

Der folgende Text ist eine vom O'Reilly Verlag nicht autorisierte, aber mit Genehmigung des O'Reilly Verlages angefertigte stellenweise sinngemäße Übersetzung von Mark Heitbrink, des Kapitels 8 aus dem Buch „Windows System Policy Editor“ (ISBN: 1-56592-649-8).

Das englische Original kann als Sample-Chapter auf der Homepage des Verlages gelesen werden.

Original-Quellen: <http://www.oreilly.de/catalog/winsyspe/>
<http://www.oreilly.de/catalog/winsyspe/chapter/ch08.html>

Diverse Begriffe wurden ins deutsche übersetzt, in Klammern wird der englische original Begriff „(fett)“ angezeigt. Im Verlauf der weiteren Übersetzung wird der englische Fachbegriff in dem Moment verwendet, wo sich sonst syntaktische Fehler, Verständnisprobleme oder andere Fehler ergeben würden, bzw. wenn sie in den branchenüblichen Sprachgebrauch eingeflossen sind. Abbildungsüberschriften sind grundsätzlich nicht übersetzt worden.

Ich empfehle jedem bei Problemen mit dieser Übersetzung auf das englische Original zurückzugreifen, da die englische Sprache für diesen technischen Gebrauch doch wesentlich klarer in ihren Ausdrücken und Formulierungen ist.

Des Weiteren sollte sich jeder im Klaren darüber sein, dass diese Übersetzung nicht 100% das Buch oder sein Sample-Chapter in seiner originalen Version ersetzen kann, darf und vor allem will.

O'REILLY™ Online Catalog

PRODUCT INDEX
SEARCH THE CATALOG



Windows System Policy Editor

By Stacey Anderson-Redick
1st Edition June 2000
1-56592-649-8, Order Number: 6498
545 pages, \$34.95

Kapitel 8

Erstellung einer eigenen administrativen Vorlage (im Verlauf des Textes wird der Begriff Template oder Adm-Template verwendet)

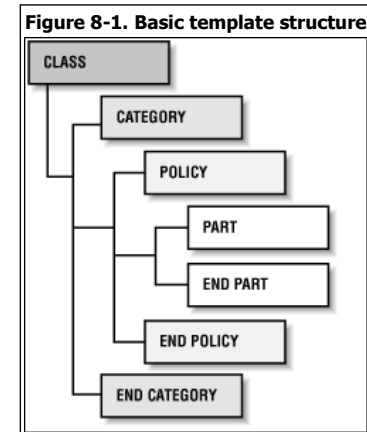
Wohl eine der flexibelsten und vielseitigsten Funktionen des SPE ist die Fähigkeit, eine eigene administrative Vorlage für jedes beliebige Programm zu nutzen, das Einträge in der Windows Registrierung (**Registry**) vornimmt und/oder benötigt. Gerade beim Einsatz verschiedener 3rd Party oder selbstprogrammierter Software ist es nötig eine homogene Verteilung der Einstellungen im Netzwerk für die jeweiligen Workstations zu realisieren. Diese Verteilung kann mit dem SPE und einer eigenen administrativen Vorlage gewährleistet werden.

In diesem Kapitel wird zuerst mal ein Einblick in die die Struktur eines Templates gegeben und es wird in seine kleinste Struktur aufgeteilt. Für jede Komponente wird dann eine vollständige Erklärung und ein Beispiel folgen. Danach werde ich ein spezielles Template erstellen, welches einige häufig verwendete Registry Einträge beinhaltet. Während des Entstehungsprozesses werde ich auf die nötigen Informationen hinweisen und erklären warum und wie man die jeweils richtige Syntax und dazu passende Schlüsselwörter (**Keywords**) auswählt.

Die vier Zonen

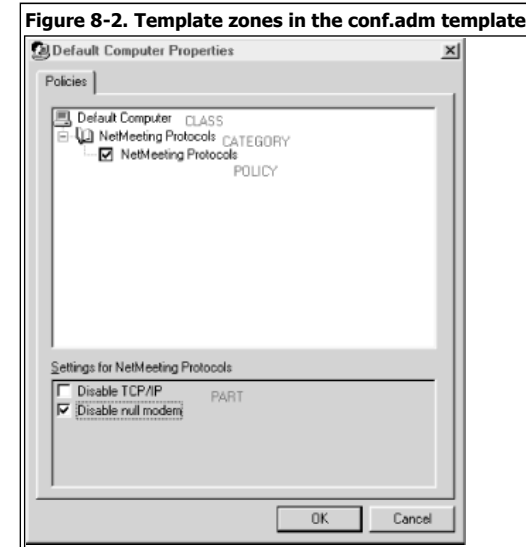
Ein Template enthält 4 verschachtelte Zonen: Klassen (**class**), Kategorien (**category**), Richtlinien (**policy**) und Teile (**part**). Parts liegen innerhalb Policies; Policies sind in Categories integriert und diese wiederum

sind Bestandteil einer einzelnen Class. [Figure 8-1](#) zeigt die Ansicht der grundlegenden Struktur.



Zwei der vier Zonen sind für die grundlegende Struktur des Templates verantwortlich. Klassen trennen Computer und Benutzer Richtlinien (**policies**), wohingegen Kategorien die Policies (Richtlinien) in logische Bereiche aufteilen. Policy und Parts beinhalten letzten Endes die Informationen und Elemente die es erlauben die Registry zu bearbeiten.

Jede Angabe innerhalb dieser einzelnen Zonen wird in einem eigenen Bereich des SPE's dargestellt. Entweder in den Eigenschaften der Computer- oder Benutzerkonfiguration. Siehe dazu [Figure 8-2](#). In den folgenden Abschnitten werde ich detaillierter auf diese Zonen eingehen.



In jeder der vier Zonen ist eine große Anzahl von Schlüsselwörtern verfügbar. Die meisten dieser Keywords sind spezifisch für ihren Bereich. Es gibt drei Keywords, die aber in jeder der vier Zonen eingesetzt werden können:

KEYNAME "value"

Bezieht sich auf eine komplette Pfadangabe in der Registry. Solange diese nicht woanders

definiert ist, wird diese auf alle folgenden Unterbereiche des Templates vererbt. Der Top-Level Schlüssel (**key**) HKLM oder HKCU muss nicht extra angegeben werden da er durch die Angabe/Bestimmung der CLASS definiert ist und diese dem Schlüsselnamen (**KEYNAME**) vorangestellt ist. Zusätzliche KEYNAME Angaben können Unterschlüssel (**subkeys**) eines vorher genannten Schlüssel sein. Als Beispiel, in der CLASS Angabe wird "Software\Microsoft\Windows\CurrentVersion" als Key benutzt, bei der darunter liegenden Policy müsste nur noch "Policies\Network." angegeben werden um folgenden Pfad zu erreichen:
Software\Microsoft\Windows\CurrentVersion\ Policies\Network.

Beispiel für einen möglichen Keyname:
KEYNAME "Control Panel\Colors"

!!string

Bezieht sich auf eine Zeichenkette (**string**) die durch einen Text im Ansichtsfenster des SPE ersetzt wird. Man kann ihn für Überschriften der categories, policies, und parts einsetzen, oder für Hilfe-Texte. Wenn ein string angegeben wird, muss er nur einmalig am Ende des Templates angegeben werden. Der Abschnitt [strings] beinhaltet die dazugehörigen Definitionen. Zum Beispiel:

```
!!Wallpaper1 !!Wallpaper2
[strings]
Wallpaper1="Fächer.bmp"
```

; comment

Kommentare können überall eingefügt werden und werden durch ein vorangestelltes Semikolon gekennzeichnet. Jede Zeile die mit einem Semikolon beginnt wird vom SPE nicht ausgewertet. Ein Beispiel:

```
;template version 2.1
```

Der Einsatz von !! string für Textüberschriften ist nicht zwingend nötig. Texte/Überschriften die ohne !! eingegeben werden erscheinen auch so im SPE. Sobald die Zeile Leerzeichen enthüllt muss diese in Anführungszeichen gesetzt werden. Das gilt für sowohl für Text, als auch für Registry Pfade.

Jede der vier Zonen hat ihre eigene Syntax und eigene Keywords, auf die ich in den folgenden Abschnitten näher eingehen werde.

Class

Es gibt nur zwei gültige Klassen: MACHINE und USER (siehe auch [Table 8-1](#)). Sie definieren die beiden möglichen Registry Schlüssel, die mit dem SPE editiert werden können. Entweder HKCU oder HKLM. HKCU wird unter einem Benutzer oder einer Gruppenrichtlinie angezeigt und ist bestimmt durch class USER. HKLM wird unter der Computer Richtlinie angezeigt und ist durch class MACHINE bestimmt.

Basic Syntax	CLASS <i>class_type</i>
Examples from <i>conf.adm</i>	CLASS MACHINE CLASS USER

Category

Auf der SPE Eigenschaftsseite, die Textüberschriften die neben den „Buch-Icons“ erscheinen sind die Categories (siehe [Figure 8-2](#)). Sie sind integriert sowohl auf der Computer- als auch auf der Benutzer/Gruppen Eigenschaftsseite, abhängig welchem CLASS Eintrag sie folgen. Wenn sie unterhalb Class Machine eingetragen sind erscheinen in den Computereigenschaften, sind sie unterhalb Class User eingetragen so erscheinen sie in den Benutzer-, bzw. Gruppeneigenschaften. Categories werden benutzt um dem Template eine organisatorische Struktur zu geben. Sie können inner- und unterhalb einander

verschachtelt sein um Haupt- und Unterkategorien zu bilden.

Basic Syntax	CATEGORY !! <i>string</i> KEYNAME " <i>value</i> " [<i>policy and part statements go here</i>] END CATEGORY
Example From <i>conf.adm</i>	CATEGORY !!NetMeeting2 KEYNAME "Software\Microsoft\Conferencing" [<i>policy and part statements</i>] END CATEGORY

KEYNAME ist optional. Wenn er nicht hier angegeben wurde, so kann er von der übergeordneten Kategorie übernommen oder in der untergeordneten Policy oder des Parts angegeben werden. Wenn ein KEYNAME angegeben wurde so kann er von allen untergeordneten subcategories, policies and parts verwendet werden. Die Vererbung von Keywords vereinfacht das Template, da es die Wiederholung der oftmals langen Registry Pfade unnötig macht.

Policy

Der Kern des Templates bildet die Policy die innerhalb der Category residiert. Diese sind die geläufigen checkboxes, drei mögliche Einstellungen annehmen können: „weiß“ (**cleared**), „ausgegraut“ (**grayed**) oder „abgehakt“ (**checked**). Die Class und Category Angaben zeigen keinen direkten Effekt in der Registry, aber die Änderung einer Policy Option verändert einen Registry Schlüssel (**key**) oder Wert (**value**). Die einfachste Policy Syntax lautet wie folgt:

```
POLICY !!string
KEYNAME "value"
    [optional keyword(s)]
    [optional PART statement(s)]
END POLICY
```

Wie schon bei der Category, ein KEYNAME ist optional und kann von der vorhergehenden / übergeordneten Policy oder Category geerbt werden.

Eine Policy kann mehrere optionale Keywords enthalten, die gleich aufgelistet werden. Alternativ können sie aber auch innerhalb des Parts angegeben werden.

VALUENAME "value"

Der VALUENAME des Registry Keys. Dieser kann nur weggelassen werden, wenn die Policy weitere Parts enthüllt. Ein VALUENAME mit Leerzeichen muss in Anführungszeichen gesetzt werden. Man sollte sich grundsätzlich angewöhnen die Valuenames in Anführungszeichen zu setzen, um von vornherein eine mögliche Fehlerquelle auszuschließen. Ein Beispiel:
VALUENAME "Background"

VALUE *value*

Der Wert (**value**) des per VALUENAME definierten Registry Eintrag. Wenn kein VALUE angegeben wird, wird dieser, wenn die Policy aktiviert, ist ein DWORD mit dem Wert 1 erstellt und bei Deaktivierung der Policy der Value inkl. Valuename aus der Registry gelöscht. Ist ein value angegeben, wird dieser per default als ein string (REG_SZ) interpretiert. Strings mit Leerzeichen müssen in Anführungszeichen gesetzt sein. Um spezielle DWORD Values zu erstellen, muss dem Value das Wort NUMERIC vorangestellt werden. Um ein VALUENAME/VALUE Registry Paar zu löschen, benutzt man DELETE. For example:

```
VALUENAME "Background" VALUE "128 128 192"
VALUENAME "Platform_Type" VALUE NUMERIC 0
VALUENAME "StaticVxD" VALUE DELETE
```

DEFCHECKED

Die Policy Checkbox grundsätzlich schon ausgegraut (checked) und somit immer aktiv gesetzt. Kann als Bedingung benutzt werden.

VALUEON *value*

Überschreibt den für checked Einstellungen gesetzten Default-Wert des **VALUE** (DWORD value = 1) mit einem anderen angegebenen Wert. Der Wert wird als REG_SZ erstellt, wenn nicht anders, z.B. per **NUMERIC** definiert. Ein Beispiel:

```
VALUEON "1"
VALUEON DELETE
```

VALUEOFF *value*

Überschreibt das Default Verhalten wenn eine Policy deaktiviert wird und setzt den angegebenen Wert ein, anstelle des Value plus Valuename zu löschen. Folgt ansonsten den gleichen Regeln wie **VALUEON**.

ACTIONLISTON

Weitere Registry Keys die mit der einen aktivierten Policy geändert werden. Zum Beispiel, im der *admin.adm* template ist eine Option das Desktop-Schema per Name auszuwählen (e.g., Rainy Day). Die **ACTIONLIST** wird jetzt benutzt, um die Farbe für jedes individuelle Bildelement zu verändern, um das gewünschte Resultat zu erzielen.

ACTIONLISTOFF

Die Veränderungen die bei der Deaktivierung der Policy vorgenommen werden. Wenn die Checkbox cleared (off) ist. Die Syntax für **ACTIONLISTON** und **ACTIONLISTOFF** lautet:

```
ACTIONLISTOFF (or ACTIONLISTON)
KEYWORD value
VALUENAME "value"VALUE value          VALUENAME
"value"VALUE value                    ... .. etc.
END ACTIONLISTOFF (or END ACTIONLISTON)
```

Es kann eine beliebige Anzahl an Policies unterhalb eine Category oder Subkategorie existieren.

TIP:

Für Windows 9x gibt es nur zwei Arten von **VALUE** Einträgen: string oder DWORD. In Windows NT können nur REG_DWORD, REG_SZ, und REG_EXPAND_SZ Registry Values verändert werden. REG_BINARY und REG_MULTI_SZ werden generell nicht von den SPE Templates unterstützt.

Ein Beispiel für eine einfache Richtlinie aus der *conf.adm* sieht so aus:

```
POLICY !!PreventAnswering
PART !!PreventAnswering CHECKBOX
VALUENAME "NoAnswering"
END PART
END POLICY
```

Sie werden bemerken, dass in diesem Beispiel der **VALUENAME** nicht direkt im Anschluss an die Policy Anweisung folgt, wie es im Gegensatz in der **PART** Anweisung zu finden ist. Nur zur Verdeutlichung, wenn ein Value nicht angegeben wird, so wird er per default als DWORD mit dem Wert „1“ erstellt.

Part

Wenn weitere Policies innerhalb dieser aktiviert (enabled/checked) oder deaktiviert (disabled/cleared) werden müssen, so kann dieses in einer Part Anweisung geschehen. Sobald die Policy Checkbox aktiviert ist,

können die Part Optionen im unteren grauen Feld der Benutzer oder Computereigenschaften genutzt werden. Es gibt sieben verschiedene Part Typen/Arten., und mit diesen Typen hat man die Möglichkeit die Auswahl des Werts in verschiedener Form zu präsentieren, es gibt z.B. Text-Felder, Zählwerke, Aufklappmenüs etc. (Siehe hierzu die unterschiedlichen Part Typen für mehr Information.).

Die minimale **PART** Syntax sieht so aus:

```
PART !!string part_type
part_type optional_keyword
KEYNAME "value"
VALUENAME "value" VALUE value
END PART
```

Der string ist die Überschrift/ Kennung die neben dem aktuellen Part Typen im Eigenschaftsfenster erscheint. Die zur Auswahl stehenden möglichen Keywords sind abhängig von dem ausgewählten Part Typ.

KEYNAME ist abermals optional, da er von der übergeordneten Policy oder Category geerbt werden kann. **Value** ist ebenfalls optional, und wenn nicht angegeben, wird wieder per default ein DWORD value = 1 erstellt. Wie auch immer, **Valuename** ist zwingend erforderlich für die meisten Part Typen (Siehe auch hierzu wieder die unterschiedlichen Part Typen für weitergehende Informationen).

Ein Beispiel für eine einfache Richtlinie aus der *conf.adm* sieht so aus:

```
PART !!EnableNullModem CHECKBOX
DEFCHECKED
KEYNAME "Software\Microsoft\Conferencing\Transports\DIRCB"
VALUENAME "Disabled"
END PART
```

In diesem Beispiel ist der Part Typ eine Checkbox, und das optionale Part Typ Keyword ist **DEFCHECKED** (die Checkbox ist per default checked / aktiviert).

TIP:

Eine schlichte Policy (ohne jegliche Part Anweisungen) hat eine Menge mit dem Part Typ **CHECKBOX** gemeinsam, in sofern, dass beide die gleiche Syntax und die wahlweise vorhandenen Keywords benutzen.

Putting It Together – Zusammenfassung

Nimmt man die vier Zonen zusammen (class, category, policy and part), dann sähe der Machine Abschnitt des *conf.adm* Templates wie dargestellt in [Figure 8-2](#) so aus:

```
CLASS MACHINE

CATEGORY !!NetMeeting2
KEYNAME "Software\Microsoft\Conferencing"
POLICY !!NetMeeting2
PART !!EnableTCPIP CHECKBOX
KEYNAME "Software\Microsoft\Conferencing\Transports\TCPIP"
VALUENAME "Disabled"
END PART

PART !!EnableNullModem CHECKBOX
DEFCHECKED
KEYNAME "Software\Microsoft\Conferencing\Transports\DIRCB"
VALUENAME "Disabled"
END PART
END POLICY
END CATEGORY
```

Bei näherer Betrachtung dieses Class Beispiels, kann man feststellen, dass nur eine Class verwendet wird, ebenso nur eine Policy die dafür aber 2 Part enthält. Ebenfalls kann man feststellen, dass der Keyname schon

in der Class angegeben wird und somit in den Parts nicht mehr extra spezifiziert werden muss. Für beide Parts gilt, der Valuename ist angegeben, der Value nicht, daraus lässt sich ersehen, dass der Default Wert: DWORD value = 1 gesetzt wird.

Part Types

Die wahre Flexibilität einer eigenen administrativen Vorlage ergibt sich aus den einzelnen Part Typen, von denen insgesamt sieben zur Auswahl stehen:

- TEXT
- EDITTEXT
- CHECKBOX
- NUMERIC
- DROPDOWNLIST
- COMBOBOX
- LISTBOX

Jeder Part teilt sich die gleichen universellen keywords, wie in der Vier Zonen universal Keyword Liste angegeben. Zusätzlich verfügt jeder Part über eigene individuelle keywords

TEXT

Ein TEXT Part editiert die Registry nicht, er dient lediglich als Kommentar, von dem kein Registry Key oder Value abhängig ist. Er erlaubt es einem zusätzlich erklärenden Text in der Policy oder dem Part zu hinterlegen. Siehe [Figure 8-3](#).

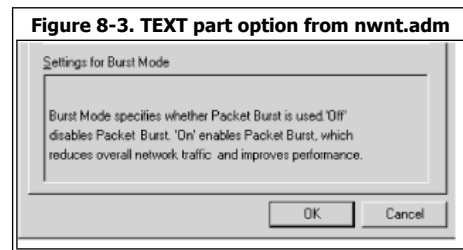


Figure 8-3. TEXT part option from nwnt.adm

Der in [Figure 8-3](#) angegebene TEXT Part wird durch folgende Anweisung erstellt:

```
PART !!BurstMode_TIP1 TEXT END PART
PART !!BurstMode_TIP2 TEXT END PART
PART !!BurstMode_TIP3 TEXT END PART
```

```
[strings]
BurstMode_TIP1="Burst Mode specifies whether Packet Burst is used.'Off'"
BurstMode_TIP2="disables Packet Burst. 'On' enables Packet Burst, which"
BurstMode_TIP3="reduces overall network traffic and improves performance."
```

Der Text wird in drei Zeilen aufgeteilt, damit er nicht die Länge/Größe des Anzeigefensters überschreitet. Im Schnitt können 65 Zeichen pro Zeile verwendet werden, bevor der Text über die Fenstergröße hinausgeht.

EDITTEXT

EDITTEXT erlaubt es ein Eingabefeld für alphanumerischen Text zu erstellen, siehe [Figure 8-4](#).

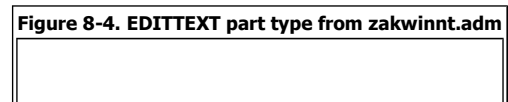
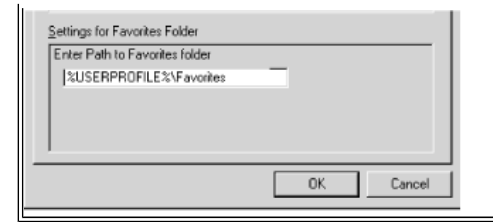


Figure 8-4. EDITTEXT part type from zakwinnt.adm



Diesem Part ist ein Registry Keyname zugeordnet, der entweder angegeben oder von der übergeordneten Policy, bzw. Category geerbt werden kann. Der EDITTEXT Anweisung können noch weitere mögliche Keywords folgen. Siehe [Table 8-1](#).

Table 8-1: EDITTEXT Keywords	
Optional Keywords	Details
VALUENAME "value"	Dieses Keyword ist <i>nicht</i> optional. Siehe auch die Policy Optional Keyword Liste für Details.
VALUE value	Siehe die Policy Optional Keyword Liste für Details.
DEFAULT value	Gibt den anfänglichen Text string. Ohne Angabe ist das Feld per Default leer. Der Value wird in der Registry als string (REG_SZ) gesetzt.
EXPANDABLETEXT	Erlaubt die Verwendung von Umgebungsvariablen. Der Wert wird als REG_EXPAND_SZ in die Registry geschrieben.
MAXLEN value	Gibt die maximale Länge des Textfeldes an. Höchster Wert 255 Zeichen.
REQUIRED	Eine Policy kann nicht aktiviert (checked) werden, ohne das in dem Textfeld eine Eingabe erfolgte. Wenn keine Eingabe erfolgt wird eine Fehlermeldung ausgegeben.
OEMCONVERT	Setzt den ES_OEMCONVERT Stil im Eingabefeld so, dass der eingegebene Text vom ANSI Zeichensatz zu dem auf dem System installierten OEM Zeichensatz konvertiert wird.

Der EDITTEXT Part der in [Figure 8-4](#) gezeigt wird ist durch folgende Anweisungen entstanden:

```
Part !!UserProfiles_FavoritesPath EDITTEXT REQUIRED EXPANDABLETEXT
Default !!UserProfiles_FavoritesPathDefault
ValueName "Favorites"
End Part
```

Sie werden feststellen, dass das Eingabefeld die Eingabe von Umgebungsvariablen durch die Angabe des Keywords EXPANDABLETEXT ermöglicht. Durch das zweite Keyword REQUIRED ist es dem Benutzer nicht erlaubt die Richtlinie ohne Eingabe eines Textes zu verlassen.

TIP:

Man kann ein Adm-Template erstellen, das ein Eingabefeld mit mehr als 255 Zeichen erstellt, es werden aber nur die ersten 255 Zeichen vom SPE in die Registry eingetragen

CHECKBOX

Eine CHECKBOX kann an 2 Formen in einem Template annehmen, checked oder unchecked (an oder aus). Per default, wenn die box checked ist, ist es ein REG_DWORD =f 1. Wenn die box unchecked ist, wird der value auf null gesetzt.

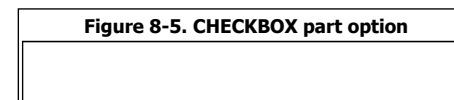
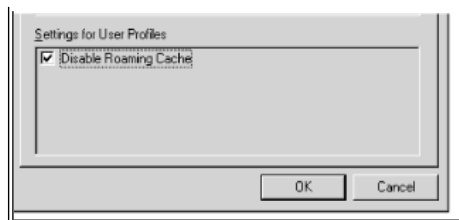


Figure 8-5. CHECKBOX part option



Diesem Part ist ein Registry Keyname zugeordnet, der entweder angegeben oder von der übergeordneten Policy, bzw. Category geerbt werden kann. Der CHECKBOX Part Type benutzt die gleichen optionalen Keywords wie die Basic Policy (Siehe auch die Policy Optional Keyword Liste für weitere Informationen), Siehe [Figure 8-5](#). Folgende Keywords sind möglich:

- VALUENAME
- VALUE
- DEFHECKED
- VALUEON/VALUEOFF
- ACTIONLISTON/ACTIONLISTOFF

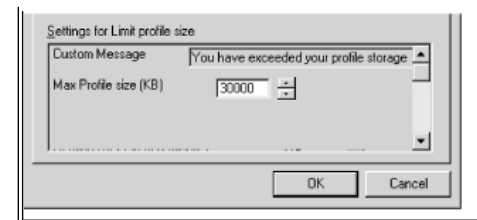
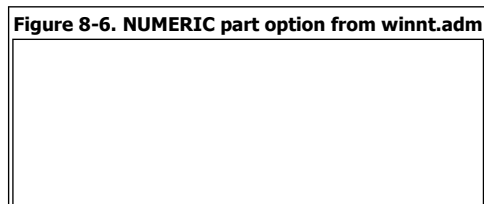
Der CHECKBOX Part der in [Figure 8-5](#) dargestellt ist, ist erstellt durch folgende Template Anweisungen:

```
PART !!DisableCache CHECKBOX
  VALUENAME "UserProfiles"
  VALUEON NUMERIC 0
  VALUEOFF NUMERIC 1
ACTIONLISTON
  KEYNAME "Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\
  Cache\Content"
  VALUENAME "PerUserItem" VALUE NUMERIC 0
END ACTIONLISTON
ACTIONLISTOFF
  KEYNAME "Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\
  Cache\Content"
  VALUENAME "PerUserItem" VALUE NUMERIC 1
END ACTIONLISTOFF
END PART
```

Sie werden feststellen, dass wenn die CHECKBOX aktiviert ist, die ACTIONLISTON Anweisungen weitere Registry Einträge erlaubt. Ebenfalls, erlaubt die ACTIONLISTOFF Anweisung weitere Registry Einträge wenn die Checkbox ist im „cleared“ Zustand ist. Typischerweise werden ACTIONLISTON und ACTIONLISTOFF nicht unabhängig von einander verwendet, selbst wenn ACTIONLISTOFF nichts anderes macht, als die VALUENAME/VALUE Registry Einträge zu löschen, die von ACTIONLISTON erstellt wurden.

NUMERIC

Der NUMERIC Part Typ erlaubt es einen numerischen Value einzutragen, oder eine Zähler (**spin dial**) zu verwenden (die hoch und runter Schaltflächen) um den Wert auszuwählen. Dem spin dial kann optional ein spezifischer Wert angegeben werden und die Schrittweite pro Hoch-/Runtertaste kann festgelegt werden.



Dieser Part ist mit einem Registry Keyname verbunden, der entweder angegeben oder von der übergeordneten Policy, bzw. Category geerbt werden kann. Der NUMERIC Anweisung kann auch ein oder mehrere optionale Keywords enthalten. Siehe [Table 8-2](#).

Optional Keywords	Details
VALUENAME "value"	Dieser Keyword ist <i>nicht</i> optional. Siehe auch die Policy Optional Keyword Liste für Details.
VALUE value	Siehe auch die Policy Optional Keyword Liste für Details. Dieser Value wird als Binary (REG_DWORD) value gelesen
DEFAULT value	Gibt den anfänglichen numerischen Wert an. Als Default, ohne Angabe eines Wertes wird ein leeres Feld angezeigt.
MIN value	Minimum value, default ist 0.
MAX value	Maximum value, default ist 9999.
SPIN value	Die Schrittweite pro Schaltfläche; default ist 1. Spin 0 entfernt die Schaltflächen.
TXTCONVERT	Schreibt die Values als Strings (REG_SZ) und nicht DWORD Values.
REQUIRED	Die Policy kann nicht aktiviert (checked) werden, ohne das in dem Eingabefeld eine Eingabe erfolgte. Wenn keine Eingabe erfolgt wird eine Fehlermeldung ausgegeben.

Der NUMERIC Part der in [Figure 8-6](#) dargestellt ist, ist erstellt durch folgende Template Anweisungen:

```
PART !!ProfileSize NUMERIC REQUIRED SPIN 100
  DEFAULT 30000
  MAX 30000
  MIN 300
  VALUENAME "MaxProfileSize"
END PART
```

Beachten Sie, dass der Spin Dial in 100 Schritten erhöht wird, von einem Minimum von 300 ausgehend, bis zu einem Maximum von 30.000, was auch der Default Value ist.

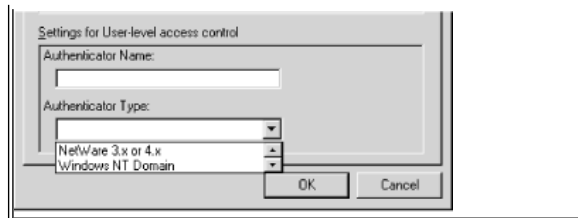
TIP:

Verursacht durch einen Bug in Poledit, Der maximal Wert, den ein Benutzer direkt in ein Numeric Feld eintragen kann, ist 9999 (oder 4 Ziffern), obwohl ein Spin Dial benutzt werden kann, um den Wert jenseits dieser Grenze einzutragen.

DROPDOWNLIST

Eine DROPDOWNLIST (siehe [Figure 8-7](#)) präsentiert dem Anwender einer festen Anzahl von Möglichkeiten, aus denen er auswählen kann. Es ist nicht möglich eine andere freie Auswahl zu treffen.





Dieser Part ist mit einem Registry Keyname verbunden, der entweder angegeben oder von der übergeordneten Policy, bzw. Category geerbt werden kann. Die DROPDOWNLIST Anweisung kann auch ein oder mehrere optionale Keywords enthalten. Siehe [Table 8-3](#).

Table 8-3: DROPDOWNLIST Optional Keywords	
Optional Keywords	Details
VALUENAME "value"	Dieser Keyword ist <i>nicht</i> optional. Siehe auch die Policy Optional Keyword Liste für Details.
VALUE value	Siehe die Policy Optional Keyword Liste für Details.
NOSORT	Sortiert die Werte nicht alphabetisch. Per Default wird sortiert.
REQUIRED	Die Policy kann nicht aktiviert (checked) werden, ohne das in dem Eingabefeld eine Eingabe erfolgte. Wenn keine Eingabe erfolgt wird eine Fehlermeldung ausgegeben.
ITEMLIST	Die Liste der Elemente die in eine drop-down Liste integriert werden Folgende Syntax wird verwendet (Siehe die Policy Optional Keyword Liste für VALUENAME und VALUE Möglichkeiten): ITEMLIST NAME !!string VALUE value ACTIONLIST KEYNAME "value" VALUENAME "value" VALUE value END ACTIONLIST END ITEMLIST

Die ITEMLIST erlaubt jede Auswahl, die in !!string benannt ist und damit jeden verbundenen zusätzlichen Registry Eintrag, die einzeln aufgelistet werden können. In diesem Beispiel, arbeitet die ACTIONLIST hauptsächlich wie CHECKBOX Keyword ACTIONLISTON (Policy Optional Keyword Liste).

Der DROPDOWNLIST Part der in [Figure 8-7](#) dargestellt ist, ist erstellt durch folgende Template Anweisungen:

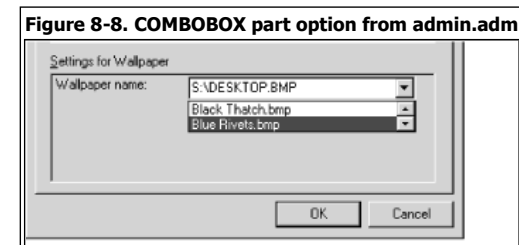
```
PART !!AuthenticatorType DROPDOWNLIST
KEYNAME Security\Provider
VALUENAME "Platform_Type" REQUIRED
ITEMLIST
  NAME !!AT_NetWare      VALUE NUMERIC 3
ACTIONLIST
  KEYNAME System\CurrentControlSet\Services\VxD\NWSP
  VALUENAME "StaticVxD" VALUE nwsp.vxd
  VALUENAME "Start"      VALUE NUMERIC 0
  KEYNAME Security\Provider
  VALUENAME "Address_Book" VALUE nwab32.dll
END ACTIONLIST
NAME !!AT_NTAS          VALUE NUMERIC 2
ACTIONLIST
  KEYNAME System\CurrentControlSet\Services\VxD\MSSP
  VALUENAME "StaticVxD" VALUE mssp.vxd
  VALUENAME "Start"      VALUE NUMERIC 0
  KEYNAME Security\Provider
```

```
VALUENAME "Address_Book" VALUE msab32.dll
END ACTIONLIST
NAME !!AT_NT          VALUE NUMERIC 1
ACTIONLIST
  KEYNAME System\CurrentControlSet\Services\VxD\MSSP
  VALUENAME "StaticVxD" VALUE mssp.vxd
  VALUENAME "Start"      VALUE NUMERIC 0
  KEYNAME Security\Provider
  VALUENAME "Address_Book" VALUE msab32.dll
END ACTIONLIST
END ITEMLIST
END PART
```

Beachten Sie, dass es drei NAME Einträge gibt, daraus resultierend ergeben sich drei Optionen in der drop-down Liste. Der Value der neben jedem Namen angegeben ist, wird dem VALUENAME in Zeile drei zugewiesen (Platform_Type). Mit jedem Name sind zusätzliche Registryeinträge in der ACTIONLIST verbunden

COMBOBOX

Die COMBOBOX (siehe [Figure 8-8](#)) ist im Wesentlichen eine Kombination aus EDITTEXT und DROPDOWNLIST. Wie bei einem EDITTEXT Feld kann man alphanumerischen Text einsetzen oder aus einer Liste, wie bei der DROPDOWNLIST, eine Auswahl treffen



Dieser Part ist mit einem Registry Keyname verbunden, der entweder angegeben oder von der übergeordneten Policy, bzw. Category geerbt werden kann. Die COMBOBOX Anweisung kann auch ein oder mehrere optionale Keywords enthalten. Siehe [Table 8-4](#)

Table 8-4: COMBOBOX Optional Keywords	
Optional Keywords	Details
VALUENAME "value"	Dieser Keyword ist <i>nicht</i> optional. Siehe auch die Policy Optional Keyword Liste für Details.
VALUE value	Siehe die Policy Optional Keyword Liste für Details.
DEFAULT value	Gibt den Anfangswert an. Als Default, ohne Angabe eines Wertes wird ein leeres Feld angezeigt.
EXPANDABLETEXT	Erlaubt die Verwendung von Umgebungsvariablen. Der Wert wird als REG_EXPAND_SZ in die Registry geschrieben.
MAXLEN value	Maximum length of the text.
REQUIRED	Die Policy kann nicht aktiviert (checked) werden, ohne das in dem Eingabefeld eine Eingabe erfolgte. Wenn keine Eingabe erfolgt wird eine Fehlermeldung ausgegeben.
OEMCONVERT	Setzt den ES_OEMCONVERT Stil im Eingabefeld so, dass der eingegebene Text vom ANSI Zeichensatz zu dem auf dem System installierten OEM Zeichensatz konvertiert wird.
NOSORT	Sortiert die Werte nicht alphabetisch. Per Default wird sortiert.

SUGGESTIONS	Führt Vorschläge an, die in der Drop-Down Box integriert werden können. Jeder Vorschlag wird durch ein Leerzeichen vom anderen getrennt. Vorschläge, die ein Leerzeichen enthalten werden in Anführungszeichen gesetzt: SUGGESTIONS !!string1 !!string2 !!string3 END SUGGESTIONS
-------------	--

Der COMBOBOX Part der in [Figure 8-8](#) dargestellt ist, wird durch folgende Template Anweisungen erstellt.

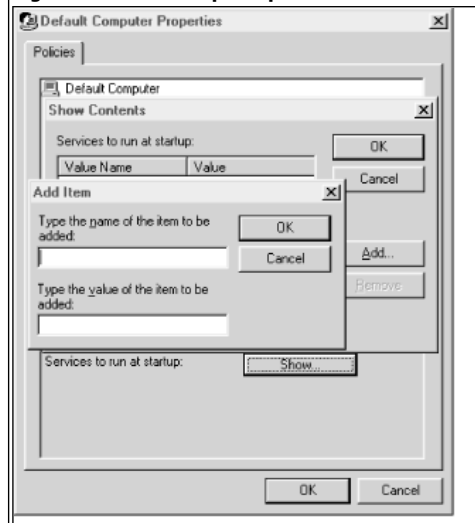
```
PART !!WallpaperName COMBOBOX REQUIRED
  SUGGESTIONS
    !!Wallpaper1 !!Wallpaper2 !!Wallpaper3 !!Wallpaper4 !!Wallpaper5
    !!Wallpaper6 !!Wallpaper7 !!Wallpaper8 !!Wallpaper9 !!Wallpaper10
  END SUGGESTIONS
  VALUENAME "Wallpaper"
END PART
```

Sie werden feststellen, dass in diesem Beispiel der VALUENAME nach der Suggestions-Liste aufgeführt ist. Es ist nicht zwingend nötig den Valuenamen in der dem Part Type folgenden Zeile anzugeben, wobei es einfacher ist einen Fehler im Script zu finden, wenn man immer die gleiche Reihenfolge der Keywords verwendet.

LISTBOX

Die LISTBOX ist der einzige PART Type, der innerhalb eines Registry Keys mehrere unterschiedliche Values sowohl hinzuzufügen, als auch zu löschen kann. Ein Beispiel für so eine Art von Registry Key ([Figure 8-9](#)) ist der Eintrag, der die Programme auflistet, die zum Startzeitpunkt von Windows geladen werden.

Figure 8-9. LISTBOX part option from admin.adm



Dieser Part ist mit einem Registry Keyname verbunden, der entweder angegeben oder von der übergeordneten Policy, bzw. Category geerbt werden kann. Wie auch immer, im Gegensatz zu den meisten anderen PART Typen werden VALUENAME und VALUE nicht unterstützt. Es kann kein einzelner Registry Valuenamen oder Value möglich sein, für einen Wert, der mehrere Einträge erlaubt.

Die LISTBOX Anweisung kann einen oder mehrere optionale Keywords enthalten. [Table 8-5](#).

Table 8-5: LISTBOX Optional Keywords

Optional Keywords	Details
VALUEPREFIX "value"	Das Präfix ist eine ansteigende Nummer um den VALUENAME zu erstellen. Zum Beispiel: Das Präfix "Directory" wird zu dem Datenwert "Directory1", "Directory2", und so weiter. Durch leere Anführungszeichen "" werden nur die Nummern generiert. Dieser Keyword kann nicht mit EXPLICITVALUE kombiniert werden.
EXPLICITVALUE	Hiermit werden zwei Reihen in der Show Contents Listbox angezeigt und erstellt dem Benutzer die Möglichkeit sowohl den Valuenamen als auch den Value Data einzugeben. Per Default entspricht der Valuenamen dem Value Data. Dieser Keyword kann nicht mit VALUEPREFIX kombiniert werden.
ADDITIVE	Hier eingetragene Values werden den schon existierenden Registry Einträgen hinzugefügt. Normalerweise würde sonst der Value den vorhandenen Wert überschreiben. ACHTUNG: Dieser Keyword ist in ADM Templates für NT4 nicht unterstützt. Die Werte werden weiterhin überschrieben.

Der LISTBOX Part, der in [Figure 8-9](#) dargestellt ist, ist erstellt durch folgende Template Anweisungen:

```
PART !!RunListbox LISTBOX EXPLICITVALUE
END PART
```

Obwohl LISTBOX eine der komplexesten Schnittstellen bietet, so hat es doch eine der einfachsten Syntax.

Part Type Keyword Zusammenfassung

[Table 8-6](#) kann als Kurzreferenz dienen, um zu überprüfen, welcher Part Typ, welches jeweilige Keyword unterstützt. Wird ein Keyword unterstützt, so ist es in der Tabelle mit einem „X“ gekennzeichnet.

Table 8-6: Part Type Keyword Summary

Keyword	CHECK BOX	COMBO BOX	DROPDOWNLIST	LISTBOX	NUMERIC	EDITTEXT	TEXT
!!	X	X	X	X	X	X	X
KEYNAME	X	X	X	X	X	X	
VALUENAME	X	X	X		X	X	
VALUE	X	X	X		X	X	
DEFCHECKED	X						
VALUEON/ VALUEOFF	X						
ACTIONLISTON/ ACTIONLISTOFF	X						
DEFAULT		X			X	X	
MIN					X		
MAX					X		
SPIN					X		
Txtconvert					X		
Required		X	X		X	X	
Expandabletext		X				X	
Maxlen		X				X	
Oemconvert		X				X	
Nosort		X	X				

Suggestions		X					
ItemList			X				
Actionlist			X				
Valueprefix				X			
Explicitvalue				X			
Additive				X			

#if version

Zur Unterstützung der Abwärtskompatibilität, kann man einen logischen Test durchführen, ob das Template mit der speziellen Version von *poedit.exe* unterstützt wird. Die original Versionen von Poedit unterstützen die Keywords *NOSORT* oder *EXPANDABLETEXT* nicht. Wenn sie wollen, dass ihr Selbsterstelltes Template für eine möglichst Große Anzahl von Umgebungen einsetzbar ist, kann man mit dem Test die Version von überprüfen lassen, um Fehlermeldungen zu unterdrücken.

Die Poedit Version die mit Windows 95 ausgeliefert wurde ist Version 1. Die Versionen die mit Windows 98, NT4 Service Packs und Office 2000 kamen haben die Version 2.

Die Syntax ist sehr einfach:

```
#if version operator poedit_version
part_keyword
#endif
```

Es gibt 5 Operatoren:

```
< less than
> greater than
<= less than or equal to
>= greater than or equal to
== equal to
```

Wenn die Anweisung als false gestestet wird, dann wird der Prozess abgebrochen und an der Stelle mit der `endif` Anweisung weitergeführt.

Ein Beispiel eine Richtlinie in der ein Versionscheck durchgeführt wird, bietet die *off97NT4.adm*:

```
PART !!MyDocText EDITTEXT
  VALUENAME "Personal"
  #if VERSION > 1
    EXPANDABLETEXT
  #endif
  REQUIRED
END PART
```

Diese Beispiel stellt eine Abfrage ob die verwendete Version von Poedit größer als 1 ist und im Falle das es so ist wird die Variable expandiert.

Erstellung eines Templates

Beobachten wir nun einmal den Entstehungsprozess eines Windows 9x Templates welches einige häufige Registry Änderungen beinhaltet. Im Folgenden werden diese Einstellungen abgehandelt:

- Änderung der Windows Lizenzierungsdaten
- Änderung des Teps des Tages
- Änderung der System- Start und Endklänge
- Änderung der Menügeschwindigkeit
- Ausblenden der Laufwerke im Arbeitsplätze

WARNUNG:

Das Ausblenden spezieller einzelner Laufwerke steht im Konflikt mit der "Ausblenden aller Laufwerke" –Richtlinie, die in der *common.adm* und *admin.adm* Templates vorhanden ist. Diese Richtlinie darf nicht aktiviert werden, wenn spezielle Laufwerke ausgeblendet werden sollen.

Erstellung der Policies und Parts

Für die Erstellung der Policy und Parts braucht man im Vorfeld die Informationen, welche Registry Werte sich verändern sollen, im speziellen:

- Der Class Type (Machine oder User)
- Der spezielle Registry Schlüssel
- Den Registry Key *VALUENAME*
- Die Registry Key *VALUE(s)*
- Den Value Typ (string or *DWORD*)

Verfolgen von Registry Änderungen

Manchmal findet man eine Dokumentation des Registry Key, den man verändern will, manchmal nicht. Wenn man sich nicht sicher ist welche Registry Keys betroffen sind, wenn man Programm-Optionen anpasst, sollt man ein Programm benutzen, welches die Registry Änderungen aufzeichnet. Eines dieser Programme ist xSnapShot von JAS Concepts (<http://www.jasconcepts.com>). Dieses Utility ähnelt stark der *sysdiff.exe*, die nur mit Windows NT4 benutzt werden kann. xSnapShot funktioniert sowohl unter 9x als auch NT, und erlaubt dem Anwender die genaue Einstellung, welche Registry Schlüssel durchsucht werden sollen und welche ausgeschlossen werden sollen. Ein Snapshot der Registry kann vor und nach der Installation von Programmen erstellt werden. Diese beiden Snapshots könne miteinander verglichen werden und eine INF Datei mit allen Registry Änderungen wird generiert. Diese Information kann dann benutzt werden, um eine eigenes ADM Template zu erstellen.

Die folgende Tabelle gibt im Einzelnen an, welche Keys nötig sind, um die erwähnten Registry Änderungen durchzuführen. Wenn man erst die nötigen Informationen hat, ist es nötig sich zu entscheiden, wie man den Eintrag des Benutzers kontrollieren möchte. Mit anderen Worten, welche Art von PART Typen ist am besten geeignet, oder würde auch eine einfache Checkbox genügen?

Windows Lizenzierungseinträge

Im "Allgemein" Reiter der Arbeitsplatz-Eigenschaften wird der Name und die Organisation des Registrierten Benutzers angezeigt. Wir möchten die Arbeitsstationen von einer Abteilung in die andere verlegen, oder wir möchten einfach nur eine einheitliche Bezeichnung in der ganzen Firma. Die Informationen für diese Umstellung ist in [Figure 8-7](#) angezeigt.

Class	Machine (HKLM)
Registry Key	Software\Microsoft\Windows\CurrentVersion
Valuename Owner	RegisteredOwner
Organization	RegisteredOrganization
Value	Entered by user
Value Type	String (REG_SZ)

Der Benutzer wird einen variablen Text eingeben, somit kann eine einfache Checkbox nicht ausreichend sein und man muss zusätzlich einen Part verwenden. Die Part Typen, die die Eingabe eines Textes erlauben sind *EDITTEXT*, *COMBOBOX* und *LISTBOX*. Da es keine standardisierten Einträge für Besitzer und Organisationen gibt, wäre die *COMBOBOX* keine gute Wahl. *LISTBOX* (welche die Benutzung eines *VALUENAME* nicht unterstützt) wäre ebenfalls keine gute Wahl, da es um festgelegte Valuenames geht. Somit bleibt *EDITTEXT*. Aus den *EDITTEXT* Optionen ([Table 8-1](#)), ist ersichtlich, dass nur *REQUIRED* eine passende Lösung bietet, da ein leeres Textfeld nicht unseren Ansprüchen genügen würde.

Es ist immer hilfreich einen erklärenden Text mit einzufügen, in unserem speziellen Fall, könnte man einfach erklären, wo sich die Änderungen auswirken und wo sie sichtbar wären (Arbeitsplatz -Eigenschaften). Der Abschnitt des Template könnte so wie folgt aussehen:

```
KEYNAME "Software\Microsoft\Windows\CurrentVersion"
POLICY !!Changereg
  PART !!Changereg1 EDITTEXT REQUIRED
  VALUENAME "RegisteredOwner"
  END PART
  PART !!changereg2 EDITTEXT REQUIRED
  VALUENAME "RegisteredOrganization"
  END PART
  PART !!changeregtext1 TEXT END PART
  PART !!changeregtext2 TEXT END PART
END POLICY
```

Diese Anweisung erstellt eine Policy mit vier Parts. Ein Part ändert den Besitzernamen, während der zweite Part den Namen der Organisation ändert. Die zwei verbleibenden Part fügen den erklärenden Text hinzu. Durch die Aufteilung des Textes in zwei Parts verhindert man, dass in der Ansicht die Zeilen aus dem sichtbaren Bereich herauslaufen.

Tipp des Tages

Man kann die die Tipps, die jedes Mal wenn Windows95 startet anzeigt, so ändern, dass sie auf die eigene Organisation angepasst sind. Es geht um die Tipps, die mit dem Willkommensbildschirm erscheinen und unter "Wussten Sie schon " angezeigt werden. Der Willkommensbildschirm von Windows 98 ist nicht kompatibel mit diesen Einstellungen, deswegen funktionieren die Einstellungen nur mit Windows 95. Abgesehen davon, ist es immer noch ein gutes Beispiel, dass man sich anschauen sollte. Die benötigten Informationen für dieses Template sind in [Table 8-10](#) zusammengefasst.

Class	Machine (HKLM)
Registry key	Software\Microsoft\Windows\CurrentVersion\Explorer\Tips
Valuename	Variable, tips are numbered sequentially
Value	Tip is entered by user
Value Type	String (REG_SZ)

Der Benutzer wird einen variablen Text eingeben. Nochmals, eine einfache Policy Checkbox wird nicht ausreichen. Die Part Typen, die eine Texteingabe erlauben sind `EDITTEXT`, `COMBOBOX` und `LISTBOX`. `COMBOBOX` ist keine gute Wahl da die Tipps im Ermessen des Users liegen und es keinen voraussichtlichen Vorschläge gibt. `EDITTEXT` ist keine gute Lösung, da es keine einzelnen Valuenames gibt. Der einzige Part Type der multiple Valuenames erlaubt ist `LISTBOX`. Für `LISTBOX` Optionen siehe [Table 8-5](#). Ich muss `VALUEPREFIX` benutzen, da die Tipps einfach durchnummeriert werden. `ADDITIVE` könnte benutzt werden wenn man die Tipps zu den schon vorhandenen Tipps hinzufügen wollte. `EXPLICITVALUE` könnte benutzt werden, wenn man dem Anwender erlauben wollte eine feste Zahl zuzuweisen. In diesem Falle, könnte der Value die Tipp Zahl bilden, während der Valuename den Tipp selber darstellt. Der Abschnitt des Templates könnte dann so aussehen:

```
POLICY !!tips
  KEYNAME "Software\Microsoft\Windows\CurrentVersion\Explorer\Tips"
  PART !!tips LISTBOX
  END PART
END POLICY
```

Es sollte erwähnt werden, dass die Tipps nicht angezeigt werden, wenn die Option "Willkommensbildschirm anzeigen" ausgeschaltet ist. Da der zuständige Registry Key ein hexadezimal Wert ist (`REG_BINARY`), ist es nicht möglich diesen mit einem ADM Template zu verändern. Wie auch immer, dafür kann eine `.reg` Datei im Loginscript verwendet werden (siehe [Chapter 4, Building the Policy File](#)) um diesen Wert in der Benutzer

Registry zu verändern. Die REG Datei würde folgende Zeilen enthalten:

```
[HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Tips]
"Show"=hex:01,00,00,00
```

Ein hexadezimaler Wert von `00,00,00,00` schaltet die Tipps aus. Alternativ kann man das `admin.adm` Template benutzen um das Programm `welcome.exe` im Autostart zu platzieren. Dieser Vorgang würde die gleichen Änderungen in der Registry herbeiführen.

Sound Ereignisse

Sound Dateien (WAV Dateien) von der lokalen Festplatte oder von einem Netzwerklaufwerk können benutzt werden für die System Start und End Klänge. Die benötigten Informationen für dieses Template sind in [Table 8-9](#) zusammengefasst.

Class	Machine (HKLM)
Registry Key	AppEvents\Schemes\Apps\.Default\SystemStart\.Current
Start	AppEvents\Schemes\Apps\.Default\SystemExit\.Current
Shutdown	
Valuename	[Default]
Value	WAV filename is entered by user
Value Type	String (REG_SZ)

Der Benutzer wird einen variablen Text eingeben. Eine einfache Policy Checkbox wird nicht ausreichen. Die Part Typen, die eine Texteingabe erlauben sind `EDITTEXT`, `COMBOBOX` und `LISTBOX`. `LISTBOX` ist keine Wahl, da es sich um festgelegt Valuenames handelt. Ein Lösung könnte es sein die von Windows vorgegebenen Systemklänge (WAV Dateien) zu nutzen, deswegen könnte `COMBOBOX` ein mögliche Lösung bieten. Ebenfalls könnte man `EDITTEXT` auswählen, wenn man weitere Vorschläge integrieren wollte, oder man sich nicht sicher ist, welche Systemklänge auf der lokalen Festplatte zur Verfügung stehen könnten. Da es das einfachste ist, werde ich `EDITTEXT` verwenden. Von unseren `EDITTEXT` Optionen ([Table 8-3](#)), ist keine wirklich nötig. Zum Schluss kann ich mich entscheiden, ob ich die Optionen als 2 Richtlinien mit jeweils einem Part darstelle, oder als eine Richtlinie mit 2 Parts. Das ist eine individuell getroffene Vorliebe; Ich wähle die Möglichkeit mit 2 Richtlinien. Der Abschnitt des Templates könnte dann so aussehen:

```
POLICY !!changestart
  KEYNAME "AppEvents\Schemes\Apps\.Default\SystemStart\.Current"
  PART !!wavfile EDITTEXT
  VALUENAME ""
  END PART
END POLICY

POLICY !!changeexit
  KEYNAME "AppEvents\Schemes\Apps\.Default\SystemExit\.Current"
  PART !!wavfile EDITTEXT
  VALUENAME ""
  END PART
END POLICY
```

Zur Beachtung, obwohl der `VALUENAME` in der Registry als `[Default]` angezeigt wird, so braucht man ihn doch nicht als `VALUENAME` angeben. Wenn man es täte, würde das Template einen zweiten Eintrag in der Registry erstellen mit dem Namen `[Default]` anstelle den wirklichen `VALUENAME` zu editieren. Wenn man einen als `[Default]` dargestellten `VALUE` editieren möchte, so muss man einen `VALUENAME` einsetzen, der nur aus leeren Anführungszeichen besteht.

Menu Geschwindigkeit

Auf dem Windows Desktop wird jedes assoziierte Submenü automatisch geöffnet, wenn der Mauszeiger sich

darüber bewegt. Es ist sowohl möglich dieses Verhalten zu beschleunigen, sowie es zu verlangsamen, sodass man den Eindruck erhält, das der ganze Vorgang stoppen würde. Im diesem Falle wäre ein Maus-Klick nötig, um das Submenü zu öffnen. Die benötigten Informationen für dieses Template sind in [Table 8-10](#) zusammengefasst.

Class	User (HKCU)
Registry key	Control Panel\Desktop
Valuename	MenuShowDelay
Value	0 to 10,000 milliseconds (Default 400)
Value Type	String (REG_SZ)

Der Value ist eine Zahl zwischen 0 bis 10,000. Der Part Type der am besten numerische Einträge anzeigt ist `NUMERIC`. Wenn man sich den Value Type ansieht, wird man feststellen, dass es kein `DWORD` ist, sondern ein `String`. Es ist nötig das optionale Keyword `NUMERIC` (siehe [Table 8-2](#)) mit `TXTCONVERT` zu verbinden, um den Wert zu einem String zu konvertieren. Die von Windows per Default verwendete Menügeschwindigkeit beträgt 400 Millisekunden. Aus diesem Grunde verwende ich die `DEFAULT` Option und verberge hier diesen Wert. Das Hinzufügen eines Spin Dail macht es dem Anwender einfacher den Wert einzutragen und da sich Schrittweiten kleiner als 500 Millisekunden optisch kaum eine Wirkung zeigen werde ich den Spin auf diese minimale Schrittweite einstellen. Zum Schluss verberge ich für den Wert ein Minimum von 0 und ein Maximum von 10,000.

Erklärender Text sollte hinzugefügt werden, da die meisten Anwender mit diesen Einstellungen nicht vertraut sind. Es sollte erklärt werden, dass eine niedrigerer Wert einen Anstieg der Menügeschwindigkeit zur Folge hat. Der Abschnitt des Templates könnte wie folgt aussehen:

```
POLICY !!menudelay
  KEYNAME "Control Panel\Desktop"
  PART !!menudelay NUMERIC SPIN 500
    VALUENAME "MenuShowDelay"
    TXTCONVERT
    DEFAULT 400
    MIN 0
    MAX 10000
  END PART
  PART !!MENUTEXT TEXT END PART
END POLICY
```

Ausblenden der Laufwerke

Spezifische Laufwerke können im Explorer oder Arbeitsplatz ausgeblendet werden. Ein Programm, welches am Explorer vorbei greift, wird immer noch den Inhalt des Laufwerkes anzeigen, somit ist diese Richtlinie nicht verlässlich, um böswillige Benutzer auszuschließen (siehe [Chapter 6, Troubleshooting](#) für weitere Information). Die benötigten Informationen für dieses Template sind [Table 8-11](#) zusammengefasst.

Class	User (HKCU)
Registry key	Software\Microsoft\CurrentVersion\Policies\Explorer
Valuename	NoDrives
Value	A: 1, B: 2, C: 4, D: 8, E: 16, F: 32, G: 64, H: 128, I: 256, J: 512, K: 1024, L: 2048, M: 4096, N: 8192, O: 16384, P: 32768, Q: 65536, R: 131072, S: 262144, T: 524288, U: 1048576, V: 2097152, W: 4194304, X: 8388608, Y: 16777216, Z: 33554432, ALL 67108863 To hide more than one drive, sum the value of each drive. For example, to hide A., B.; and C., add 1 + 2 + 4 =7
Value Type	Binary (REG_DWORD)

Eine einfache Checkbox kann für jeden Laufwerksbuchstaben benutzt werden; wie auch immer, um alle Laufwerke zu integrieren, würde es 26 Policy Checkboxes unterhalb einer Category erfordern. Es würde keinen unaufgeräumten Eindruck hinterlassen, wenn man einen Part Typ einsetzt, der es dem Anwender ermöglicht eine Auswahl von einer möglichen Liste von Vorschlägen zu treffen. Part Typs die diese Anforderung erfüllen sind `COMBOBOX` und `DROPDOWNLIST`. Die `COMBOBOX` wäre keine gute Wahl, da es dem eine Eingabe vom Anwender erlaubt, wobei dieser aber eventuell keine Idee hat, welchen Wert er überhaupt eintragen soll. Deswegen wähle ich die `DROPDOWNLIST`. Aus den `DROPDOWNLIST` Optionen ([Table 8-3](#)) kann man die `NOSORT` Option auswählen, um die Liste in einer vorgegebenen Reihenfolge anzuzeigen. Ich werden ebenfalls einen Default Wert mit Value 0 (keine ausgeblendeten Laufwerke) setzen, für den Fall, dass die Checkbox versehentlich ausgewählt wird.

Es kann nur ein Laufwerk oder eine einzelne Kombination von Laufwerken ausgewählt werden. Es ist nicht möglich alle 26 Laufwerke einzublenden und es dem Anwender zu ermöglichen mehr als einen Laufwerksbuchstaben für die Ausblendung zu markieren, da es keine Template Syntax gibt, die es ermöglicht die einzutragenden Werte zu addieren. Das zuletzt ausgewählte Laufwerk würde ausgeblendet, die anderen würden schlicht ignoriert. Um es dem Anwender nun zu ermöglichen mehr als ein Laufwerk auszublenden muss ich konkrete Optionen in der drop-down Liste vorgeben. Zusätzlich könnte im erklärenden Text erwähnt werden, dass die Laufwerke nur in bestimmten Bereichen der Programme ausgeblendet werden. Der Abschnitt in dem Template könnte sich so darstellen:

```
POLICY !!HideDrives
  KEYNAME Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
  PART !!HIDEDRIVES DROPDOWNLIST REQUIRED
    #if version > 1
    NOSORT
    #endif
    VALUENAME "NoDrives"
    ITEMLIST
      NAME !!A VALUE NUMERIC 1
      NAME !!B VALUE NUMERIC 2
      NAME !!C VALUE NUMERIC 4
      NAME !!AB VALUE NUMERIC 3
      NAME !!ABC VALUE NUMERIC 7
      NAME !!CLEARALL VALUE NUMERIC 0 DEFAULT
    END ITEMLIST
  END PART
  PART !!HIDEDRIVESTEXT1 TEXT END PART
  PART !!HIDEDRIVESTEXT2 TEXT END PART
```

In diesem Beispiel ist es möglich nur das Laufwerk A, nur B, nur C, alle drei A, B, C oder nur Laufwerk A und B auszublenden.

Zusammenfassung

Wenn man erst einmal die Parts und Policies hat, muss man sie in Categories gruppieren und danach die Categories in die passenden Classes integrieren. Ich habe bereits darauf hingewiesen, zu welcher Klasse welche Richtlinie gehört. Im Speziellen sind die Richtlinien in class `MACHINE`, die die Windows Registrierungsdaten und die Tips anzeigen. Die anderen Richtlinien (Klang Ereignisse, Menügeschwindigkeit und ausgeblendete Laufwerke) gehören in die Class `USER`.

Gruppierung der Richtlinien in Kategorien ist schlicht der Vorgang, zu entscheiden was der sinnvollste Weg wäre sie zu gruppieren. In diesem Beispiel werde ich eine Category pro Policy bis hin zur Tabelle [Table 8-11](#) erstellen.

Zum Schluss muss noch eine Strings Sektion erstellt werden für jeden Wert, dem zwei !! vorangestellt sind, in dem die Text Definition dieses Wertes festgelegt ist. Das komplette Policy Template würde dann so aussehen:

```
; chapter8.adm
!!!!!!!!!!!!!!!!!!!!
```

```

CLASS MACHINE ;;;;;
;;;;;;;;;;;;;

CATEGORY !!changereg
KEYNAME "Software\Microsoft\Windows\CurrentVersion"
  POLICY !!Changereg
    PART !!Changereg1 EDITTEXT REQUIRED
    VALUENAME "RegisteredOwner"
    END PART
    PART !!changereg2 EDITTEXT REQUIRED
    VALUENAME "RegisteredOrganization"
    END PART
    PART !!changeregtext1 TEXT END PART
    PART !!changeregtext2 TEXT END PART
  END POLICY
END CATEGORY

CATEGORY !!TIPS
POLICY !!tips
KEYNAME "Software\Microsoft\Windows\CurrentVersion\Explorer\Tips"
PART !!tips LISTBOX EXPLICITVALUE
END PART
END POLICY
END CATEGORY

;;;;;;;;;;;;;
CLASS USER ;;;;;
;;;;;;;;;;;;;

CATEGORY !!changesounds
POLICY !!changestart
KEYNAME "AppEvents\Schemes\Apps\.Default\SystemStart\.Current"
  PART !!wavfile EDITTEXT
  VALUENAME ""
  END PART
END POLICY

POLICY !!changeexit
KEYNAME "AppEvents\Schemes\Apps\.Default\SystemExit\.Current"
  PART !!wavfile EDITTEXT
  VALUENAME ""
  END PART
END POLICY
END CATEGORY

CATEGORY !!menudelay

POLICY !!menudelay
KEYNAME "Control Panel\Desktop"
  PART !!menudelay NUMERIC SPIN 500
  VALUENAME "MenuShowDelay"
  TXTCONVERT
  DEFAULT 400
  MIN 0
  MAX 10000
END PART
  PART !!MENUTEXT TEXT END PART
END POLICY
END CATEGORY

CATEGORY !!HIDEDRIVES
POLICY !!HideDrives
KEYNAME Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
  PART !!HIDEDRIVES DROPDOWNLIST REQUIRED

```

```

# if version > 1
NOSORT
#endif
VALUENAME "NoDrives"
  ITEMLIST
  NAME !!A VALUE NUMERIC 1
  NAME !!B VALUE NUMERIC 2
  NAME !!C VALUE NUMERIC 4
  NAME !!AB VALUE NUMERIC 3
  NAME !!ABC VALUE NUMERIC 7
  NAME !!CLEARALL VALUE NUMERIC 0 DEFAULT
  END ITEMLIST
END PART
PART !!HIDEDRIVESTEXT1 TEXT END PART
PART !!HIDEDRIVESTEXT2 TEXT END PART
END POLICY
END CATEGORY

[STRINGS]
changereg="Change Registration"
changereg1="Registered Owner"
changereg2="Registered Organization"
changeregtext1="Change registration names under the "General" tab of"
changeregtext2="My Computer" properties window"
tips="Write out your personalized "Tips of the Day""
changesounds="Change Windows Sound Events"
changestart="Change Windows Start Sound"
wavfile=".WAV file to use"
changeexit="Change Windows Exit Sound"
menudelay="Delay in milliseconds"
menutext="Change the delay before submenus automatically cascade open"
hidedrives="Hide Drives"
A="Restrict Drive A"
B="Restrict Drive B"
C="Restrict Drive C"
AB="Restrict Drive A and B"
ABC="Restrict Drive A, B, and C"
clearall="Do not restrict any drives"
hidrivestext1="Hide one or more drives from My Computer and Explorer"
hidrivestext2="Note: Do not use with other "hide drives" policies"

```

Fehlerbeseitigung im Template

Fehlerbeseitigung im Template, so weit es die Syntax betrifft ist erstaunlicherweise einfach. Sobald die Textdatei erstellt ist, speichert man sie mit der Dateieindung ADM. Danach lädt man das Template in den SPE. Sobald ein Fehler gefunden wird, wird er vom SPE inklusive der Zeile, dem verursachendem Fehler, dem unerwartetem Keyword oder Text angezeigt und eine mögliche Liste der passenden Keywords für diesen Bereich wird mit angegeben.

Der SPE kann nicht untersuchen ob die eingetragenen KEYNAMES oder VALUES Sinn machen. Das kann nur durch Aktivierung der Richtlinie erfolgen und der Kontrolle ob die aktuelle Einstellung die Erwartungen erfüllt und das geplante Ergebnis in dem Programm bringt. Es sein angemerkt, dass wenn man Änderungen am Template vornimmt, solange es noch vom SPE geöffnet ist man das Template neu importieren muss, um die Änderungen im SPE darzustellen. Dafür kann man über Optionen → Richtlinienvorlagen die OK Schaltfläche wählen. Alternativ kann der SPE geschlossen werden, die Änderungen durchgeführt werden und danach das Programm wieder gestartet werden. Das hätte denselben Effekt, da das Template bei jedem Start des SPE automatisch neu geladen wird.

WARNUNG:

Es sollte immer ein Registry Backup gemacht werden, bevor man das neue Template ausprobiert!

Zusammenfassung

Das Template kann auf die vier Hauptzonen aufgeschlüsselt werden (Class, Category, Policy und Part). Die Class und Category Zonen dienen hauptsächlich der organisatorische Struktur, die Policy und Part Zonen enthalten die aktuellen Registry Änderungen. Die Verwendung eines Templates wird durch die Anwendung von einem oder mehreren der sieben Part Typen (TEXT, EDITTEXT, CHECKBOX, NUMERIC, DROPDOWNLIST, COMBOBOX and LISTBOX) erleichtert. Diese Part Typen erlauben es die Daten in ihrer logischsten Folge zusammenzufassen, basierend auf ihren Value Types und des Einsatzbereiches des Templates.

Wenn sie wie ich sind, dann sind sie begierig darauf alle ihre häufig angewendeten Registry Einträge auf ihre Einsatzmöglichkeit in einem selbst erstellten Template zu überprüfen. Wenn der Eintrag in einem Unterschlüssel von HKCU oder HKLM ist, der Wert vom Typ einem String (REG_SZ und REG_EXPAND_SZ) oder DWORD (REG_DWORD) entspricht, dann kann es möglich sein. Wenn der Key unglücklicherweise außerhalb der Top-Level Key HKCU and HKLM liegt, ein Binary (REG_Binary) oder anderer Windows Registry Typ ist, so müssen diese weiterhin, per Hand editiert werden oder man benutzt den Registrierungs- Editor um eine REG Datei zu importieren. Abgesehen von diesen Einschränkungen wird eine selbst erstellte Richtlinienvorlage die Funktionalität der SPE erweitern, auch in dem Firmen die nur hausinterne Software nutzen.