

Group Policy Structure and Processing

In Chapter 1, I discussed the history of policy and described the enhancements Microsoft delivered in the Windows 2000 (Win2K) *Group Policy Object (GPO)*. Of course, these enhancements add complexity. The complexity of GPOs is generally hidden from the administrator by the simple interface used to control the settings. This interface was described in Chapter 1.

In fact, pieces of GPOs can be found in Active Directory (AD) and in the shared SYSVOL folder found on each AD domain controller. Understanding the structure of GPOs and how GPO processing works in conjunction with AD and SYSVOL will help you more effectively deploy, manage and troubleshoot Group Policy.

In this chapter, I'll start by describing how GPOs are structured and how they interact with AD and SYSVOL. Then I'll move into a discussion of how GPOs are processed by users and computers in your AD infrastructure.

Structure of the Group Policy Object

As I implied above, the GPO doesn't exist as a single object. Rather, it's a collection of settings stored in AD and a corresponding set of files stored in SYSVOL. As I've described in Chapter 1, the administrator typically views a GPO using the Microsoft Management Console (MMC) Group Policy snap-in (see Figure 2.1).

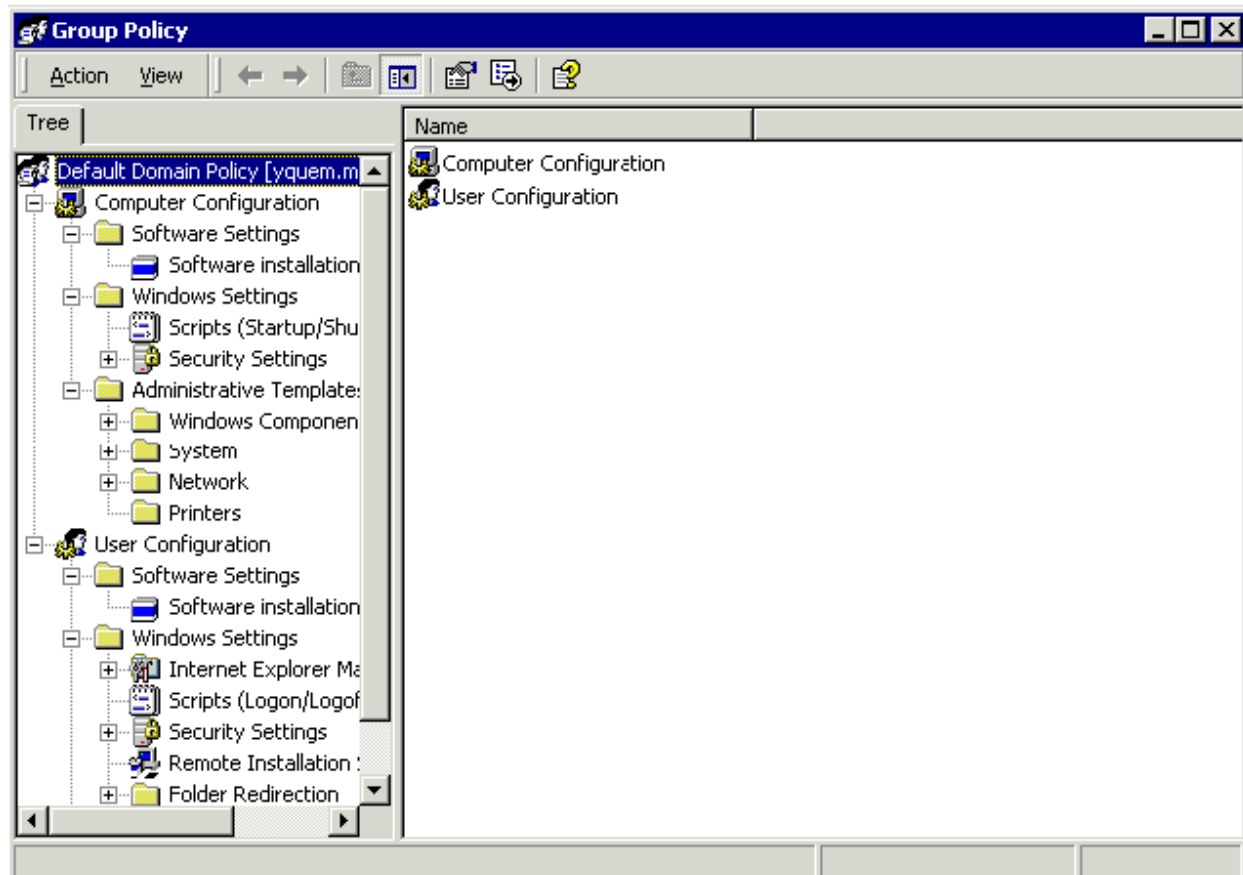


Figure 2.1: The Group Policy snap-in showing the GPO Default Domain Policy.

The GPO really consists of two components. The first part is stored in AD as a container object. This part of the GPO is referred to as the *Group Policy Container (GPC)*. The GPC AD object is derived from a special AD class called, as you would expect, **groupPolicyContainer**. The second part of the GPO is stored as files in the file system in the shared SYSVOL folder on every domain controller. This second part is called the *Group Policy Template (GPT)*.

The exception to this model is the local GPO. As I mentioned in Chapter 1, the local GPO resides on every Win2K device that you install. It doesn't have a GPC in AD or a GPT in SYSVOL. Rather, it simply stores what would correspond to its GPT in the local file system of a workstation or server. In this chapter, I'm going to focus on AD-based GPOs rather than the local variety, so that's all I'll say about local GPOs for now.

Given this reliance of GPOs on AD and a domain controller infrastructure, it's easy to understand why, as I explained in Chapter 1, you need to have an AD infrastructure in place to take full advantage of GPO functionality. But this reliance implies something else as well—namely, that GPOs “belong” to a particular domain where they're created. That is, in a multi-domain AD forest, each GPO you create must be stored in a single domain.

However, it's also important to remember that while GPOs live in a single domain, they can be linked to objects outside the domain. As you can imagine, this “cross-domain” linking can have an impact on the performance of GPO processing. I'll discuss that topic later in this chapter in “Group Policy Processing.”

Next, let's dive into the details of the two parts of the GPO—the GPC and GPT.

Group Policy Container (GPC)

The *Group Policy Container (GPC)* is critical to ensuring that computers and users in AD environments correctly process policies. To view the GPC, open the MMC AD Users and Computers snap-in. Make sure you have the advanced features view enabled. Select the system container, and beneath that, the Policies container in the domain. There you'll find all of the policies defined in the domain. Each GPO is represented as a container named with a 128-bit *Globally Unique ID (GUID)* (see Figure 2.2).

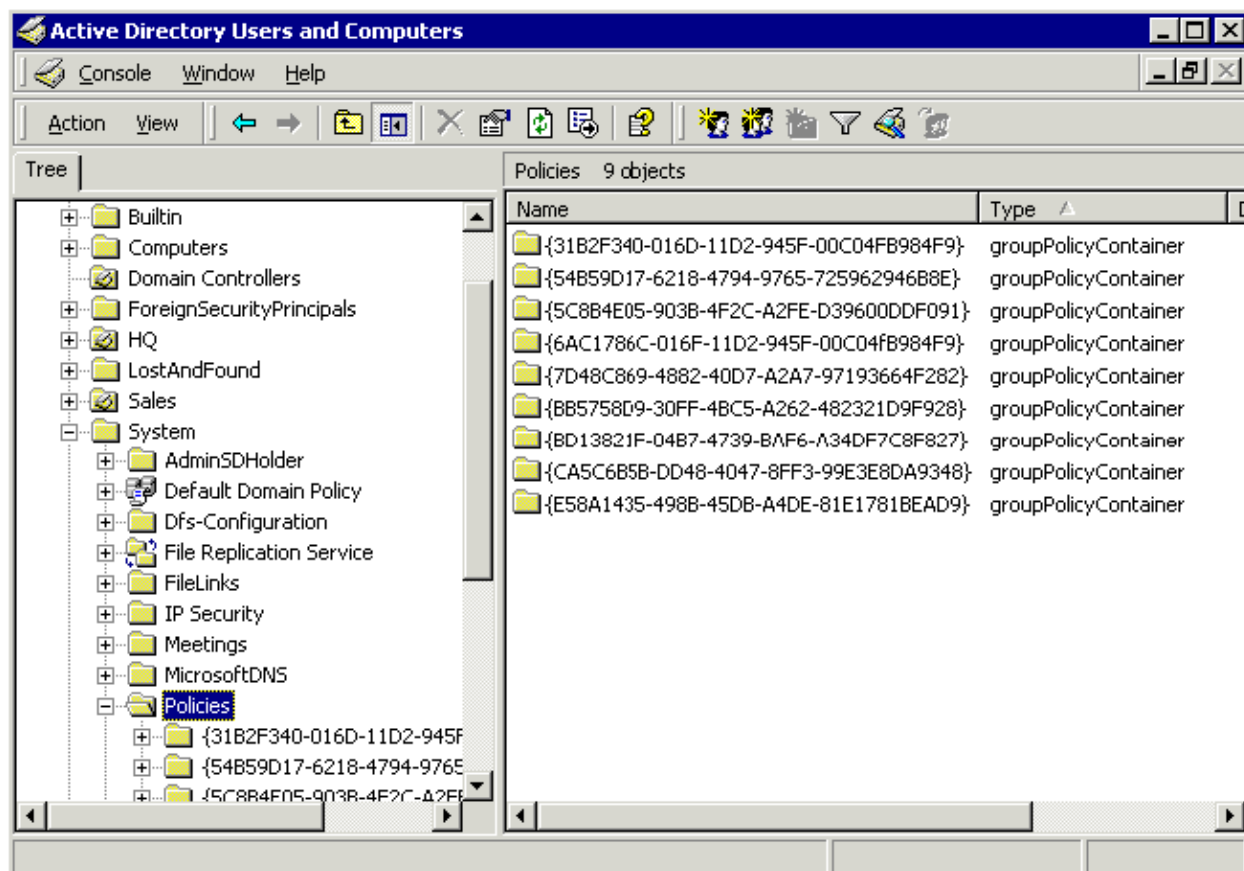


Figure 2.2: GPCs exposed through the AD Users and Computers tool.

How to View the Policies Container

By default, you cannot see the System\Policies container when you first start the AD Users and Computers tool. To see the container, choose View>Advanced Features (shown in Figure 2.3).

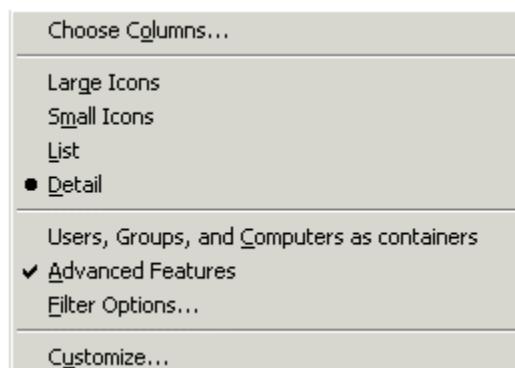


Figure 2.3: Advanced Features selected on the MMC View menu.

Now let's think about which GPC object in Figure 2.2 maps to a recognizable GPO, like the one shown in Figure 2.1.

Typically, we compare the GUID string in the Policies container with a GUID string on the GPO, as seen in the MMC Group Policy tool. If you right-click the GPO name in the Group Policy tool, then choose Properties from the context menu, you can view the properties of the GPO in the Properties dialog box. You'll see its GUID listed on the General tab as Unique Name (see Figure 2.4).

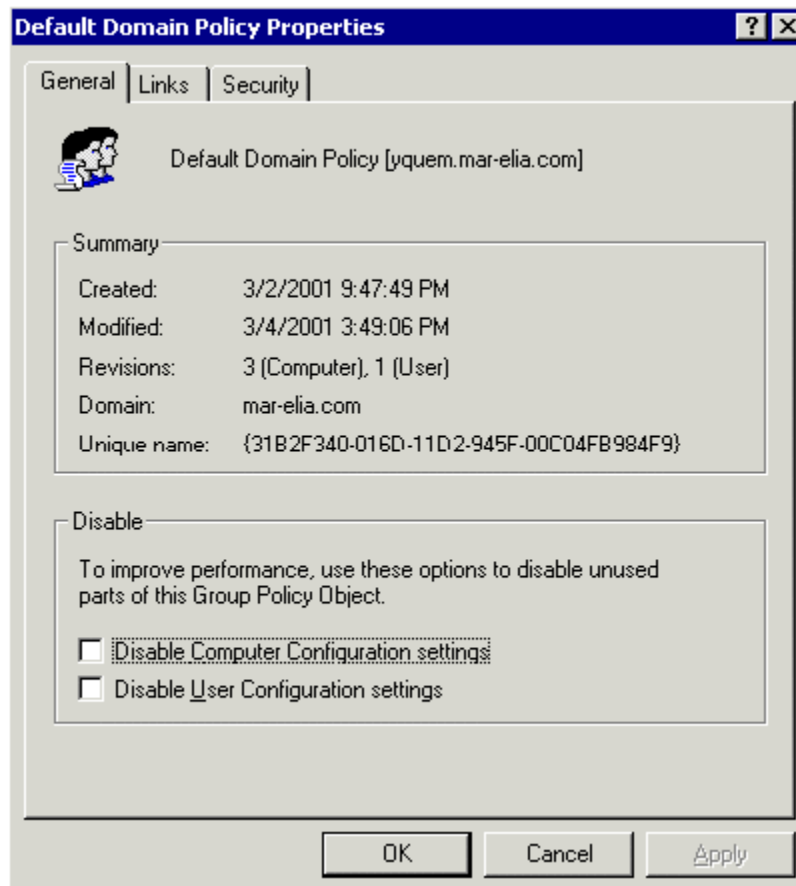


Figure 2.4: Viewing the GUID of a GPO using the Group Policy tool.

Once you know the GUID of the GPO you're editing, it's easy to find the corresponding **GPC** object in the Policies container of the domain where the policy is stored. In addition, if you're reviewing a GPO and are unsure which domain is mastering it, just look at the value for Domain on the General tab of the policy's Properties dialog box. The domain shown there is the domain where the GPC, and thus the GPO, is stored.

Finding GPO Friendly Names from Their GUIDs

The need to derive a GPO's friendly name from its GUID (that is, to perform a reverse policy name lookup) doesn't occur very often, but if you need to perform such an operation, you can use a tool called ADSIEdit.



You can install ADSIEdit as a part of the Support Tools from the Windows 2000 Professional, Server, Advanced Server, or Data Center CDs. It's also available in the Windows 2000 Resource Kit.



Registering ADSIEdit Without Installing the Support Tools

You may not have security clearance to install Support Tools on your servers. In this case, consider registering the `adsiedit.dll` for one-time use. Simply find a copy of `adsiedit.dll` (on any system where you've installed the previous tool set). Copy the `adsiedit.dll` to your system's `%systemroot%\system32` folder, and run the following code to register the dll:

```
Regsvr32 %systemroot%\system32\adsiedit.dll
```

Un-registering is as simple as executing the following command:

```
Regsvr32 /u %systemroot%\system32\adsiedit.dll
```

Once you've installed ADSIEdit, you can start it by choosing Start>Run, then typing `adsiedit.msc` in the dialog box or by loading the ADSIEdit snap-in into an MMC console. When you start the tool, three folders are listed—Domain NC, Configuration Container, and Schema. These are AD's three *naming contexts*.

Naming contexts are partitions in AD. A full discussion of them is beyond the scope of this book, but for our purposes, GPCs are always stored in the Domain NC folder. Expand this folder, and you'll see a structure very similar to that shown in the AD Users and Computers tool. If you select CN=System, then CN=Policies, you'll see a list of folders, named with their GUIDs. These are the GPCs that reside in the domain (see Figure 2.5).

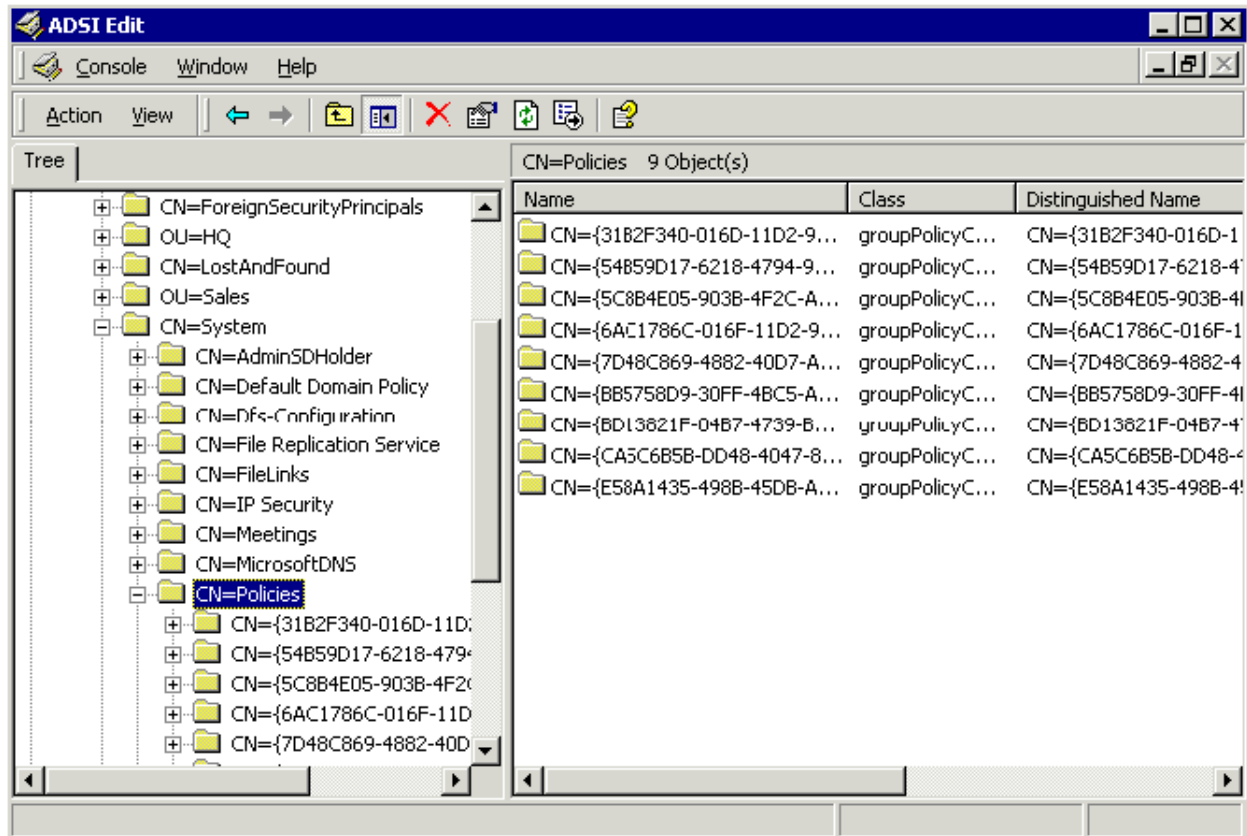


Figure 2.5: Using ADSIEdit to view GPC objects.

Once you've exposed the GPC objects using ADSIEdit, you can review the properties of each GUID to find a GPO by its friendly name. Right-click a GUID and choose Properties from the context menu. The Properties dialog box appears. For example, suppose you're looking for the Default Domain Controller policy. In the Select a Property to View text box, choose the displayName property from the drop-down list (see Figure 2.6). The Value(s) text box at the bottom of the dialog box displays the friendly name for that GPO.

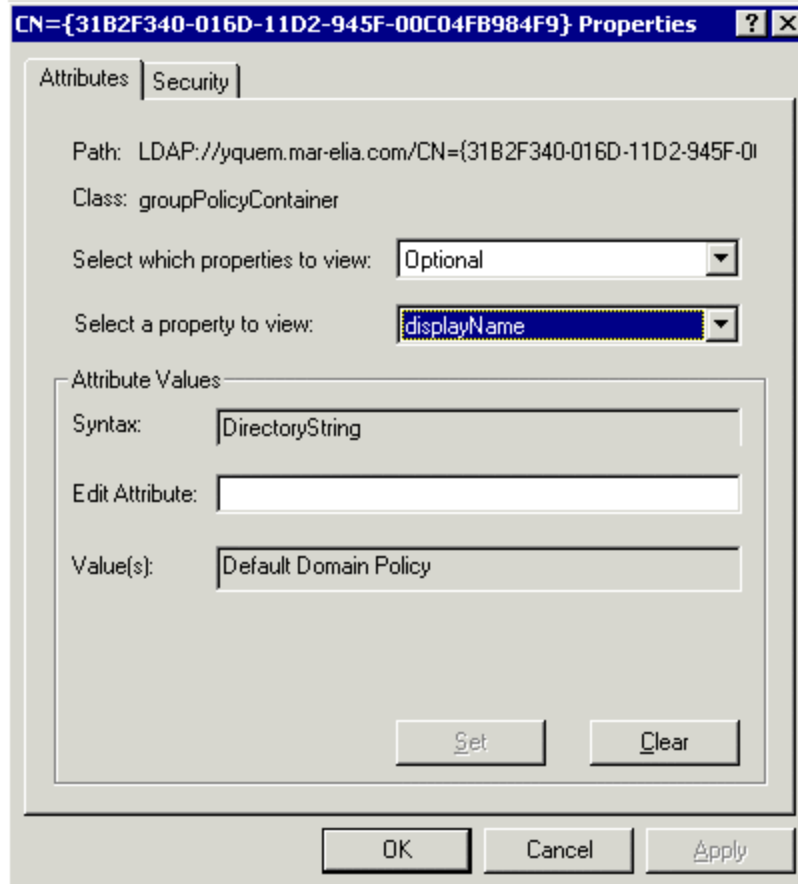


Figure 2.6: Reviewing GPC property information in ADSIEdit.

In this next section, I'll take a look at the GPO's template information, or what is commonly referred to as the file-system part of a GPO.

Group Policy Template (GPT)

The *Group Policy Template (GPT)* provides an additional level of storage as it pertains to the GPC. The GPT stores policy settings that don't store well in AD. The GPT is a template or a suite of files stored on a Win2K domain controller. A particular GPT is tied to its associated GPC through a special property on the GPC object under the Policies container.

 GPTs and GPCs have a one-to-one relationship. In other words, you generally can't (and shouldn't) have a particular GPT referenced by more than one GPC.

To see the relationship between a GPT and a GPC, display the Properties dialog box for a GPC in ADSIEdit. In the Select a Property to View text box, choose the gPCFileSysPath property from the drop-down list. This property translates to the path to the GPT for this particular GPC. (See Figure 2.7.)

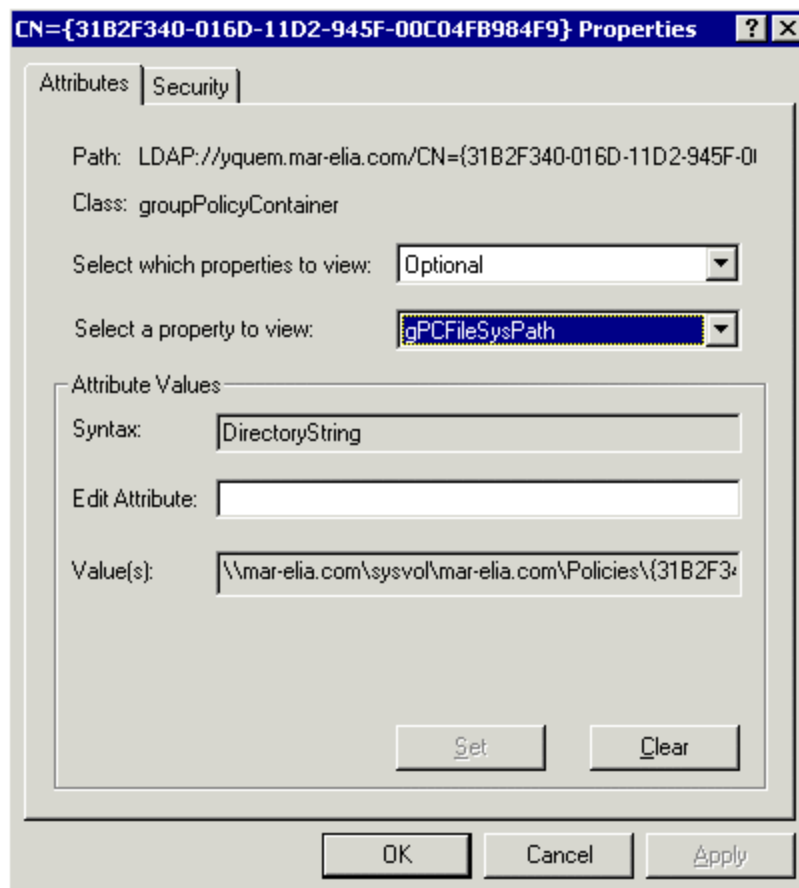


Figure 2.7: Viewing the gPCFileSysPath property on a GPC object.

As you look at the Value(s) text box in Figure 2.7, you'll see a rather long entry. The following is a typical example:

```
\\domain.tld\sysvol\domain.tld\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}
```

Note that the GUID in the entry above is the same as the title bar in Figure 2.7. This shows that the GPC and the GPT share the use of the GUID to ensure consistency, regardless of the GPO's friendly name.

The value of the GPT is that it gives software vendors (Microsoft included) who want to Group Policy-enable their applications the flexibility to provide policy-related configuration data (for example, files and file folders) that isn't conducive to being stored in AD. This apparent disconnection between the GPT and the GPC can cause many problems with GPO processing that you'll need to be aware of. (This is discussed in greater detail in Chapter 6.) Later in this chapter, I'll discuss GPO versioning, which is the principal method by which Microsoft attempts to ensure that for a particular GPO, the contents of both the GPC and the GPT are synchronized before being processed by a user or computer. (See "GPO Versioning.")

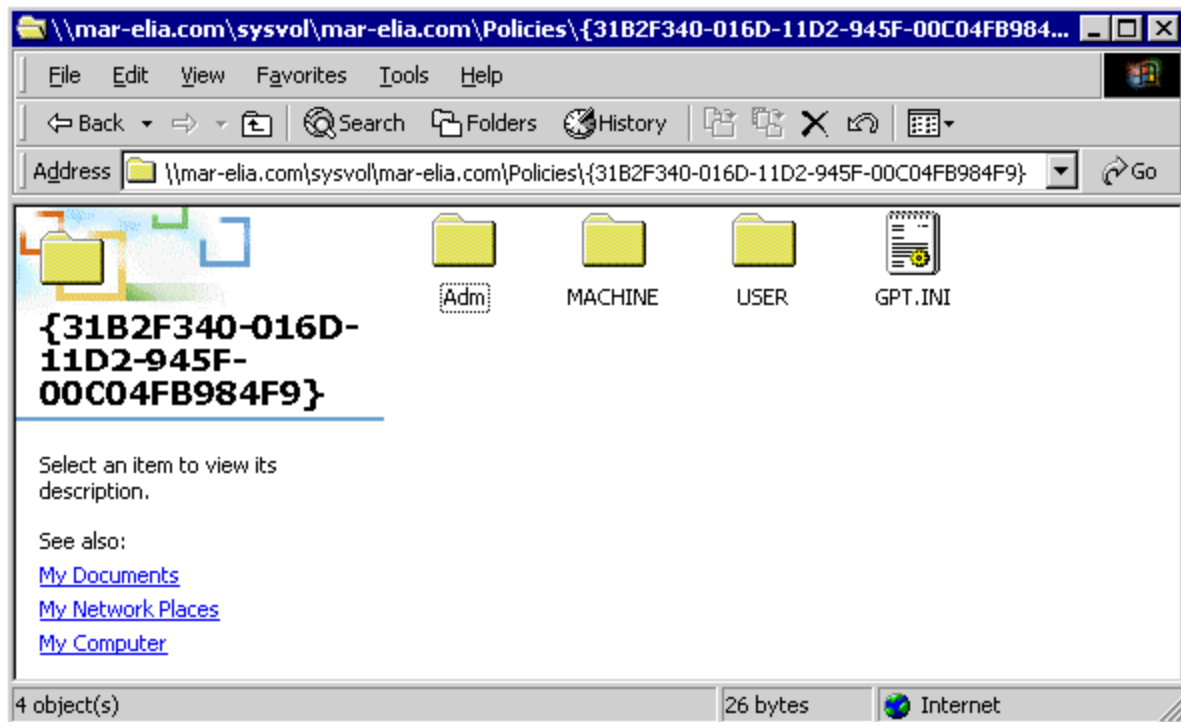


Figure 2.8: Viewing GPT file-system objects.

Compare Figure 2.8 with Figure 2.7, and you'll see that the title bars show the same GUID, but the content in Figure 2.8 is a view of the file system on a Win2K domain controller. Specifically, Figure 2.8 is the SYSVOL share point found on all domain controllers, and it shows the contents of a GPT. As you can see, it's composed of a number of files and folders, which I'll describe in a later section. (See "GPT Storage.")

Because a GPT is stored in SYSVOL, it's replicated to all domain controllers in a particular domain. When you create or edit a GPO, in effect, you're creating or editing a GPC on the domain controller that you're currently focused on, and you're correspondingly creating or editing a GPT on the same domain controller in SYSVOL. As the domain controller replicates AD and SYSVOL, so is your GPO distributed throughout your AD infrastructure.

 The NT File Replication Service (NTFRS) is responsible for replicating the SYSVOL structure among domain controllers. However, AD is replicated using a different mechanism and can replicate on a different schedule from NTFRS.

In the next section, I'll drill into GPC and GPT storage. Namely, I'll take a close look at the contents and subcontainers of the GPC and the GPT.

Storing Group Policy Settings

The mechanism used by Group Policy for storing settings is versatile. However, this versatility breeds some complications. It's critical that you understand the underlying storage mechanisms in order to effectively manage and troubleshoot GPOs in your infrastructure.

As I mentioned earlier and in Chapter 1, Group Policy allows software vendors to Group Policy-enable their applications. When making the decision to integrate with Group Policy, developers must consider their storage options.

Group policy settings can be stored in three places; these are shown in Table 2.1.

Location	Content Should Be	Container
AD	Typically less than 100KB in size, binary, alpha-numeric, and so on. Doesn't change very often.	GPC
SYSVOL\policies\<GUID>\<User or Machine>\Registry.pol	Administrative Template (that is, Registry) policy.	GPT
SYSVOL\policies\<GUID>\<User or Machine>\<vendor or custom folder>\customFile	Any policy-related configuration data.	GPT

Table 2.1: Options for storing settings in GPOs.

I'll start with a more detailed description of GPC storage, then move into what's stored on the file system under the SYSVOL share point.

Storing GPC Information

A GPC object has two important subcontainers in AD. These two subcontainers (see Figure 2.9) correspond to the AD objects we envision Group Policy managing—namely, people (users) and machines (computers).

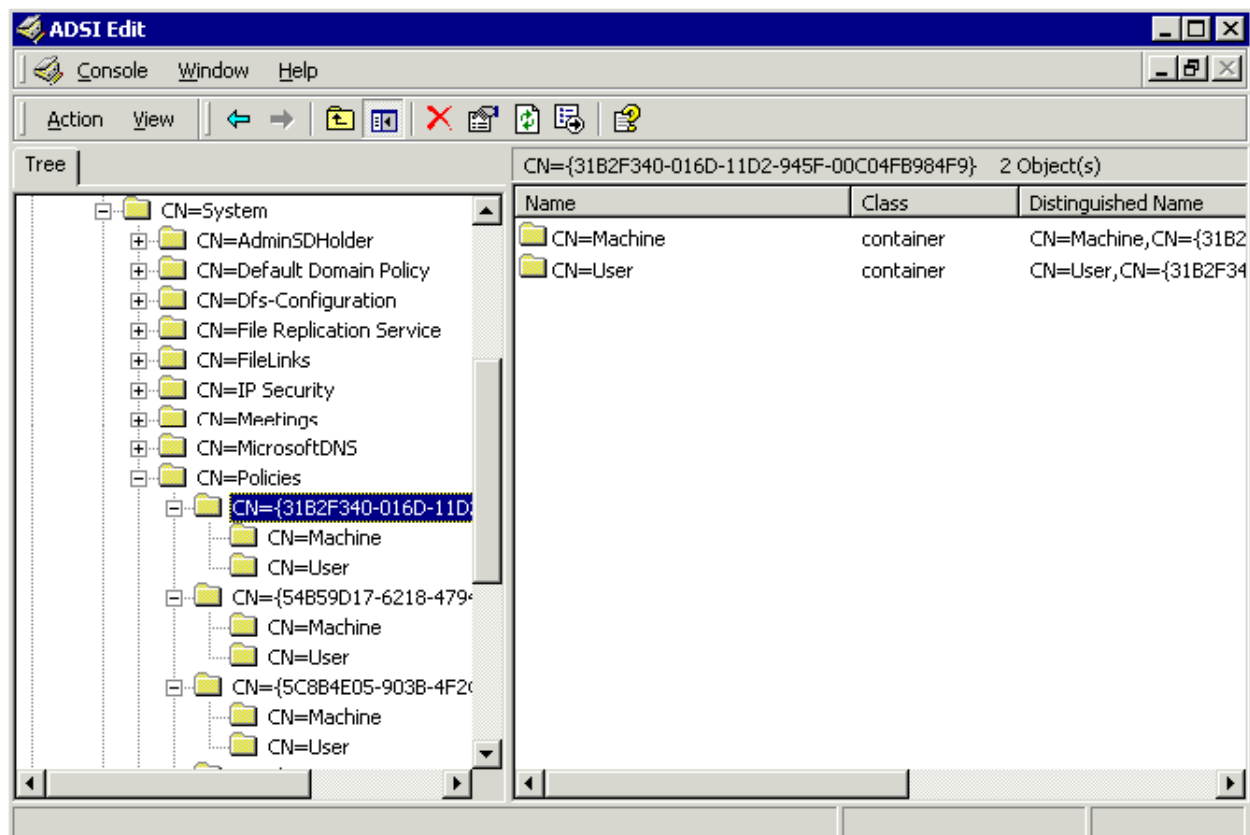


Figure 2.9: User and Machine subcontainers in a GPC.

As with the GPT, any software vendor has the ability to extend the user or machine subcontainers to provide AD-based policy storage for their applications. Earlier in this chapter, I discussed using ADSIEdit to view some of the properties on a GPC object. Table 2.2 provides a description of some of the more interesting properties in a GPC.

Property	Description
displayName	The Group Policy friendly name shown in the MMC Group Policy tool and on the Group Policy tab of Sites, Domains and OUs.
gPCFileSysPath	The file-system location of the GPT. It typically points to the SYSVOL share point.
gPCFunctionalityVersion	Defines the version of the MMC Group Policy extension that created this GPO.
GPCMachineExtensionNames	Describes, in GUID form, the extension dynamic link libraries (DLLs), or Group Policy functionality, defined for computers on this GPO. If Disable Computer Configuration Settings is selected in the policy's Properties dialog box, this is blank.
GPCUserExtensionNames	Describes, in GUID form, the extension DLLs, or Group Policy functionality, defined for users. If Disable User Configuration Settings is selected in the policy's Properties dialog box, this is blank.
VersionNumber	The version of the policy in the directory. It's compared to the version in the gpt.ini file in the GPT to determine how a client extension might react to the GPC or GPT being out-of-sync. (For details about GPO versioning, see "GPO Versioning" further on in this chapter.)

Table 2.2: A list of some important properties on GPC objects in the Policies container.

The Class Store Container

In Chapter 1, I described the GPO software installation feature, which lets an administrator publish or assign applications to users and computers. If you look at a GPC in ADSIEdit where an application has been published or assigned using software settings, you'll see the Class Store container listed under either the user or the machine subcontainers. The Class Store container has yet another subcontainer called Packages. The Packages container stores objects of class **packageRegistration**, which represent each application that has been assigned or published on this GPO (see Figure 2.10).

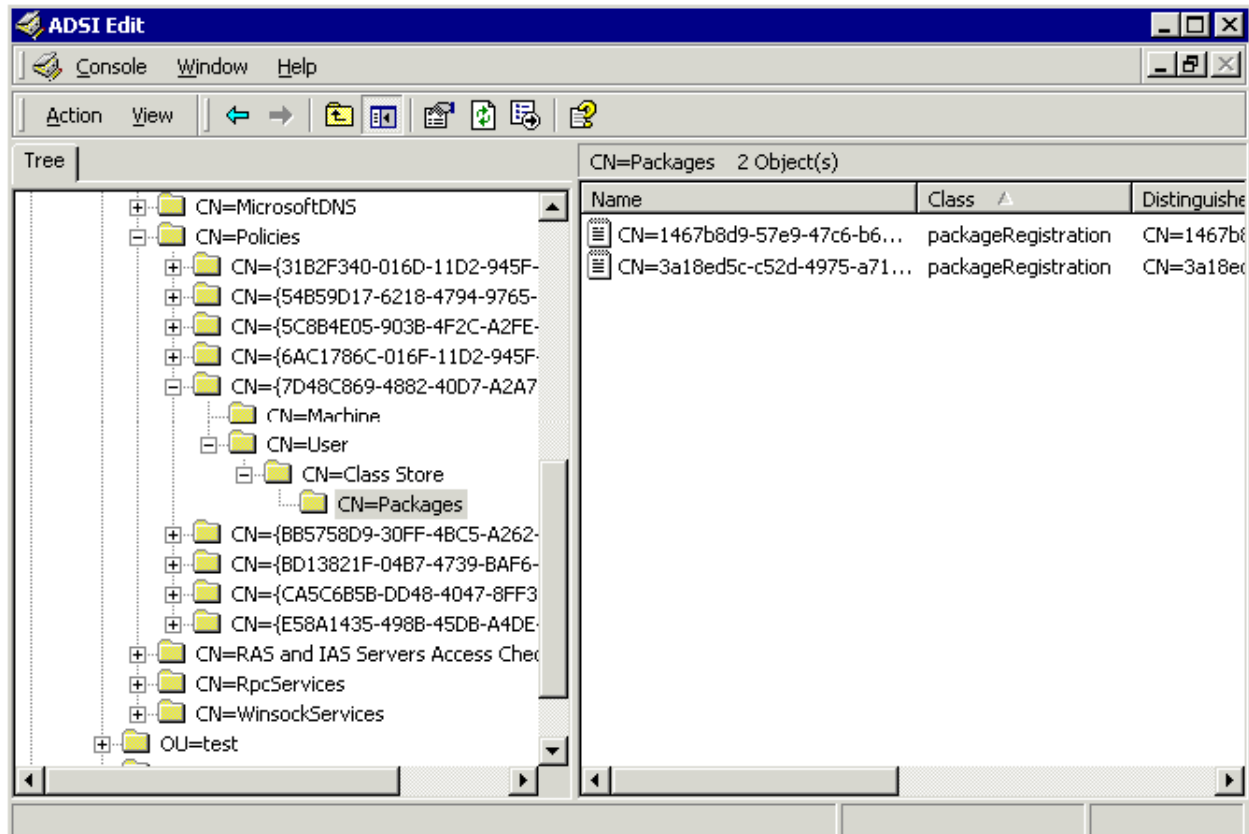


Figure 2.10: Viewing the Class Store subcontainer of a GPC.

The Class Store provides a powerful tool for developers and administrators. Think of the Class Store as an AD-based version of the Win2K Registry. More specifically, it's an AD version of HKEY_CLASSES_ROOT, the Registry subtree that holds useful configuration information about file-extension associations and Component Object Model (COM) objects that are installed on a particular system.

As discussed in Chapter 1, when you assign an application to a user, the application registers up to three advertisements on the user's computer. These advertisements include file shortcuts, file-extension associations, and COM interfaces (if any). For this last form of advertisement, COM class registration, the Class Store provides a central repository of registration that lets a computer or user processing GPOs know what COM objects have been deployed.

In addition, when a user wants to use the Add/Remove Programs (ARP) applet in the Control Panel to review which GPO-published applications are available from the network, the Winlogon process on the computer uses an application programming interface (API) called GetGPOList to query the associated GPCs for GPOs that apply to them. Once the API has identified the applicable GPCs, it queries the Class Store package container to determine which applications are available to the user, and the ARP applet is populated with a list of available applications. Likewise, when GPOs are processed when a computer starts up or a user logs on, the GetGPOList API methods review all assigned applications.



GetGPOList is an API documented by Microsoft that allows the system to pass in user or computer information and have returned a list of applicable GPOs. GetGPOList is discussed later in this chapter in “Group Policy Processing.”

As I mentioned in Chapter 1, Group Policy makes it possible to deploy applications that you install using file-extension activation. Normally, when a user double-clicks a file with an extension that hasn’t been registered to an application with HKEY_CLASSES_ROOT, the familiar Open With dialog box appears, prompting the user to choose an application to open the file.

Group Policy uses a combination of the Class Store and a file called an application assignment script (.aas file, stored in the GPT) to guarantee that file-extension associations are advertised when an application has been assigned to a computer or user. When a computer or user accesses a file using that extension, Win2K then automatically installs applications with registered file-extension associations as part of a GPO-deployed application.

For example, if you assign Microsoft Office using Group Policy, a number of file extensions are associated with it in the GPC and GPT (for example, .doc, .xls, .ppt). The next time a user logs on or a computer restarts, these file extensions are registered in the computer’s local Registry. If a user then clicks a file with a registered extension—for example, a Word .doc file—the Office package will be installed at that time.

There are several reasons why developers can take advantage of the Class Store too. Developers can create applications installed using the Windows Installer engine and packaging format. These application packages contain information about DLL files, or COM objects, that might be needed by other applications. When an application is published or assigned in a GPO, references to any COM objects that are exposed by that application are stored in the GPC in the **packageRegistration** object for that application. These references are stored, using their associated GUID, in the **cOMClassID** property on the object.

If an application running on the workstation makes a subsequent call to a COM object that has been published or assigned, the API that calls the object queries the **packageRegistration** objects in the GPC to determine whether the necessary COM object has been registered. If it has, the application package that contains the COM object is installed automatically from the MSI package defined in the GPO.



The Class Store provides a valuable way for developers to use AD to make their COM components widely and automatically available to their users.

Table 2.3 lists some of the more interesting properties in the **packageRegistration** object.

Property	Description
canUpgradeScript	The script that would be called if the package had an upgrade associated with it. (See Chapter 3 for a discussion of establishing upgrade relationships.) It’s typically in the form \\domain.tld\\sysvol\\policies\\domain.tld\\policies\\GUID.
categories	Contains references to application categories that an administrator can define to help organize applications that appear in the Control Panel ARP applet.
cOMClassID	The GUID(s) associated with the COM object(s) that this package contains. When an instance of an object is created, this property is queried by APIs to determine if the application package contains the required object.

Property	Description
cOMInterfaceID	Defines COM interface IDs available from this package (similar to cOMClassID).
cOMProgID	Defines the COM ProgID contained in the package. (A ProgID is the user-friendly textual representation of a COM object's Class ID.) For example, you see ProgIDs used when a Visual Basic (or VBScript) application creates a new object instance.
cOMTypelibID	Lists any COM type libraries available with this package.
displayName	Is the display name of the application as it's written in the Windows Installer package.
installUiLevel	Is a numeric value that describes whether the basic or maximum installation user interface (UI) is used. Basic is 3. Maximum is 5.
localeID	A numerical code representing the language of the Windows Installer package. You can find these codes in the Win2K Resource Kit documentation (w2rkbook.chm) by searching on the phrase <i>language codes</i> .
machineArchitecture	Defines the type of system on which the package can be deployed. If the policy were multi-platform, you could deploy packages to users on different systems.
msiFileList	Identifies the location on the network of the MSI package that installs this application. If you changes the server where your applications were deployed from, you'd modify the 0:<File Location> value to reflect that change. Ironically, once you've deployed the package, you can't change this value in the Group Policy snap-in, so modifying this property is your only choice. (See the caution below this table for information about making changes to the GPC using ADSIEdit.) The format of this property is <n:file location>, where n can be 0...∞ and <file location> is a valid network resource like a Distributed File System (Dfs) share point. Any value of 1 or higher identifies a Windows Installer transform file that was identified on the Modifications tab when the application was deployed in the GPO.
msiScriptPath	Identifies the application assignment script (.aas file) that should be called when installing or removing an application. (This is discussed in more detail in the next section, "GPT Storage.")
packageFlags	A computed value based on settings on the Deployment tab of a GPC.
packageName	The actual name of the package in the Group Policy tool.
packageType	A numeric value differentiating Windows Installer (.msi) packages from custom script packages called .zap files. (For more information on .zap files, see Chapter 3.) Windows Installer is 5. ZAP is 3.
productCode	Octet representation of the product code, shown in the form of xxxxx-xxx-xxxxxxx-xxxxx.
setupCommand	The value used in .zap files for installation.
url	The application-vendor URL that is listed on the general page of a package.
versionNumberHi	Because version numbers use dotted decimal notation, this describes the whole number, or the tens place. For example, in <i>Version 2.0</i> , 2 is the versionNumberHi.
versionNumberLow	Related to versionNumberHi— Represents everything to the right of the dotted decimal notation version number. In the example of <i>Version 2.0</i> ,.0 is the versionNumberLow.

Table 2.3: Important properties on packageRegistration objects in the GPC.



Be careful about editing the properties of a GPC directly in ADSIEdit. As I'll explain later in this chapter, GPO versioning is the mechanism by which Win2K computers validate whether the GPC and GPT are in sync when processing a GPO. (See the section entitled "GPO Versioning.") Versioning is normally controlled using the MMC Group Policy snap-in. When you make a change to a GPO, the GPC and GPT version information is incremented to reflect the change. When you make a change to a GPO, it may require changes to both the GPT and the GPC or to one but not the other, but the Group Policy tool will increment version information on both. If you make a change to a GPC using a tool such as ADSIEdit, you may be circumventing GPO versioning, creating a situation where your Win2K computers think your GPOs are in sync when in fact, they're not.

Storing GPT Information

Recall from the last section the property on the GPC called **gPCFileSysPath**; it points to a folder on the [domain's](#) SYSVOL share point that references the associated GPT for that GPC. An example is \\domain.tld\\sysvol\\domain.tld\\Policies\\{6AC1786C-016F-11D2-945F-00C04fB984F9}.

If we step up one level from the path shown above, we see that SYSVOL contains a number of GUIDs representing the GPTs for each GPO defined in the domain. Remember, all policy has to be stored somewhere. It's always stored on domain controllers in some domain in the forest.

In a new AD domain, the Policies folder contains at least two GUIDs relating to the default GPOs available for the domain. Table 2.4 shows the relationship between GUID and policy for default domain policies.

GUID	Policy Name
{31B2F340-016D-11D2-945F-00C04FB984F9}	Default Domain Policy
{6AC1786C-016F-11D2-945F-00C04fB984F9}	Default Domain Controller Policy

Table 2.4: GUID-to-policy mapping for default domain policies.

Once you open a GUID folder using file explorer, whether it's a default policy or one that you've created, you start to see the guts of the GPT. If you refer back to Figure 2.8, you'll see the default top-level folders for a GPT. Table 2.5 describes the purpose of these folders.

This Object	Stores This
Adm	All of the Administrative Templates (.adm files) for a policy. If you add an administrative template (.adm) file to a policy it's stored here and replicated throughout the domain.
Machine	All machine policy, including Registry policy.
User	All user policy, including Registry policy.
GPT.ini	Version information about the GPT. It allows extensions to gracefully fail if there is a synchronization problem between the GPC and the GPT.

Table 2.5: A description of the contents of the Objects folder in a GPT.

To expand on the information in Table 2.5, the Adm folder stores Administrative Templates (.adm files) used by the Group Policy editor. The default files in the Adm folder are:

- **conf.adm:** manages Net Meeting settings using policy

- **inetres.adm:** manages Internet Explorer settings
- **system.adm:** manages all Registry settings associated with the operating system (OS)

The Machine and User folders are very similar in structure, so I'll discuss them in tandem. If you look at the User folder in the GPT GUID, you'll find something similar to Figure 2.11.

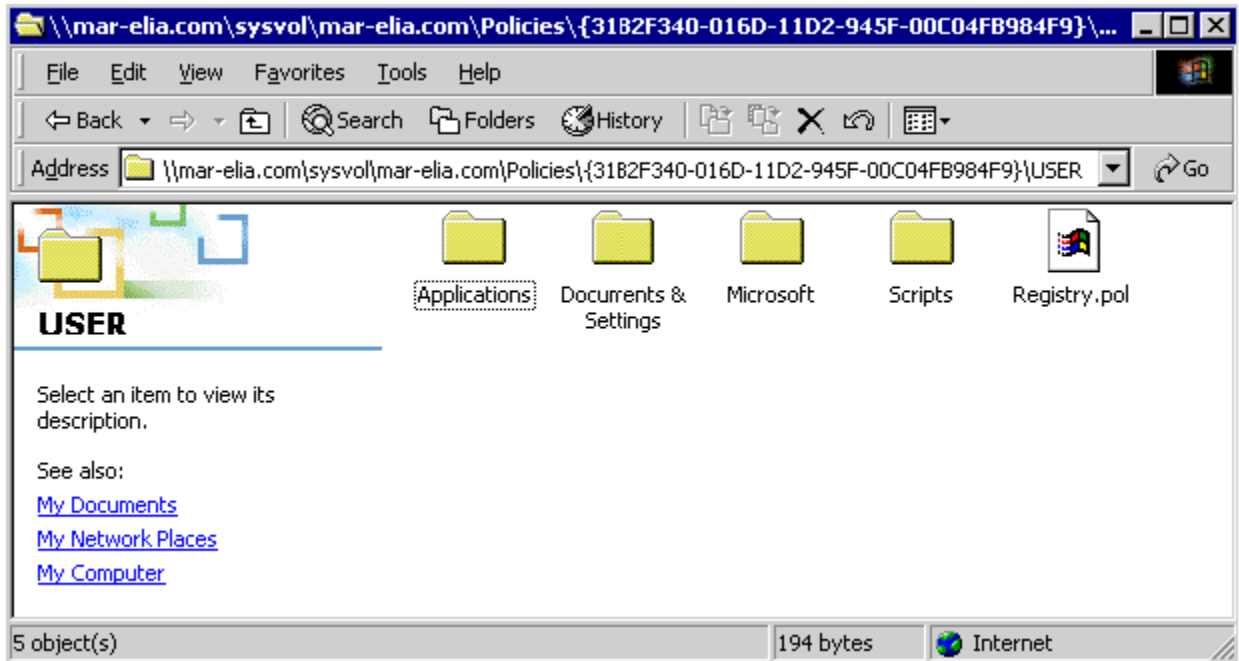


Figure 2.11: The folder structure of a User GPT.

I'll now drill down into each folder in the User and Computer folders and describe its function.

Applications Folder

The Applications folder in Figure 2.11 stores files relating to applications being deployed using the Software Installation feature in Group Policy. The files in this folder contain application assignment scripts using the file-name format <GUID>.aas. The GUID refers to the **packageRegistration** object in the Class Store of the GPC. The files can be viewed in Notepad, but the .aas extension has no default association. As I mentioned above, the .aas script allows advertisements to be registered for a particular application deployed using Group Policy.

Documents & Settings Folder

Figure 2.11 shows the Documents and Settings folder. This folder appears only when you've configured the Folder Redirection feature in Group Policy. The Folder Redirection extension code on the client reads this folder when the user logs on, and the extension determines the folder redirection settings for the user.

The Documents and Settings folder contains a special text file named fdeploy.ini that contains configuration-related information. An example of the file is shown in Listing 2.1.

```
[FolderStatus]
```

```

Application Data=39
Desktop=39
My Documents=39
My Pictures=2
Start Menu=28
Programs=2
Startup=2
[Application Data]
s-1-5-21-970468228-843086746-623648099-1714=\\dks\corp\m-
users\%username%\appdata
[Desktop]
s-1-5-21-970468228-843086746-623648099-1714=\\dks\corp\m-
users\%username%\desktop
[My Documents]
s-1-5-21-970468228-843086746-623648099-1714=\\dks\corp\m-
users\%username%\docs-n-pics
[My Pictures]
[Start Menu]
s-1-5-21-970468228-843086746-623648099-1714=\\dks\corp\m-
users\%username%\startmenu
[Programs]
[Startup]

```

Listing 2.1: The *fdeploy.ini* file, which is located in a user's Documents and Settings folder.

The FolderStatus section describes the options selected on the Settings tab of each redirected folder. The sections beneath FolderStatus refer the workstation to the correct network or file-system location for the redirected folder. In the example above, members of a user group (represented by their security identifiers, or SIDs) are being redirected to various folders under a Dfs share point (for example, \\dks\corp).

The SIDs that appear within the file (e.g. s-1-5-21....etc.) refer to the SIDs of users and groups and are used instead of friendly names because such names can change. Using SIDs ensures that the user folders are always redirected regardless of the user or group names.

Microsoft Folder

The Microsoft folder is really just a vendor's folder and is found under both the machine and user sub-folders within a GPT. Any application vendor can, and by design should, create a folder within the GPT for its applications that are Group Policy-enabled. In Microsoft's case, several applications are Group Policy-enabled and are stored in this folder—namely the Security Configuration Editor, the IE Admin. Kit and Remote Install Server (RIS) settings.

- The Security Configuration Editor settings are stored in the Machine sub-folder. The Security Configuration Editor tool was first introduced in Windows NT 4.0. In Win2K, we commonly refer to it as Secedit. Its data is stored in a template file in SYSVOL\domain\policies\<GPO GUID>\Machine\Microsoft\Windows NT\Secedit\gpttmpl.inf.

- The Microsoft folder also stores settings used by the Internet Explorer Administration Kit (IEAK). Its data is found in a series of folders under SYSVOL\domain\policies\<GPO GUID>\User\Microsoft\IEAK and contains information that the IEAK would typically create and that an administrator would deploy. Having these settings integrated into Group Policy is a convenient way of enforcing Internet Explorer (IE) configuration and branding settings across multiple machines simultaneously.
- Finally, the RemoteInstall folder, also found under SYSVOL\domain\policies\<GPO GUID>\User\Microsoft, provides settings to a client machine starting up without an OS and is trying to run a Win2K installation from the Remote Installation Services (RIS) server. The settings in the folder are very basic and can be discerned by reviewing the oscfilter.ini file.



Any vendor can create a companyName folder in the User or Machine folder to store file-based configuration data for its applications.

Scripts Folder

The Scripts folder exists in both the User and Machine folders and provides a mechanism for distributing logon, logoff, shutdown, and startup scripts to users and computers in your environment. The Scripts folder in the User folder stores logoff scripts in the Logoff folder and logon scripts in the Logon folder. In the Machine folder, the Scripts folder stores shutdown scripts in the Shutdown folder and startup scripts in the Startup folder.

When you're adding a script to a GPO, if you select the script file from a local file system, such as your administrative workstation, the Group Policy tool doesn't automatically copy the script file to the appropriate Scripts folder in the GPT. Before you add a reference to the script file in the Group Policy tool, you must copy the file manually to the appropriate folder in the GPT.

Once you create references to scripts in a GPO, they automatically appear in a scripts.ini file. This file is created in the Scripts folder; it controls the processing order of the scripts and references their locations in the file-system. It's important to note that just because a script exists in the Logon or Logoff folder, it isn't necessarily processed. The scripts must be assigned in the scripts.ini file, which is populated using the Group Policy snap-in UI. The following is an example of a User scripts.ini file:

```
[Logon]
0CmdLine=\\DCServer\sysvol\domain.tld\Policies\{5DE00B1E-2918-4B82-8CEF-790296F4DFF8}\User\Scripts\Logon\startup.cmd
0Parameters=

[Logoff]
0CmdLine=\\DCServer\sysvol\domain.tld\Policies\{5DE00B1E-2918-4B82-8CEF-790296F4DFF8}\User\Scripts\Logoff\cleanup.cmd
0Parameters=
```

The file above can be interpreted slash for slash. In this example, the first of possibly many logon scripts is referenced by a numeric prefix (that is, 0). The script that is executed is provided

by the `0CmdLine` key. The `0Parameters` key lists any optional command-prompt parameters that you provide for starting the script.

Registry.pol

The `registry.pol` file exists in both User and Machine folders. It stores all of the settings that have been activated in the Administrative Templates section in a GPO. The `Adm` folder mentioned earlier has an impact on this folder. When you open the Group Policy tool, it loads the Administrative Templates (`.adm` files) stored in the `Adm` folder, and these files control the settings that you see in the Administrative Templates section. When you expand on the Administrative Templates folder in the Group Policy tool, the `registry.pol` file is loaded for the appropriate node you're viewing (User or Machine).

The `registry.pol` file is also read by the client OS when a computer starts up and when a user logs on. It's then reread periodically thereafter.

GPO Versioning

GPOs track version numbers each time you make a change to them. This version data is referenced by the code that processes a GPO on the client machine. Namely, it's used to determine whether something in the GPO has changed from the last time it was processed and whether the GPT and GPC are in sync (and can therefore be reliably processed). If the version numbers of the GPT and GPC for a GPO aren't the same, a Win2K system querying that GPO will simply not attempt to process it. In this section, I'm going to explain how GPOs track version numbers.

In the GPT, the version number is kept in a file in `SYSVOL\<domain>\policies\<GUID>` called `gpt.ini`. This is a simple text file like the example below, which shows that the version of the GPT is 65:

```
[General]
Version=65
```

In the case of the GPC, version number information is stored as a property in AD called `versionNumber`. As you'd expect, you can view this property using the `ADSIEdit` tool described earlier in the chapter. (See "Finding GPO Friendly Names from Their GUIDs.")

For a GPO to be considered in sync, the version numbers of the GPC and GPT must be identical on each domain controller in the domain. Replication problems in AD or `SYSVOL` can cause a GPO to get out of sync on some domain controllers.

You can easily see which GPOs in a given domain are in sync on a domain controller by using the Replication Monitor (`Replmon.exe`) tool that comes with the Win2K Support Tools. Right-click a domain controller in `Replmon` and choose `Show Group Policy Object Status` from the context menu. You'll see a listing of all the GPOs in your domain for that domain controller (see Figure 2.12) as well as the current versions of the GPC (listed under `Version`) and GPT (listed under `SysVol Version`). If the GPC and GPT versions of any GPO are out of sync, a check mark will appear in the `Sync` column on the left of the table.

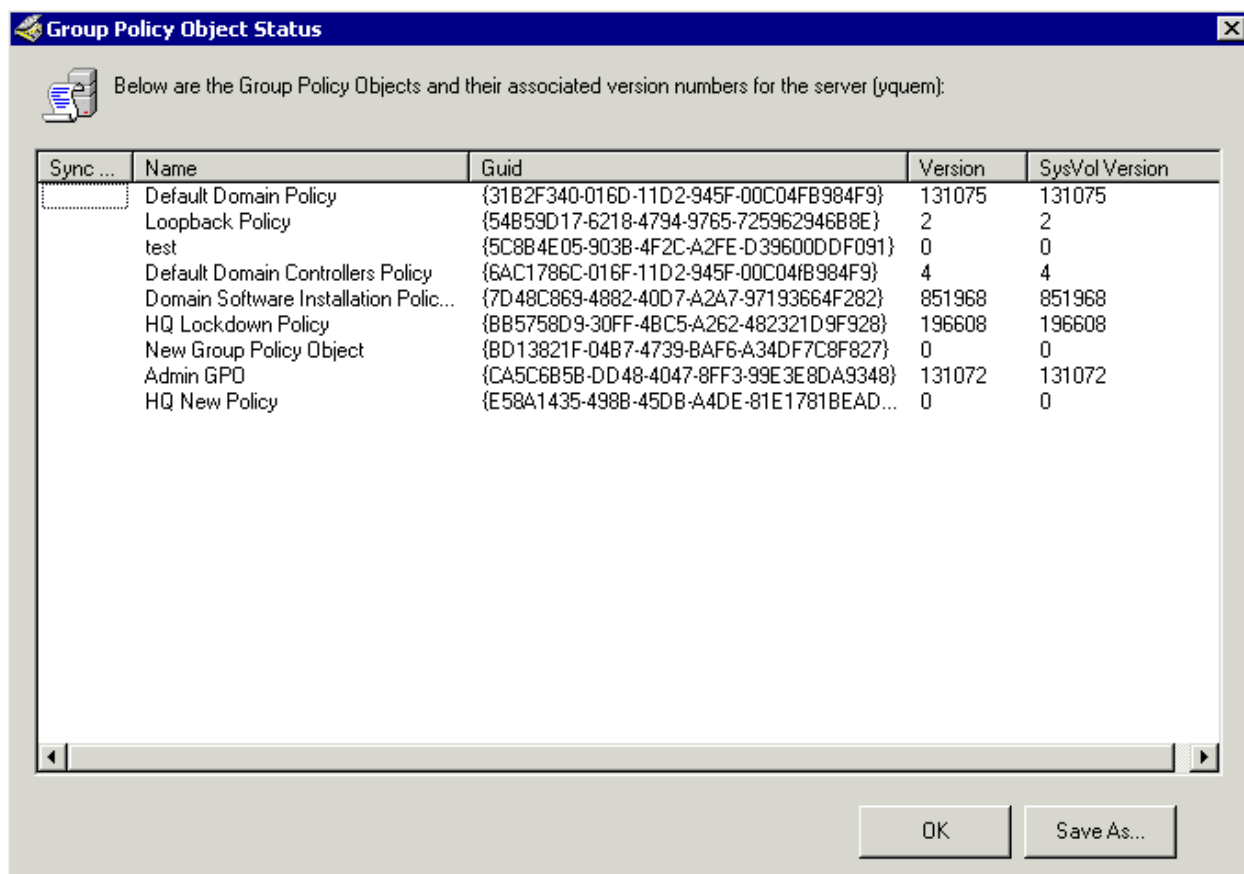


Figure 2.12: Viewing Group Policy replication status using ReplMon.

You'd think that whenever you edited a GPO, the version numbers for the GPT and GPC would be incremented by 1. However, that's not the case, and the actual versioning process isn't the slightest bit intuitive. As you know, there are two sections in a GPO—computer and user. Each of these sections actually maintains its own version information. This is because when a Win2K device is processing a GPO, it needs to know, for computer and user sections, whether there has been a change from the last time the GPO was processed. If only one version number were kept for both computer and user sections, there would be no way to know if a change was made to one section but not the other.

Because a separate version number is kept for each section, only the section of the GPO that has changed will be processed the next time the system reads the GPO. You can view version information for each section in a GPO by opening the GPO in the Group Policy MMC snap-in, selecting the GPO name, right-clicking, then selecting Properties from the context menu. On the General tab, the Revisions property displays values for both *Computer* and *User* (see Figure 2.13).

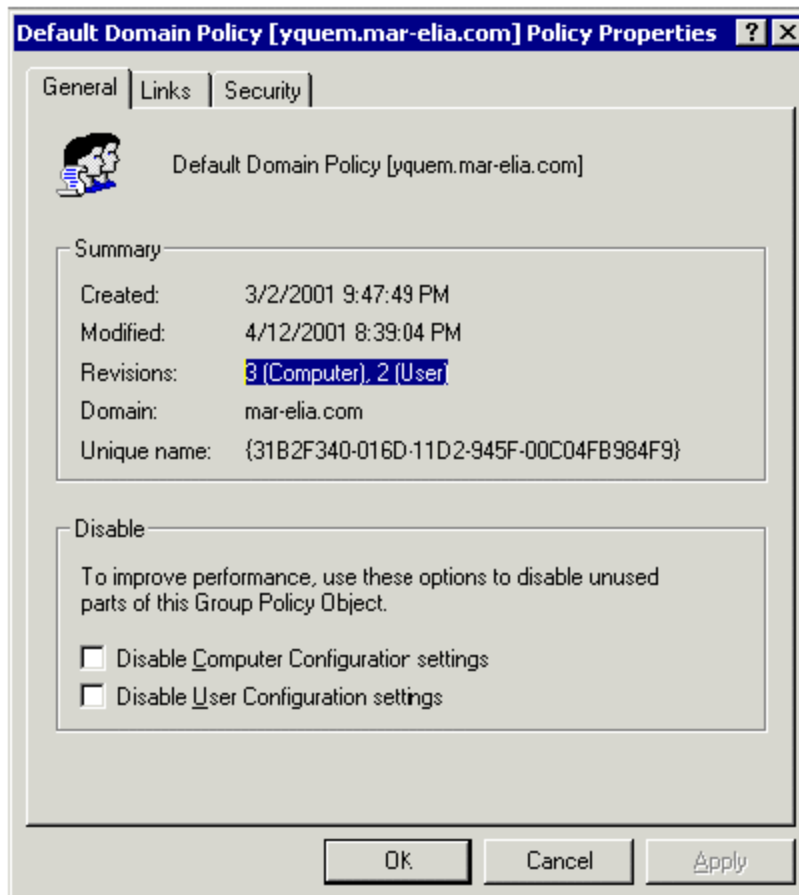


Figure 2.13: Viewing the revision information for a GPO.

These revision numbers ~~can will often~~ be different from the actual version numbers stored in the GPC and GPT, and that's where GPO versioning starts to become counterintuitive. Version numbers for each section of a GPO increment at different rates. Specifically, each change to the computer section of a GPO increments the version number by 1. However, each change to the user section of a GPO increments the version number by 65536! If only one section of the GPO is edited (for example, the computer section), the GPC and GPT version numbers are exactly the same as the number of revisions that that section has undergone. (When you edit a GPO in the MMC snap-in, a change is registered each time you click Apply.) For example, if I've made ten changes to the user section of a GPO and none to the computer section, the Revisions property in the General tab of the GPO will read:

0 (Computer), 10 (User)

The corresponding version numbers for the GPT (in GPT.ini) and GPC (in the versionNumber property) will be 10 x 65536 or 655360. However, when I make the first edit to the computer section on that GPO, ~~the versioning scheme becomes more complicated. From then on, each change I make increments the GPT and GPC version numbers by 65536!~~

~~After I edit the computer section of that GPO,~~ the Revisions property will show:

1 (Computer), 10 (User)

However, the version number stored in gpt.ini and the versionNumber property in the GPT and GPC, respectively, will show 65536146 (655360+10). Each additional change I make to computer section of that that GPO from then on will also increase the version number by 655361, and each additional user section change increases it by 65536.

To reiterate, the revision numbers you see on the GPO's General properties tab is the **number** of changes made to either the user or the computer section of the GPO since it was created. The version number of the GPO is a calculated value that is kept for both the GPC and GPT; it determines whether each is in sync with the other.

Group Policy Processing

Group Policy processing flows as follows: identifying which policies are associated with a user or computer, calling the appropriate client-side extension (CSE) DLLs, reading the policy information from the GPT and GPC, and updating the computer and user settings appropriately. Before we get into that, let's first think about how the system determines which policies are associated with a user or computer.

I discussed in Chapter 1 that Win2K Group Policy can be linked at three different levels: by site, domain, and Organizational Unit (OU). Whenever you link a policy to a site, domain, or OU, the policy's distinguished name (DN) becomes an entry in a special property of the site, domain, or OU called gPLink (see Figure 2.14).

 For more information on gPLink, see Chapter 3, "Creating Group Policies Step by Step."

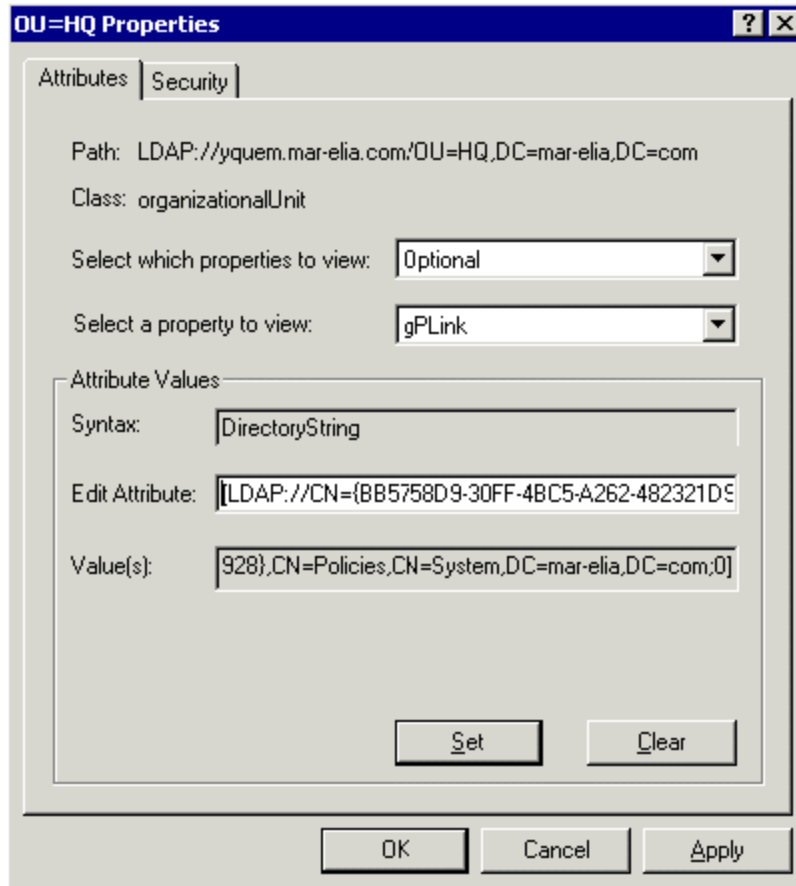


Figure 2.14: Viewing the gPLink property with the DN of the GPC object. The first half of the DN is shown in the Edit Attribute text box, while the second half is shown in the Value(s) text box.

Win2K uses the previously described GetGPOList API to identify which GPOs should be processed on the client. While GetGPOList is being executed on a computer, the computer name and Internet Protocol (IP) address are used to determine the correct site of the GPOs that may be associated with the computer. Similarly, wherever the computer (or user) exists in the domain and in an OU, GetGPOList uses this info to determine whether any additional GPOs apply.

Next, once the list of applicable GPOs is discovered, version information and other GPO options are examined to determine which of the required GPOs must actually be processed during the current cycle. For example, GPOs that haven't changed since the last time they were processed won't be processed again unless the administrator has forced such an operation.

Finally, the required CSE DLLs will process policy for each applicable GPO. These DLLs are installed by default on a Win2K system and registered in the Registry under HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\GPExtensions.

Of course, Group Policy is extensible by any software vendor, so the CSEs registered on a given Win2K system may not be identical to a default Win2K installation. It's important to remember that CSEs do the real work of enforcing GPO policy. They read the settings stored in the GPT and GPC and use that data to apply policy for whichever feature they're responsible for.

Bringing It All Together

As users start their computers and proceed to log on to a Win2K system, Group Policy processing begins. As the system is starting, it obtains an IP address statically or from the Dynamic Host Configuration Protocol (DHCP) server. This IP address allows the system to communicate with a domain controller for authentication and secure channel connection to the domain.

Once a secure channel is created, the client calls GetGPOList. The client identifies its site based on its IP address and site information stored in AD. Having a computer account allows GetGPOList to find the domain the account belongs in as well as the OU. Once the client has obtained the list of GPOs, it prioritizes all of the GPOs, then queries gPCFileSysPath for each GPO so that it can find the GPT.

Once the final computer-specific GPO is processed, the user is presented with the CTRL-ALT-DEL dialog box. When the user types in a user name and password, a login is sent across the secure channel the computer has with a domain controller. The domain controller replies to the user with Kerberos tickets, and the policy processing for the user begins. Once all of the GPOs are processed by the appropriate CSE DLLs, policy processing is complete.

Summary

In this chapter, I began by describing the architecture of Group Policy. The GPO is really composed of a GPC, stored in AD, and a GPT, stored with the SYSVOL share on each domain controller. Later in the chapter, I tied together the details of the GPC, GPT, and settings storage in the form of policy processing.