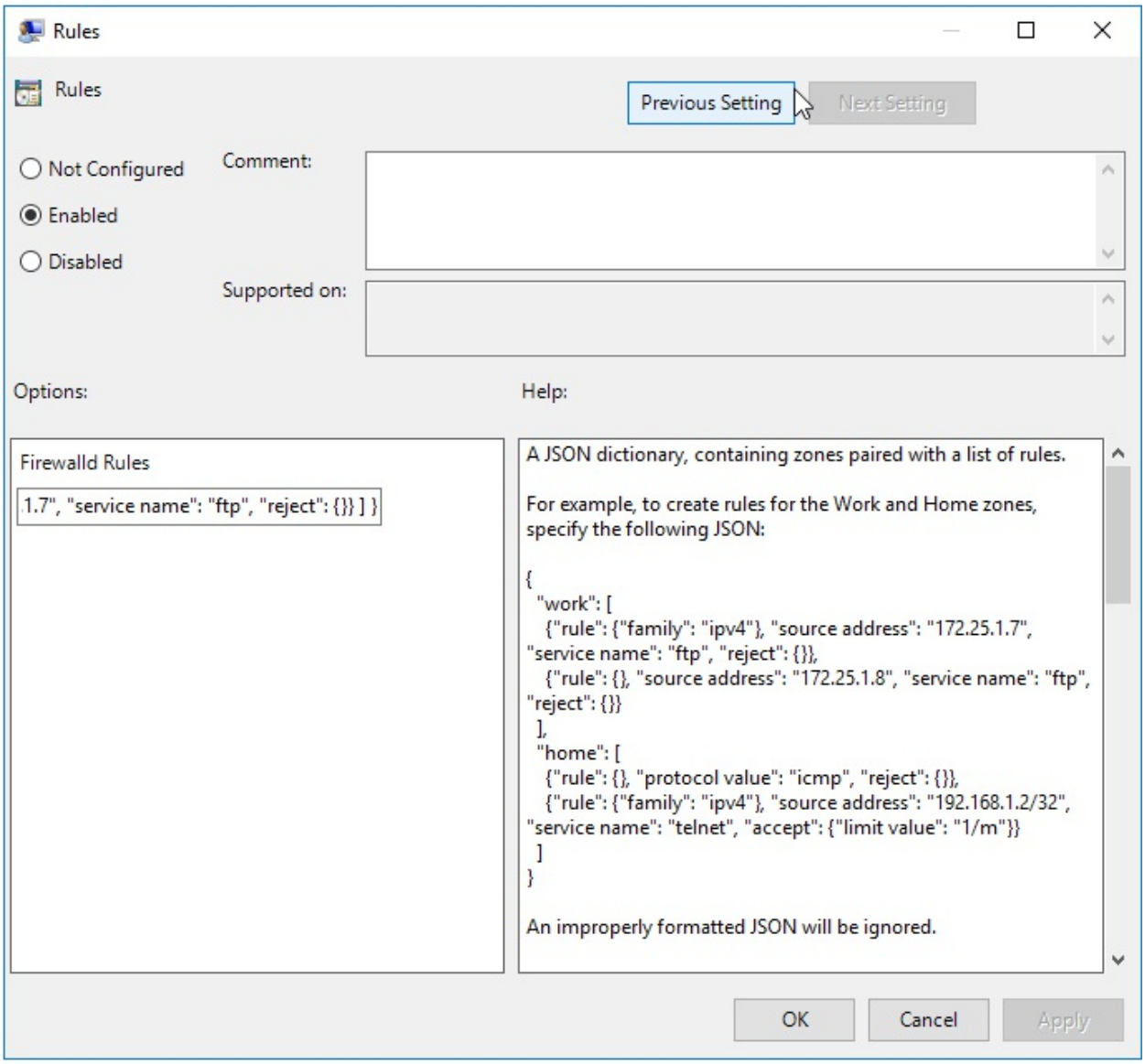


Figure 19.3: New Rule

19.1.1.1 Firewalld Policy Rule Definitions

The policy provides for the creation of rules and zones. Zones are defined as a list in the Zones setting in the Firewalld policy. Existing zones on the host will be unaffected.

Rules are defined using a JSON dictionary, containing zones paired with a list of rules.

For example, to create rules for the Work and Home zones, specify the following JSON:

```
{
  "work": [
    {
      "rule": {"family": "ipv4"},
      "source address": "172.25.1.7",
      "service name": "ftp",
      "reject": {}
    },
    {
      "rule": {},
      "source address": "172.25.1.8",
      "service name": "ftp",
      "reject": {}
    }
  ],
  "home": [
    {
      "rule": {},
      "protocol value": "icmp",
      "reject": {}
    },
    {
      "rule": {"family": "ipv4"},
      "source address": "192.168.1.2/32",
      "service name": "telnet",
      "accept": {"limit value": "1/m"}
    }
  ]
}
```

The rule structure loosely follows the Firewalld Rich Language Documentation. The general rule structure follows.

```
{
   "rule": {
     "family": "ipv4 | ipv6",
```

```
  "priority": "priority"
},
"source [not] address | mac | ipset":
  "address[/mask] | mac-address | ipset",
"destination [not] adress": "address[/mask]",
"service name": "service name",
"port": {
  "port": "port value",
  "protocol": "tcp | udp"
}
"protocol value": "protocol value",
"icmp-block name": "icmptype name",
"Masquerade": true|false,
"icmp-type": "icmptype name",
"forward-port": {
  "port": "port value",
  "protocol": "tcp | udp",
  "to-port": "port value",
  "to-addr": "address"
},
"source-port": {
  "port": "port value",
  "protocol": "tcp | udp"
},
"log": {
  "prefix": "prefix text",
  "level": "emerg | alert | crit | error
            | warning | notice | info | debug",
  "limit value": "rate/duration"
},
"audit": {
  "limit value": "rate/duration"
},
"accept" : {
  "limit value": "rate/duration"
} | "reject": {
  "type": "reject type",
  "limit value": "rate/duration"
} | "drop": {
  "limit value": "rate/duration"
} | "mark": {
  "set": "mark[/mask]",
  "limit value": "rate/duration"
```

```
    }
  }
```

19.2 Client Side Extension

The Firewalld Client Side Extension (CSE) accepts the rules specified by the SSE, and applies them using the firewalld command `firewall-cmd`. For example, new zones are added using the `firewall-cmd --permanent --new-zone=work` command. New firewall rules are added using the `firewall-cmd --permanent --zone=work --add-rich-rule 'rule family=ipv4 source address=172.25.1.7 service name=ftp reject'`.

Let’s check the Resultant Set of Policy on our Linux client.

```
> sudo /usr/sbin/samba-gpupdate --rsop
[sudo] password for root:
Resultant Set of Policy
Computer Policy

GPO: Default Domain Policy
============================================================
  CSE: gp_firewalld_ext
  ------------------------------------------------------------
    Policy Type: Rules
    ------------------------------------------------------------
    [
      [ work ] =
        [
          [ rule ] =
            [ family ] = ipv4
          [ source address ] = 172.25.1.7
          [ service name ] = ftp
          [ reject ] =
        ]
    ]
    ------------------------------------------------------------
    Policy Type: Zones
    ------------------------------------------------------------
    [ work ]
    ------------------------------------------------------------
  ------------------------------------------------------------
============================================================
```

Let’s now force an apply, and verify that rule gets applied.

```
> sudo /usr/sbin/samba-gpupdate --force
> sudo tdbdump /var/lib/samba/gpo.tdb -k "TESTSYSDM$" \
 | sed -r "s/\\\22/\"/g" | sed -r "s/\\\5C/\\\\/g" \
 | xmllint --xpath "//gp_ext[@name='Security/Firewalld']" - \
 | xmllint --format -
<gp_ext name="Security/Firewalld">
  <attribute name="rule:work:f480...e3a8">
    rule family=ipv4 source address=172.25.1.7
      service name=ftp reject
  </attribute>
</gp_ext>
```

Unfortunately, at the time of this writing, `firewall-cmd` appears to be unable to list installed rich rules. We can verify that our rule has applied though by trying to apply it again.

```
> sudo firewall-cmd --permanent --zone=work --add-rich-rule \
 'rule family=ipv4 source address=172.25.1.7
  service name=ftp reject'
Warning: ALREADY_ENABLED: rule family=ipv4
 source address=172.25.1.7 service name=ftp reject
success
```

Attempting to add the rule, we can see that firewalld warns us that the rule is already enabled.

20 Writing Group Policy Extensions

The chapter will explain how to write a Group Policy Extension for Samba's Winbind. Group Policy is a delivery mechanism for distributing system settings and company policies to machines joined to an Active Directory domain. Unix/Linux machines running Samba's Winbind can also deploy these policies.

20.1 Creating the Server Side Extension

20.1.1 Administrative Templates

The first step to deploying Group Policy is to create a Server Side Extension (SSE). There are multiple ways to create an SSE, but here we'll only discuss Administrative Templates (ADMX). The purpose of the SSE is to deploy policies to the SYSVOL share. Theoretically, you could manually deploy any file (even plain text) to the SYSVOL and then write a Client Side Extension that parses it, but ADMX can be read and modified by the Group Policy Management Editor, which makes administration of policies simpler.

ADMX files are simply XML files which explain to the Group Policy Management Console how to display and store a policy in the SYSVOL. AMDX files always store policies in Registry.pol files. Samba provides a mechanism for parsing these, which we'll discuss later.

Below is a simple example of an ADMX template, and it's corresponding ADML file.

samba.admx:

```
<policyDefinitions revision="1.0" schemaVersion="1.0">
  <policyNamespaces>
    <using prefix="windows"
           namespace="Microsoft.Policies.Windows" />
  </policyNamespaces>
  <supersededAdm fileName="" />
  <resources minRequiredRevision="1.0" />
  <categories>
    <category name="CAT_SAMBA"
              displayName="$(string.CAT_SAMBA)" />
    <category name="CAT_UNIX_SETTINGS"
              displayName="$(string.CAT_UNIX_SETTINGS)">
      <parentCategory ref="CAT_SAMBA" />
    </category>
  </categories>
  <policies>
    <policy name="POL_DAILY_SCRIPTS" class="Machine"
            displayName="$(string.POL_DAILY_SCRIPTS)"
        explainText="$(string.POL_DAILY_SCRIPTS_Help)"
        presentation="$(presentation.POL_DAILY_SCRIPTS)"
        key="Software\Policies\Samba\Unix Settings">
      <parentCategory ref="CAT_UNIX_SETTINGS" />
      <supportedOn ref="windows:SUPPORTED_WindowsVista" />
      <elements>
        <list id="LST_DAILY_SCRIPTS"
          key="Software\Policies\Samba\
               Unix Settings\Daily Scripts"
          valueName="Daily Scripts" />
      </elements>
    </policy>
  </policies>
</policyDefinitions>
```

en-US/samba.adml:

```
<policyDefinitionResources revision="1.0" schemaVersion="1.0">
  <displayName>
  </displayName>
  <description>
  </description>
  <resources>
    <stringTable>
      <string id="CAT_SAMBA">Samba</string>
```

```
      <string id="CAT_UNIX_SETTINGS">Unix Settings</string>
      <string id="POL_DAILY_SCRIPTS">Daily Scripts</string>
      <string id="POL_DAILY_SCRIPTS_Help">
        This policy setting allows you to execute commands,
        either local or on remote storage, daily.
      </string>
    </stringTable>
    <presentationTable>
      <presentation id="POL_DAILY_SCRIPTS">
        <listBox refId="LST_DAILY_SCRIPTS">
      Script and arguments
        </listBox>
      </presentation>
    </presentationTable>
  </resources>
</policyDefinitionResources>
```

The meaning of the various tags are explained in Microsoft's Group Policy documentation at https://docs.microsoft.com/en-us/previous-versions/windows/desktop/policy/admx-schema. Before the endless documentation and confusing XML scares you away, be aware there is an easier way!

20.1.1.1 ADMX Migrator

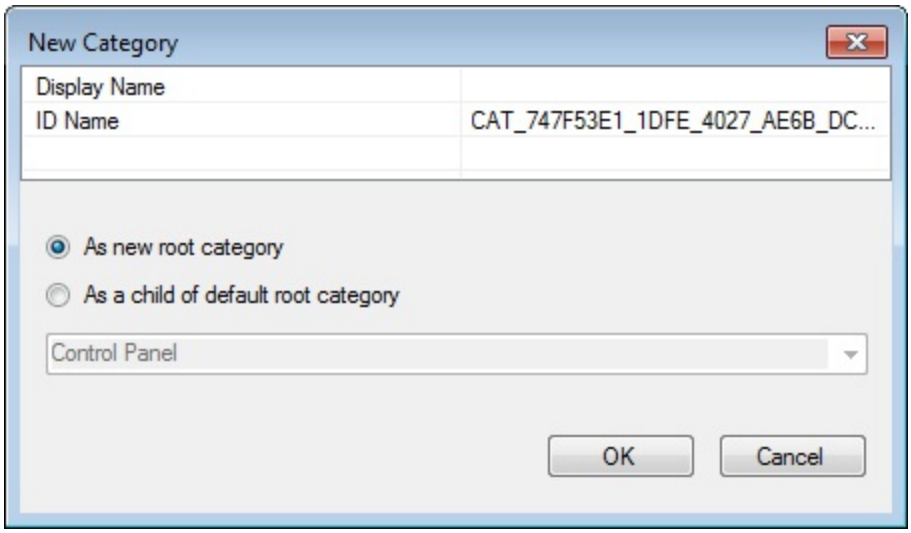
FullArmor created the ADMX Migrator to simplify the shift for system administrators from the old ADM policy templates to ADMX. Fortunately, this tool also serves our purpose for assisting us in easily creating these templates for our SSE. Unfortunately, the tool hasn't seen any development in the past 8 years, and wont run in Windows 10 (or any Unix/Linux platform, for that matter). I had to dredge up a Windows 7 VM in order to install and use the tool.

20.1.1.1.1 Creating the Administrative Template

1. Open ADMX Migrator
2. Right click on ADMX Templates in the left tree view, and click New Template.

3. Give your template a name, and click OK.

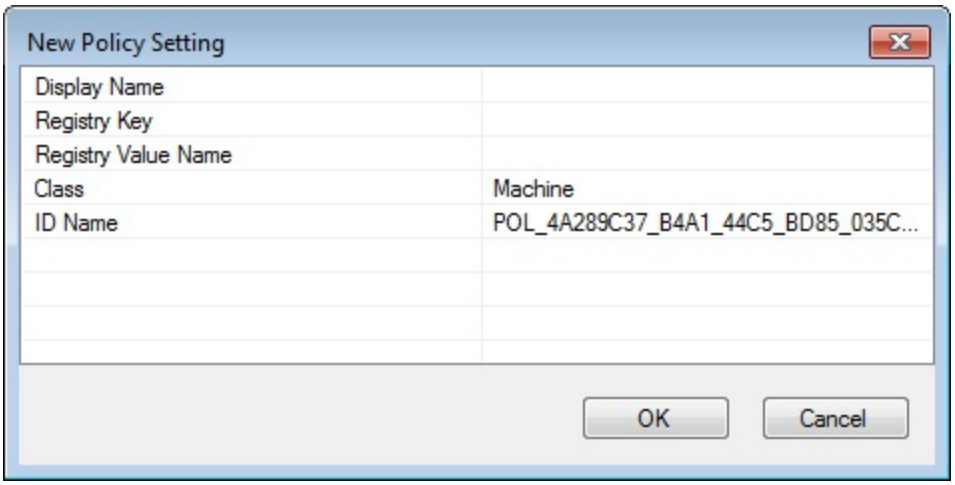
4. Right click on the new template in the left tree view, and click New Category.



5. Give the Category a name. This name will be displayed in the Group Policy Management Editor under Administrative Templates. You can choose to nest template under an existing category, or simply add it as a new root.

Note: You can also add sub-categories under this category. After clicking OK, right click the category you created and select New Category.

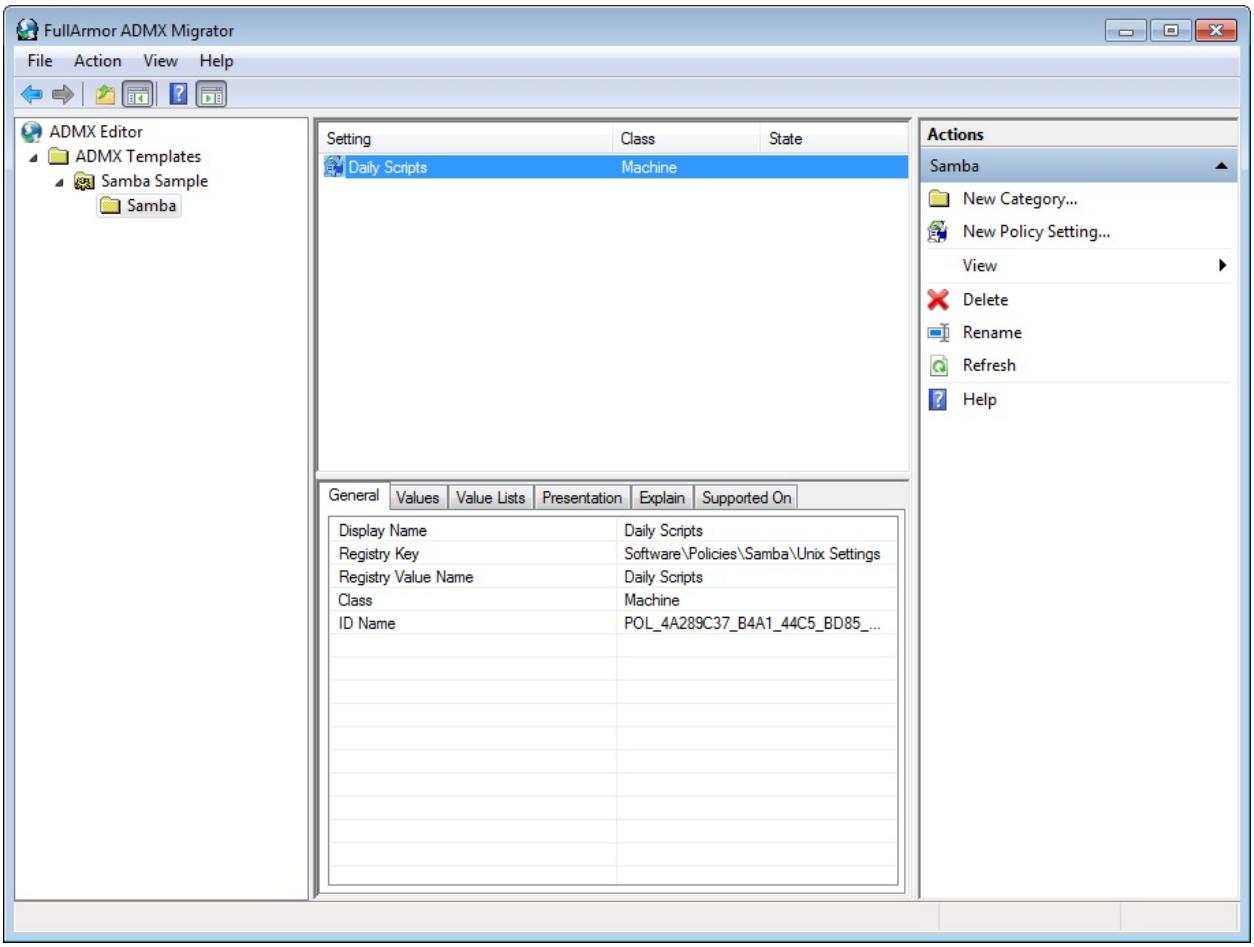
6. Next, create your policy by right clicking on your new category, and selecting New Policy Setting.



7. Because we'll be applying these settings to a Linux machine, the Registry fields are mostly meaningless, but they are required. Your policies will be stored under these keys on the SYSVOL in the Registry.pol file. Choose some sensible Registry Key, such as 'Software\Policies\Samba\Unix Settings', and a Registry Value Name, such as 'Daily Scripts' (these are the values used for Samba's cron.daily policy). The Display Name is the name that will be displayed for this policy in the Group Policy Management Editor. I usually make this the same as the Registry Value Name, but it doesn't need to be.

8. Select whether this policy will be applied to a Machine, a User, or to Both in the Class field. In our example, we could potentially set Both, then our Client Side Extension would need to handle both cron.daily scripts (the Machine) and also User crontab entries. Click OK for your policy to be created.

9. Your new policy will appear in the middle list view. Highlight it, and you will see a number of tabs below for configuring the policy.



10. Select the Values tab and set the Enabled Value Type. In this case, we'll use String, since our cron commands will be saved to the Registry.pol as a string. In the Value field, you can set a default enabled value (this is optional).

11. Select the Presentation tab, right click in the Elements view, and click New Element > ListBox (or a different presentation, depending on the policy). If you look at the samba.adml file from the previous section, you'll notice that the presentationTable contains a listBox item. That's what we're creating here.

12. Choose an element Label, this will be the name for the list displayed in the Group Policy Management Editor.

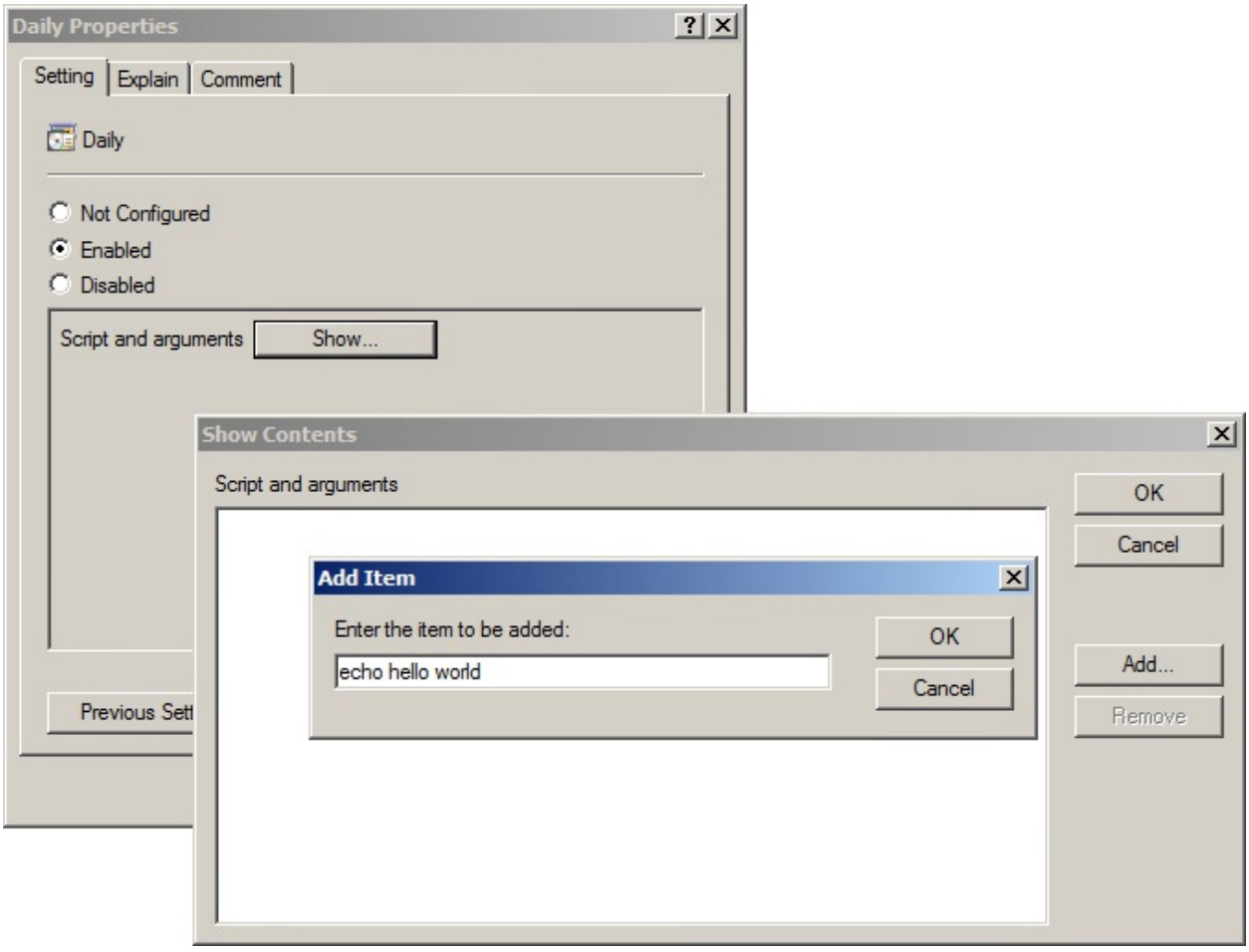
13. Choose a Registry Key. This will be pre-populated with the parent Registry Key you gave when creating the policy. Append something to

the key to make it unique. We'll use 'Software\Policies\Samba\Unix Settings\Daily Scripts' for our cron.daily policy.

14. Navigate to the Explain tab, and add an explanation of what this policy is and what it does. This will be displayed to users in the Group Policy Management Editor.

15. Now right click on your template name in the left tree, and select Save As.

16. Finally, you'll need to deploy your new policy definition to the SYSVOL. It should be saved to the Policies\PolicyDefinitions (the Group Policy Central Store) directory. These instructions from Microsoft can assist you in setting up your Group Policy Central Store.



20.1.2 samba-tool gpo manage

The `samba-tool gpo manage` command is a tool provided by the Samba team for managing Group Policy Objects (GPOs). This command provides a number of subcommands that allow you to add, remove, and list policies within a GPO.

Adding subcommands to `samba-tool gpo manage` is one way to create a Server Side Extension (SSE) for Group Policy. Each `samba-tool gpo manage` command generally provides 3 subcommands for each policy; add, remove, and list. These subcommands allow you to add new policies to a GPO, remove existing policies from a GPO, and list the policies that are currently configured in a GPO.

Group Policy SSEs should be added to `samba-tool` in the `python/samba/netcmd/gpo.py` file.

20.1.2.1 Subcommands

To add python subcommands using the `SuperCommand` class, you will need to create a new class that inherits from the `SuperCommand` class, and define a subcommands attribute that lists the subcommands that are supported by your command. The subcommands attribute will be a dictionary containing keys with new command names, paired with instances of `Command` classes which implement the command.

For example:

```
class cmd_scripts(SuperCommand):
    """Manage Scripts Group Policy Objects"""
    subcommands = {}
    subcommands["add"] = cmd_add_script()
    subcommands["list"] = cmd_list_script()
    subcommands["remove"] = cmd_remove_script()
```

Your `SuperCommand` will then need to be tied into an existing `samba-tool` command, for example:

```
class cmd_manage(SuperCommand):
    """Manage Group Policy Objects"""
    subcommands = {}
```

```
    subcommands["sudoers"] = cmd_sudoers()
    subcommands["security"] = cmd_security()
    subcommands["smb_conf"] = cmd_smb_conf()
    subcommands["symlink"] = cmd_symlink()
    subcommands["files"] = cmd_files()
    subcommands["openssh"] = cmd_openssh()
    subcommands["motd"] = cmd_motd()
    subcommands["issue"] = cmd_issue()
    subcommands["access"] = cmd_access()
    subcommands["scripts"] = cmd_scripts()
```

Notice that the `cmd_scripts` `SuperCommand` from earlier has been appended to the `cmd_manage` list of `subcommands`.

20.1.2.1.1 Implementing an Add Subcommand

To write a command that adds a new policy to a Group Policy Object (GPO), you will need to create a class that inherits from the Command class provided by the samba.netcmd module, and define a run method that takes a series of arguments and options, and contains the code that will be executed when the command is run.

The run method should begin by connecting to a Domain Controller (DC) using the `smb_connection` function defined in `python/samba/netcmd/gpo.py`. It can then retrieve the data from the GPO's Registry.pol file using the loadfile method of the connection object returned by the `smb_connection` function, or create a new file object if the file does not exist.

Next, the method can parse the data in the `Registry.pol` file using the `ndr_unpack` function from the `samba.ndr` module, which will return a file object representing the data in the file. We will then add a key to the list of entries in `Registry.pol` file object.

Finally, we save the modified file object back to the GPO's `Registry.pol` file on the DC using the savefile method of the connection object.

The class should also define a synopsis attribute, which provides a brief summary of the command.

Here is an example of what it might look:

```
class cmd_add_script(Command):
    """Adds Script Group Policy to the sysvol

This command adds a script policy to the sysvol.

Example:
samba-tool gpo manage scripts add \
 {31B2F340-016D-11D2-945F-00C04FB984F9} MACHINE daily \
 test_script.sh '\\-n \\-p all'

policy_class is defined as either MACHINE or USER.
freq is defined as either Daily, Monthly, Weekly, or Hourly.
    """

    synopsis = "%prog <gpo> <policy_class> <freq> <script> "
               "[args] [options]"

    takes_optiongroups = {
        "sambaopts": options.SambaOptions,
        "versionopts": options.VersionOptions,
        "credopts": options.CredentialsOptions,
    }

    takes_options = [
        Option("-H", "--URL",
               help="LDB URL for database or target server",
           type=str, metavar="URL", dest="H"),
    ]

    takes_args = ["gpo", "policy_class", "freq", "script",
                  "args?"]

    def run(self, gpo, policy_class, freq, script, args=None,
            H=None, sambaopts=None, credopts=None,
        versionopts=None):
        policy_class = policy_class.upper()
        if policy_class not in ['MACHINE', 'USER']:
            raise CommandError("'%s' is not a valid policy_class.
                   " Choose from MACHINE or USER" % policy_class)
```

```
freq = freq.title()
if freq not in ['Daily', 'Monthly', 'Weekly', 'Hourly']:
    raise CommandError("'%s' is not a valid frequency. "
        "Choose from Daily, Monthly, Weekly, or "
"Hourly" % freq)

self.lp = sambaopts.get_loadparm()
self.creds = credopts.get_credentials(self.lp,
                                      fallback_machine=True)

if not os.path.exists(script):
    raise CommandError(
        "Script '%s' does not exist" % script)

# We need to know writable DC to setup SMB connection
if H and H.startswith('ldap://'):
    dc_hostname = H[7:]
    self.url = H
else:
    dc_hostname = netcmd_finddc(self.lp, self.creds)
    self.url = dc_url(self.lp, self.creds, dc=dc_hostname)

# SMB connect to DC
conn = smb_connection(dc_hostname,
                      'sysvol',
                      lp=self.lp,
                      creds=self.creds)

realm = self.lp.get('realm')
pol_file = '\\'.join([realm.lower(), 'Policies', gpo,
                      '%s\\Registry.pol' % policy_class])
try:
    pol_data = ndr_unpack(preg.file,
                          conn.loadfile(pol_file))
except NTSTATUSError as e:
    if e.args[0] in [NT_STATUS_OBJECT_NAME_INVALID,
                         NT_STATUS_OBJECT_NAME_NOT_FOUND,
                         NT_STATUS_OBJECT_PATH_NOT_FOUND]:
        # The file doesn't exist, so create it
        pol_data = preg.file()
    elif e.args[0] == NT_STATUS_ACCESS_DENIED:
        raise CommandError("The authenticated user does "
```

```
                                "not have sufficient privileges")
            else:
                raise

        reg_key = b'Software\\Policies\\Samba\\Unix Settings'
        keyname = b'%s\\%s Scripts' % (reg_key, get_bytes(freq))
        entry = '%s %s' % (script,
                           args if args is not None else '')
        e = preg.entry()
        e.keyname = keyname
        e.valuename = reg_key
        e.type = 1
        e.data = get_bytes(entry)
        entries = list(pol_data.entries)
        entries.append(e)
        pol_data.num_entries = len(entries)
        pol_data.entries = entries

        try:
            conn.savefile(pol_file, ndr_pack(pol_data))
        except NTSTATUSError as e:
            if e.args[0] == NT_STATUS_ACCESS_DENIED:
                raise CommandError("The authenticated user does "
                                   "not have sufficient privileges")
            raise
```

20.1.2.1.2 Implementing a List Subcommand

To write a command that lists the policies for a Group Policy Object (GPO), you would need to create a class that inherits from the `Command` class provided by the `samba.netcmd` module, and define a `run` method that takes a series of arguments and options, and contains the code that will be executed when the command is run.

The run method should begin by connecting to a Domain Controller (DC) using the `smb_connection` function defined in `python/samba/netcmd/gpo.py`. It can then retrieve the data from the GPO's Registry.pol file using the loadfile method of the connection object returned by the `smb_connection` function.

Next, the method can parse the data in the `Registry.pol` file using the `ndr_unpack` function from the `samba.ndr` module, which will return a file object representing the data in the file. The method can then iterate over the keys and values in the file object, printing out the names and data for each key.

The class should also define a synopsis attribute, which provides a brief summary of the command.

Here is an example of what it might look:

```
class cmd_list_script(Command):
    """List Script Group Policy from the sysvol

This command lists the script policies currently set on the sysvo

Example:
samba-tool gpo manage scripts list \
 {31B2F340-016D-11D2-945F-00C04FB984F9}
    """

    synopsis = "%prog <gpo> [options]"

    takes_optiongroups = {
        "sambaopts": options.SambaOptions,
        "versionopts": options.VersionOptions,
        "credopts": options.CredentialsOptions,
    }

    takes_options = [
        Option("-H", "--URL",
               help="LDB URL for database or target server",
           type=str, metavar="URL", dest="H"),
    ]

    takes_args = ["gpo"]

    def run(self, gpo, H=None, sambaopts=None, credopts=None,
            versionopts=None):
        self.lp = sambaopts.get_loadparm()
```

```
self.creds = credopts.get_credentials(self.lp,
                                      fallback_machine=True)

# We need to know writable DC to setup SMB connection
if H and H.startswith('ldap://'):
    dc_hostname = H[7:]
    self.url = H
else:
    dc_hostname = netcmd_finddc(self.lp, self.creds)
    self.url = dc_url(self.lp, self.creds, dc=dc_hostname)

# SMB connect to DC
conn = smb_connection(dc_hostname,
                      'sysvol',
                      lp=self.lp,
                      creds=self.creds)

realm = self.lp.get('realm')
pol_file = '\\'.join([realm.lower(), 'Policies', gpo,
                      '%s\\Registry.pol'])
for policy_class in ['MACHINE', 'USER']:
    self.outf.write('%s:\n' % policy_class)
    try:
        pol_data = ndr_unpack(preg.file,
                    conn.loadfile(pol_file % policy_class))
    except NTSTATUSError as e:
        if e.args[0] in [NT_STATUS_OBJECT_NAME_INVALID,
                         NT_STATUS_OBJECT_NAME_NOT_FOUND,
                         NT_STATUS_OBJECT_PATH_NOT_FOUND]:
            # The file doesn't exist,
    # so there is nothing to list
            continue
        elif e.args[0] == NT_STATUS_ACCESS_DENIED:
            raise CommandError("The authenticated user "
                "does not have sufficient privileges")
        else:
            raise

    reg_key = 'Software\\Policies\\Samba\\Unix Settings'
    for e in pol_data.entries:
        if e.valuename == "**delvals.":
            continue
```

```
            if e.keyname.startswith(reg_key) and \
                    e.keyname.endswith('Scripts'):
                self.outf.write("\t%s:\n" % e.keyname)
                self.outf.write("\t\t%s\n" % e.data)
```

20.1.2.1.3 Implementing a Remove Subcommand

To write a command that removes a policy from a Group Policy Object (GPO), you would need to create a class that inherits from the `Command` class provided by the `samba.netcmd` module, and define a `run` method that takes a series of arguments and options, and contains the code that will be executed when the command is run.

The run method should begin by connecting to a Domain Controller (DC) using the `smb_connection` function defined in `python/samba/netcmd/gpo.py`. It can then retrieve the data from the GPO's Registry.pol file using the loadfile method of the connection object returned by the `smb_connection` function.

Next, the method can parse the data in the Registry.pol file using the `ndr_unpack` function, which will return a file object representing the data in the file. It then checks whether the entry specified in the "script" variable exists in the list of entries contained in the `pol_data` object. If the entry does exist, it is removed from the list and the number of entries in the list is updated.

Finally, we save the modified file object back to the GPO's `Registry.pol` file on the DC using the savefile method of the connection object.

The class should also define a synopsis attribute, which provides a brief summary of the command.

Here is an example of what it might look:

```
class cmd_remove_script(Command):
    """Removes Script Group Policy from the sysvol

This command removes a script policy from the sysvol.
```

```
Example:
samba-tool gpo manage scripts remove \
 {31B2F340-016D-11D2-945F-00C04FB984F9} MACHINE daily \
 'test_script.sh \\-n \\-p all'

policy_class is defined as either MACHINE or USER.
freq is defined as either Daily, Monthly, Weekly, or Hourly.
    """

    synopsis = "%prog <gpo> <policy_class> <freq> <script>"
               "[options]"

    takes_optiongroups = {
        "sambaopts": options.SambaOptions,
        "versionopts": options.VersionOptions,
        "credopts": options.CredentialsOptions,
    }

    takes_options = [
        Option("-H", "--URL",
               help="LDB URL for database or target server",
           type=str, metavar="URL", dest="H"),
    ]

    takes_args = ["gpo", "policy_class", "freq", "script"]

    def run(self, gpo, policy_class, freq, script, H=None,
            sambaopts=None, credopts=None, versionopts=None):
        policy_class = policy_class.upper()
        if policy_class not in ['MACHINE', 'USER']:
            raise CommandError("'%s' is not a valid policy_class."
                   " Choose from MACHINE or USER" % policy_class)
        freq = freq.title()
        if freq not in ['Daily', 'Monthly', 'Weekly', 'Hourly']:
            raise CommandError("'%s' is not a valid frequency. "
                               "Choose from Daily, Monthly, "
                   "Weekly, or Hourly" % freq)

        self.lp = sambaopts.get_loadparm()
        self.creds = credopts.get_credentials(self.lp,
```

```
                                        fallback_machine=True)

        # We need to know writable DC to setup SMB connection
        if H and H.startswith('ldap://'):
            dc_hostname = H[7:]
            self.url = H
        else:
            dc_hostname = netcmd_finddc(self.lp, self.creds)
            self.url = dc_url(self.lp, self.creds, dc=dc_hostname)

        # SMB connect to DC
        conn = smb_connection(dc_hostname,
                              'sysvol',
                              lp=self.lp,
                              creds=self.creds)

        realm = self.lp.get('realm')
        pol_file = '\\'.join([realm.lower(), 'Policies', gpo,
                              '%s\\Registry.pol' % policy_class])
        try:
            pol_data = ndr_unpack(preg.file,
                                  conn.loadfile(pol_file))
        except NTSTATUSError as e:
            if e.args[0] in [NT_STATUS_OBJECT_NAME_INVALID,
                                 NT_STATUS_OBJECT_NAME_NOT_FOUND,
                                 NT_STATUS_OBJECT_PATH_NOT_FOUND]:
                raise CommandError("Cannot remove script '%s' "
                    "because it does not exist" % script)
            elif e.args[0] == NT_STATUS_ACCESS_DENIED:
                raise CommandError("The authenticated user does "
                                   "not have sufficient privileges")
            else:
                raise

        script = script.strip()
        if script in ([e.data.strip() for e in pol_data.entries] \
                      if pol_data else []):
            entries = [e for e in pol_data.entries \
                       if e.data.strip() != script]
            pol_data.num_entries = len(entries)
            pol_data.entries = entries
```

```
            try:
                conn.savefile(pol_file, ndr_pack(pol_data))
            except NTSTATUSError as e:
                if e.args[0] == NT_STATUS_ACCESS_DENIED:
                    raise CommandError("The authenticated user "
                                       "does not have sufficient"
                                       " privileges")
                raise
        else:
            raise CommandError("Cannot remove '%s' because it"
                               " does not exist" % script)
```

20.2 Creating the Client Side Extension

The following script defines a Group Policy Client Side Extension (CSE) in Python, which will be called by Samba's Winbind to deploy our newly created policy. A CSE is a program that runs on a client machine and processes Group Policy Objects (GPOs) that are applied to the machine. The CSE processes the GPOs by applying the policies they contain to the client machine.

```
#!/usr/bin/python3
import os, re
from samba.gpclass import gp_pol_ext, gp_file_applier,
    register_gp_extension, unregister_gp_extension,
    list_gp_extensions
from tempfile import NamedTemporaryFile
from samba.gp.util.logging import log
from samba import getopt as options
import optparse

intro = '''
### autogenerated by samba
#
# This file is generated by the gp_scripts_ext Group Policy
# Client Side Extension. To modify the contents of this file,
# modify the appropriate Group Policy objects which apply
# to this machine. DO NOT MODIFY THIS FILE DIRECTLY.
#
```

```
'''

class gp_scripts_ext(gp_pol_ext, gp_file_applier):
    def __str__(self):
        return 'Unix Settings/Scripts'

    def process_group_policy(self, deleted_gpo_list,
                             changed_gpo_list):

        # Iterate over GPO guids and their previous settings,
        # reverting changes made by this GPO.
        for guid, settings in deleted_gpo_list:

            # Use the unapply() function from the base class
            # gp_file_applier to remove the files.
            if str(self) in settings:
                for attribute, script in \
                        settings[str(self)].items():
                    # Delete the applied policy
                    self.unapply(guid, attribute, script)

        # Iterate over GPO objects, applying new policies found
        # in the SYSVOL
        for gpo in changed_gpo_list:
            if gpo.file_sys_path:
                reg_key = 'Software\\Policies\\' + \
                          'Samba\\Unix Settings'
                sections = { '%s\\Daily Scripts' % reg_key :
                               '/etc/cron.daily',
                             '%s\\Monthly Scripts' % reg_key :
                               '/etc/cron.monthly',
                             '%s\\Weekly Scripts' % reg_key :
                               '/etc/cron.weekly',
                             '%s\\Hourly Scripts' % reg_key :
                               '/etc/cron.hourly'
                           }

                # Load the contents of the Registry.pol
                # from the SYSVOL
                pol_file = 'MACHINE/Registry.pol'
                path = os.path.join(gpo.file_sys_path, pol_file)
                pol_conf = self.parse(path)
```

```
if not pol_conf:
    continue

# Gather the list of policies to apply
policies = {}
for e in pol_conf.entries:
    if e.keyname in sections.keys() and \
            e.data.strip():
        if e.keyname not in policies:
            policies[e.keyname] = []
        policies[e.keyname].append(e.data)

# Specify the applier function, which will be
# used to apply the policy.
def applier_func(keyname, entries):
    ret = []
    cron_dir = sections[e.keyname]
    for data in entries:
        with NamedTemporaryFile(prefix='gp_',
                                mode="w+",
                                delete=False,
                                dir=cron_dir) as f:
            contents = '#!/bin/sh\n%s' % intro
            contents += '%s\n' % data
            f.write(contents)
            os.chmod(f.name, 0o700)
            ret.append(f.name)
    return ret

# For each policy in the Registry.pol,
# apply the settings
for keyname, entries in policies.items():
    # Each GPO applies only one set of each type
    # of script, so so the attribute matches the
    # keyname.
    attribute = keyname
    # The value hash is generated from the script
    # entries, ensuring any changes to this GPO
    # will cause the scripts to be rewritten.
    value_hash = self.generate_value_hash(*entries)
    self.apply(gpo.name, attribute, value_hash,
               applier_func, keyname, entries)
```

```
                # Cleanup any old scripts that are no longer
                # part of the policy
                self.clean(gpo.name, keep=policies.keys())

    def rsop(self, gpo):
        output = {}
        pol_file = 'MACHINE/Registry.pol'
        if gpo.file_sys_path:
            path = os.path.join(gpo.file_sys_path, pol_file)
            pol_conf = self.parse(path)
            if not pol_conf:
                return output
            for e in pol_conf.entries:
                key = e.keyname.split('\\')[-1]
                if key.endswith('Scripts') and e.data.strip():
                    if key not in output.keys():
                        output[key] = []
                    output[key].append(e.data)
        return output

if __name__ == "__main__":
    parser = optparse.OptionParser('gp_scripts_ext.py [options]')
    sambaopts = options.SambaOptions(parser)
    parser.add_option_group(sambaopts)

    parser.add_option('--register',
                      help='Register extension to Samba',
                      action='store_true')
    parser.add_option('--unregister',
                      help='Unregister extension from Samba',
                      action='store_true')

    (opts, args) = parser.parse_args()

    # We're collecting the Samba loadparm simply to
    # find our smb.conf file
    lp = sambaopts.get_loadparm()

    # This is a random unique GUID, which identifies this CSE.
    # Any random GUID will do.
```

```
    ext_guid = '{5930022C-94FF-4ED5-A403-CFB4549DB6F0}'
    if opts.register:
        # The extension path is the location of this file. This
        # script should be executed from a permanent location.
        ext_path = os.path.realpath(__file__)
        # The machine and user parameters tell Samba whether to
        # apply this extension to the computer, to individual
        # users, or to both.
        register_gp_extension(ext_guid, 'gp_scripts_ext',
                              ext_path, smb_conf=lp.configfile,
                              machine=True, user=False)
    elif opts.unregister:
        # Remove the extension and do not apply policy.
        unregister_gp_extension(ext_guid)

    # List the currently installed Group Policy Client Side
    # Extensions
    exts = list_gp_extensions(lp.configfile)
    for guid, data in exts.items():
        print(guid)
        for k, v in data.items():
            print('\t%s: %s' % (k, v))
```

The CSE is defined by the `gp_scripts_ext` class, which is derived from the `gp_pol_ext` and `gp_file_applier` classes. The `gp_pol_ext` class provides a framework for processing GPOs, and the `gp_file_applier` class provides functions for applying GPO policies to files on the client machine.

20.2.1 The gp_ext and gp_applier Python Classes

Your CSE must be a class that inherits from subclasses of `gp_ext` and `gp_applier`. The `gp_pol_ext` is a subclass of `gp_ext` that provides simplified parsing of Registry.pol files. If you choose to store your policies in ini/inf files in the SYSVOL (instead of using Administrative Templates), then you can inherit from the `gp_inf_ext` instead. The `gp_file_applier` is a subclass of `gp_applier` which provides convenience functions for applying and unapplying policies which add files to the machine.

If your class inherits from either `gp_pol_ext` or `gp_inf_ext`, it has a `parse()`

function defined, which takes a single filename. The parse() function will parse the contents of the policy file and return it in a sensible format.

If for some reason you choose to store data on the SYSVOL in some other format (such as in XML, etc), you'll need to subclass `gp_ext`, then implement a `read()` function, like this:

```
import xml.etree.ElementTree
class gp_xml_ext(gp_ext):
    def read(self, data_file):
        return xml.etree.ElementTree.parse(data_file)
```

The `read()` function is called by `parse()`, passing it a local filename tied to the systems SYSVOL cache. Then within `process_group_policy()` you can call `parse()` to fetch the parsed data from the SYSVOL.

The `gp_file_applier` class implements the helper functions `apply` and `unapply`. If you instead inherit from `gp_applier` directly, you'll need to implement `apply` and `unapply` yourself. The `gp_applier` class provides various helper functions for assisting you in creating the `apply` and `unapply` functions. The `apply` and `unapply` functions are responsible for both adding and removing policy, as well as managing the Group Policy Cache contents.

20.2.2 Process Group Policy

The CSE has two main functions: `process_group_policy` and `rsop`. The `process_group_policy` function is called by the Group Policy engine on the client machine to process the GPOs that apply to the machine. It takes two arguments: `deleted_gpo_list` and `changed_gpo_list`. The `deleted_gpo_list` argument is a list of GPOs that MUST be removed from the machine, and the `changed_gpo_list` argument is a list of GPOs that have been changed (or are new) and MUST be re-applied to the machine.

The `process_group_policy` function serves two primary purposes; it applies new policy, and it removes old policy. It first iterates over the `deleted_gpo_list`, using the unapply function from the `gp_file_applier` class to remove the files that were applied by the GPOs.

```
for guid, settings in deleted_gpo_list:
    if str(self) in settings:
        for attribute, script in settings[str(self)].items():
            self.unapply(guid, attribute, script)
```

The `deleted_gpo_list` is a dictionary which contains the guids of Group Policy Objects, with associated settings which were previously applied. This list of applied settings is generated by the second loop (`changed_gpo_list`) while it is applying policy.

The `process_group_policy` function then iterates over the `changed_gpo_list`, applying the policies contained in the GPOs to the client machine. This second loop is a little more involved. When we iterate over `changed_gpo_list`, we're actually iterating over a list of GPO objects. The attributes of the object are:

- `gpo.name`: The GUID of the GPO.
- `gpo.file_sys_path`: A physical path to a cache of GPO on the local filesystem.

There are other methods and attributes, but these are the only ones important to a CSE.

The primary purpose of this loop is to iterate over the GPOs, read their policy in the SYSVOL, then check the sections for the Registry Key we created in our Server Side Extension. If our policy Registry Key exists, then we read the entry and apply the policy.

In our example, we find the 'Software\Policies\Samba\Unix Settings\Daily Scripts' policy, then read the script contents from Registry.pol entry and write the script to a local file.

The `applier_func` function is called to apply the policies contained in the GPOs to the client machine. It takes two arguments: keyname and entries. The keyname argument is the name of the policy being applied, and the entries argument is a list of the policy entries. The function writes the policy entries to randomly generated file names in the appropriate cron directories on the client machine, and returns the names of the temporary files. The file names will be stored in the Group Policy Cache, for retrieving later for the

`deleted_gpo_list`.

20.2.3 Resultant Set of Policy

The `rsop` function in the extension is optional. It should return a dictionary containing key/value pairs of what our current policy will apply or has applied. The function is passed a list of GPO objects (similar to our `changed_gpo_list`), and we should parse the list similar to how we did in `process_group_policy`.

The `rsop` function generates the output for our Resultant Set of Policy. RSoP is a feature in Group Policy that allows you to determine the effective settings that are applied to a user or a computer as a result of Group Policy processing. RSoP provides a report that shows the policies that have been applied to the user or computer, as well as any conflicts or errors that may have occurred during Group Policy processing. RSoP can be used to troubleshoot Group Policy issues, to verify that the correct policies are being applied, and to determine the impact of Group Policy on a particular user or computer.

This function enables the `samba-gpupdate --rsop` command (see Chapter 23.1).

20.2.4 Registering/Unregistering a Client Side Extension

The CSE also includes a function for registering and unregistering the CSE with the Group Policy engine on the client machine. While the example code provides a detailed example of how to register an extension, the basic requirement is simply to call `register_gp_extension()`.

```
ext_guid = '{5930022C-94FF-4ED5-A403-CFB4549DB6F0}'
ext_path = os.path.realpath(__file__)
register_gp_extension(ext_guid, 'gp_scripts_ext', ext_path,
    smb_conf='/etc/samba/smb.conf', machine=True, user=False)
```

The extension guid can be any random guid. It simply must be unique among all extensions that you register to the host. The extension path is literally just the path to the source file containing your CSE.

You must pass your smb.conf file to the extension, so it knows where to store the list of registered extensions. You also must specify whether to apply this extension to the machine, or to individual users (or to both).

Unregistering the extension is simple. You call the `unregister_gp_extension()` and pass it the unique guid you previously chose which represents this CSE.

20.2.4.1 Registering/Unregistering a Client Side Extension via samba-tool

Alternatively, as of Samba 4.18, Client Side Extension registration can be managed using `samba-tool`.

The `samba-tool gpo cse register` command is used to register a Group Policy Client Side Extension (CSE) on a Linux client. The command takes two arguments: the path to the CSE file and the name of the CSE. It also accepts two options: `--machine` or `--user`. The command does not enable the CSE for either Machine or User policy by default.

To enable a CSE for Machine policy, you must use the `--machine` option when running the `samba-tool gpo cse register` command. For example, to register a CSE file located at `/root/policies/gp_test_ext.py` with the name `gp_test_ext` and enable it for Machine policy, the command would be:

```
samba-tool gpo cse register /root/policies/gp_test_ext.py \
  gp_test_ext --machine
```

When registering a CSE, `samba-tool` will automatically generate a unique random GUID to identify the extension. To find the unique GUID of your extension, you can use the `samba-tool gpo cse list` command.

The output of the `samba-tool gpo cse list` command shows the GUID of the CSE, the file name, the extension name, and whether machine policy or user policy are enabled. For example:

```
> samba-tool gpo cse list
UniqueGUID          : {5d159033-f613-4a60-90e7-87e0f6847fbf}
FileName            : /root/policies/gp_test_ext.py
```

```
ProcessGroupPolicy : gp_test_ext
MachinePolicy      : True
UserPolicy         : False
```

To unregister a CSE, the `samba-tool gpo cse unregister` command is used, with the unique GUID of the CSE as the argument. For example, to unregister a CSE with the GUID {5d159033-f613-4a60-90e7-87e0f6847fbf}, the command would be:

```
samba-tool gpo cse unregister \
  {5d159033-f613-4a60-90e7-87e0f6847fbf}
```

Note that the above commands are run on the Linux client machine (not from a Samba ADDC), and must be executed as the root user.

These commands are useful for registering custom CSEs, but can also be utilized to backport CSEs from newer versions of Samba. For example, if you'd like to utilize the Chrome policy which isn't available in the installed version of Samba, you can fetch the `python/samba/gp/gp_chromium_ext.py` file from the Samba master branch, then activate the policy via:

```
wget bit.ly/3H3vMba -O /root/policies/gp_chromium_ext.py
samba-tool gpo cse register /root/policies/gp_chromium_ext.py \
  gp_chrome_ext --machine
samba-tool gpo cse register /root/policies/gp_chromium_ext.py \
  gp_chromium_ext --machine
```

Notice that we ran the register command twice. This is because the `gp_chromium_ext.py` contains two CSEs, `gp_chrome_ext` and `gp_chromium_ext`.

21 Modifying a Registry.pol File

21.1 Using samba-tool

Samba provides the `samba-tool gpo load`, `samba-tool gpo remove` and `samba-tool gpo show` commands for manipulating Registry.pol policies. These commands format the registry policies as **json** to simplify the process. For example, a policy which sets the Firefox homepage would like like so:

```
[
  {
    "keyname": "Software\\Policies\\Mozilla\\Firefox\\Homepage",
    "valuename": "StartPage",
    "class": "MACHINE",
    "type": "REG_SZ",
    "data": "homepage"
  },
  {
    "keyname": "Software\\Policies\\Mozilla\\Firefox\\Homepage",
    "valuename": "URL",
    "class": "MACHINE",
    "type": "REG_SZ",
    "data": "samba.org"
  }
]
```

To set this policy on a GPO, we either put it in a file, or pass it `samba-tool gpo load` in standard input.

```
> sudo samba-tool gpo load -UAdministrator --content=test.json
```

21.2 Scripting with python

Samba provides python libraries for manipulating a Registry.pol on Linux. The following python code snippet demonstrates how to open one of these files.

```
from samba.ndr import ndr_unpack
from samba.dcerpc import preg

raw = open('Registry.pol', 'rb').read()
pol_conf = ndr_unpack(preg.file, raw)
```

The parsed file contains a list of entries, which you can iterate over. Each entry contains a keyname, valuename, and data.

```
for e in pol_conf.entries:
    print(e.keyname, e.valuename, e.data)
```

Writing to the `pol_conf` can be tricky. If you write the length of the entries prior to writing the entries, it will actually cause memory corruption (this is a bug). So ensure you write to the entries, then to the length. You can create an entry using the `preg` import from `samba.dcerpc`.

```
e = preg.entry()
e.keyname = b'Software\\Policies\\Samba\\smb_conf'
e.valuename = b'apply group policies'
e.type = 4 # REG_DWORD, an integer
e.data = 1

entries = list(pol_data.entries)
entries.append(e)
pol_data.entries = entries
# Ensure you set the new num_entries last
pol_data.num_entries = len(entries)
```

The data type refers to Microsoft defined registry types:

Registry.type.name	**Registry.type.value**
REG_NONE	0
REG_SZ	1
REG_EXPAND_SZ	2
REG_BINARY	3
REG_DWORD	4
REG_DWORD_BIG_ENDIAN	5

REG_LINK	6
REG_MULTI_SZ	7
REG_RESOURCE_LIST	8
REG_QWORD	11

To write your changes back to the Registry.pol file, you'll use the following:

```
from samba.ndr import ndr_pack

with open('Registry.pol', 'wb') as w:
    w.write(ndr_pack(pol_data))
```

22 Installing Administrative Templates

Administrative Templates allow you to define policies that can be administered from the Group Policy Management Editor.

The `samba-tool gpo admxload` command copies ADMX templates to the `<domain>/Policies/PolicyDefinitions` directory on the SYSVOL share. After installing any ADMX templates, you MUST install Microsoft's ADMX templates also, otherwise you will be unable to administer Windows domain members (see section 22.4).

The following is instructions on how to obtain and install the various ADMX templates that are used by Samba.

22.1 Install Samba ADMX Templates

The Samba ADMX templates are available in the Samba source tree, and can be downloaded from https://download.samba.org/pub/samba/samba-latest.tar.gz, and can then be installed using the `samba-tool gpo admxload` command.

```
> tar -xf samba-latest.tar.gz
> samba-tool gpo admxload \
 --admx-dir=./samba-4.18.0/libgpo/admx -UAdministrator
```

Warning: There are several bugs in the GNOME Settings ADMX templates in Samba versions less than 4.18, which prevents them from being displayed in some versions of the Group Policy Management Editor (GPME). Please use the templates from a newer version of the Samba sources.

22.2 Installing Firefox ADMX Templates

Download the Firefox ADMX templates from

https://github.com/mozilla/policy-templates/releases, then extract and install them to your SYSVOL using the `samba-tool gpo admxload` command.

```
> tar -xf v4.4.tar.gz
> samba-tool gpo admxload \
 --admx-dir=./policy-templates-4.4/windows -UAdministrator
```

22.3 Installing Chromium ADMX Templates

Download the Chromium ADMX templates from https://support.google.com/chrome/a/answer/187202, then extract and install them to your SYSVOL using the `samba-tool gpo admxload` command.

```
> unzip policy_templates.zip
> samba-tool gpo admxload --admx-dir=./windows/admx \
 -UAdministrator
```

22.4 Installing Windows ADMX Templates

Download the Windows ADMX templates from https://www.microsoft.com/en-us/download/102157, then extract and install them to your SYSVOL using the `samba-tool gpo admxload` command.

```
> msiextract Administrative\ Templates\ \(.admx\)\ for\
            Windows\ 10\ October\ 2020\ Update.msi
> cd ./Program\ Files/Microsoft\ Group\ Policy
> cd Windows\ 10\ October\ 2020\ Update\ \(20H2\)
> samba-tool gpo admxload  --admx-dir=./PolicyDefinitions \
 -UAdministrator
```

23 Automatic Policy Refresh

The `samba-gpupdate` command is typically executed on a regular interval between 90 and 120 minutes in order to ensure that all policy settings are up to date. This interval is known as the Group Policy refresh interval.

There are two main ways that the `samba-gpupdate` command can be executed automatically on a regular basis: via winbind or by oddjob-gpupdate. Regardless of which method is used, the `samba-gpupdate` command is automatically executed on a regular basis to ensure that all policy settings are up to date. This helps to ensure that all users and computers in the network are following the same set of policies and helps to prevent issues with policy inconsistencies.

23.1 The samba-gpupdate command

The `samba-gpupdate` command is used to refresh Group Policy settings on an Active Directory domain member. Group Policy allows an administrator to specify settings for users and computers in an Active Directory domain. When these settings are changed, the `samba-gpupdate` command can be used to apply the changes on the domain member.

To use the `samba-gpupdate` command, open a terminal window and simply type the following:

```
samba-gpupdate
```

This will refresh all Group Policy settings on the local machine. You can also specify specific options to refresh only certain settings. For example, to refresh only the computer settings, you can use the `--force` option:

```
samba-gpupdate --force
```

To refresh only the user settings, use the `--force` option combined with the `--target` and `-U` options to specify the user:

```
samba-gpupdate --force --target=User -U tux
```

To unapply Group Policy settings, you can use the `--unapply` option:

```
samba-gpupdate --unapply
```

To print the Resultant Set of Policy (RSOP) for a particular target, you can use the `--rsop` option:

```
samba-gpupdate --rsop --target=Computer
```

It is important to note that the `samba-gpupdate` command can only be used on a machine that is a member of an Active Directory domain. It will not work on a standalone machine or on a machine that is part of a different type of domain.

23.2 Automatic Policy Refresh via winbind

To configure winbind Automatic Policy Refresh, you will set the `apply group policies` smb.conf parameter.

To set this parameter manually, you will need to add the following line to the `global` section of the smb.conf file:

```
apply group policies = Yes
```

This will enable winbind to automatically apply Group Policy settings on the *Group Policy refresh interval*.

Alternatively, you can deploy this setting automatically using smb.conf Group Policies. See chapter 5 section 5.1 for instructions how to deploy this setting via Group Policy. The `samba-gpupdate` command will need to be executed manually to deploy this setting the first time.

23.3 Automatic Policy Refresh via oddjob-gpupdate

Using oddjob-gpupdate to provide Automatic Policy Refresh allows you to run Samba's Group Policy with the System Security Services Daemon

(SSSD). SSSD is a system service that provides access to remote identity and authentication providers, such as Active Directory.

To install oddjob-gpupdate, you'll need to find the appropriate packages for your distribution. In openSUSE, for example, you can install oddjob-gpupdate via:

```
sudo zypper in oddjob oddjob-gpupdate
```

Some distributions may not have oddjob-gpupdate packaged, in which case you can build the sources from https://github.com/openSUSE/oddjob-gpupdate.

Beware that the package named *oddjob-gpupdate* in the ALT Linux distribution *is not the correct package*. This package is meant for Group Policy application using ALT Linux's custom Group Policy implementation.

After installing oddjob-gpupdate, you can start and enable the oddjob service to begin refreshing policy.

```
sudo systemctl enable oddjobd
sudo systemctl start oddjobd
```

Once the oddjobd service is running, it will automatically execute the oddjob-gpupdate command on the *Group Policy refresh interval* to update user and computer Group Policies.