System Policy Templates

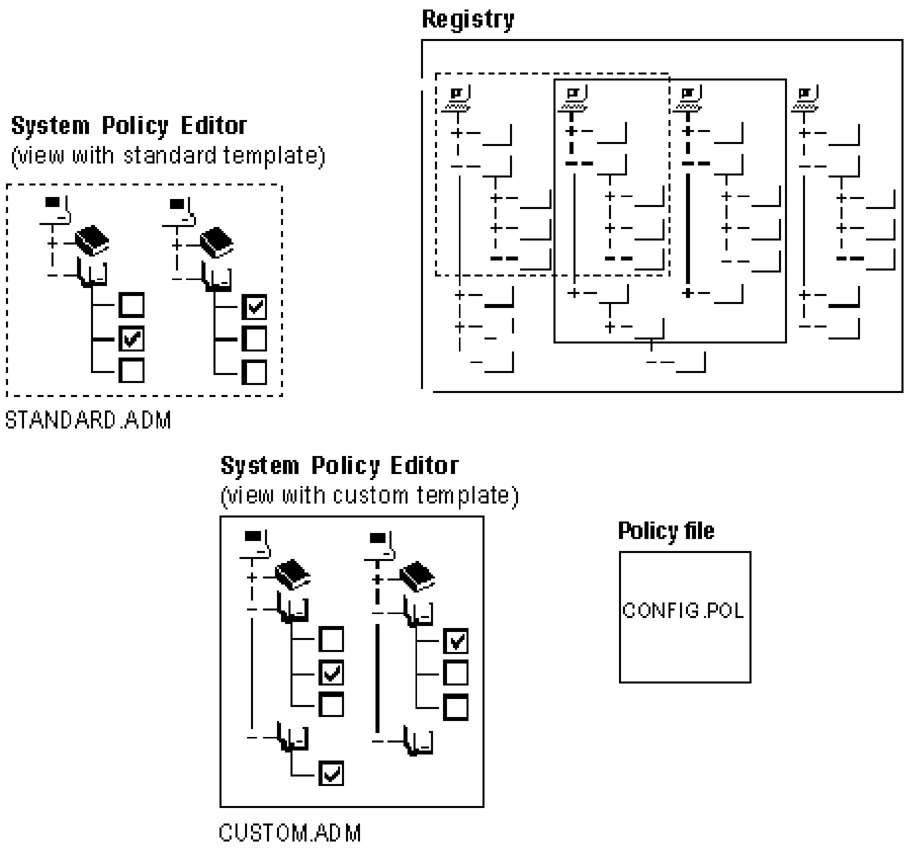
When you run System Policy Editor, Windows 95 opens the default policy template, which contains existing policies that you can enable or modify. A template is a listing of the possible policies that you can use. By default, this template file is named ADMIN.ADM and is stored in the Windows INF directory.

This section describes how you can create custom system policy templates (.ADM files) and switch between multiple templates in System Policy Editor.

For example, it might be helpful to have system policy settings for corporate-specific applications, such as an in-house database, custom front end, or electronic mail package. After a template has been customized, you can then load the template and use it to set values in the Registry.

> **Note** If you want to define system policies for applications, the applications must be able to read the Windows 95 Registry.

Creating your own template is helpful when you want to define a specific set of Registry settings in your system policies, including settings not definable by default through System Policy Editor. As shown in the following illustration, the template defines the policies you can set through System Policy Editor. Changes you make there are reflected in the policy file (shown in the example as



▸ To use a template other than the default template

1.
2. On the Options menu, click Template.
3. Click Open Template, and select an .ADM file to be your template to begin setting system policies.

3. Click Open Template, and select an .ADM file to be your template to begin setting system policies. Click Open.
4. Click Close to return to System Policy Editor.

You can create your own templates that can be read by System Policy Editor. Users can then load the template and use it to set values in the Registry. To create a template, use a text editor such as WordPad to edit or write an .ADM file. You can open the default template named ADMIN.ADM in the Windows INF directory to use as an example.

A template uses several key words, syntaxes, and symbols, as summarized in the following list.

- Class:

```
CLASS category_type
```

- Category:

```
CATEGORY name
  [KEYNAME key_name]
  [... policy definition statements ...]
END CATEGORY
```

- Policy:

```
POLICY name
  [KEYNAME key_name]
  [... part definition statements ...]
END POLICY
```

- Part:

```
PART name part_type
  type-dependent data
  [KEYNAME key_name ]
  VALUENAME value_name
END PART
```

The following table describes the keywords in system policy templates. Following this table are lists of the controls and values that can be defined in templates.

System Policy Template Key Words

Template key word	**Description**
CLASS	Defines the Registry key that can be edited; the value must be USER or MACHINE, corresponding to Hkey_Current_User or Hkey_Local_Machine, respectively.
CATEGORY *name*	Defines a category in System Policy Editor. If a *name* contains spaces, it must be enclosed in quotes. A category statement can appear only once for each category name.
END CATEGORY	Defines the end of a category and all of its policies.
POLICY *name*	Defines a policy within a category. Policy names that contain spaces must be enclosed in quotes.
END POLICY	Defines the end of a policy and all its parts.
PART *name*	Defines one or more controls that can be used to set the values of a policy. Part names that contain spaces must be enclosed in quotes. Policy part types and type-dependent data are described in the following tables.

END PART	Defines the end of the control list.
VALUEON	Specifies the setting to assign to the value when the policy is checked.
VALUEOFF	Specifies the setting to assign to the value when it is not checked.
KEYNAME	Specifies the full path of the Registry key. This is an optional Registry key name to use for the category or policy. If there is a key name specified, it is used by all child categories, policies, and parts, unless they define a key name of their own.
VALUENAME	Defines the Registry value entry name.
VALUE	Specifies the Registry value to set to a *VALUENAME*.
!!	Indicates a string value.
[strings]	Defines a section containing string values.

System Policy Template Part Control Indicators

Part Control Indicator	**Description**
CHECKBOX	Displays a check box. The value is nonzero if is unchecked.
NUMERIC	 accepts a numeric value.
EDITTEXT	Displays an edit field that accepts alphanumeric text.
COMBOBOX	Displays a combo box, which is an edit field plus a drop-down list for suggested values.
TEXT	Displays a line of static (label) text. There is no Registry value associated with this part type.
DROPDOWNLIST	Displays a drop-down list. The user can choose from only one of the entries supplied. The main advantage of a drop-down list is that, based on the user's selection, a number of extra Registry edits can be performed.
LISTBOX	Displays a list box with Add and Remove buttons. This is the only part type that can be used to manage multiple values under one key.

System Policy Template Type-Specific Information

Type-specific modifier	**Description**
CHECKBOX:	
DEFCHECKED	Causes the check box initially to be checked.
VALUEON	 check box. For example: **VALUEON "On"** writes “On” to the Registry.
VALUEOFF	 check box. For example: **VALUEOFF "Off"** writes

	check box. For example: **VALUEOFF "Off"** writes “Off” to the Registry.
ACTIONLISTON	
	“on.”
ACTIONLISTOFF	
	“off.”
NUMERIC:	
DEFAULT *value*	
	statement is not specified, the edit field is initially empty.
MIN *value*	Specifies minimum value for number. Default value is 0.
MAX *value*	Specifies maximum value for number. Default value is 9999.
SPIN *value*	Specifies increments to use for a spin control. Specifying **SPIN 0** removes the spin control; **SPIN 1** is the default.
REQUIRED	
	containing this part to be enabled unless a value has been entered.
TXTCONVERT	Writes values as strings rather than binary values.
EDITTEXT:	
DEFAULT *value*	
	this is not specified, the field is empty initially.
MAXLEN *value*	Specifies the maximum length of the string in the edit field.
REQUIRED	
	containing this part to be enabled unless a value has been entered.
COMBOBOX:	
	Accepts all the key words that EDITTEXT does, plus SUGGESTIONS.

SUGGESTIONS	Begins a list of suggestions to be placed in the drop-down list. Suggestions are separated with spaces and can be enclosed by quotes. The list is terminated with END SUGGESTIONS. For example:
	`SUGGESTIONS`
	`Alaska  Alabama  Mississippi  "New York"`
	`END SUGGESTIONS`

TEXT:	Contains no type-specific data.
DROPDOWNLIST:	
REQUIRED	
	containing this part to be enabled unless a value has been entered.

ITEMLIST	Begins a list of the items in the drop-down list. The end of the list must be terminated by END ITEMLIST. Each item in the list is specified as follows:
	`NAME` *name* `VALUE` *value*
	`[ACTIONLIST` *actionlist*`]`
	`...`

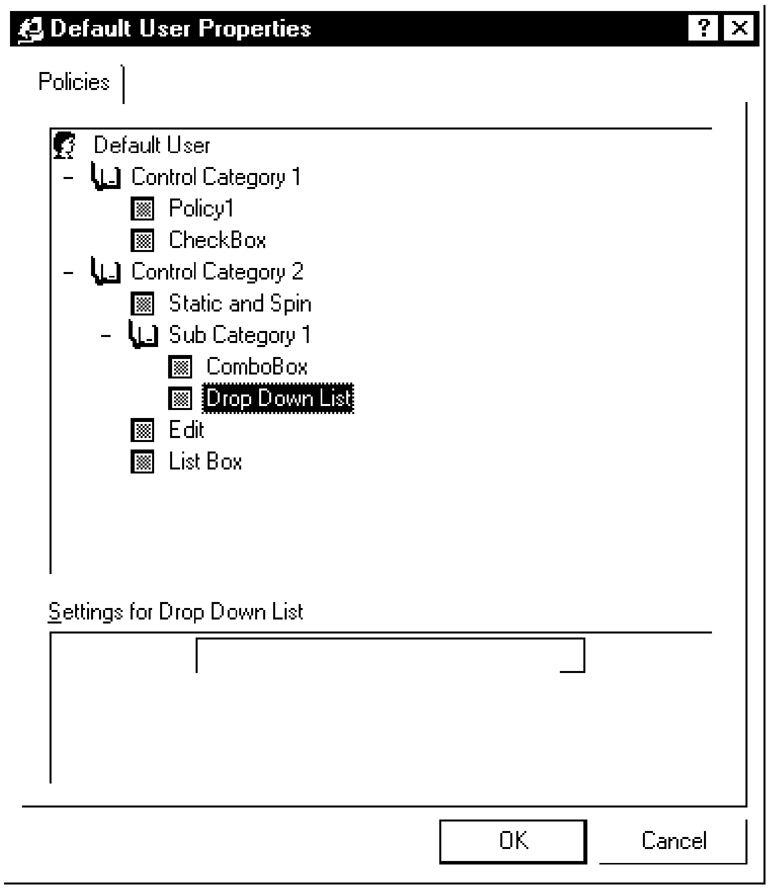
	name is the text to be displayed in the related drop-down list.
	value is the value to be written for the part’s value if this item is selected. Values are assumed to be strings, unless they are preceded by the key word NUMERIC. For example:
	`VALUE "Some value"`
	`VALUE NUMERIC 1`
	If the VALUE key word is followed by the DELETE key word (that is, VALUE DELETE), then this Registry name/value pair will be deleted.
	actionlist is an optional list to be used if this value is selected.
LISTBOX:	
VALUENAME	Cannot be used with the list box type, because there is no single value name associated with this type. By default, only one column appears in the list box, and for each entry a value is created with an identical value name and value data. For instance, the **List Entry** value in the list box would create a value named “List Entry” containing “List Entry” as data.
VALUEPREFIX *prefix*	Defines the prefix to be used in determining value
	naming scheme listed earlier in this table. The prefix can be empty (" "), which will cause the value names to be “1,” “2,” and so on. A prefix of **SomeName** will generate value names “SomeName1,” “SomeName2,” and so on.
EXPLICITVALUE	Causes the user to specify the value data and the value name. The list box shows two columns for each item, one for the name and one for the data. This key word cannot be used with the VALUEPREFIX key word.
ADDITIVE	If specified, values set in the list box are added to whatever values exist in the target Registry. Existing values are not deleted; by default, the content of list boxes will “override” whatever values are set in the
	in the policy file which causes existing values to be deleted before the values set in the policy file are merged.
Strings:	
!!	Indicates a string value. For example:
	`!!StrConst`
[*strings*]	Defines a section of string values; the values are defined in the following format:
	var_name=string value
	For example:
	`StrConst="Control Name"`
Comments	Can be added by preceding the line with a semicolon

Comments	Can be added by preceding the line with a semicolon (;).

The following example shows a template that uses all the types of controls. This sample .ADM file is included with the *Windows 95 Resource Kit* utilities.

```
CLASS USER
CATEGORY "Control Category 1"
KEYNAME KeyName1
  POLICY "Policy1"
    ; actions to take when policy is checked
    ACTIONLISTON
      KEYNAME KeyName1
      VALUENAME Checked1   VALUE "AAA"
      VALUENAME Checked2   VALUE "BBB"
      VALUENAME Checked3   VALUE "CCC"
      KEYNAME KeyName2
      VALUENAME Unchecked1   VALUE DELETE
      VALUENAME Unchecked2   VALUE DELETE
      VALUENAME Unchecked3   VALUE "not checked"
    END ACTIONLISTON
    ; actions to take when policy is unchecked
    ACTIONLISTOFF
      KEYNAME KeyName1
      VALUENAME Checked1   VALUE ""
      VALUENAME Checked2   VALUE ""
      VALUENAME Checked3   VALUE ""
      KEYNAME KeyName2
      VALUENAME Unchecked1   VALUE "AAA"
      VALUENAME Unchecked2   VALUE "BBB"
      VALUENAME Unchecked3   VALUE "CCC"
    END ACTIONLISTOFF
  END POLICY
  POLICY "CheckBox"
    PART "CheckBox1:" CHECKBOX DEFCHECKED
      VALUENAME "CheckBox Control"
      VALUEON "is checked" VALUEOFF "is not checked"
    END PART
  END POLICY
END CATEGORY
CATEGORY "Control Category 2"
KEYNAME KeyName3
  POLICY "Static and Spin"
    PART "Below is a spin control" TEXT
    END PART
    PART "Spin:" NUMERIC SPIN 10 REQUIRED
    MAX 110
    VALUENAME "Spin"
    END PART
  END POLICY
  CATEGORY "Sub Category 1"
  KEYNAME KeyName4
    POLICY "ComboBox"
      PART "Combo:" COMBOBOX
      SUGGESTIONS
        One Two Three Four
      END SUGGESTIONS
      VALUENAME "Combo Control"
      END PART
    END POLICY
    POLICY "Drop Down List"
      PART "DropDown" DROPDOWNLIST
      VALUENAME DropDown REQUIRED
      ITEMLIST
        NAME "Name One" VALUE "Value One"
        ACTIONLIST
          VALUENAME "Value Name 1"   VALUE "Value 1"
          VALUENAME "Value Name 2"   VALUE "Value 2"
        END ACTIONLIST
        NAME "Name Two" VALUE "Value Two"
        ACTIONLIST
          VALUENAME "Value Name 1"   VALUE DELETE
          VALUENAME "Value Name 2"   VALUE DELETE
        END ACTIONLIST
        NAME "Name Three" VALUE NUMERIC 333
        NAME "Name Four" VALUE "Value Four"
      END ITEMLIST
      END Part
    END POLICY
  END CATEGORY
  POLICY "Edit"
    PART "Edit" EDITTEXT
    MAXLEN 10
    VALUENAME Edit
    DEFAULT "Edit Default"
    END Part
  END POLICY
  POLICY "List Box"
  KEYNAME KeyName5
    PART "List Box Control" LISTBOX EXPLICITVALUE
    END PART
  END POLICY
END CATEGORY
```

The following shows the policies created by this sample .ADM file as they appear in System Policy Editor.


Default User Properties

Policies

- Default User
  - Control Category 1
    - Policy1
    - CheckBox
  - Control Category 2
    - Static and Spin
    - Sub Category 1
      - ComboBox
      - Drop Down List
    - Edit
    - List Box

Settings for Drop Down List