Knowledge Base

Writing Custom ADM Files for System Policy Editor

PSS ID Number: 225087

Article Last Modified on 6/24/2004

The information in this article applies to:

- Microsoft Windows NT Server 4.0 Terminal Server Edition
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows NT Server 4.0
- Microsoft Windows NT Server, Enterprise Edition 4.0

This article was previously published under Q225087
IMPORTANT: This article contains information about modifying the registry. Before you modify the registry, make sure to back it up and make sure that you understand how to restore the registry if a problem occurs. For information about how to back up, restore, and edit the registry, click the following article number to view the article in the Microsoft Knowledge Base:

256986 Description of the Microsoft Windows Registry

SUMMARY

This article is a guide for writing custom ADM files for use with System Policy Editor. System Policy Editor's interface, comprised of all of the books, check boxes, and text boxes you see when you open a computer or user of group policy, is created with a template. The system policy template, or ADM file, is a simple program that instructs System Policy Editor what books, check boxes, and other input controls to present to the administrator.

For more information about writing an ADM file for group policy, see the white paper ("Implementing Registry-Based Group Policy") at the following Microsoft Web site:

http://www.microsoft.com/WINDOWS2000/techinfo/howitworks/management/rbppaper.asp

MORE INFORMATION

Templates Specify the Registry Change

There are two default templates included when you install Windows NT. They are Common.adm and Winnt.adm. These files use the Class, Keyname, and Valuename variables to allow you to manipulate specific Windows NT policy activities on computers running Windows NT. The following is an explanation of these variables and how they can be used to create your own policy files.

- CLASS (Machine or User)

  The Machine or User keywords are used in the following manner:

  - The Machine keyword controls the entries in the **HKEY_LOCAL_MACHINE** hive.
  - The User keyword controls the entries in the **HKEY_CURRENT_USER** hive.

  CLASS (Machine or User): The CLASS variable specifies the handle registry key where the policy is implemented. There are two handle keys that can be modified by System Policy Editor: **LOCAL_MACHINE** and **CURRENT_USER**. CLASS MACHINE specifies a **LOCAL_MACHINE** policy change and CLASS USER specifies a **CURRENT_USER** policy change.

  When you open the Winnt.adm file, the CLASS MACHINE entry appears at the beginning. Every policy after that is implemented in **HKEY_LOCAL_MACHINE** and appears as a computer policy in the System Policy Editor interface. Below the CLASS MACHINE entry in the Winnt.adm file, the CLASS USER command appears. From this point forward, policies are implemented in **HKEY_CURRENT_USER** and appear as a user or group policy in the System Policy Editor interface.

- KEYNAME

  - Remaining path to change a registry value

KEYNAME: The KEYNAME variable specifies the remaining path to the location where the registry value is added or changed. For example:

```
CATEGORY !!Login_Policies
    POLICY !!LogonBanner

        KEYNAME "Software\Microsoft\Windows NT\CurrentVersion\Winlogon"
    PART !!LogonBanner_Caption
            EDITTEXT
            VALUENAME "LegalNoticeCaption"
            MAXLEN 255
            DEFAULT !!LogonBanner_DefCaption
            END PART
    PART !!LogonBanner_Text
            EDITTEXT
            VALUENAME "LegalNoticeText"
            MAXLEN 255
            DEFAULT !!LogonBanner_DefText
            END PART
    END POLICY
```

The location of this change in the registry is the **HKEY_LOCAL_MACHINE** hive as specific by the CLASS variable. The key location, Software\Microsoft\Windows NT\CurrentVersion\Winlogon, is specified by the KEYNAME variable.

- VALUENAME
  - The Value keyword is created or changed in the registry.
    - REG_SZ (default)
    - NUMERIC = DWORD or BINARY data change

  VALUENAME: The VALUENAME variable specifies the values of the registry keys that are added or changed. In the example above, the logon banner policy requires two registry changes. The value **LegalNoticeCaption** is added or changed and the value **LegalNoticeText** is added or changed in the example above.

  By default, the string value type is REG_SZ. You can override that setting by adding the keyword NUMERIC. All numeric values are typed into a policy template in a decimal format. The value is then stored in the registry as BINARY and DWORD.

  Remember that the data stored in those values is determined by the administrator and whatever that person types into the text boxes in System Policy Editor.

Converting a Registry Change into ADM Keywords

The biggest challenge may be finding a useful registry change that you want to distribute. For example, take the following change that allows you to move the printer spool folder. Remember that before you point the spool to a new folder, that folder must be created. You can then make the following change to the registry:

WARNING: If you use Registry Editor incorrectly, you may cause serious problems that may require you to reinstall your operating system. Microsoft cannot guarantee that you can solve problems that result from using Registry Editor incorrectly. Use Registry Editor at your own risk.

1. Start Registry Editor (Regedt32.exe).
2. Locate the **DefaultSpoolDirectory** value under the following key in the registry:
   ```
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Printers
   ```

   NOTE: The above registry key is one path; it has been wrapped for readability.
3. On the **Edit** menu, click **String**, type `X:\`*`Pathname`* (for example, D:\Printing), and then click **OK**.
4. Quit Registry Editor.

This change to the registry must be converted into a template format so that HKEY becomes CLASS (Machine or User), Key becomes KEYNAME, and Value becomes VALUENAME (followed by NUMERIC if the type is BINARY or DWORD).

Creating an ADM File

ADM files can be created with any text editor.

Create a file following the block-structure syntax of the *.adm language:

- Quotation marks around terms with spaces
- Block structures have a beginning and an end
- Save with *.adm extension (not .txt)

To create a ADM file template:

1. Start NotePad or any text editor to create your template.
2. Create an .ADM file following the rules outlined above. For example:

```
CLASS MACHINE

CATEGORY "How To Test"

    POLICY "Change Spool Directory"

        KEYNAME "System\CurrentControlSet\Control\Print\Printers"
        PART "Spool Directory"
        EDITTEXT
        VALUENAME "DefaultSpoolDirectory"
        END PART

    END POLICY

END CATEGORY
```

Spacing does not matter. If a name has a space in it, it must be surrounded by quotation marks. As a good practice, surround all labels, keynames, and valuenames with quotation marks. Remember to save the file with a .ADM extension.

Loading the Template into System Policy Editor

1. Start the System Policy Editor tool. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **System Policy Editor**. An empty window is displayed. If any icons are displayed, click **Close** on the **File** menu. You must quit the program as you cannot load another .ADM file while any policy files (or the registry) are open for editing or viewing purposes.
2. On the **Options** menu, click **Policy Template** to display the Policy Template Options window. Notice the two .ADM files discussed earlier are already loaded.
3. Click **Add**, go to the location of the file to be loaded, click the file, click **Open**, and then click **OK**.
4. On the **File** menu, click **New Policy** to display the **System Policy Editor** window.
5. Double-click **Default Computer** because the .ADM file uses the CLASS MACHINE variable and keyword, which is for the HKEY_LOCAL_MACHINE registry hive.

The How To Test entry that was created earlier is now displayed. You can now set this entry with a system policy that affects whatever computers you choose.

REFERENCES

Terms and Abbreviations

System Policy Editor (SPE): The tool needed to load the .pol file and apply .adm files.

ADM files: The templates used to change current registry settings.

!!Strings: The text variables in the .adm file.

CATEGORY: The "Books" of Policy Groupings.

POLICY: Used to create the check box entry needed to alter registry values.

PART: Used for the Input Control of the required policy.

The Winnt.adm file creates the interface used in the example above. The following list explains the keywords that create the interface.

!!STRINGS: The are several lines that begin with "!!". The "!!" denotes a string variable and is followed by the variables name, *!!stringname*. The strings are defined at the bottom of the policy template. If you scroll to the bottom of the file, you see a section similar to the following:

```
[strings]
System = Windows NT System
Login_Policies = Logon
LogonBanner_DefText = User created text
```

The variable !!SYSTEM in the body of the template maps to Windows NT System, which is visible in the interface. Having a section of strings allows the descriptions to be replaced easily. This is particularly useful when you translate the files into other languages. Otherwise, you may just want to embed the text in the body of the template.

CATEGORY: This defines a grouping of similar policies. The CATEGORY keyword creates an expandable and collapsible book in the SPE interface, and CATEGORIES can nest within each other. CATEGORY is used solely for organizational purposes, and you can create as many or as few categories as you require.

POLICY: This keyword creates the check box entry which, if selected, creates an instruction for a registry change. If cleared, it creates an instruction for a different registry change (usually a deletion). If the check box is unavailable, it does not create an instruction in the policy file.

PARTs: Some registry changes are accomplished by implementing or clearing a policy. Other registry changes are more complex. For example, to create a logon banner (the policy), an administrator must indicate what goes in the title bar of the window and what goes in the body of the window. These additional pieces of information are gathered by the PARTs variable.

PART: This creates an input control in the lower part of the policy dialog box. A large variety of control windows can be created with this.

CATEGORY, POLICY, and PART create the backbone of the SPE interface. Each is a container. CATEGORY can contain categories and policies. POLICIES can contain parts. Because this is a structured programming language, when you begin one structure or container, you also need to end it.

END - **END CATEGORY, END POLICY, and END PART**: An END command is required to designate the end of each of the respective structures. You do not need to name the category that is ending. It is determined based on the nesting level. Starting and ending each container properly provides for the correct nested structures.

Types of PARTs: There are numerous types of PARTs, which allows you to be creative in designing system policies. For example, the logon banner caption and text are both text boxes. This input control is created with the keyword EDITTEXT, which itself is modified with the optional keywords MAXLEN (to designate the maximum length of input), and DEFAULT (to create a default, suggested data for the administrator's input). So the SPE interface is created with CATEGORIES, POLICIES, and PARTS. An administrator indicates preferences by implementing or clearing policies, and sometimes typing information into input controls called PARTs.

Additional query words: terminalsrv write

Keywords: kbinfo KB225087
Technology: kbNTTermServ400 kbNTTermServSearch kbwin2000AdvServ kbwin2000AdvServSearch kbwin2000Search kbwin2000Serv kbwin2000ServSearch kbWinAdvServSearch kbWinNT400search kbWinNTS400 kbWinNTS400search kbWinNTsearch kbWinNTSEnt400 kbWinNTSEntSearch kbWinNTSsearch