ИВАН КОРОБКО

# АДМИНИСТРИРОВАНИЕ СЕТЕЙ WINDOWS С ПОМОЩЬЮ СЦЕНАРИЕВ

Windows Script Host, Visual Basic Script Edition, ASP, ASP.NET

Основы программирования Active Directory: WinNT и LDAP

Microsoft Windows Management Instrument

Сценарии регистрации пользователей в сети

Автоматическая установка программного обеспечения

СИСАДМИН
СИСТЕМНЫЙ АДМИНИСТРАТОР

**Иван Коробко**

# АДМИНИСТРИРОВАНИЕ СЕТЕЙ WINDOWS С ПОМОЩЬЮ СЦЕНАРИЕВ

Санкт-Петербург

«БХВ-Петербург»

2007

УДК 681.3.06
ББК 32.973.26-018.2
К68

**Коробко И. В.**

К68 Администрирование сетей Windows с помощью сценариев. — СПб.: БХВ-Петербург, 2007. — 368 с.: ил. — (Системный администратор)

ISBN 978-5-9775-0140-8

Книга представляет собой подробное руководство по автоматизации различных процессов в сети с помощью сценариев, а также содержит большое количество справочной информации. Приведенные примеры наглядно иллюстрируют возможные способы решения задач, возникающих перед пользователями и системными администраторами. Описываются достоинства и недостатки каждого способа. Рассмотрены основы разработки сценариев с использованием Windows Script Host, Visual Basic Script Edition, ASP и ASP.NET. Показано, как программно управлять реестром и файловой системой. Рассмотрены инструменты WMI. Уделено большое внимание созданию сценариев регистрации пользователей в сети на базе языка программирования KIXTart. Описано программное управление Active Directory с помощью LDAP и WinNT. Рассмотрены вопросы автоматизации процесса установки ОС и ПО, клонирования жестких дисков и др.

*Для системных администраторов, сотрудников службы технической поддержки*

УДК 681.3.06
ББК 32.973.26-018.2

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Игорь Шишигин* |
| Зав. редакцией | *Григорий Добин* |
| Редактор | *Екатерина Капалыгина* |
| Компьютерная верстка | *Натальи Караваевой* |
| Корректор | *Виктория Пиотровская* |
| Дизайн серии | *Инны Тачиной* |
| Оформление обложки | *Елены Беляевой* |
| Зав. производством | *Николай Тверских* |

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.08.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 29,67.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0140-8

© Коробко И. В., 2007
© Оформление, издательство "БХВ-Петербург", 2007

# Оглавление

**Благодарности** ..... **1**

**Предисловие** ..... **3**

**Введение** ..... **5**

Ответственность ..... 8

**Глава 1. Основы программирования сценариев** ..... **9**

Программирование с использованием объектов ..... 9
- Понятие объекта в программировании ..... 9
- Наборы ..... 11
- Методы ..... 11

Обзор систем исчислений, используемых в программировании ..... 12
- Десятеричная система исчисления ..... 12
- Двоичная система исчисления ..... 12
- Шестнадцатеричная система исчисления ..... 13

Обзор языков программирования ..... 15
- Сценарии на базе командной строки ..... 16
- WinBatch ..... 17
- JavaScript ..... 17
- Visual Basic Script Edition ..... 18
- Visual Basic for Application ..... 18
- Windows Scripting Host ..... 18
- Hiddensof AutoIt ..... 19
- KIXtart ..... 20
- Анализ предлагаемых решений ..... 20

ActiveX-компоненты ..... 23

Инструмент разработки сценариев. Primal Script 4.0 ..... 24
- Интегрирование собственных шаблонов ..... 24
- Создание новой объектной модели ..... 28

Добавление новых функций в существующие языки программирования ... 30
Шифрование скриптов ... 31
Шифрование файлов из командной строки ... 31
Использование Microsoft Script Encoder ... 32
Шифрование KIX-сценариев ... 32

**Глава 2. Visual Basic Script Edition ... 33**

Основы синтаксиса ... 33
Переменные ... 33
Соглашение об именах ... 33
Комментарии ... 34
Непрерывные строки ... 34
Типы данных ... 34
Оператор *Option Explicit* ... 36
Константы ... 37
Объявление переменных ... 37
Присвоение значений переменным ... 37
Объявление массивов ... 38
Переопределение размеров массива ... 39
Определение границ массива ... 39
Операторы ... 39
Операторы конкатенации ... 40
Преобразование типов ... 41
Инструкции ... 42
Инструкции *If…Then* ... 42
Инструкция *Select…Case* ... 43
Инструкция безусловного перехода ... 43
Управляемые циклы ... 44
Циклы с параметрами: оператор *For…Next* ... 44
Циклы с условием ... 45
Цикл *Do…While* ... 45
Цикл *While…Wend* ... 46
Оператор *For…Each* ... 46
Подпрограммы и классы ... 47
Функция ... 47
Процедуры ... 48
Передача значений параметров с помощью ключевых слов *ByRef* и *ByVal* ... 49
Классы ... 49
Получение свойств обработчика языка ... 51

Управление свойствами объектов ... 52
Создание методов ... 54
Использование инструкции *With* ... 54
Интерактивная работа скриптов ... 55
Ввод информации с помощью функции *InputBox()* ... 55
Отображение информации с помощью функции *MsgBox()* ... 56
Определение вида иконки и кнопок диалогового окна ... 57
Определение возвращаемого значения функции *MsgBox()* ... 59

**Глава 3. Windows Script Host ... 61**

Сервер сценариев Windows Script ... 61
Установка Windows Script Host 5.6 ... 61
Запуск сценариев WSH из командной строки ... 61
Возможности WSH-сценариев ... 63
Объектная модель WSH ... 63
Объект *WScript* ... 64
Свойства объекта *WScript* ... 64
Свойство *Arguments* ... 65
Свойства *FullName*, *Name*, *Path*, *Version* ... 67
Свойства *StdErr*, *StdIn*, *StdOut* ... 67
Методы объекта *WScript* ... 67
Метод *CreateObject(ObjectID, Prefix)* ... 68
Метод *GetObject(Object, ObjectID, Prefix)* ... 68
Метод *DisconnetObject(ObjectID)* ... 69
Объект *WshArguments* ... 69
Объект *WshShell* ... 70
Метод *AppActivate* ... 71
Управление ярлыками методом *CreateShortcut* ... 72
Объект *WshShortcut* ... 72
Создание ярлыка ... 73
Чтение/изменение свойств ярлыка ... 74
Удаление ярлыка ... 75
Объект *WshURLShortcut* ... 75
Объект *Wscript.Shell* ... 76
Метод *Environment* ... 76
Операции с переменными окружения ... 79
Создание переменных окружения с помощью сценария ... 80
Удаление переменных окружения с помощью сценария ... 80
Метод *ExpandEnvironmentStrings* ... 80
Метод *LogEvent* ... 81

Метод *Popup* ..... 81
Определение возвращаемого значения методом *Popup* ..... 82
Метод *Run* ..... 83
Метод *SendKeys* ..... 85
Метод *SpecialFolders* ..... 86
Объект *WshNetwork* ..... 88
Методы *AddWindowsPrinterConnection* и *AddPrinterConnection* ..... 89
Обработка ошибок ..... 90
Метод *RemovePrinterConnection* ..... 91
Метод *EnumPrinterConnections* ..... 91
Метод *SetDefaultPrinter* ..... 92
Метод *MapNetworkDrive* ..... 92
Метод *EnumNetworkDrives* ..... 93
Метод *RemoveNetworkDrive* ..... 94

**Глава 4. На пути от VBScript к .NET ..... 95**

Переход от VBScript к ASP ..... 95
Настройка IIS для ASP ..... 96
Переход от VBScript и ASP к ASP.NET ..... 99
Установка Visual Studio ..... 99
Microsoft FRAMEWORK ..... 100
Управление доверительными отношениями в ASP.NET ..... 101
Изменения в синтаксисе ..... 104
Файловая структура ..... 104
Исполнение ASP-файлов в ASP.NET ..... 105
Оператор *Option Explicit* ..... 106
VBScript, ASP ..... 106
ASP.NET ..... 106
Преобразование типов данных ..... 107
Использование скобок при передаче параметров подпрограммам и методам ..... 108
Операторы *SET* и *LET* ..... 109
Поддержка многопоточных компонентов ..... 109
Обработка ошибок в ASP.NET ..... 110

**Глава 5. Программное управление реестром ..... 111**

Основы построения реестра ..... 112
Редакторы реестра ..... 113
Программное управление реестром ..... 114
Редактирование реестра сценарием на базе командной строки ..... 114
Редактирование реестра с помощью WSH ..... 115

Редактирование реестра с помощью KIXTart ... 117
Групповые политики ... 120
Административные шаблоны. Синтаксис ... 120
Комментарии ... 121
Строки ... 121
*CLASS* ... 122
*CATEGORY* ... 122
*POLICY* ... 124
Программирование интерфейса политик безопасности ... 125
*CHECKBOX* ... 127
*COMBOBOX* ... 128
*DROPDOWNLIST* ... 129
*EDITTEXT* ... 130
*LISTBOX* ... 130
*NUMERIC* ... 131
*TEXT* ... 132
Практика использования административных шаблонов ... 132
Внедрение административных шаблонов ... 134
Удаление административного шаблона ... 135

**Глава 6. Управление файловой системой ... 137**

Команды ввода/вывода (*FSO*) ... 137
Работа с дисками. Набор *Drive* ... 138
Работа с папками и файлами. Наборы *Folders* и *Files* ... 140
Формирование списка вложенных объектов в папке ... 141
Чтение характеристик файлов ... 144
Изменение атрибутов файлов ... 146
Операции над файлами и папками ... 147
Управление текстовыми файлами ... 149
Управление правами доступа на файлы и папки ... 152
Библиотека ADsSequrity.dll ... 152
Организация доступа к библиотеке ... 152
Управление правами на папку ... 154
Основные операции: просмотр, добавление, удаление ... 158
Просмотр списка доступных объектов ... 159
Добавление объекта ... 161
Удаление существующих объектов ... 163

**Глава 7. Основы программирования Active Directory ... 165**

Active Directory изнутри ... 165
DNS и Active Directory ... 166

Архитектура службы каталогов Active Directory ... 168
Объектная модель ADSI ... 170
Провайдеры ADSI ... 171

**Глава 8. Программное управление ADSI: WinNT ... 173**

Объектная модель провайдера WinNT ... 173
Класс *Domain* ... 177
Определение доступных доменов ... 177
Чтение параметров класса *Domain* ... 178
Обновление параметров класса *Domain* ... 179
Перечисление объектов класса *Domain* ... 179
Создание, переименование и удаление объектов в домене ... 180
Создание объектов ... 181
Удаление объектов ... 182
Переименование объектов ... 183
Подкласс *User* ... 183
Манипулирование пользовательскими флагами функцией *UserFlags()* ... 184
Подкласс *Group* ... 187
Взаимосвязь учетных записей пользователей и групп ... 187
Добавление и удаление учетной записи пользователя в группу ... 187
Получение списка всех учетных записей, входящих в группу ... 188
Просмотр списка групп ... 188
Класс *Computer* ... 189
Подклассы PrintQueue, PrintJob, PrintJobsCollection ... 189
Управление принтерами и очередями принтеров ... 189
Управление принтером ... 190
Просмотр состояния принтера ... 191
Чтение свойств заданий в очереди принтера ... 192
Управление очередью печати ... 193
Класс *Service* ... 194
Подкласс *FileService* и *FileShare* ... 194
Чтение свойств совместно используемых ресурсов ... 194
Программное создание и удаление совместно используемого ресурса ... 195
Подкласс *Service* ... 196
Перечисление и чтение свойств служб на выбранном компьютере ... 196
Связывание служб на выбранном компьютере ... 197

**Глава 9. Программное управление ADSI: LDAP ... 199**

Структура объектной модели провайдера LDAP ... 199
Имена LDAP URL ... 199
Развернутая форма записи ... 200

Сокращенная форма записи ....201
Определение имени домена. Обзор способов ....201
Объекты Active Directory ....202
Поиск объектов в AD ....203
Поиск всех учетных записей пользователей в домене ....203
Объектная модель провайдера LDAP ....204
Active Directory Viewer (Microsoft) ....204
Просмотр и редактирование объектной модели программой ADV в режиме *ObjectViewer* ....205
LDAP Browser 2.6. (Softerra) ....207
Различия провайдеров LDAP и WinNT ....211
Выполнение команд и программ от имени определенного пользователя ....211
Выполнение сценариев от имени конкретного пользователя ....212
Импорт и экспорт данных из AD ....213
LDIF-файлы ....214
CSVDE-файлы ....215
Действия над объектами ....215
Создание объекта ....215
Удаление объекта ....217
Перемещение объектов ....218
Чтение атрибутов объектов ....219
Изменение атрибутов объекта ....220
Поиск объекта ....220

**Глава 10. Microsoft Windows Management Instrument ....221**

Средства управления WMI ....222
Внутреннее устройство WMI ....223
Язык запросов WQL ....224
Безопасность и WMI ....225
Имперсонация ....225
Аутентификация ....226
Привилегии ....227
Способы доступа к объектам WMI (VBScript) ....232
Продукты, использующие WMI ....234
SMS 2003 SP1 ....234
MOM 2005 SP1 ....236

**Глава 11. Сценарий регистрации пользователей в сети ....237**

KIXTart ....237
Системные требования ....237
Комплект поставки KIXTart ....238

Установка KIXTart ....238
Синтаксис KIX32 ....238
Запуск KIX32 ....239
Режим отладки сценариев ....239
Синтаксис KIXTart ....239
Адаптация листингов VBScript и WSH к KIXTart ....240
Задачи, решаемые сценарием ....241
Решение задачи инвентаризации ....241
Сбор информации об аппаратном обеспечении с помощью WMI ....242
Сбор информации об учетной записи пользователя с помощью Acctive Directory ....247
Формирование файла отчета в формате XML ....248
Экспорт данных в SQL ....250
Подключение к базе данных ....250
Операции с SQL-таблицей ....251
Решение задачи подключения сетевых принтеров ....252
Соглашение об именах групп безопасности и принтеров ....252
Предварительная настройка принтера и Active Directory ....253
Формирование списка принтеров, которые необходимо подключить пользователю ....255
Формирование списка сетевых принтеров, подключенных пользователю ....258
Приведение списков принтеров в соответствие ....259
Решение задачи подключения сетевых дисков ....260
Определение членства в соответствующих группах безопасности ....262
Чтение данных из AD ....262
Отключение сетевых дисков ....264
Подключение необходимых сетевых дисков ....264
Корректировка описаний дисков в папке Мой компьютер ....266
Переименование описаний сетевых дисков для Windows XP ....267
Переименование описаний сетевых дисков для Windows 2000 ....268
Визуализация работы сценария KIXTart ....273
Визуализация работы скрипта с помощью стандартных диалоговых окон ....273
Визуализация работы скрипта с помощью DLL-библиотеки ....274
KIXForms ....274
KIXTart Script Packager ....274
Визуализация работы скрипта с помощью DHTML ....275
Синтаксис KIXWin ....276
Основы формирования файла DHTML ....276
Внедрение скрипта в эксплуатацию ....278

**Глава 12. Подготовка рабочей станции к функционированию в сети ..... 283**

Клонирование жестких дисков ..... 283
Автоматизация процесса установки: ОС ..... 286
Обзор инсталляторов ..... 286
Пакетная установка ПО ..... 286
Windows Installer ..... 287
Взгляд изнутри: файл msiexec.exe ..... 290
Параметры командной строки для Msiexec ..... 290
Inno Setup ..... 292
Nullsoft Scriptable Install System (NSIS) ..... 293
Wise Installer ..... 295
Создание дистрибутива ..... 295
Подготовка файловой структуры будущего диска ..... 295
Создание файла ответов для установки Windows ..... 296
Файловая структура дистрибутивного диска Windows ..... 297
Механизм инсталляции быстрых исправлений ..... 299
Подготовка дистрибутива Windows ..... 299
Интеграция пакета исправлений Service Pack в Windows ..... 299
Установка пакетов быстрых исправлений в автоматическом режиме ..... 300
Принципы именования hotfix ..... 300
Копирование hotfix из Интернета ..... 300
Интеграция hotfix в дистрибутив Windows ..... 302
Ключи, используемые при установке hotfix ..... 302
Установка MUI в автоматическом режиме ..... 303
Интеграция драйверов в дистрибутив ..... 303
Автоматизация процесса установки: ПО ..... 303
Установка антивируса: Norton Antivirus ..... 303
Установка навигатора: WinCMD ..... 304
Автоматическая установка архиватора: WinRar 3.x ..... 304
Установка программы записи CD/DVD: Nero Burning Rom 6.3.0.x ..... 305
Установка Adobe Acrobat Reader 6 ..... 306
Автоматическая установка Microsoft Office 2003 и необходимых дополнений ..... 306
Способы автоматической установки Office ..... 306
Подготовка дистрибутива Office ..... 307
Создание загрузочного диска ..... 309
Создание файла-образа диска ..... 309
Тестирование ISO-файла ..... 311
Запись файла-образа на диск ..... 312

**ПРИЛОЖЕНИЯ ..... 315**

**Приложение 1. Управление сетевой печатью ..... 317**

Сетевой принтер ..... 317
Сервер печати ..... 318
Соглашение об именах ..... 318
Установка и настройка сетевого принтера ..... 319
Описание установки принтера ..... 322

**Приложение 2. Ошибки выполнения сценариев в WSH ..... 329**

**Приложение 3. Объектная модель провайдера WinNT ..... 331**

objectClass *Domain* ..... 331
objectClass *User* ..... 332
objectClass *Group* ..... 334
objectClass *Computer* ..... 334
objectClass *PrintQueue* ..... 335
objectClass *PrintJob* ..... 335
objectClass *FileService* ..... 336
objectClass *FileShare* ..... 336
objectClass *Service* ..... 337

**Приложение 4. Таблица ASCII (American Standard Code for Information Interchange) ..... 338**

Назначение специализированных символов ..... 339
Форматирование ..... 339
Передача данных ..... 340
Разделительные знаки при передаче информации ..... 340
Другие символы ..... 341

**Приложение 5. Файл ответов Winnt.sif ..... 342**

**Приложение 6. Файл ответов Install.inf ..... 344**

**Предметный указатель ..... 347**

*С любовью посвящается Елене Ходовой*

# Благодарности

Создание этой книги — труд не одного дня. Я хочу выразить особую благодарность Зыкову Филиппу за ценные идеи, Алексеенко Александру за дельные советы, Тачилину Юрию за неоценимую помощь и всему "Центр-ИТ" за огромное терпение во время тестирования созданных мною скриптов.

Мои благодарности сотрудникам издательства "БХВ-Петербург", в особенности Шишигину Игорю и Капалыгиной Екатерине за их вклад в редактировании плана и текста книги.

Отдельное спасибо Ходовой Елене — первому читателю-редактору этой книги перед ее печатью, за массу дельных предложений и поддержку при работе над книгой.

# Предисловие

Человек полетит, опираясь на силу разума,
а не на силу мышц…

*Н. Е. Жуковский*

Современные компьютерные сети становятся все сложнее и сложнее, а значит, управлять ими вручную все труднее. В больших сетях ошибки обслуживающего персонала обходятся очень дорого, поэтому необходимо свести влияние человеческого фактора к минимуму, максимально повысив производительность труда. Одним из решений является автоматизация функций IT-службы с помощью различных скриптов.

Книга задумывалась как подробное руководство-справочник по автоматизированию различных процессов в сети. Она имеет выраженную практическую направленность и написана на основе собственного опыта. Примеры в этой книге наглядно иллюстрируют возможные способы решения поставленной задачи, описываются достоинства и недостатки каждого из них. Прочитав эту книгу, вы узнаете, как создать файл ответов для автоматической установки операционной системы Windows и Microsoft Office, обеспечить автоматическую установку большинства приложений.

Научитесь разрабатывать интеллектуальные сценарии регистрации пользователей в сети, управляющие подключением сетевых ресурсов (дисков и принтеров). Скрипт подключит пользователю только те ресурсы, с которыми он работает. Узнаете, как сформировывать отчеты, содержащие в себе информацию о регистрирующемся в сети пользователе на основе данных из Active Directory. А также сохранять информацию об аппаратной и программной конфигурации рабочей станции на основе данных из реестра, WMI и в XML-файле, SQL-базе.

Книга будет полезна не только опытным администраторам и программистам, но и всем сотрудникам службы технической поддержки, которые хотят повысить эффективность своей работы, автоматизировав часть возложенных на них задач.

# Введение

Начиная свой путь со специалиста системной поддержки, автор старался автоматизировать рутинные действия — установку операционной системы, различного программного обеспечения, настройку рабочих станций для того, чтобы сэкономить время на выполнение повторяющихся операций и повысить результативность своей работы. Все это позволило снизить TCO.

Приобретя опыт в создании файлов-ответов и несложных скриптов, автор приступил к созданию сценариев для регистрации пользователей в сети. Идея, лежащая в основе сценария, — создать для каждого пользователя персональное окружение, подключив только необходимые для его работы ресурсы и убрать все лишние. Одновременно, осуществлять при загрузке мониторинг настроек и незаметно для пользователя изменять их. Создание скрипта была очень интересной и плодотворной работой целого коллектива. Параллельно разрабатывались различные административные сценарии по управлению Active Directory.

Основная проблема, с которой столкнулся автор, — отсутствие различной справочной информации, подробного описания объектных моделей, наглядных примеров. Сейчас появилось достаточно много книг — сборников рецептов, в которых можно найти часто используемые решения и воспользоваться ими. Однако иногда необходимо создавать сложные скрипты. Для этого нужно обладать хорошими теоретическими знаниями и понимать идею, заложенную в основе той или иной объектной модели. В большинстве книг теоретический блок как таковой отсутствовал и рассматривался лишь узкий круг вопросов и практических решений. Поэтому родилась идея создания такой книги, которая содержала бы не только конкретные скриптовые рецепты, но и теоретические основы создания скриптов и не требовала обращения к другим источникам информации.

Каждая глава написана на основе личного опыта, снабжена характерными примерами.

В книге вы найдете:

- информацию об основных подходах объектно-ориентированного программирования;
- обзор основных скриптовых языков (VBScript, WSH, KIXTart) и рекомендации по выбору языка для решения поставленной задачи;
- основы программного управления реестром с помощью различных скриптовых языков;
- основы программирования Active Directory, описание объектных моделей основных провайдеров — WinNT и LDAP;
- основы программирования Windows Management Instrument (WMI);
- способы создания интеллектуального сценария регистрации пользователей в сети;
- методы работы с XML, чтения и записи данных в SQL;
- принципы создания файлов ответов для обеспечения автоматической установки ОС;
- обзор существующих инсталляторов с точки зрения автоматизации процесса установки ПО.

Здесь есть ответы на эти и многие другие вопросы, а с помощью приведенной теории можно пополнить свои знания, разрешить оставшиеся вопросы и сделать еще один шаг в программировании скриптов.

Книга состоит из 12 глав.

*Глава 1 "Основы программирования сценариев"* содержит базовые знания по объектно-ориентированному программированию; обзор систем исчислений, используемых в программировании.

Первый шаг создания скрипта — анализ поставленной задачи и выбор оптимального для ее решения языка программирования (VBScript, WSH, KIXTart). В первой главе читатель сможет ознакомиться с возможностями основных существующих скриптовых языков программирования, выбрать наиболее подходящий язык для решения поставленной перед ним задачи.

*Глава 2 "Visual Basic Script Edition"* содержит описание не только базовых возможностей одного из самых используемых скриптовых языков — VBScript, но и дополнительных возможностей: обработку ошибок, классы и т. д.

*Глава 3 "Windows Script Host"* познакомит читателя с основами программирования сценариев для Microsoft Windows Script Host 2.0. Научит управлять се-

тевыми ресурсами, ярлыками программ, осуществлять запуск сценариев из командной строки, управлять процессами ОС, спецпапками, получать доступ к различным OLE-объектам, работать с системным реестром.

*Глава 4 "На пути от VBScript к .NET"* рассказывает о практике перехода с VBScript на более мощные языки — ASP и ASP.NET, на базе полученных знаний во второй и третьих главах.

*Глава 5 "Программное управление реестром"* целиком посвящена реестру и групповым политикам, которые органично дополняют его возможности. Существует несколько скриптовых языков, в том числе VBScript, WSH, KIXTart, скрипты на базе командной строки. В главе описано не только, что такое системный реестр и как создавать REG-файлы, но и рассматриваются приемы работы, характерные для каждого из приведенных скриптовых языков программирования, проиллюстрированные наглядными примерами.

*Глава 6 "Управление файловой системой"* посвящена управлению файловой системой из VBScript, ASP и ASP.NET. Прочитав ее, читатель сможет не только осуществлять различные операции с файлами и папками, но и программно управлять их безопасностью.

*Глава 7 "Основы программирования Active Directory"* содержит обзор воспринимаемых Active Directory форматов имен, доступных провайдеров. Описана объектная модель ADSI.

*Глава 8 "Программное управление ADSI: WINNT"* и *глава 9 "Программное управление ADSI: LDAP"* познакомит читателя с основными принципами программирования Active Directory. Здесь приведены описания объектных моделей провайдеров WINNT и LDAP, рассмотрены различные способы доступа к ним и характерные примеры.

*Глава 10 "Microsoft Windows Management Instrument"* посвящена решению задачи накопления и сохранения информации об аппаратной конфигурации рабочей станции с помощью Microsoft Windows Management Instrument (WMI).

*Глава 11 "Сценарий регистрации пользователей в сети"* — пожалуй, самая большая глава, в которой используются все накопленные знания в этой книге: чтение данных из реестра, использование WMI, получение доступа к Active Directory, запись данных в SQL-таблицы и многое другое. Сценарии формируются на базе специально созданного для этих целей скриптового языка KIXTart.

*Глава 12 "Подготовка рабочей станции к функционированию в сети"* расскажет читателю о клонировании жестких дисков и подготовке рабочей станции к работе в сети. Вы узнаете много нового об инсталляторах и научи-

тесь автоматизировать установку различных программ, в том числе Microsoft Ofiice, WinRar, Nero и т. д. Сможете создавать новые файлы ответа, позволяющие обеспечить автоматическую установку операционной системы.

В *приложениях* приведена различная справочная информация, не вошедшая в главы.

Создавая эту книгу, автор надеялся, что она станет настольной.

# Ответственность

В книге описан очень мощный инструментарий по управлению серверов и настройке программного обеспечения. Все скрипты перед введением в эксплуатацию настоятельно рекомендуется многократно и всесторонне протестировать в тестовой зоне. Перед внедрением в эксплуатацию следует делать резервные копии. В неумелых руках сценарии могут привести к необратимым последствиям.



# Глава 1

# Основы программирования сценариев

Создание различных сценариев базируется на знании основ объектно-ориентированного программирования, синтаксисе языка, на котором он создается, используемых систем исчисления и их взаимодействии. Первый шаг в создании сценария — выбор оптимального языка программирования. От того, насколько правильно сделан этот шаг, будет зависеть не только удобство создания скрипта программистом, но и скорость его работы и надежность. Повысить скорость создания скрипта можно, выбрав один из предлагаемых редакторов, который обеспечит контроль вводимой информации, позволит сделать пошаговую его трассировку.

## Программирование с использованием объектов

### Понятие объекта в программировании

Примерами *объектов* реального мира могут быть автомобиль, дом, книга, стол. У каждого объекта есть характеризующие его *свойства*. Например, у автомобиля — цвет, габариты, вес и т. д. Число свойств зависит от объекта. Над объектом реального мира можно совершать различные действия: книгу — читать, машину — водить, одежду — носить, стирать и т. д.

В мире разработки программного обеспечения в программах также можно использовать *объектно-ориентированный подход* при определении объектов, которые имеют набор свойств и над которыми можно производить определенные действия. Рассмотрим простой объект — диалоговое окно (рис. 1.1). Его можно представить как объект, поскольку оно содержит данные (информацию) и совершает действие (выводит информацию). Свойствами объекта

"диалоговое окно" являются его положение на рабочем столе, цвет фона, размер шрифта заголовка и т. д.

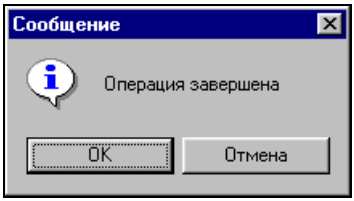



**Рис. 1.1.** Пример диалогового окна

Идея построения объектной модели основана на том, что объект может содержать другие объекты, так называемые *"дочерние" объекты*.

Приведенное в качестве примера диалоговое окно имеет две кнопки (**OK** и **Отмена**) и иконку знака информации. Эти элементы диалогового окна являются его *субобъектами* ("дочерние" объекты).

Рассмотрим в качестве объекта текстовый редактор. У него есть данные (документы, слова, предложения, числа и т. д.), над которыми можно выполнять действия (открывать, закрывать, сохранять документ, вставлять и удалять текст и т. д.).

Для определения структуры объектов в рамках приложения рекомендуется использовать иерархический подход. Объект `Текстовый редактор` в качестве субобъектов содержит `Документы`, которые включают в себя `Абзацы`, состоящие из `Предложений`, которые в свою очередь состоят из `Слов`, а `Слова` из `Букв`. В программе можно ссылаться на эту иерархию следующим образом (рис. 1.2):

```
Приложение.Документ.Абзац.Предложение.Слово.Буква.
```




**Рис. 1.2.** Пример объектной модели

Имена объектов выстроены в иерархическом порядке и разделены точками. Левее всех находится "родительский" объект, в данном случае `Приложение`. Зная иерархию объектов, можно обращаться к любому из них, определяя его позицию в иерархии, известной также как *объектная модель* (object model).

## Наборы

Показанная цепочка субобъектов представляет собой простейшую иерархическую модель. При этом документы состоят из целого набора абзацев, абзацы из набора предложений, а предложение из набора слов и т. д. (рис. 1.3).

*Набор* (collection) — это совокупность субобъектов одного типа. Чтобы получить доступ к одному из элементов набора, нужно указать номер необходимого элемента:

```
Приложение.Документы(0).Абзацы(5).Предложения(2).Слова(9).Буквы(3).
```

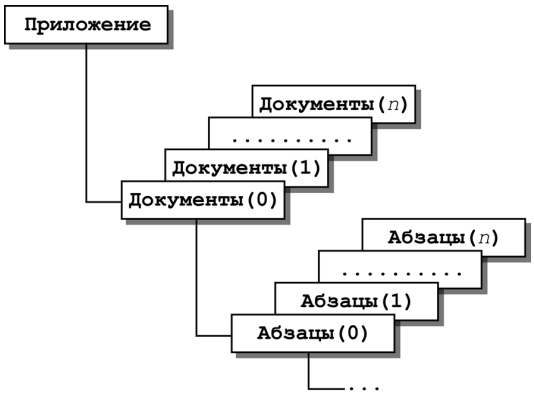



**Рис. 1.3.** Пример набора объектов

Нумерация элементов начинается с нуля, а имена объектов в данном случае употребляются во множественном числе. Объект `Документы` является набором объектов `Документ`. Первый член набора `Документы` (первый `Документ`), обозначен как `Документы(0)`.

## Методы

*Метод* представляет собой вызов процедуры или функции, ассоциированной с конкретным объектом. Свойства в свою очередь являются переменными, принадлежащими объектам.

Преимущество применения объектов в том, что программисту не нужно учитывать внутренних принципов работы. Все, что необходимо знать — это имя объекта и какие свойства и методы он поддерживает.

Например, для вывода простого диалогового окна на экран можно записать оператор на VBScript (см. рис. 1.1) — листинг 1.1.

**Листинг 1.1. Диалоговое окно**

```
Set WshShell=CreateObject("Wscript.Shell")
result=WshShell.Popup("Операция завершена", 5, "Сообщение", 65)
Wscript.Echo result
```

В данном случае объектом является `Wscript`, а `Echo` — именем метода.

# Обзор систем исчислений, используемых в программировании

Программистам и системным администраторам часто приходится иметь дело с данными, записанными в различных системах исчисления: десятеричной, двоичной и шестнадцатеричной, переводить данные из одной системы в другую. Рассмотрим каждую из этих систем исчисления и методику пересчета чисел между ними.

## Десятеричная система исчисления

В десятеричной системе исчисления знаменателем является число 10. Для задания значений используют числа от 0 до 9. Цифры, суммируясь, умножаются справа налево на увеличивающуюся степень десяти $10^0, 10^1, 10^2, ..., 10^n$.

Так, число 856 представлено как $8 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0 = 8 \cdot 100 + 5 \cdot 10 + 6 \cdot 1$.

Десятичная система исчисления является самой удобной системой для использования в повседневной жизни, однако она малоприменима в компьютерной системе, в которой используется 0 (нет сигнала) и 1 (есть сигнал). В компьютерном мире обычно применяется двоичная система исчисления.

## Двоичная система исчисления

Знаменателем двоичной системы является число 2. Значение формируется с помощью двух цифр — нуля и единицы. Соответственно, число 100110 в десятичной системе выглядит следующим образом:

$$1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 32 + 0 + 0 + 4 + 2 + 0 = 38.$$

Переведем число 50 из десятеричной в двоичную систему исчисления с помощью табл. 1.1. Найдем в ней наибольшее из меньших чисел — это число 32, которому соответствует пятый разряд. Сразу можно сказать, что двоичное число будет иметь шесть цифр, первая из которых 1. Определим остаток разности сравниваемого числа с найденным, т. е. 50 – 32 = 18. Поступаем аналогично, ищем наибольшее из меньших чисел для получившегося числа — это число 16, соответствующее четвертому разряду: пятым символом двоичного числа также будет единица. Разность 18 – 16 = 2.

***Таблица 1.1.*** *Пересчет чисел из десятеричной в двоичную систему исчисления*

| **Разряд** | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Значение** | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| | $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^1$ | $2^1$ | $2^0$ |

Повторяя описанную последовательность действий, получаем, что ближайшее число — 2 соответствует первому разряду. Знак равенства сигнализирует об окончании процедуры перевода. Тем разрядам, которые не равны единице, присваиваются нули. Таким образом, числу 50 в десятеричной системе соответствует число 110010 в двоичной системе исчисления.

## Шестнадцатеричная система исчисления

Преобразовывать числа в двоичном формате в десятеричную систему и обратно — задача довольно сложная. Гораздо удобнее использовать шестнадцатеричную систему, которая имеет знаменатель 16. Для обозначения числа в этой системе исчисления используются цифры от 0 до 9 и буквы от A до F. Приведем в табл. 1.2 эквиваленты шестнадцатеричной, десятеричной и двоичной систем.

***Таблица 1.2.*** *Соотношения между системами исчисления*

| **Двоичная система** | **Шестнадцатеричная система** | **Десятеричная система** |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |

***Таблица 1.2*** *(окончание)*

| Двоичная система | Шестнадцатеричная система | Десятеричная система |
|---|---|---|
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | A | 10 |
| 1011 | B | 11 |
| 1100 | C | 12 |
| 1101 | D | 13 |
| 1110 | E | 14 |
| 1111 | F | 15 |

Шестнадцатеричное число B35F в десятеричной системе представляется как:

$$11 \cdot 16^3 + 3 \cdot 16^2 + 5 \cdot 16^1 + 15 \cdot 16^0 = 11 \cdot 4096 + 3 \cdot 256 + 5 \cdot 16 + 15 \cdot 1 = 106511.$$

Обратный перевод из десятеричной в шестнадцатеричную систему исчисления осуществляется в соответствии с приведенным далее алгоритмом. Представим десятеричное число 356 в шестнадцатеричной системе: в табл. 1.3 найдем ближайшее меньшее число и определим соответствующий этому числу разряд. Таким числом является 256, принадлежащее ко второму разряду.

***Таблица 1.3.*** *Пересчет числел из 10-ной в 16-ную систему исчисления*

| **Разряд** | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| **Значение** | 4096 | 256 | 16 | 1 |
| | $16^3$ | $16^2$ | $16^1$ | $16^0$ |

Найдем целую часть от отношения исходного числа и только что определенного значения ближайшего меньшего к исходному числу числа $\frac{356}{256} = 1{,}390625$. Целая часть равна единице. Поэтому множителем при $16^2$ будет единица. Теперь определим произведение числа, соответствующего

второму разряду — $1 \cdot 16^2 = 256$. Найдем разность между первоначальным и получившимся числом: 356 – 256 = 100. Теперь это число считаем исходным и действуем по описанному алгоритму: ближайшее меньшее число — 16, соответствующее первому разряду. Целая часть от деления исходного числа на определенное ближайшее меньшее число равна 6 ($\frac{100}{6} = 6{,}25$). Числом, соответствующим первому разряду, является 96 ($16 \cdot 6 = 96$). Разность между первоначальным и получившимся числом — 4 ($100 - 96 = 4$). Новое исходное число, соответствующее последнему разряду — 4. Множителем при первом разряде является 4. Таким образом, десятеричному числу 356 соответствует шестнадцатеричное число 164.

Однако существует проблема — как идентифицировать систему исчисления числа, если оно содержит в себе числа от 0 до 9. Было принято решение — добавить к шестнадцатеричным числам префикс 0x.

Рассмотрим методику преобразования чисел из шестнадцатеричной системы в двоичную и обратно на примере числа 0xB35F. Оно состоит из четырех частей. Используя табл. 1.2, находим соответствия каждому из элементов.

$$\overbrace{1011}^{B}\overbrace{0011}^{3}\overbrace{0101}^{5}\overbrace{1111}^{F}.$$

Из двоичной в шестнадцатеричную систему перевод аналогичен. Если количество знаков в числе двоичного формата кратно четырем, то перед числом дописывается нужное количество нулей. Например, число 110011111 имеет 9 символов. Ближайшее кратное четырем число — двенадцать. В соответствии с этим правилом следует дописать три нуля и разбить получившееся число на части по четыре символа:

$$\underbrace{0001}_{1}\underbrace{1001}_{2}\underbrace{1111}_{3}.$$

Первой части соответствует число 1, второй — 9, третьей — 6. Таким образом, числу 110010111 в двоичной системе исчисления соответствует 0x19F в шестнадцатеричной.

# Обзор языков программирования

Существует несколько языков для создания скриптов. Среди них в операционную систему Windows интегрированы:

- сценарии на базе командной строки (файлы с расширением bat, pif);
- Windows Scripting Host (WSH);

- Microsoft Java Script (JScript);
- Microsoft Visual Basic Script Edition (VBScript).

Есть также языки, специально разработанные для создания скриптов, такие как KIXtart, AutoIT, WinBatch.

Языки программирования можно условно разделить на компилируемые и интерпретируемые.

*Компилируемый язык* — это язык, программный код которого преобразуется в машинный до его выполнения. Такое преобразование необходимо выполнять только один раз. Его выполняет компилятор.

*Интерпретируемый язык* является противоположностью компилируемого. Каждый раз при выполнении сценария код необходимо интерпретировать, т. е. преобразовывать в машинный. Для выполнения этой задачи предназначен интерпретатор. При одновременном интерпретировании на сервере некоторого количества сценариев происходит падение скорости выполнения скрипта.

Все приведенные ранее языки программирования являются интерпретируемыми, за исключением AutoIT и WinBatch — это компилируемые языки.

## Сценарии на базе командной строки

Сценарии на базе командной строки появились в Disk Operating System (DOS) версии 1.0. Файлы сценариев представляют собой обычный текстовый файл с расширением bat. DOS была создана компанией Microsoft в 1981 году, совместно с IBM. В настоящее время сценарии на основе командной строки претерпели незначительные изменения по сравнению с первыми своими версиями. Ограниченное количество поддерживаемых функций (около 40) делает сценарии громоздкими. Работа сценария осуществляется с помощью командного интерпретатора. В Windows 9x командный интерпретатор представлен исполняемым файлом command.com, в Windows 2k — cmd.exe. Интерпретатор поддерживает внутренние и внешние команды. Команды, распознающиеся и выполняющиеся непосредственно интерпретатором, называются *внутренними*. Команды операционной системы, представляющие собой отдельные подпрограммы, называются *внешними*.

Внутренние команды интерпретатора:

`AT`, `BREAK`, `CALL`, `CHCP`, `CHDIR`, `CLS`, `CONVERT`, `COPY`, `DATE`, `DEL`, `DIR`, `ECHO`, `EXIT`, `FINDSTR`, `FOR`, `FOTO`, `IF`, `LOADHIGH`, `MKDIR`, `PATH`, `PAUSE`, `PROMPT`, `REM`, `RENAME` `RMDIR`, `SET`, `SHIFT`, `TIME`, `RYPE`, `VER`, `VERIFY`, `VOL` и др.

Все остальные команды — внешние и представлены выполняемыми файлами. Запуск внешних команд осуществляется с помощью команды `START`.

Команды для работы с файловой системой:

`CD`, `ATTRIB`, `COPY`, `XCOPY`, `DIR`, `MKDIR`, `RMDIR`, `DEL`, `DELTREE`, `REN`, `MOVE`, `SUBST`, `VOL`, `LABEL`.

Команды для работы с сетью: `NET`, `PING`, `IPCONFIG`.

Сценарии на базе командной строки обладают ограниченными возможностями и с их помощью очень трудно реализовать необходимый функционал.

Пакет поддержки сценариев на базе командной строки встраивается в ОС Microsoft Windows. Его цена входит в стоимость операционной системы.

## WinBatch

В настоящее время компанией Wilson WindowsWare (**http://www.winbatch.com**) создана программа WinBatch, которая является расширенной версией сценариев на основе командной строки. Данная программа поддерживает около 400 функций, которые можно разбить логически на следующие группы:

- системное администрирование — управление учетными записями домена, восстановление и архивирование ОС, установка ПО, управление планировщиком, управление панелью управления Windows, управление почтой;
- сетевое администрирование — управление учетными записями, определение статуса сервера, архивирование системы, установка принтеров, запуск и остановка сервисов, администрирование ADSI, удаленная установка программ;
- управление данными — манипуляции с файлами, создание PDF-файлов, чтение BAR-кода с COM-порта, поиск и обработка данных, ведение журнала.

WinBatch — платная программа, поставляется в двух вариантах. Варианты отличаются друг от друга поддержкой сетевых технологий и возможностью компиляции написанного скрипта.

## JavaScript

JavaScript — интерпретируемый язык. Интерпретатор JScript интегрирован в интернет-браузеры (Internet Explorer, Opera, Netscape Navigator и др.).

Достоинство JScript — интеграция в ОС и простота программирования. Для создания сценария достаточно иметь любой текстовый редактор. JScript не обладает средствами поддержки объектно-ориентированного программирования и является объектно-базированным языком.

Основной недостаток JScript — отсутствие доступа к файловой системе компьютера. Как следствие, с помощью JScript невозможно получить доступ

к реестру, к аппаратной части рабочей станции. Однако, используя встроенную поддержку объектов, можно получить доступ к файловой системе через ActiveX-компоненты.

Сценарии могут быть не только встроенными в код Web-страницы, но и могут быть автономными файлами с расширением js.

## Visual Basic Script Edition

В настоящее время компанией Microsoft представлено несколько версий Basic для Windows: Visual Basic (VB), Visual Basic for Application (VBA) и Visual Basic Script Edition (VBScript). VBScript — сокращенная версия VB, которую можно рассматривать как пакетный язык (по аналогии с пакетными файлами MS-DOS), используя базовый синтаксис VB. В отличие от VBA, VBScript является интерпретируемым языком программирования.

До создания языка VBScript стандартом скрипта для Web-страниц служил JavaScript, но сходство с C++ сдерживало его широкое применение. При разработке языка VBScript задачей компании Microsoft было создание языка для Web-страниц, совместимого с промышленным стандартом фирмы, для разработок приложений под Windows, т. е. Micorosft Visual Basic. Необходимо было создать язык, который интерпретировался бы так же быстро, как JScript. Чтобы увеличить скорость интерпретации, синтаксис языка был упрощен. Еще одной причиной упрощения VBScript является забота о безопасности; значительным упрощением стал отказ от многочисленных констант, определенных в библиотеке типов VBA.

VBScript не обладает средствами поддержки объектно-ориентированного программирования и является объектно-базированным языком. Язык имеет встроенную поддержку объектов. Сценарии могут быть автономными файлами с расширением vbs или встроенными в код Web-страницы.

## Visual Basic for Application

Visual Basic for Application (VBA) разрабатывается компанией Microsoft с 1993 года. VBA является попыткой компании Microsoft создать общий макроязык для всех продуктов, работающих под Windows на основе VB. Основное отличие VBA от VBScript — поддержка OLE-объектов и библиотеки типов с базовым подмножеством VB в VBA.

## Windows Scripting Host

В 1998 году Microsoft предложила в качестве инструмента разработки и выполнения сценариев для операционной системы Microsoft Windows сервер

сценариев Windows Script Host (WSH) (Windows Script Host). В комплект поставки Microsoft Windows 98 входит WSH версии 1.0, в Microsoft Windows 2000 — WSH 2.0. Сценарии WSH поддерживают два встроенных в Microsoft Windows языка программирования — Microsoft Visual Basic Script Edition (VBScript) и Miscrosoft Java Script Edition (JScript).

WSH предъявляет минимальные требования к объему ОП и является удобным инструментом для создания сценариев регистрации пользователей и автоматизации повседневных задач. В настоящее время используют WSH 5.6, который отличается поддержкой XML. Последнюю версию WSH можно скачать с сервера компании Microsoft **http://msdn.microsoft.com/scripting**.

Сценарии могут запускаться одним из двух способов: с помощью проводника Windows (WSCRIPT.EXE) или из командной строки (CSCRIPT.EXE).

Возможности WSH-сценариев:

- ❒ вывод информации в виде диалоговых окон;
- ❒ считывание информации из потока данных;
- ❒ управление процессами Microosft Windows;
- ❒ работа с локальной сетью — подключение сетевых дисков и принтеров;
- ❒ работа с переменными среды;
- ❒ работа со специальными папками;
- ❒ создание ярлыков;
- ❒ работа с системным реестром;
- ❒ эмуляция нажатия клавиш клавиатуры;
- ❒ управление OLE-объектами.

## Hiddensof AutoIt

Программа AutoIt создана Джонатаном Беннетом (Jonathan Bennett). Разработка данного продукта началась в 1999 году. AutoIt интерпретирует файл-скрипт с расширением aut. Скрипт позволяет симулировать нажатие клавиш клавиатуры, перемещение курсора мыши, выполнение операций с окнами. Язык AutoIt поддерживает около 90 функций. Программа AutoIt имеет следующие отличительные возможности:

- ❒ небольшой размер интерпретатора;
- ❒ самодостаточность интерпретатора;

- наличие компилятора (AUT->EXE);
- наличие декомпилятора (EXE->AUT);
- блокировка клавиатуры и мыши для пользователя на время выполнения скрипта;
- DLL-библиотека, позволяющая вызывать команды AutoIt из VBS и JS.

Программу можно бесплатно загрузить с сайта **http://hiddensoft/AutoIt**.

## KIXtart

Первая версия KIXtart (**http://kixtart.org**) была разработана в 1991 году для создания сценариев загрузки. KIXTart — язык, работающий в среде Microsoft LAN Manager. Простота, скорость и отсутствие конкурентов быстро сделали KIXtart популярным среди администраторов. KIXtart бесплатная программа, поставляющаяся в комплекте с Microsoft Resoure Kit.

В последней версии KIXtart поддерживается Microsoft Windows Server 2003, Windows Vista, Windows XP, Windows 2000, Windows NT 3.x/4.x, всеми версиями Windows 95/98/ME.

Язык KIXtart поддерживает около 50 команд и столько же макросов, более 100 функций. Обладает следующими возможностями:

- вывод информации в виде диалоговых сообщений;
- считывание информации из входного потока;
- подключение сетевых ресурсов;
- расширенная поддержка редактирования реестра;
- поддержка INI-файлов;
- расширенная поддержка операций со строками и массивами;
- сбор информации о пользователе и рабочей станции;
- поддержка OLE-объектов;
- операции с файлами и каталогами;
- создание ярлыков Windows.

## Анализ предлагаемых решений

Сравнение возможностей языков программирования по автоматизации процесса установки приложений (табл. 1.4).

***Таблица 1.4.*** *Сравнение решений для реализации автоматической установки программного обеспечения*

| | **AutoIT** | **WSH+VBScript** |
|---|---|---|
| Управление окнами | + | – |
| Блокировка мыши, клавиатуры | + | – |
| Самодостаточность[1] | + | + |
| Интерпретируемый язык | + | + |
| Компилируемый язык | +/– | – |
| Стоимость продукта, $ | 0 | 0 |

Для автоматической установки любого программного обеспечения рекомендуется использовать AutoIT, поскольку он обладает возможностью на время выполнения скрипта блокировать мышь и клавиатуру, реализовано управление окнами. Одним из достоинств является возможность компилировать и декомпилировать скрипт.

Сравнение инструментов для создания сценариев регистрации пользователей в сети см. в табл. 1.5.

***Таблица 1.5.*** *Сравнение решений для создания сценариев загрузки*

| | **Сценарии на базе командной строки** | **WinBatch + Compiler** | **WSH + VBScript** | **WSH + JScript** | **KIXtart** |
|---|---|---|---|---|---|
| Поддержка ActiveX (OLE) | – | + | + | + | + |
| Стоимость, $ | 0 | 495 | 0 | 0 | 0 |
| Интерпретируемый | + | – | + | + | + |
| Компилируемый | – | + | – | – | – |
| Самодостаточность | + | + | + | + | + |
| Поддержка INI-файлов | – | – | – | – | + |
| Возможность редактирования реестра | – | + | + | + | + |

[1] Здесь и далее под термином "самодостаточность" следует понимать то, что для корректной работы данного продукта не требуется дополнительной инсталляции; достаточно одного файла-приложения.

***Таблица 1.5*** *(окончание)*

| | Сценарии на базе командной строки | Win-Batch+ Compiler | WSH + VBScript | WSH + JScript | KIXtart |
|---|---|---|---|---|---|
| Поддержка массивов | – | + | + | + | + |
| Операции с файлами и каталогами | + | + | + | + | + |
| Инвентаризация | – | + | + | + | + |
| Расширение | BAT | EXE | VBS | JS | KIX |
| Встроен в ОС | + | – | + | + | – |
| Графический интерфейс | – | – | – | – | – |
| Функциональность (1—5) | 1 | 5 | 3 | 3 | 5 |

Сравнение решений, позволяющих обеспечить графический интерфейс и интерактивность работы скрипта, см. в табл. 1.6.

***Таблица 1.6.*** *Сравнение решений, обеспечивающих интерактивную работу*

| | ASP + VBScript (JScript) | HTML + VBScript (JScript) | KIXtart + KIXForms + KIX2EXE | KIXtart + KIXWin |
|---|---|---|---|---|
| Поддержка OLE | + | – | + | + |
| Отображение информации в реальном времени | – | – | + | – |
| Самодостаточность | – (для работы ASP требуется установка IIS на сервере) | + | – (регистрация DLL - библиотеки на каждом PC) | + (KixWin.exe на сервере) |
| Интерактивность | + | + | + | + |
| Передача параметров из KIXtart | – | – | + | + |
| Стоимость, $ | 0 | 0 | 0 | 0 |

Использование одного из перечисленных инструментов не позволит реализовать весь спектр поставленных задач. Для каждого из уже описанных типов сценариев, рекомендуется выбрать свой инструмент. Для создания сценария регистрации пользователей в сети целесообразнее всего использовать KIXTart и как его дополнение — KIXWin, а для создания административных шаблонов по управлению AD, например, — VBScript + WSH.

KIXWin (**http://kixtart.org**) — бесплатная программа, предназначенная для передачи параметров из KixTart в HTML-файл.

# ActiveX-компоненты

Доступ к объектным моделям различных приложений (таких как Word, MSN, Nero и т. д.) осуществляется с помощью ActiveX-компонентов — COM-объектов. В настоящее время Microsoft постепенно отказывается от них и активно использует FrameWork, библиотеки которого не являются COM-объектами. В результате в сценариях интерпретируемых языков напрямую невозможно получить доступ к объектной модели библиотеки, не являющейся COM-объектом. Одним из возможных решений проблемы может быть использование Microsoft Assembly Registration Tool (RegAsm.exe), которая позволяет преобразовать библиотеку. Утилиту можно загрузить с сайта компании Miscrosoft или найти в каталоге `FrameWork: c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727`.

Утилита имеет следующий синтаксис:

```
regasm assemblyFile [/regfile [:regFile]][/tlb [:typeLibFile]]
[/unregister] [/?]
```

Где `assemblyFile` — имя библиотеки, которую необходимо зарегистрировать как COM-объект.

`/regfile [:regFile]` — создает REG-файл для регистрации COM-объекта; не регистрирует библиотеку.

`/tlb [:typeLibFile]` — создает файл-таблицу (*.tlb), в котором описана точка входа в библиотеку, и регистрирует ее как COM-объект.

`/unregister` — удаляет внедренный COM-объект из системы.

`/?` — выводит справку на экран.

Дополнительная информация об утилите RegAsm и полный список ключей находится на сайте Microsoft по адресу:

**http://msdn2.microsoft.com/en-us/library/tzat5yw6(vs.71).aspx**.

# Инструмент разработки сценариев. Primal Script 4.0

В настоящее время существует множество редакторов, позволяющих создавать скрипты. Как правило, редакторы такого рода узкоспециализированы, т. е. поддерживают один или два языка программирования. Редактор Primal Script 4.0, созданный компанией Sapien (**http://www.sapien.com**), имеет встроенную поддержку более чем 30 языков программирования, начиная от HTML, XML и VBScript, заканчивая узкоспециализированными, такими как KIXTart.

## Интегрирование собственных шаблонов

Каждый программист имеет собственные наработки, которые позволяют ему быстро и качественно создавать различные скрипты. Получив возможность интегрировать их в редактор, он сможет еще быстрее их создавать, не тратя время на поиск нужного файла, содержащего шаблон.

Создадим шаблон, в котором осуществляется соединение с Active Directory (далее AD) с помощью ADODB-соединения (листинг 1.2). За его основу возьмем сценарий чтения всех пользователей в AD.

**Листинг 1.2. Чтение данных из Active Directory с помощью ADODB-соединения**

```
domain = "LDAP://" + GetOb-
ject("LDAP://RootDSE").Get("defaultNamingContext")
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.CommandTimeout = 120
objConnection.Provider = "ADsDSOObject"
objConnection.Properties("ADSI Flag")=1+2
objConnection.Properties("User ID")="msk\adminstrator"
objConnection.Properties("Password")="password"
objConnection.Properties("Encrypt Password")=TRUE
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection
objCommand.properties("Page size")=10000
objCommand.properties("Timeout")=300
objCommand.properties("Cache Results")=false
Set st=objconnection.execute("SELECT Samaccountname, description FROM ' "
& Domain & " ' WHERE objectClass='person')
```

```
st.Movefirst
Temp=""
Do Until st.EOF
SamAccountName=""
SamAccountName= St.Fields("Samaccountname").Value

Description=""
A_Description= St.Fields("Description").Value
For Each AA in A_Description
Description= Description+AA
Next

Temp=Temp+ "Имя: "+ SamAccountName + "Описание: " + Description
+chr(13)+chr(10)
st.MoveNext
Loop

Wscript.Echo Temp
```

Из приведенного сценария можно сделать несколько шаблонов: шаблон определения длинного имени домена, соединения с AD, SQL-запрос обращения к объекту, чтения данных из переменной, имеющей тип данных, строка и массив.

Однако ограничимся созданием одного шаблона. По местоположению в скрипте соединение с AD располагается в самом начале, и сразу после него формируется SQL-запрос. Выбор создания именно такого шаблона не случаен. Дело в том, что для успешного соединения с AD и чтения/записи данных необходимы административные права доступа. Как следствие, эти значения параметров необходимо указать, но они не статичны, поэтому для формирования шаблона скрипт необходимо трансформировать (листинг 1.3).

**Листинг 1.3. Шаблон соединения с Active Directory**

```
Admin_Name=" "
Password=" "
Set Domain= GetObject("LDAP://RootDSE").Get("defaultNamingContext")
Short_Domain = mid(Domain, instr(Domain,"=")+1,instr(Domain,",")-
instr(Domain,"=")-1)
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.CommandTimeout = 120
```

```
objConnection.Provider = "ADsDSOObject"
objConnection.Properties("ADSI Flag")=1+2
objConnection.Properties("User ID")= Short_Domain + "\"+Admin_Name
objConnection.Properties("Password")=Password
objConnection.Properties("Encrypt Password")=TRUE
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection
objCommand.properties("Page size")=10000
objCommand.properties("Timeout")=300
objCommand.properties("Cache Results")=false
```

Шаблон, в котором имя и пароль пользователя перенесены в начало скрипта, будет удобно использовать. Он представляет собой текстовый файл с расширением snippet, расположенным в папке C:\Program Files\SAPIEN\PrimalScript Professional\Snippets, если программа установлена по умолчанию. Иерархическую структуру расположения шаблонов и описания функций различных языков определяет структура каталогов и файлы с расширением snippet (рис. 1.4).

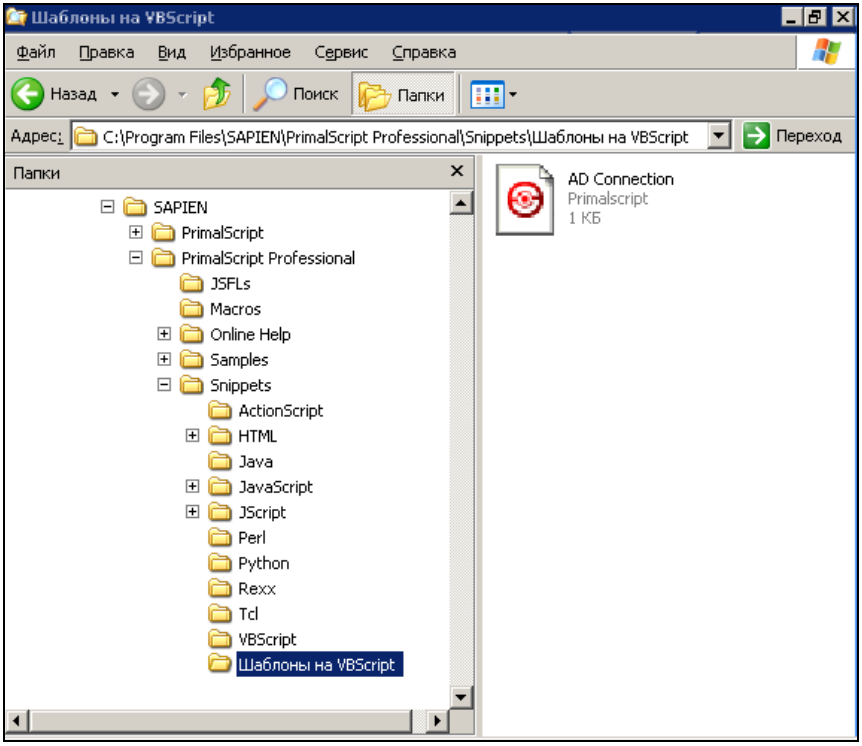



**Рис. 1.4.** Расположение шаблонов в файловой системе

Создадим в каталоге Snippets подкаталог Шаблоны на VBScript, а в нем текстовый файл AD Connection.snippet. Все его содержимое представляет собой шаблон. После создания файла и заполнения его содержимым, необходимо перезапустить Primal Script, чтобы внесенные в файловую структуру изменения вступили в силу.

В редакторе вызовите меню **Snippets Windows**, в котором находится список шаблонов. Для этого необходимо в меню **Tools** выбрать пункт **Options** (рис. 1.5), а в появившемся окне войти в **Environment/Nexus Windows** и сделать активным меню **Snippet Browser**, как это показано на рис. 1.5. Чтобы сделанные изменения вступили в силу, необходимо перезапустить редактор.

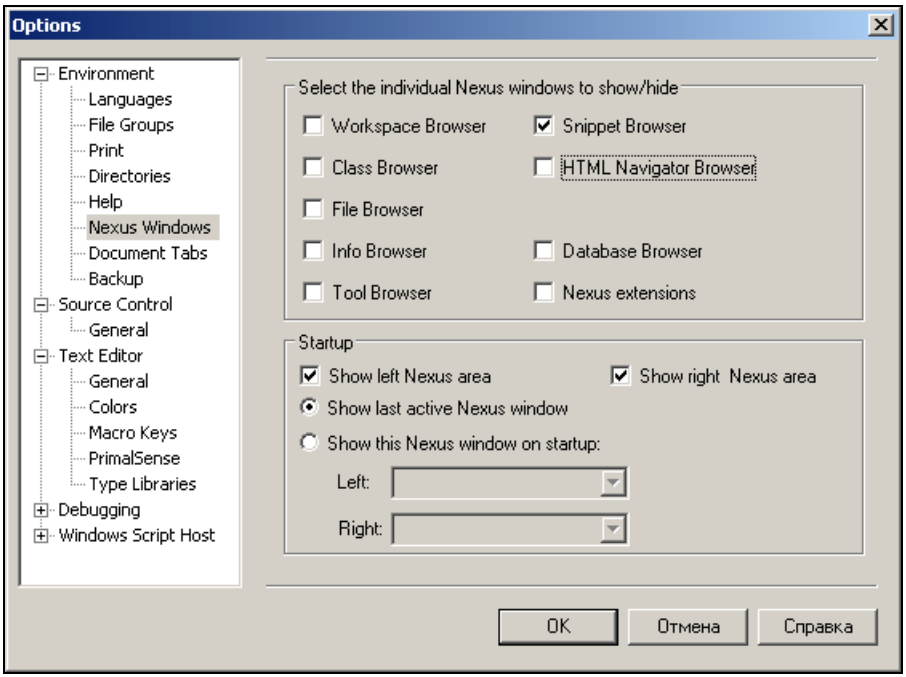



**Рис. 1.5.** Настройка свойств Primal Script 4.0

В навигаторе редактора **Snippets Browser** (рис. 1.6) помимо 10 встроенных подразделов появится еще один — **Шаблоны на VBScript**, в котором будет единственный шаблон — **AD Connection**. Для его использования следует создать файл с расширением vbs и два раза щелкнуть левой кнопкой мыши по имени файла. Шаблон из файла AD Connection.Snippet будет вставлен в сценарий.

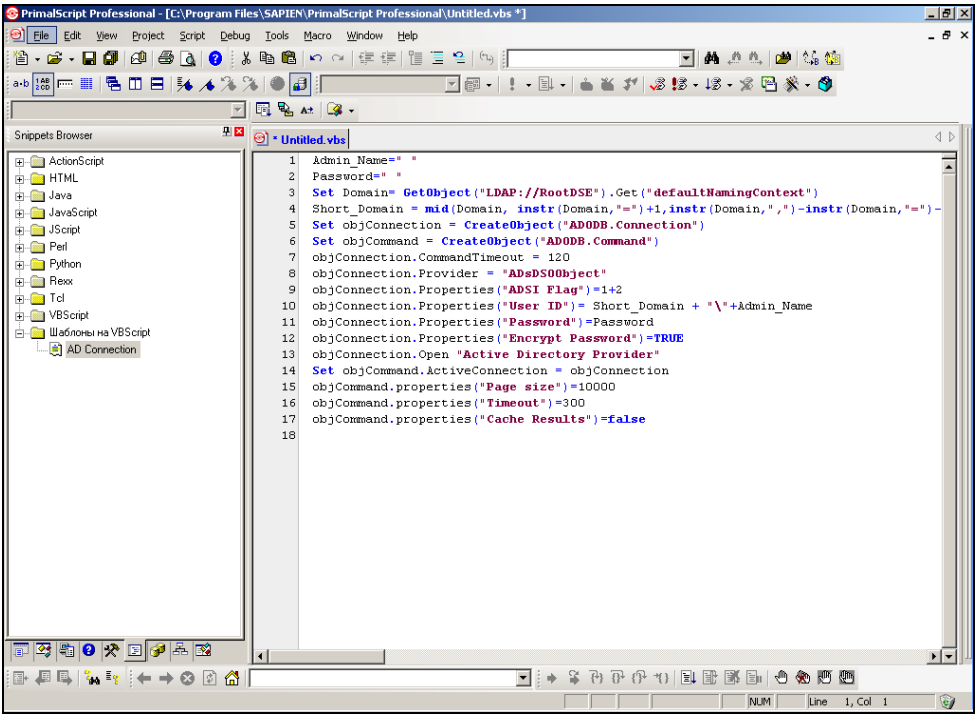



**Рис. 1.6.** Пример использования созданного шаблона

## Создание новой объектной модели

Создание собственных объектных моделей поможет вам избежать ошибок при создании кода. Разберем простой пример. На рис. 1.7 изображен пример объектной модели, которую предстоит описать. В ней присутствует ряд объектов, обладающих наборами свойств.

Существует несколько правил описания модели в файле. По структуре он представляет собой INI-файл. Иерархия описывается с помощью имен разделов. Параметры, которые могут быть объектами или свойствами, различаются символами `P` или `M`, соответственно. Синтаксис строки выглядит следующим образом: сначала пишется идентификатор `P` или `M`, затем — название объекта или свойства, после него — уточняющие параметры. В названии раздела и описании его значения между идентификатором и именем объекта символ пробела не допускается.

Объектная модель описывается в текстовом файле с расширением sense, расположенном в корневом каталоге редактора. Как правило, объектная модель

используется при создании скрипта на одном из языков программирования. В корневом каталоге редактора присутствует несколько файлов с расширением sense. Чтобы описать новый язык программирования, необходимо создать файл с таким расширением и описать его в настройках программы. Иерархическая связь объектов показана на рис. 1.8. Связанные объекты обозначены одинаковыми цифрами. Стрелка, указывающая вверх, обозначает родительский объект, вниз — дочерний.

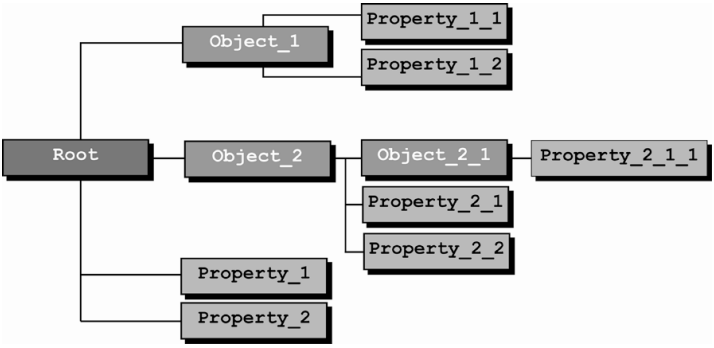



**Рис. 1.7.** Пример объектной модели

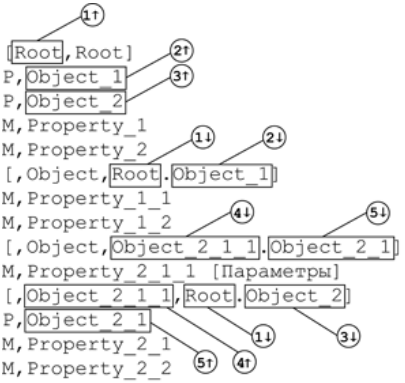



**Рис. 1.8.** Пример описания иерархической модели в SENSE-файле

После того как созданный текст добавлен в конец файла VBScript.sense, необходимо перезагрузить Primal Script и создать или открыть любой файл с расширением vbs. В редакторе: набираем `Root` и видим, что можем выбрать один из объектов: `Object_1` или `Object_2` или одно из свойств объекта `Root` — `Property_1` или `Property_2` (рис. 1.9). Выберем объект `Object_2`. В нем об-

наружим еще два свойства и объект — выберем его. У выбранного объекта `Object_2_1` есть единственное свойство `Property_2_1_1`, которое также выбираем.

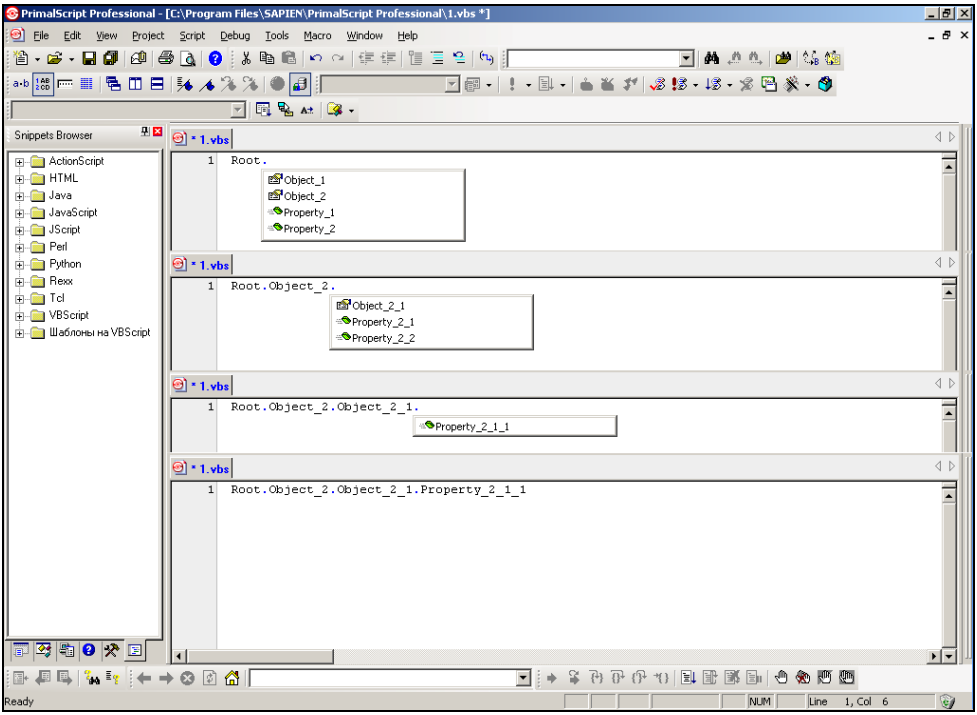



**Рис. 1.9.** Пример использования в Primal Script созданной объектной модели

## Добавление новых функций в существующие языки программирования

Существует множество языков программирования, набор функций которых постоянно расширяется — выходят новые версии. Ярким примером такого языка программирования является KIXTart, который уже несколько лет находится в активной разработке. С выходом новой версии в нем становятся доступны новые функции, которые хотелось бы сделать "понятными" для редактора. Для этого необходимо отредактировать соответствующий файл с расширением sense, например, KIXTart.sense.

Приведем пример добавления абстрактной функции `WriteToXml()`, которая имеет три параметра: имя файла, записываемый текст и кодировку. Причем

кодировка — необязательный параметр. Исходя из этих условий, функция имеет синтаксис:

```
WriteToXml("filename", "doby" [, "encoding"])
```

Для того чтобы редактор воспринял `WriteToXml` как служебное слово, необходимо в файл Kixtart.sense добавить строку

```
M,WriteToXml("filename", "doby" [, "encoding"])
```

Пробел между `M` и названием функции не допускается.

После сохранения файла необходимо перезагрузить редактор, чтобы внесенные в файл изменения вступили в силу.

# Шифрование скриптов

Хотя NTFS позволяет обезопасить файлы от постороннего доступа, однако с помощью некоторых методов (например, утилиты NTFSDOS) можно обойти эти разрешения. В Windows 2k используется файловая система с шифрованием (Encrypting File System, EFS).

## Шифрование файлов из командной строки

Утилита CIPHER позволяет производить шифрование/дешифрование файлов из командной строки. Поддерживаются ключи:

- `/A` — указывает на выполнение команды в отношении всех файлов;
- `/D` — выполнить дешифрование;
- `/E` — выполнить шифрование;
- `/F` — выполнить шифрование, даже если файлы уже зашифрованы;
- `/H` — выполнить операцию также в отношении системных и скрытых файлов;
- `/I` — игнорировать ошибки;
- `/K` — создать новый ключ шифрования для данного пользователя;
- `/Q` — запустить в фоновом режиме;
- `/S` — выполнить операцию в отношении файлов текущего каталога и всех его подкаталогов.

Зашифрованные файлы не могут считываться до загрузки ОС. Файлы, необходимые для запуска ОС, нельзя шифровать.

Для шифрования файлов и папок каталога в фоновом режиме выполните команду:

```
CIPHER /E /A /S /F /Q /H "directory"
```

Где `directory` — каталог, который будет зашифрован.

## Использование Microsoft Script Encoder

Шифровальщик Microsoft Script Encoder позволяет защитить сценарии. По умолчанию он поддерживает следующие типы файлов: ASA, ASP, CDX, HTM, HTML, JS, SET и VBS. Для шифрования используйте следующий синтаксис:

```
SCRENC inputfile outputfile
```

`inputfile` — шифруемый файл, а `outputfile` — зашифрованный файл. Кроме того, этот шифровальщик поддерживает ряд параметров командной строки (табл. 1.7).

***Таблица 1.7.*** *Параметры утилиты Microsoft Script Encoder*

| Параметр | Описание |
|---|---|
| `/E` | Определить известные расширения для файлов неопознанных типов |
| `/F` | Перезаписать входной файл его зашифрованной версией |
| `/L` | Использовать язык JScript или VBScript |
| `/S` | Работать в фоновом режиме |
| `/X` | Не включать в ASP-файл директиву `@language` |

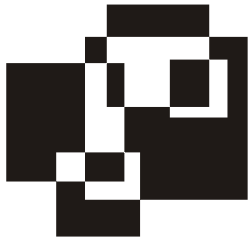
Перед зашифровкой сценариев, рекомендуется выполнить резервное копирование сценариев, поскольку вернуть файл в исходное состояние после зашифровки невозможно.

## Шифрование KIX-сценариев

Созданные сценарии регистрации пользователей в сети на языке KIXTart можно шифровать, компилируя сценарий в бинарный файл. Эта возможность реализована в KIXTart, начиная с версии 4.50:

```
Kix32.exe -t script.kix
```

Результат действия команды — бинарный файл с расширением kx. Преимущество заключается в увеличении скорости работы сценария и в защите, т. к. из бинарного файла невозможно получить первоисточник.



# Глава 2

# Visual Basic Script Edition

Один из основных скриптовых языков — Visual Basic Script Edition (VBScript). Он обладает обширными возможностями, при этом очень легок и понятен. Обычно с него начинают изучать программирование в Windows. Для создания функциональных скриптов необходимо хорошо знать синтаксис языка, изучить его тонкости.

## Основы синтаксиса

### Переменные

Важнейшая составляющая любого языка программирования — переменные. Их основная задача — постоянное/временное хранение данных, используемых в программе. *Переменная* (Variable) — именованный фрагмент памяти, выделяемый или резервируемый для сохранения данных. Главной характеристикой переменной является тип, который определяет ее содержание и объем памяти.

### Соглашение об именах

При присвоении имен сценариям, константам, переменным и аргументам необходимо учитывать следующие условия и требования:

- в VBScript не различаются строчные и прописные буквы, поэтому переменные, например, `Indentificator` и `idenTifiCator`, указывают на одну область памяти;
- имена переменных в скрипте должны начинаться с букв;

- имена переменных не могут содержать пробел, точку (.), восклицательный знак (!) и следующие спецсимволы: @, &, $, #;
- имена не должны содержать более 255 символов;
- нельзя присваивать переменным зарезервированные имена; совпадение может привести к ошибке в работе программы или к ее зависанию.

## Комментарии

Комментарии полезно использовать для документирования содержимого создаваемого сценария. Комментарий предваряют апострофом (‘). Его можно помещать в любом месте строки (листинг 2.1).

**Листинг 2.1. Использование комментария**

```
‘ Это комментарий
Count_Printers=10 ‘ Кол-во принтеров
```

## Непрерывные строки

Длинные строки снижают читабельность сценариев. Если в программе их слишком много, то при редактировании листинга теряется много времени на горизонтальную прокрутку строк во время поиска нужного фрагмента. Для улучшения восприятия текста сценария рекомендуется разбивать строку на несколько подстрок с помощью символа "_" (листинг 2.2).

**Листинг 2.2. Разбиение длинной строки на несколько подстрок**

```
WScript.Echo "SubString1" &_
"SubString 2."
```

Обнаружив в конце строки символ "_", интерпретатор языка считает ее непрерывной и рассматривает следующую строку как часть текущего оператора.

## Типы данных

VBScript поддерживает только один тип данных — Variant, имеющий несколько подтипов (табл. 2.1).

***Таблица 2.1.*** *Поддерживаемые в VBScript подтипы данных Variant*

| Подтип | Описание |
|---|---|
| `Empty` | Пустое значение. Равно нулю для цифровых значений и пустой строке (`""`) для строковых переменных |
| `Null` | Не содержит данных |
| `Boolean` | Булевы значения `True` (1) и `False` (0) |
| `Byte` | Целые числа в диапазоне от 0 до 255 |
| `Integer` | Целые числа в диапазоне от –32 768 до 32 767 |
| `Currency` | Числа в диапазоне от –922 337 203 685 477.5808 до 922 337 203 685 477.5807 |
| `Long` | Целые числа в диапазоне от –2 147 483 648 до 2 147 483 647 |
| `Single` | Числа с плавающей точкой от –3.402 823 E38 до –1.401 298 E–45 для отрицательных значений; от 1.401 298 E–45 до 3.402 823 E38 для положительных значений |
| `Double` | Числа с плавающей точкой в диапазоне от –1.79 769 313 486 232 E308 до –4.94 065 645 841 247 E–324 для отрицательных значений и от 4.94 065 645 841 247 E–324 до 1.79 769 313 486 232 E308 для положительных значений |
| `Date` (`Time`) | Формат даты в диапазоне от 1 января 100 года до 31 декабря 9999 года |
| `String` | Строка: длина может достигать примерно двух биллионов символов |
| `Object` | Содержит объекты |
| `Error` | Содержит номер ошибки |

В дальнейшем, программируя Active Directory (AD), часто необходимо определить тип данных переменной, которая хранится в том или ином поле AD. Для определения типа переменной используйте функцию `VarType()`, возвращающую число, которому соответствует подтип (табл. 2.2):

```
VarType(VarName)
```

Где `Varname` — имя переменной, содержащей значение.

***Таблица 2.2.*** *Соответствие строковых констант подтипов числовым значениям*

| Константа | Значение | Описание |
|---|---|---|
| vbEmpty | 0 | Empty (пустое значение) |
| vbNull | 1 | Null (не содержит данных) |
| vbInteger | 2 | Integer |
| vbLong | 3 | Long |
| vbSingle | 4 | Single (число с плавающей точкой) |
| vbDouble | 5 | Double (число с плавающей точкой) |
| vbCurrency | 6 | Currency |
| vbDate | 7 | Date |
| vbString | 8 | String |
| vbObject | 9 | Object |
| vbError | 10 | Error |
| vbBoolean | 11 | Boolean |
| vbVariant | 12 | Variant (используется только с массивами) |
| vbDataObject | 13 | Object |
| vbByte | 17 | Byte |
| vbArray | 8192 | Array |

Работая с массивами, функция `VarType()` никогда не возвращает значение 8192. Это связано с тем, что элементы массива содержат данные, которые также соответствуют одному из типов данных. Например, если элементами массива являются числа, относящиеся к типу `Integer`, то функция будет возвращать значения 2+8192, т. е. 8194.

## Оператор *Option Explicit*

В VBScript допускается неявное объявление переменных. При этом переменная создается без предварительного объявления операторами `Dim`, `Private`, `Public` или `ReDim`. Разрешив неявное объявление переменных в сценариях, велик риск пропустить синтаксическую ошибку в имени переменной во время программирования. В том случае, если допущена ошибка, VBScript просто объявит новую переменную и создаст ее, вследствие чего программа может работать некорректно. Для обнаружения ошибок-опечаток такого рода в пер-

вую строку программы помещают оператор `Option Explicit`. Он сигнализирует интерпретатору о переменных, не объявленных явно.

Если в скрипте используется оператор `On Error Resume Next`, то инструкция `Option Explicit` игнорируется. Инструкция `On Error Resume Next` чаще всего применяется, если в программе осуществляется упорядочивание массивов.

## Константы

В VBScript константы объявляются с помощью зарезервированного слова `Const` (листинг 2.3). Они могут быть строковыми или числовыми. Значения констант не переопределяются. Для задания даты в виде константы используйте символ "#" по краям даты.

**Листинг 2.3. Объявление констант**

```
Const String = "HP LaserJet 4100"
Const Number = 25
Const Today_Data=#01-02-2004#
```

## Объявление переменных

Объявляемую переменную в скрипте предваряют ключевым словом `Dim`, `Public` или `Private` (листинг 2.4).

**Листинг 2.4. Объявление одной переменной в строке**

```
Dim Printer_Name
```

Объявление нескольких переменных одного типа осуществляется следующим образом: после ключевого слова через запятую перечисляют переменные (листинг 2.5).

**Листинг 2.5. Объявление нескольких переменных в одной строке**

```
Dim Printer_Name, Server_Name, Driver_Name
```

## Присвоение значений переменным

Для присвоения значения переменной сначала указывается имя переменной, затем знак равенства, за которым следует значение переменной. Оно может быть задано в явном или неявном виде.

Переменной можно присвоить строковое или числовое значение (листинг 2.6), значение элемента массива.

**Листинг 2.6. Явное присвоение значений переменным**

```
Count_Printers=10
Printer_Name="HP LaserJet 1200 PS"
```

**Листинг 2.7. Неявное присвоения значений переменным**

```
t=10
b=t
```

Приведенный пример в листинге 2.7 эквивалентен явному объявлению переменной:

```
b: b=10
```

## Объявление массивов

Различают статические и динамические, одномерные и многомерные массивы (табл. 2.3). Для статического массива память выделяется один раз в полном объеме, а динамический растет или уменьшается, в зависимости от объема записанных в него данных.

Одномерный массив является частным случаем многомерного массива. Рассмотрим общий случай — многомерный массив. Массивы объявляются с помощью оператора `Dim`. После имени массива в круглых скобках через запятую указывают границы каждого измерения.

Особенности объявления массивов:

- индексация элементов массивов начинается с нуля;
- в многомерном массиве может быть до 60 измерений;
- динамические одномерные и многомерные массивы объявляются одинаково.

***Таблица 2.3.*** *Примеры объявления многомерного и одномерного массивов*

| Типы массивов | Одномерный массив | Многомерный массив |
|---|---|---|
| Статический | `Dim Array(9)` | `Dim Array(9,19,5)` |
| Динамический | `Dim Array()` | `Dim Array()` |

## Переопределение размеров массива

Размер массива может быть переопределен, если он динамический. Эта процедура осуществляется с помощью команды `ReDim`, имеющей следующий синтаксис:

```
ReDim [Preserve] array_name(subscripts) [,array_name(subscripts)]…
```

Где `Preserve` — переопределяет размер массива, сохраняя существующие данные в массиве, если параметр не указан, то все его ячейки обнуляются.

`Array_name` — название массива.

`Subscripts` — количество элементов в массиве.

При изменении размера многомерного массива, корректировке поддается только размер последнего измерения.

## Определение границ массива

Определение размера выделенной памяти для массива осуществляется с помощью команд `lbound(array_name)` и `ubound(array_name)`, где `array_name` — название массива, границы которого необходимо определить. Команда `lbound` определяет нижнюю границу массива, а `ubound` — верхнюю границу. Данный прием часто используется для чтения всех элементов любого массива.

**Листинг 2.8. Перебор всех элементов массива**

```
For i=Lbound(Array_name) to Ubound(Array_name)
       Msgbox Array_name(i)
Next
```

## Операторы

*Оператор* (Operator) — это символ (операнд), с помощью которого осуществляется комбинация простых выражений в более сложные. Большинство операндов обозначаются специальными символами. Операторы логически делятся на несколько групп: арифметические, сравнения, логические, конкатенации. В свою очередь эти группы относятся к унарным и бинарным операторам. Унарные операторы применяются к одному операнду, а бинарные к двум. К унарным операндам относятся унарный плюс и унарный минус, характеризующие положительные и отрицательные числа. Унарные и бинарные операнды, за исключением операндов конкатенации, приведены в табл. 2.4.

***Таблица 2.4.*** *Операторы VBScript*

| Арифметические | | Сравнения | | Логические | |
|---|---|---|---|---|---|
| **Описание** | **Символ** | **Описание** | **Символ** | **Описание** | **Символ** |
| Возведение в степень | ^ | Эквивалентность | = | Логическое Не | Not |
| Унарное вычитание | - | Неравенство | <> | Логическое И | And |
| Умножение | * | Меньше чем | < | Логическое Или | Or |
| Деление | / | Больше чем | > | Логическое исключающее Или | Xor |
| Деление нацело | \ | Меньше или равно | <= | Логическая эквивалентность | Eqv |
| Арифметический модуль | Mod | Больше или равно | >= | Логическая импликация | Imp |
| Сложение | + | Эквивалентность объектов | Is | | |
| Вычитание | - | | | | |

## Операторы конкатенации

К операторам конкатенации относятся строковые операторы, позволяющие соединять строки. В VBScript для соединения строковых выражений используются два операнда: сложение (+) и амперсанд (&). Результатом объединения двух строк является строковое выражение типа `String`. Часто в строку необходимо добавлять спецсимволы (табл. 2.5).

***Таблица 2.5.*** *Константы VBScript*

| Константа | Значение | Описание |
|---|---|---|
| vbCr | Chr(13) | Перевод каретки |
| VbCrLf | Chr(13) & Chr(10) | Эквивалентно нажатию клавиши <Enter> |
| vbLf | Chr(10) | Переход на новую строку |

***Таблица 2.5*** *(окончание)*

| Константа | Значение | Описание |
|---|---|---|
| vbNewLine | Chr(13) & Chr(10) или Chr(10) | Формирование новой строки |
| vbTab | Chr(9) | Горизонтальная табуляция |
| | Chr(…) | Для вставки спецсимвола вместо троеточия указывают номер символа, соответствующий таблице ASCII1 или ASCII2 |

Если одно из складываемых выражений не является выражением типа String, то его необходимо преобразовать к типу String с помощью функции Cstr().

## Преобразование типов

Поскольку VBScript не осуществляет автоматического преобразования типов данных, то, создавая скрипт, необходимо использовать функции преобразования типов при использовании операторов конкатенации с арифметическими операторами. Основные функции преобразования типов приведены в табл. 2.6.

***Таблица 2.6.*** *Преобразование типов в VBScript*

| Функция | Аргумент | Возвращаемые типы данных |
|---|---|---|
| CBool() | Строка или числовое выражение | Boolen |
| CByte() | Целое число от 0 до 255 | Byte |
| CCur() | Число в диапазоне от –92 237 720 368 547 765 808 до 92 237 720 368 547 765 807 | Currency |
| CDate() | Дата | Data |
| CInt() | Число от –32 768 до 32 767 | Integer |
| CLng() | Число от –2 147 483 648 до 2 147 483 647 | Long |
| CSng() | Любое число с дробной частью | Single |
| CStr() | Любая допустимая строка, числовое выражение | String |
| CVar() | Весь диапазон значений для строковых переменных типа String и чисел типа Integer | Variant |
| CVErr() | Любой допустимый код ошибки | Variant |

# Инструкции

Для реализации сложной алгоритмической логики используются инструкции перехода и циклы. Различают условные и безусловные инструкции перехода. К условным относятся инструкции `If…Then` и `Select…Case`.

## Инструкции *If…Then*

```
If String1 Then
  [Blok1]
[Else If String2 Then
  [Block2]]
…
[[Else]
  [BlockN]]
End If
```

Где `String` — выражение проверки условия, в котором используются логические операторы сравнения. В выражении могут использоваться логические операторы `Or` или `And`.

`Block` — исполняемый оператор. Может содержать функции и процедуры.

`Else If` — блок проверки дополнительного условия. Количество инструкций `Else If` не ограничено.

`Else` — блок операторов, выполняемых в случаях несоблюдения всех предыдущих условий. Инструкция `Else` может быть только одна.

**Листинг 2.9. Использование инструкции `If…Then`**

```
If n1<10 And n2>=50 Then
  Result="1"
Else If n1>100 Or n2=10 Then
  Result = Detect()
Else
  Result="3"
End If
…
Function Detect()
Detect="2"
End Function
```

В приведенном в листинге 2.9 примере осуществляется анализ значений переменных n1 и n2. Если n1 меньше 10 и n2 больше или равно 50, то переменная Result принимает значение 1. Если же n1 больше 100 или n2 равно 10, то Result=2. Во всех остальных случаях переменной Result присваивается значение 3.

## Инструкция *Select…Case*

Инструкция Select…Case используется для проверки переменной на соответствие нескольким условиям. При этом в зависимости от значения переменной можно исполнить один из блоков кода (листинг 2.10).

```
Select Case Var
[Case Value1
  [String1]]
[Case Value2
  [String2]]
…
[Else Case
  [StringN]]
End Case
```

Где Var — переменная, значение которой является проверяемым условием выбора.

Value — значение или текст, относящийся к проверяемому условию.

String — инструкция, выполняемая при соблюдении условия.

**Листинг 2.10. Использование инструкции Select…Case**

```
Select Case A
Case 1
  A="1"
Case 2
  A="2"
Else Case
  A="A>2, A<1"
End Select
```

## Инструкция безусловного перехода

Инструкция безусловного перехода GoTo позволяет изменить последовательность выполнения программного кода без постановки предварительных ус-

ловий (листинг 2.11). Инструкция работает в пределах только одной функции или процедуры. В программном коде метка, обозначающая место, куда будет выполнен переход, отделяется от основного текста двоеточием (:). Данной инструкцией не рекомендуется пользоваться, поскольку она сильно снижает скорость работы программы. Листинг, изобилующий операторами `GoTo`, не читаем.

**Листинг 2.11. Использование оператора `GoTo`**

```
begin:
If t>10 Then
  T=5
  GoTo begin
End if
```

# Управляемые циклы

Управляемые циклы позволяют выполнять последовательность инструкций необходимое количество раз. Для прерывания циклов используются условия. Инструкции, предназначенные для многократного выполнения, называются телом цикла, а инструкция, определяющая число повторений — заголовком. В VBScript различают три вида циклов: с параметрами, с предусловием, с постусловием.

## Циклы с параметрами: оператор *For…Next*

Оператор `For…Next` относится к циклам с параметрами. Он позволяет организовывать заранее оговоренное количество повторов (листинг 2.12).

```
For counter=start To end
  block
Next
```

Где `counter` — переменная цикла.

`start`, `end` — числа, описывающие диапазон изменения переменной.

`block` — исполняемые операторы в цикле.

**Листинг 2.12. Использование оператора `For…Next`**

```
b=0
       For a=1 to 10
             b=b+1
       Next
```

# Циклы с условием

Цикл с предусловием/постусловием позволяет задать повторение инструкции, если заранее не известно количество циклов. К таким циклам относятся `Do…While`, `For While…Wend` и `Each…Next`.

## Цикл *Do…While*

Цикл `Do…While` позволяет выполнять группу операторов до тех пор, пока не будет нарушено заданное условие. Циклы могут быть двух категорий: проверяющие выполнение условия в начале цикла (с предусловием) и после цикла (с постусловием). Циклы по своей структуре идентичны, за исключением того, что в циклах с постусловием значение условия выхода из цикла должно быть `False`.

- Синтаксис (с предусловием):

```
Do {While|Until} string
Block
Exit Do
Loop
```

- Синтаксис (с постусловием):

```
Do
Block
Exit Do
Loop {While|Until} string
```

Где `String` — выражение проверки условия, в которой используются логические операторы сравнения. В выражении также могут использоваться логические `Or` или `And`.

`Block` — тело цикла; может содержать функции и процедуры.

`Exit Do` — оператор безусловного выхода из цикла.

**Листинг 2.13. Использование оператора `Do…While`**

```
Do While i<10
  t=detect_value()
  If t>6 Then
  Exit Do
  End If
Loop
…
```

```
Function detect_value()
…
detect_value=…
End Function
```

В приведенном в листинге 2.13 примере цикл выполняется до тех пор, пока значение переменной `t` не превысит 6. Значение переменной передается в тело программы функций `detect_value()`.

## Цикл *While…Wend*

```
While string
       [block]
Wend
```

Где `String` — выражение проверки условия, в котором используются логические операторы сравнения.

`Block` — тело цикла; может содержать функции и процедуры.

При каждом выполнении тела цикла осуществляется проверка условия, описанного в строке `string`. Если `string=True` (истина), то осуществляется чтение тела, если же нет — содержимое цикла игнорируется и осуществляется переход к следующей итерации. Таким образом, если условие `string` не было удовлетворено на первой итерации, то цикл ни разу не выполняется. Он выполняется до тех пор, пока условие `string` не примет значения `False`.

## Оператор *For…Each*

Оператор `For…Each` позволяет создать цикл для обработки всех элементов массива (листинг 2.14).

```
For Each var In array_name
      block
Next
```

Где `var` — переменная, содержащая последовательно значения всех элементов массива.

`array_name` — имя массива.

`block` — исполняемые операторы в цикле.

**Листинг 2.14. Использование оператора `For…Each`**

```
Dim array1(2)
…
string=""
```

```
For Each a In array1
        string=string+ctr(a)+chr(13)
Next
```

# Подпрограммы и классы

В любой большой программе существуют подпрограммы, которые призваны сократить программный код и повысить производительность программы. Известны два типа подпрограмм: процедуры и функции. Синтаксическая структура подпрограмм обоих типов одинакова, однако каждая из них предназначена для решения своего круга задач и обладает уникальными свойствами. Описание подпрограммы в общем случае состоит из трех основных частей:

- заголовок в теле программы, с помощью которого осуществляется вызов подпрограммы;
- описание подпрограммы, в которой содержатся различные переменные;
- тело подпрограммы, в которой осуществляются операции с переданными или передаваемыми параметрами.

## Функция

Подпрограмма функция используется в том случае, если вызывающая подпрограмма должна вернуть в тело основной программы переменную или массив (листинг 2.15). Возвращенное функцией значение всегда ассоциируется с ее именем.

```
…
t=name(parameters)
…
Function name(parameters)
        …
        name=…
        Exit Function
        …
End Function
```

Где `name` — описывает название функции; совпадает с именем переменной, значение которой передается в тело основной программы.

`parameters` — содержит названия передаваемых параметров.

`Exit Function` — безусловный выход из функции.

**Листинг 2.15. Использование функций**

```
For i=0 to 30
       a=a+2
       b=b+1
       c=c+ Mathematic(a,b)
       c=c-3
Next
…
Function Mathematic(t,q)
       If t<0 Then
              Mathematic =t+q
       Else
              Exit Function
       End If
End Function
```

## Процедуры

Процедуры могут вызываться из любого места программы. В отличие от функции, процедура не может передавать никаких параметров в тело основной программы, однако самой процедуре можно передавать параметры двумя способами (листинг 2.16):

- косвенный — передаваемая переменная является глобальной переменной;
- прямой — переменная передается в виде параметра самой функции; значения можно передавать по сслылке (`ByRef`) и по значению (`ByVal`).

Синтаксис процедуры:

```
…
name parametres
…
       Sub name(parametres)
              …
       End Sub
```

**Листинг 2.16. Использование процедур**

```
Dim result ' Глобальная переменная
Summa 16,100
Msgbox result
```

```
…
Sub Summa(t1,t2)
       result=(t1+t2)/10
End Sub
```

Для вызова процедуры не обязательно указывать ключевое слово `Call`. Однако, если оно используется, необходимо заключать параметры в скобки. Операторы

```
Call Summa(t1,t2)
```

и

```
Summa t1, t2
```

эквивалентны.

## Передача значений параметров с помощью ключевых слов *ByRef* и *ByVal*

Если необходимо указать, что значение параметра передается процедуре по ссылке (by reference, `ByRef`) или по значению (by value, `ByVal`), в объявлении функции или процедуры перед именами параметров размещают соответствующее ключевое слово — `ByRef` или `ByVal`. По умолчанию, если не указан способ передачи параметра, то интерпретатор VBScript считает, что значение параметра передано по ссылке `ByRef`. Разница между чтением значений по ссылке и по значению заключается в том, что, передавая данные по ссылке, функция или процедура получает возможность изменять значения переменных (листинг 2.17).

**Листинг 2.17. Передача значений параметров по ссылке и по значению**

```
Summa 16,100
MsgBox result
       Sub Summa(ByVal t1, ByValt2, ByRef result)
              result=(t1+t2)/10
       End Sub
```

## Классы

В VBScript поддержка классов реализована начиная с пятой версии (листинг 2.18).

**Листинг 2.18. Описание класса и обращения к его объектам**

```
Class UserProperty
       Dim FName
       Dim SName
       Dim Email
End Class

Set Upr0=new UserProperty ' создание экземпляра класса UserProperty
Upr0.FName="First Name of U0"
Upr0.SName="Second Name of U0"
Upr0.EMail="Email Address of U0"

Set Upr1=new UserProperty ' создание еще одного экземпляра класса User-
Property
Upr1.FName="First Name of U1"
Upr1.SName="Second Name of U1"
Upr1.EMail="Email Address of U1"

MsgBox "Пользователь 1: " & chr(13) & "Имя: " & Upr0.FName & chr(13) &
"Фамилия: " & Upr0.SName& chr(13) & "Почта: " & Upr0.EMail

MsgBox "Пользователь 2: " & chr(13) & "Имя: " & Upr1.FName & chr(13) &
"Фамилия: " & Upr1.SName& chr(13) & "Почта: " & Upr1.Email
```

Приведенный пример имеет один недостаток: в программе создается огромное количество экземпляров класса. Объявляя в классах массивы, нет необходимости создавать множество экземпляров одного класса. Использование массивов позволит эффективно управлять памятью (листинг 2.19).

**Листинг 2.19. Использование статических массивов при работе с классами**

```
Class UserProperty
       Dim FName(99)
       Dim SName(99)
       Dim Email(99)
End Class

Set Upr=new UserProperty ' создание экземпляра класса UserProperty

Upr.FName(0)="First Name of User1"
```

```
Upr.SName(0)="Second Name of User1"
Upr.EMail(0)="Email Address of User 1"

Upr.FName(1)="First Name of User2"
Upr.SName(1)="Second Name of User2"
Upr.EMail(1)="Email Address of User 2"

MsgBox "Пользователь 1: " & chr(13) & "Имя: " & Upr.FName(0) & chr(13) &
"Фамилия: " & Upr.SName(0) & chr(13) & "Почта: " & Upr.EMail(0)
MsgBox "Пользователь 2: " & chr(13) & "Имя: " & Upr.FName(1) & chr(13) &
"Фамилия: " & Upr.SName(1) & chr(13) & "Почта: " & Upr.EMail(1)
```

В конце работы сценарий выводит два сообщения, содержащих имя, фамилию и почту для каждого из пользователей (рис. 2.1 и 2.2).




**Рис. 2.1.** Вывод свойств пользователя `User1`

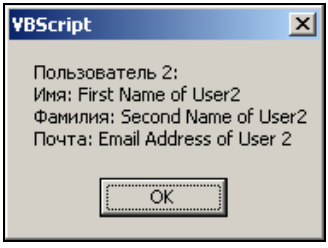



**Рис. 2.2.** Вывод свойств пользователя `User2`

Использование классов позволяет оптимизировать работу программы и сократить программный код.

## Получение свойств обработчика языка

Иногда бывает необходимо получить свойства обработчика языка, такие как версия языкового интерпретатора (табл. 2.7).

***Таблица 2.7.*** *Свойства интерпретатора языка VBScript*

| Параметр | Описание |
|---|---|
| `ScriptEngine` | Возвращает строку, идентифицирующую язык, поддерживаемый обработчиком |
| `ScriptEngineMajorVersion` | Возвращает старший номер версии разработчика языка |
| `ScriptEngineMinorVersion` | Возвращает младший номер версии разработчика языка в виде строки |
| `ScriptEngineBuildVersion` | Возвращает номер сборки разработчика языка |

Следующий скрипт на VBScript запрашивает свойства обработчика языка и выводит результат в диалоговом окне (листинг 2.20, рис. 2.3).

**Листинг 2.20. Скрипт получения свойств обработчика языка**

```
t1 = "Язык программирования " & ScriptEngine & chr(13)
t2 = "Версия: " & ScriptEngineMajorVersion & "."
&ScriptEngineMinorVersion
 t3 = " Build " & ScriptEngineBuildVersion
MsgBox t1 & t2 & t3
```

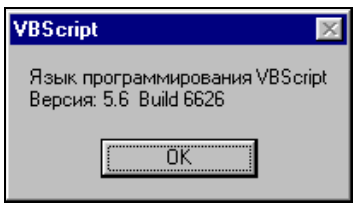



**Рис. 2.3.** Свойства обработчика языка

## Управление свойствами объектов

Для любого класса можно определить собственные свойства и методы. Свойства полей объектов класса описываются с помощью инструкций `Property Let` и `Property Get`. Процедура `Property Let` позволяет получить данные в описанном формате, а `Property Get` — записать их. Пример использования обеих инструкций показан в листинге 2.21, результат — на рис. 2.4.

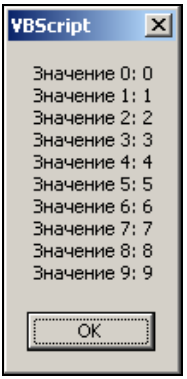



**Рис. 2.4.** Использование инструкций `Property Let` и `Property Get`

**Листинг 2.21. Использование инструкций `Property Let` и `Property Get`**

```
Class UserArray
Dim P_Array()

 Public Property Get ArrayItem(index)
              ArrayItem = P_Array (index)
 End Property

 Public Property Let ArrayItem(index, value)
              P_Array(index) = "Значение " & index & ": "& value &
chr(13)
 End Property

 Private Sub Class_Initialize()
              ReDim Preserve P_Array(Q-1)
 End Sub
End Class

Q=IputBox( "Введите размер массива (кол-во элементов)")
Set CLS = new UserArray
For i=0 To Q-1
CLS.ArrayItem(i)=i
Next
t=""
```

```
For i=0 To Ubound(CLS.P_Array)
t=t+cstr(CLS.P_Array(i))
Next
MsgBox t
```

## Создание методов

Использование методов — наилучшее решение для оптимизации кода и сокращения количества ошибок. Приведем пример создания и использования метода `Add`, позволяющего добавить параметры пользователя комплексно (листинг 2.22, рис. 2.5). Через запятую указываются имя пользователя и его фамилия, затем данные распределяются в соответствующие массивы.

**Листинг 2.22. Создание и использование метода `Add`**

```
Class UserProperty
       Dim FName
       Dim SName
       Sub Add(FirstName, SecondName)
              FName=FirstName
              SName= SecondName
       End Sub
End Class

Set User1=new UserProperty
User1.Add "Ivan", "Ivanov"
MsgBox User1.FName & " " & User1.SName
```

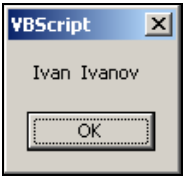



**Рис. 2.5.** Использование свойств класса

## Использование инструкции *With*

Для обращения к свойствам или методам объекта обычно используется следующий синтаксис, построенный по шаблону `объект.свойство` — листинги 2.23 и 2.24.

**Листинг 2.23. Скрипт без использования инструкции `With`**

```
User1.FName=…
User1.SName=…
User1.EMail=…
User1….=…
```

**Листинг 2.24. Скрипт с использованием инструкции `With`**

```
With User1
        .FName=…
        .SName=…
        .Email=…
        ….=…
End With
```

Оператор `With` не поддерживает VBScript версии ниже 5.0.

# Интерактивная работа скриптов

## Ввод информации с помощью функции *InputBox()*

Вызов диалогового окна ввода информации осуществляется с помощью функции `InputBox()`.

```
result=InputBox(text[, title][, default][, xpos][, ypos][, helpfile, context])
```

Где `text` — текстовое сообщение, выводимое в диалоговом окне; если текст необходимо расположить на нескольких строках, используйте перевод каретки `chr(13)`.

`title` — необязательный параметр, определяющий заголовок окна сообщения.

`default` — значение по умолчанию, выводящееся в тестовом поле сразу после выполнения функции `InputBox()`.

`xpos`, `ypos` — координаты верхнего левого угла диалогового окна; если значения не заданы, то VBScript размещает диалоговое окно в центре рабочего стола.

`helpfile`, `context` — ссылка на файл, содержащий справку с указанием индекса, по которому находится необходимое сообщение.

`result` — строка, введенная в текстовом поле.

Все параметры, кроме `result` и `text`, обязательны. Если какой-либо необязательный параметр не задан, то VBScript использует значение по умолчанию. Чтобы пропустить один из параметров, необходимо оставить его место между запятыми пустым.

**Листинг 2.25. Использование функции `InputBox()`**

```
String1= InputBox("Hello",,"Welcome")
MsgBox String1
```




**Рис. 2.6.** Работа функция `InputBox()`

В приведенном в листинге 2.6 и рис. 2.6 название окна не задано, поэтому интерпретатор VBScript подставит значение по умолчанию (VBScript). Остальные необязательные параметры, кроме `default` (`Welcome`), также будут подставлены по умолчанию, т. к. они не описаны.

## Отображение информации с помощью функции *MsgBox()*

Существует два синтаксиса функции `MsgBox()`. Отличие заключается в том, что один из них применяется, если необходимо считать код нажатой клавиши выводимого окна. Приведем оба синтаксиса:

```
MsgBox text[, buttons][, title][, helpfile, index]
```

или

```
result= MsgBox (text[, buttons][, title][, helpfile, index])
```

Где `text` — текстовое сообщение, выводимое в диалоговом окне.

`buttons` — необязательный параметр, определяющий набор кнопок и вид значка, отображающегося в диалоговом окне; если этот параметр не задан, то VBScript отображает диалоговое окно без значков с единственной кнопкой **OK**.

`title` — необязательный параметр, определяющий заголовок окна.

`helpfile`, `index` — ссылка на файл, содержащий справку с указанием индекса, по которому находится необходимое сообщение.

`result` — код, возвращаемый функцией.

Приведем пример вывода окна сообщений (листинг 2.26). Как видно из рис. 2.7, на экран выводится окно сообщений с двумя кнопками. Если нажата кнопка **OK**, то переменная `ReturnValue` принимает значение, равное 1, если **Отмена**, то — 2.

**Листинг 2.26. Окно сообщений, вызываемое с помощью функции *MsgBox()***

```
ReturnValue=Msgbox("Операция завершена", 1, "Сообщение")
If ReturnValue=1 Then
      Msgbox "Нажата кнопка "+chr(34)+"OK"+chr(34)
Else
Msgbox "Нажата кнопка "+chr(34)+"Отмена"+chr(34)
      End If
```

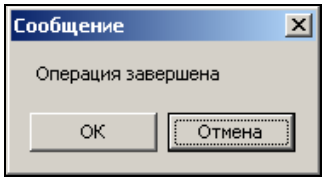



**Рис. 2.7.** Пример окна сообщений, выводимого с помощью функции `MsgBox()`

## Определение вида иконки и кнопок диалогового окна

Вид иконки в диалоговом окне, количество и названия кнопок регулируются параметром `buttons`. Значения этого параметра предопределены в Windows и являются составными. Объединение значений осуществляется с помощью знака конкатенации "+". Само значение складывается из трех параметров, возможные значения которых описаны в табл. 2.8—2.10.

***Таблица 2.8.*** *Константы, определяющие вид иконки диалогового окна*

| Константа | | Описание |
|---|---|---|
| 0 | – | Нет знака (по умолчанию) |
| 16 | vbCritical | Знак "Стоп" |
| 32 | vbQuestion | Вопросительный знак |
| 48 | vbExclamation | Восклицательный знак |
| 64 | vbInformation | Знак "Информация" |

***Таблица 2.9.*** *Константы, определяющие набор кнопок диалогового окна*

| Константа | | Описание |
|---|---|---|
| 0 | vbOKOnly | Кнопка **OK** (по умолчанию) |
| 1 | vbOKCancel | Кнопки **OK** и **Cancel** |
| 2 | vbOKRetryIgnore | Кнопки **OK**, **Retry**, **Ignore** |
| 3 | vbYesNoCancel | Кнопки **Yes**, **No**, **Cancel** |
| 4 | vbYesNo | Кнопки **Yes**, **No** |
| 5 | vbRetryCancel | Кнопки **Retry**, **Cancel** |

***Таблица 2.10.*** *Константы, управляющие фокусом кнопок*

| Константа | | Описание |
|---|---|---|
| 0 | vbDufaultButton1 | Фокус на первой кнопке (по умолчанию) |
| 256 | vbDufaultButton2 | Фокус на второй кнопке |
| 512 | vbDufaultButton3 | Фокус на третьей кнопке |
| 768 | vbDufaultButton4 | Фокус на четвертой кнопке |

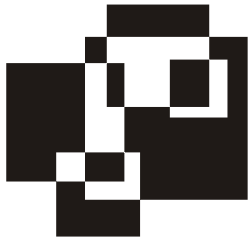
Константы могут быть заданы через переменные или числами, объединенными знаком конкатенации. Если константы задаются числами, то можно написать их сумму. В этом случае VBScript сам определит параметры диалогового окна. Например, указав в качестве параметра `buttons 292`. Имеется в виду 292 = 256 + 04 + 32, т. е. в диалоговом окне в виде ярлыка указывается знак вопроса, присутствуют две кнопки — **Yes** и **No**, а фокус установлен на второй кнопке — **No**.

# Определение возвращаемого значения функции *MsgBox()*

Значение, возвращаемое функцией `MsgBox()`, зависит от кнопки, нажатой пользователем. Анализируя значение переменной `result` по возвращаемому коду можно определить нажатую в диалоговом окне кнопку (табл. 2.11).

***Таблица 2.11.*** *Возвращаемые значения функции* `MsgBox()`

| **Константа** | | **Описание нажатой кнопки** |
|---|---|---|
| 1 | `vbOK` | **OK** |
| 2 | `vbCancel` | **Cancel** |
| 3 | `vbAbort` | **Abort** |
| 4 | `vbRetry` | **Retry** |
| 5 | `vbIgnore` | **Ignore** |
| 6 | `vbYes` | **Yes** |
| 7 | `vbNo` | **No** |



# Глава 3

# Windows Script Host

Windows Script Host (WSH) был предложен Microsoft в качестве инструмента разработки сценариев для операционной системы Windows. Сценарии WSH поддерживают два встроенных в Microsoft Windows языка программирования —Visual Basic Script Edition (VBScript) и Java Script Edition (JScript), и предъявляют минимальные требования к объему ОП. В этой главе рассмотрены основы синтаксиса WSH на примере языка VBScript.

## Сервер сценариев Windows Script

С сайта Microisoft можно загрузить выпуск Microsoft Windows Script 5.6, содержащий Visual Basic Script (VBScript) 5.6, JScript 5.6, Windows Script Components, Windows Script Host 5.6 и Windows Script Runtime 5.6 (**http://www.microsoft.com/downloads/details.aspx?familyid=E74494D3-C4E1-4E18-9C6C-0EA28C9A5D9D&displaylang=ru**).

### Установка Windows Script Host 5.6

Для использования WSH на компьютере должен быть установлен Microsoft Internet Explorer версии не ниже 3.0. Сервер сценариев использует обработчики сценариев для языков Visual Basic Script и Java Script, встроенные в Internet Explorer.

### Запуск сценариев WSH из командной строки

Для выполнения сценариев с помощью сервера сценариев для командной строки (Cscript.exe) используется следующий синтаксис:

```
cscript [имя_сценария] [параметры_сервера] [аргументы_сценария]
```

Где `имя_сценария` — имя файла сценария, включая путь и расширение.

параметры_сервера — ключи командной строки, задающие различные свойства сервера сценариев Windows. Параметр сервера всегда начинается с двух слэшей (//).

аргументы_сценария — ключи командной строки, которые передаются в сценарий. Аргумент сценария всегда начинается с одного слэша (/).

Все параметры являются необязательными, однако нельзя задать аргументы сценария, не задав сценарий. При отсутствии аргументов сценария или самого сценария программа Cscript.exe выведет описание синтаксиса команды и предусмотренные ключи. Ключи сервера сценариев для командной строки (табл. 3.1).

***Таблица 3.1.*** *Параметры командной строки Cscript.exe*

| Параметр | Действие |
|---|---|
| //B | Включение пакетного режима, при котором не выводятся подсказки и сообщения об ошибках |
| //D | Включение отладчика |
| //E:обработчик | Указание обработчика, используемого для выполнения сценария |
| //H:cscript<br>или<br>//H:wscript | Регистрация Cscript.exe или Wscript.exe в качестве сервера сценариев, применяемого по умолчанию. Если ключ не задан, используется Wscript.exe |
| //I | Включение интерактивного режима, в котором выводятся подсказки и сообщения об ошибках. Применяется по умолчанию; отменяет ключ //B |
| //Job:xxxx | Выполнение задания xxxx, указанного в файле сценария WSF |
| //Logo | Отображение перед выполнением сценария эмблемы сервера сценариев Windows. Применяется по умолчанию; отменяет режим //Nologo |
| //Nologo | Скрытие перед выполнением сценария эмблемы сервера сценариев Windows |
| //S | Сохранение параметров текущей командной строки для данного пользователя |
| //T:nnnnn | Указание максимального времени (в секундах), отведенного на выполнение сценария. Можно указать значение времени до 32 767 секунд. По умолчанию время выполнения не ограничено |
| //X | Запуск сценария в отладчике |
| //? | Вывод предусмотреных параметров командной строки и встроенной справки (аналогично запуску программы Cscript.exe без аргументов и без задания сценария) |

Параметр времени ожидания (`//T:nnnnn`) ограничивает максимальное время выполнения сценария. Если время выполнения превышает заданную величину, Cscript.exe прерывает работу обработчика и останавливает процесс.

Для использования с сервером сценариев Windows можно создать файл WSF, позволяющий вызвать несколько обработчиков сценариев и выполнить несколько заданий, в том числе, написанных на разных языках сценариев.

# Возможности WSH-сценариев

К возможностям WSH-сценариев относятся:

- вывод информации в виде диалоговых сообщений;
- считывание информации из потока данных;
- управление процессами Microosft Windows;
- работа с локальной сетью — подключение сетевых дисков и принтеров;
- работа с переменными среды;
- работа со специальными папками;
- создание ярлыков;
- работа с системным реестром;
- эмулирование нажатия клавиш;
- управление OLE-объектами.

# Объектная модель WSH

Объектная модель WSH представляет собой иерархически упорядоченную коллекцию объектов, содержащуюся в файле WSHOM.OCX, который располагается в каталоге %WinDir%\System32.

Объектную модель WSH составляют следующие объекты:

- `Wscript` — основной объект WSH, предназначенный для создания объектов и связи с ними, определения имени запускаемого скрипта и версии WSH; позволяет отображать данные с клавиатуры и выводить информацию на экран;
- `WshArguments` — обеспечивает доступ к параметрам командной строки запущенного сценария или ярлыка Microsoft Windows;
- `WshShell` — позволяет запускать процессы, создавать ярлыки, работать с системным реестром и специальными папками Microsoft Windows;

- `WshEnvironment` — предназначен для работы с переменными окружения;
- `WshSpecialFolders` — позволяет получить доступ к спецпапкам Windows;
- `WshNetwork` — используется для доступа к локальной сети; позволяет подключать сетевые устройства: принтеры и диски;
- `WshShortcut` — обеспечивает работу с ярлыками Windows;
- `WshUrlShortcut` — обеспечивает работу с ярлыками сетевых ресурсов Windows.

# Объект *WScript*

Свойства объекта `WScript` позволяют получить полный путь к используемуся серверу сценариев (wscript.exe или cscript.exe), параметрам командной строки, с которыми он запущен и т. д. (см. табл. 3.2). В сценарии WSH-объекты используются без какого-либо предварительного описания. Для доступа к остальным объектам используют метод `CreateObject()` или определенное свойство другого объекта.

## Свойства объекта *WScript*

Поддерживаемые объектом `WScript` свойства приведены в табл. 3.2.

***Таблица 3.2.*** *Свойства объекта* `WScript`

| Свойство | Описание |
|---|---|
| `Arguments` | Содержит указатель на список параметров командной строки, с которой запущен сценарий |
| `FullName` | Содержит полный путь к исполняемому файлу сервера сценариев |
| `Name` | Содержит название объекта `WScript` (Windows Script Host) |
| `Path` | Содержит путь к каталогу, в котором расположен сервер сценариев |
| `ScriptFullName` | Содержит полный путь к исполняемому сценарию |
| `ScriptName` | Содержит имя запущенного сценария |
| `StdErr` | Позволяет запущенному сценарию записывать сообщения в стандартный поток для вывода ошибок |
| `StdIn` | Позволяет запущенному сценарию считывать данные из стандартного входного потока |
| `StdOut` | Позволяет запущенному сценарию записывать данные в стандартный выходной поток |
| `Version` | Содержит версию WSH |

## Свойство *Arguments*

Свойство `Arguments` используется для передачи скрипту параметров из командной строки. Например:

```
welcome.vbs Petrov Nikolay
```

С помощью этого свойства при выполнении сценария осуществляется анализ заданных параметров, выясняется их количество.

В листинге 3.1 приведен файл who.vbs, который определяет, с какими параметрами был запущен скрипт (рис. 3.1).

**Листинг 3.1. Определение параметров запуска скрипта**

```
Set objArgs=Wscript.Arguments
t="Количество заданных параметров - " & Wscript.Arguments.Count & chr(13)
& "Параметры:" & chr(13)
For Each arg in objArgs
       t = t & arg & chr(13)
Next
MsgBox t
```

Команда запуска файла следующая:

```
Who.vbs Petrov Nikolay
```

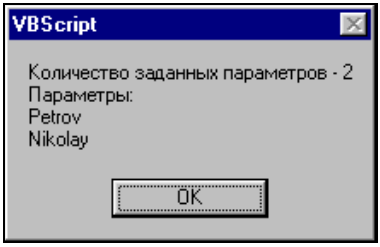



**Рис. 3.1.** Сообщение о параметрах запуска скрипта

Аргументы скрипта можно считывать другим способом (листинг 3.2). Оба способа равнозначны. Отметим лишь, что первый вариант компактнее, второй — предпочтительнее, если необходимо использовать нумерацию объектов, поскольку в первом отсутствует счетчик.

**Листинг 3.2. Определение параметров запуска скрипта со счетчиком**

```
Set objArgs=Wscript.Arguments
Coun = Wscript.Arguments.Count
t="Количество заданных параметров - " & Coun & chr(13) & "Параметры:" &
chr(13)
For i=0 to Coun-1
       t = t & i+1 & ": "& objArgs(i) & chr(13)
Next
MsgBox t
```

Для определения значения заданного аргумента используется следующий сценарий — листинг 3.3.

**Листинг 3.3. Определение значения заданного аргумента**

```
Set objArgs=Wscript.Arguments
Msgbox objArgs(2)                    ' Чтение третьего аргумента.
```

Нумерация элементов начинается с 0. Приведенный пример не является лучшим образцом программирования, поскольку в нем отсутствует обработчик ошибок. Использование обработчика ошибок, имитация простейшего интеллекта у программы, является хорошим тоном программирования. Создадим для этого примера обработчик ошибок. Ошибка может возникнуть, если предпринята попытка получить несуществующее значение (листинг 3.4).

**Листинг 3.4. Обработчик ошибок**

```
Set objArgs=Wscript.Arguments
Coun = Wscript.Arguments.Count
q = InputBox ("Введите номер считываемого аргумента", "Чтение номера ар-
гумета")-1
t = ""
if q> Coun then
       t= objArgs(q)
else
       t="ERROR"
End if
Msgbox t
```

## Свойства *FullName*, *Name*, *Path*, *Version*

Набор свойств `FullName`, `Name`, `Path`, `Version` позволяет получить информацию о версии и месторасположении сервера сценариев (листинг 3.5, рис. 3.2).

**Листинг 3.5. Определение свойств интерпретатора WSH**

```
Product=Wscript.Name & " " & Wscript.Version & chr(13)
Path= Wscript.Path & chr(13) & WScript.Fullname
MsgBox Product & Path
```




**Рис. 3.2.** Свойства интерпретатора WSH

## Свойства *StdErr*, *StdIn*, *StdOut*

Доступ к стандартным входным и выходным потокам с помощью перечисленных ранее свойств можно получить, если сценарий запущен в консольном режиме с помощью утилиты cscript.exe. Если сценарий был запущен с помощью wscript.exe, то при попытке обратиться к этим свойствам возникает ошибка `Invalid Handle`. Перечень ошибок, выдаваемых сценариями WSH, приведен в приложении. Используя сценарии регистрации пользователей в сети и ASP-файлы, осуществляется работа с wscript.exe, поэтому данные свойства описываться не будут.

# Методы объекта *WScript*

Объект `WScript` поддерживает несколько методов, которые описаны в табл. 3.3.

***Таблица 3.3.*** *Методы объекта* `WScript`

| Свойство | Описание |
|---|---|
| `CreateObject(ObjectID, Prefix)` | Создает объект, заданный параметром `Object` |
| `GetObject(Object,StrProgID, Prefix)` | Активизирует объект, определяемый заданным файлом или параметром |

***Таблица 3.3*** *(окончание)*

| Свойство | Описание |
|---|---|
| `DisconnetObject(ObjectID)` | Отсоединяет объект, с которым ранее была установлена связь |
| `ConnectObject(ObjectID, Prefix)` | Устанавливает соединение с объектом, позволяющее писать функции-обработчики его событий. Имена этих функций должны начинаться с префикса `Prefix` |
| `Echo […[[Arg1], Arg2]…]` | Выводит текстовую информацию на консоль или в диалоговое окно. Является упрощенным аналогом функции `MsgBox()` в VBScript |
| `Quit(ErrorCode)` | Прерывает выполнение сценария с заданным параметром `ErrorCode` кодом выхода. Если параметр не задан, то объект WScript.exe установит код выхода равным нулю |
| `Sleep(Time)` | Приостанавливает выполнение сценария на заданный промежуток времени. Время указывается в миллисекундах |

Приведем дополнительные пояснения и примеры наиболее часто используемых методов.

## Метод *CreateObject(ObjectID, Prefix)*

Если объект не предоставляется WSH автоматически, как в случае WScript, то его необходимо создать с помощью метода `CreateObject()`, т. е. загрузить его экземпляр в память и присвоить переменной-объекту ссылку. Используя этот метод, не забывайте об иерархии объектов. Первый параметр содержит ID объекта, например, `Wscript.Shell`; второй параметр является необязательным и содержит строку префикса, если префикс задан, то после создания этого объекта метод соединяет его исходящий интерфейс со сценарием. Такое соединение позволяет осуществлять обработку событий функциями обратного вызова, объявленными в сценарии. Например, если префиксом является `_ErrorCode`, а у объекта срабатывает событие `OnBegin`, то WSH вызывает функцию `OnBegin_ErrorCode()`. Функция должна быть описана в теле файла. На практике префиксами пользуются крайне редко. Создание объекта:

```
Set objShell= WScript.CreateObject(Wscript.Shell)
```

## Метод *GetObject(Object, ObjectID, Prefix)*

Наряду с `CreateObject()`, доступ к объектам также дает метод `GetObject()`, который получает объект из файла или из другого объекта, заданного параметром `ObjectID`. Этот метод используется в том случае, если текущий объект

экземпляра уже есть или его необходимо вызвать из файла. Синтаксис метода аналогичен методу `CreateObject()`. Как уже отмечалось, первый параметр задает путь к файлу, содержащему объект, или к объекту, второй и третий параметры аналогичны по своей сути параметрам, используемым в методе `CreateObject()`. Второй параметр позволяет загружать объект с помощью записей реестра, а третий — связывать сценарий и внешний интерфейс. Пример вызова объекта:

```
Set rootDSE_ = WScript.GetObject("LDAP://RootDSE")
```

### Метод *DisconnetObject(ObjectID)*

При создании объекта `DisconnetObject` методом `CreateObject()` или `GetObject()`, он загружается в память и остается соединенным с WSH до конца сеанса. Сценарий может подключить свой обработчик событий, используя второй параметр функции `CreateObject` или третий — для `GetObject()`. Объект может генерировать события, которые будут обслужены обработчиком событий сценария. Метод `DisconnectObject()` отсоединяет объект от процедуры обработки события во время работы сценария:

```
WScript.DisconnetObject(ObjectID)
```

или

```
WScript.DisconnetObject ObjectID
```

В данном случае, переменная `ObjectID` — имя переменной объекта. На VBScript также можно сбросить значение переменной-объекта следующим образом (листинг 3.6):

```
ObjectID = Nothing
```

**Листинг 3.6. Сброс значения переменной-объекта**

```
Set Wshell = WScript.CreateObject("wscript.shell")
…
WScript.DiconnectObject(Wshell)
```

или

```
Set Wshell = WScript.CreateObject("wscript.shell")
…
Wshell = Nothing
```

## Объект *WshArguments*

Обеспечивает доступ к параметрам командной строки запущенного сценария или ярлыка Microsoft Windows. Доступ к объекту `WshArguments` осуществляется с помощью свойства `Arguments` объекта `WScript`, которое было описано ранее.

# Объект *WshShell*

Позволяет запускать процессы, создавать ярлыки, работать с системным реестром и специальными папками Microsoft Windows. Поддерживаемые свойства и методы объекта `WSHell` представлены в табл. 3.4 и 3.5.

***Таблица 3.4.*** *Свойства объекта* `WSHell`

| Свойство | Описание |
|---|---|
| `Environment` | Обеспечивает доступ к объекту `WshEnvironment`, содержащий переменные среды для Windows 9x и 2k |
| `SpecialFolders` | Обеспечивает доступ к объекту `SpecialFolders`, содержащий набор путей к спецпапкам Windows 9x и 2k |

***Таблица 3.5.*** *Методы объекта* `WSHell`

| Метод | Описание |
|---|---|
| `AppActivate(title)` | Активизирует заданное параметром `title` окно ранее запущенного приложения. Строка `title` задает название окна |
| `CreateShortcut(strPathName)` | Создает объект `WshShortcut` для связи с ярлыком Windows (*.lnk) или объект `URLWshShortcut` для связи с сетевым ярлыком (*.url). Параметр `strPathName` содержит полный путь к создаваемому или изменяемому ярлыку |
| `Environment(strType)` | Возвращает объект `WshEnvironment`, содержащий переменные среды заданного типа |
| `ExpandEnvironmentStrings(strString)` | Возвращает значение переменной среды текущего командного окна заданной строкой `strString`. Имя переменной должно быть окружено знаками `%` |

***Таблица 3.5*** *(окончание)*

| Метод | Описание |
|---|---|
| `LogEvent (intType, strMessage [,strTarget])` | Протоколирует произошедшие события в журнал событий Windows или в файл Wsh.log. Первый параметр определяет тип сообщения, второй — его содержание. Третий параметр (`strTarget`) может быть задан только в Windows 2k. С его помощью определяют название системы, в которой протоколируются события. По умолчанию это локальная система. Метод возвращает `True`, если событие успешно занесено в журнал событий, и `False` в противном случае |
| `Popup(strText [,secWait] [,strTitle] [,nType])` | Выводит окно с заданным текстом `strText`. Аналог функции `MsgBox()` в VBScript. Параметр `secWait` — количество секунд, по истечении которых окно будет закрыто; `strTitle` определяет заголовок окна; параметр `nType` определяет конфигурацию окна (значок и кнопки) |
| `RegDelete(strName)` | Выполняют операции с реестром: удаление, чтение и запись данных, соответственно |
| `RegRead(strName)` | |
| `RegWrite(strName, Value [,strType])` | |
| `Run(strCommand [,WindowStyle], [,WaitOnReturn])` | Запускает сторонние приложения из сценария |
| `SendKeys(string)` | Эмулирует нажатие клавиш клавиатуры |
| `SpecialFolders(strType)` | Возвращает объект `WshSpecialFolders`, содержащий пути к специальным папкам Windows |

## Метод *AppActivate*

С помощью метода `AppActivate`, устанавливая фокус, активизируют ранее запущенное приложение; никаких изменений размера окна не осуществляется (листинг 3.7). Для первоначального запуска приложения и определения

размеров окна, используйте метод Run, который будет описан позднее. Для того чтобы определить, какое именно приложение необходимо активизировать, строка title поочередно сравнивается с названиями окон всех запущенных приложений. Если точного совпадения не найдено, то осуществляется запуск приложения, начало заголовка окна которого совпадает с title. Если такое окно с таким заголовком не найдено, то будет запущено приложение, у которого заголовок и строка title совпадают с конца строки.

**Листинг 3.7. Использование метода AppActivate**

```
Set Wshell = WScript.CreateObject("wscript.shell")
Wshell.Run("calc")
Wshell.AppActivate("Calculator")
```

## Управление ярлыками методом *CreateShortcut*

Существует два вида ярлыков: одни являются ссылками на файлы или папки, а другие — на интернет-страницы. Для создания ярлыков программ используют метод CreateShortcut объекта WshShortcut, для ссылок на сайты — WshUrlShortkut. С помощью обоих методов можно изменять настройки соответствующих ярлыков.

# Объект *WshShortcut*

Данный объект можно создать с помощью метода CreateShortcut объекта WshShell. Свойства объекта WshShortcut приведены в табл. 3.6.

***Таблица 3.6.*** *Свойства объекта* WshShortcut

| Свойство | Описание |
|---|---|
| Arguments | Содержит строку, задающую параметры командной строки для ярлыка |
| Description | Включает описание ярлыка |
| FullName | Содержит полный путь к ярлыку |
| HotKey | Задает комбинацию горячих клавиш для ярлыка |
| IconLocation | Задает путь к иконке ярлыка |
| TargetPath | Устанавливает путь к файлу, на который указывает ярлык |
| WindowStyle | Определяет вид окна, в котором будет выводиться приложение |
| WorkingDirectory | Задает рабочий каталог приложения |

Для записи указанных свойств в таблице используется функция `Save()`. Чтобы назначить комбинацию клавиш ярлыку, следует названия необходимых клавиш разделить символом "+", например `CTRL+ALT+W`. Приведем пояснения и примеры использования каждого из свойств. Значением свойства `WindowStyle` является целое число (табл. 3.7).

***Таблица 3.7.*** *Стили свойства* `WindowStyle`

| Значение | Описание |
|---|---|
| 1 | Стандартный размер окна, принятый по умолчанию. Окно располагается в центре экрана |
| 3 | Окно при запуске приложения будет развернуто на весь экран |
| 7 | Окно при запуске приложения будет свернуто |

## Создание ярлыка

Приведем пример, в котором используются все описанные свойства (листинг 3.8, рис. 3.3). Создадим на рабочем столе ярлык к стандартному приложению Windows notepad.exe (Блокнот), которое расположено в каталоге %windir%. Зададим клавиатурное сокращение `CTRL+ALT+B`. После запуска приложения оно должно разворачиваться на весь экран.

**Листинг 3.8. Создание ярлыка Windows-приложения**

```
Set WshShell=CreateObject("Wscript.Shell")
StrDesctop=WshShell.SpecialFolders("Desktop") ' определение пути
                                              ' к папке Desktop
Set oShk=WshShell.CreateShortcut(StrDesctop+"\\notepad.lnk")
oShk.Arguments="1.txt"
oShk.Description="Блокнот"
oShk.TargetPath= "%Windir%\\notepad.exe"
oShk.HotKey="CTRL+ALT+B"
oShk.IconLocation="notepad.exe, 0"
oShk.WindowStyle=3
oShk.WorkingDirectory="%Windir%"
oShk.Save()
wscript.echo "Ярлык создан"
```

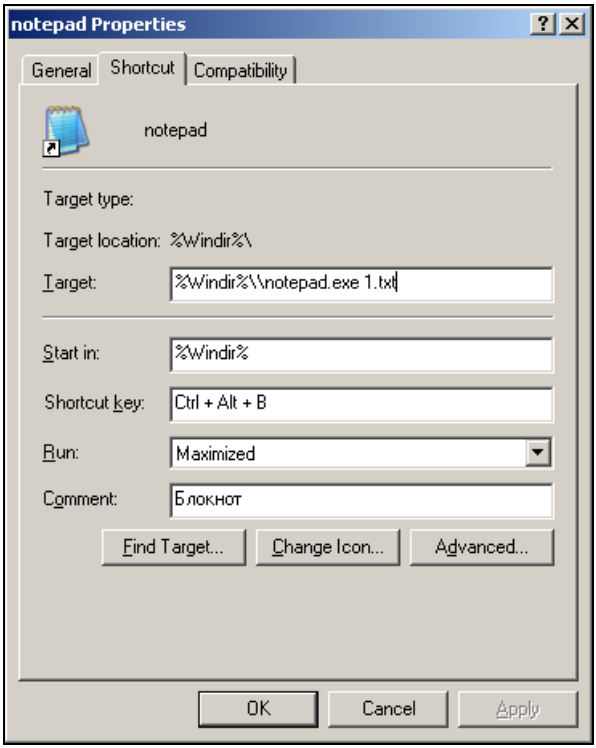



**Рис. 3.3.** Свойства ярлыка объекта

## Чтение/изменение свойств ярлыка

Метод `CreateShortcut` позволяет не только создавать ярлыки, но и считывать и изменять их свойства. Сценарий работает по следующему алгоритму: сначала получают доступ к файлам с расширением lnk с помощью объекта `Scripting.FileSystemObject` и определяют тип файла, используя свойство `Type`. Если его тип `Shortcut`, то к свойствам ярлыка получают доступ с помощью инструкции `WshShell.CreateShortcut(oFile.path)`, где `oFile.path` — путь к ярлыку. Затем осуществляют чтение и запись данных (листинг 3.9).

**Листинг 3.9. Изменение свойств ярлыка Windows-приложения**

```
Set WshShell=CreateObject("Wscript.Shell")
StrDesctop=WshShell.SpecialFolders("Desktop")
Path= StrDesctop+"\\notepad.lnk"
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Set oFile = fso.GetFile(path)
If strcomp (oFile.type,"Shortcut",0 )=0 Then
       Set oShk=WshShell.CreateShortcut(oFile.path)
       If StrComp(ucase(oShk.Description), UCase("Блокнот"))=0 Then
              t1=oShk.Description
              oShk.Description="Текстовый редактор"
              oShk.Save()
              t2=oShk.Description
       End If
End If

Wscript.echo "Описание ярлыка " + ucase(t1) + " изменено на " + ucase(t2)
```

## Удаление ярлыка

Для удаления ярлыка используют свойство `Delete` объекта `Script.FileSystemObject` (листинг 3.10).

**Листинг 3.10. Удаление ярлыка**

```
Set WshShell=CreateObject("Wscript.Shell")
StrDesctop=WshShell.SpecialFolders("Desktop") ' определение пути
                                              ' к папке Desktop
Set Path= StrDesctop+"\\notepad.lnk"
Set fso = CreateObject("Script.FileSystemObject")
Set oFile = fso.GetFile(path)
o.Delete
```

# Объект *WshURLShortcut*

У объекта `WshURLShortcut` только два свойства: `FullName` и `TargetPath`, которые полностью аналогичны одноименным свойствам метода `WshShortcut`. Также поддерживается метод `Save`, позволяющий сохранять ярлык в указанном каталоге (листинг 3.11, рис. 3.4).

**Листинг 3.11. Создание Web-ярлыка**

```
Set WshShell=CreateObject("Wscript.Shell")
StrDesctop=WshShell.SpecialFolders("Desktop") ' определение пути
                                              ' к папке Desktop
```

```
Set oShk=WshShell.CreateShortcut(StrDesctop+"\\adobe.url")
oShk.TargetPath= "http://www.adobe.com"
oShk.Save()
wscript.echo "Ярлык создан"
```

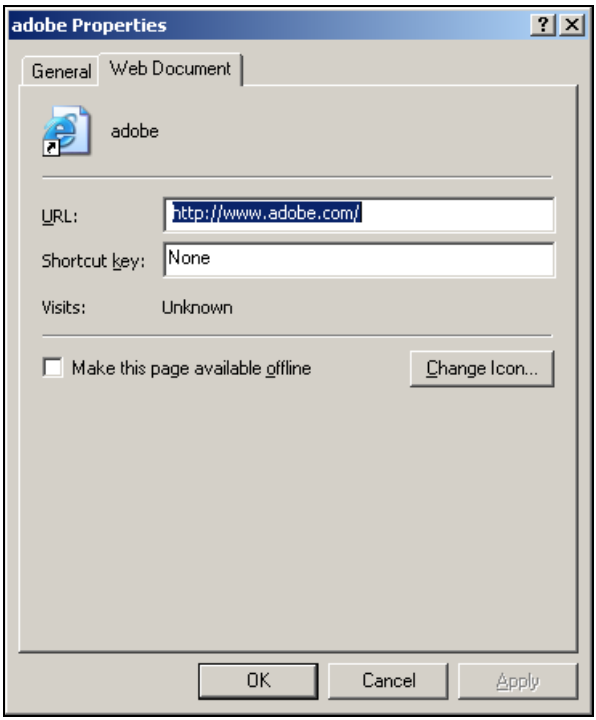



**Рис. 3.4.** Создание Web-ярлыка

# Объект *Wscript.Shell*

## Метод *Environment*

Метод `Environment` предназначен для работы с переменными окружения. Они содержат строковые значения, задаваемые командой `set`. Чтобы получить доступ к свойствам объекта `Environment`, необходимо подключиться к пространству объекта-родителя `Wscript.Shell`:

```
Set Wshell=CreateObject("Wscript.Shell")
Set variable=Wshell.Environment([StrType])
```

Индекс `StrType` задает категорию переменной окружения. В Windows 2k переменные окружения разделены на несколько категорий — `System`, `User`, `Volatile`, `Process` (табл. 3.8).

***Таблица 3.8.*** *Переменные окружения различных ОС*

| Имя | Описание | W2k | | | |
|---|---|---|---|---|---|
| | | System | Volatile | User | Process |
| COMSPEC | Путь к командному процессору (cmd.exe или command.com) | + | | | + |
| HOMEDIVE | Главный локальный диск. Как правило, c:\ | | | | + |
| HOMEPATH | Каталог по умолчанию для пользователей | | | | + |
| NUMBER_OF_PROCESSORS | Количество процессоров, установленных в материнской плате | + | | | + |
| OS | Название и версия ОС | + | | | + |
| PATH | Путь | + | | | + |
| PATHEXT | Расширения исполняемых файлов | + | | | + |
| PROCESSOR_ARHITECTURE | Тип процессора | + | | | + |
| PROCESSOR_IDENTIFIER | Идентификатор процессора | + | | | + |
| PROCESSOR_LEVEL | Уровень процессора | + | | | + |
| PROCESSOR_REVISION | Версия процессора | + | | | + |
| PROMPT | Приглашение командной строки MS-DOS | | | | + |
| SYSTEMDRIVE | Логический диск, на котором установлена ОС | | | | + |
| SYSTEMROOT | Системный каталог, в котором расположена ОС | | | | + |
| TEMP | Каталог для хранения временных файлов | | + | + | + |
| TMP | Каталог для хранения временных файлов | | + | + | + |
| WINDIR | То же, что и SYSTEMROOT | | + | | + |

**Листинг 3.12. Чтение значения одной из переменных окружения**

```
Set Wshell=CreateObject("Wscript.Shell")
Set variable=Wshell.Environment("Process")
Wscript.Echo variable("WINDIR")
```




**Рис. 3.5.** Чтение переменной окружения

Опишем действия, выполняемые в сценарии (листинг 3.12, рис. 3.5) построчно. В первой строке создается ссылка на объект `WshShell`, во второй выполняется обращение к свойству `Environment`, содержащему набор всех переменных окружения из категории `Process`. С помощью последней строки осуществляется вывод значения переменной `WINDIR` из коллекции `Process`.

***Таблица 3.9.*** *Свойства объекта* `Environment`

| Свойство | Описание |
|---|---|
| `Count` | Возвращает число аргументов командной строки |
| `Item` | Свойство по умолчанию, определяющее n-ный аргумент командной строки, который вызывает сценарий |
| `Length` | Возвращает число аргументов командной строки. Используется для совместимости с JScript |

Чтобы считать все существующие на данной рабочей станции переменные окружения, необходимо запустить сценарий из листинга 3.13. Отличие от предыдущего примера заключается в том, что, получив доступ ко всей коллекции с помощью свойства `count` (табл. 3.9), можно определить имена и значения всех переменных среды окружения.

**Листинг 3.13. Чтение всех переменных окружения раздела `Process`**

```
Set Wshell=CreateObject("Wscript.Shell")
Set variable=Wshell.Environment("Process")
for var=0 to variable.count
```

```
        temp=temp+variable.item(var)+chr(13)
next
Wscript.Echo temp
```

## Операции с переменными окружения

С помощью сценария, написанного на WSH, можно управлять переменными окружения: удалять их и создавать. В MS-DOS управление переменными осуществляется с помощью команды `SET`, которая имеет следующий синтаксис:

```
Set [Variable[=Value]]
```

Где `Variant` — имя переменной.

`Value` — значение, сопоставляемое переменной среды.

При вызове команды `Set` без параметров осуществляется вывод уже известных переменных и их значений (рис. 3.6).

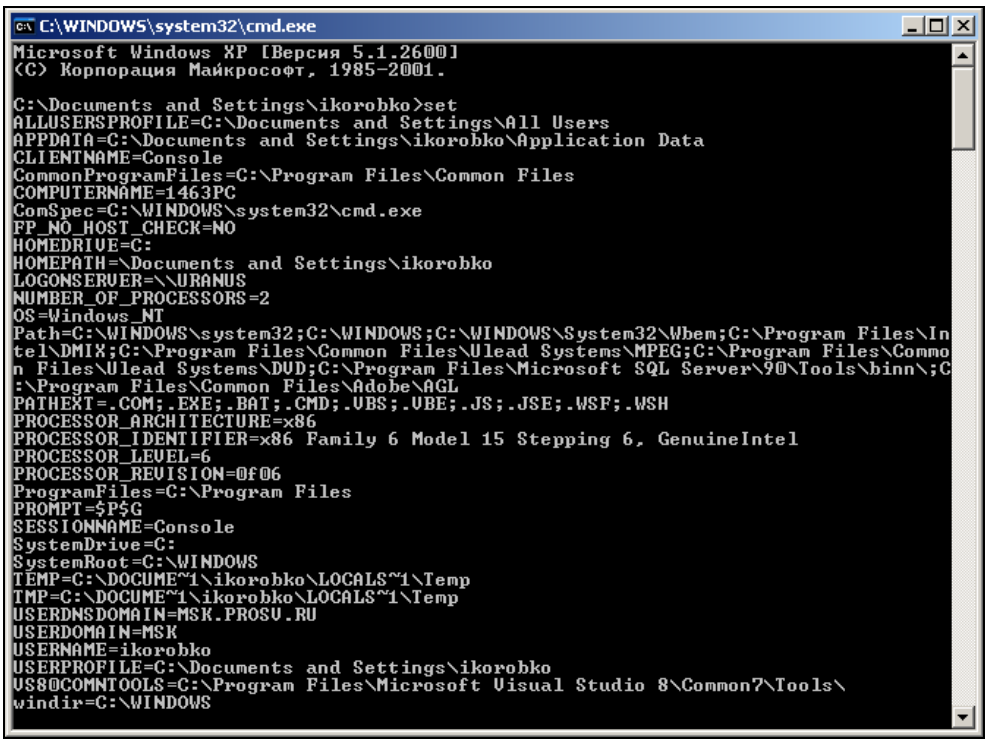



**Рис. 3.6.** Значения переменных `Set`

### Создание переменных окружения с помощью сценария

Создание переменной окружения осуществляется присвоением значения какой-либо переменной в заранее оговоренной коллекции (листинг 3.14).

**Листинг 3.14. Создание переменной окружения**

```
Set Wshell=CreateObject("Wscript.Shell")
Set variable=Wshell.Environment("Process")
variable("NewSet")="This Is New Set Variable"
Wscript.Echo variable("NewSet")
```

Если вызвать сценарий с помощью метода `RUN` из другого сценария, то переменная, объявленная в первом, будет доступна во втором сценарии.

### Удаление переменных окружения с помощью сценария

Удаление переменной окружения осуществляется с помощью метода `Remove`. Этому методу доступны только локальные переменные окружения (листинг 3.15).

**Листинг 3.15. Удаление переменной окружения**

```
Set Wshell=CreateObject("Wscript.Shell")
Wshell.Environment("Process"). Remove("NewSet")
```

## Метод *ExpandEnvironmentStrings*

Этот метод возвращает значение переменной среды текущего командного окна заданной строкой `strString` (листинг 3.16). Имя переменной должно быть окружено знаками `%`.

**Листинг 3.16. Получение значения указанной переменной окружения**

```
Set Wshell=CreateObject("Wscript.Shell")
Wscript.Echo "Каталог Windows: "&
Wshell.ExpandEnvironmentStrings("%Windir%")
```

Если операционная система установлена в каталог Windows на диске C, то переменная `%Windir%` возвратит путь `C:\Windows`.

## Метод *LogEvent*

Этот метод протоколирует произошедшие события. В Windows 2k события записываются в системный журнал. Каждая запись в файле отчетов содержит время события, его тип и описание события (табл. 3.10).

***Таблица 3.10.*** *Расшифровка значений файла *.log*

| Код | Значение | Код | Значение |
|---|---|---|---|
| 0 | SUCCESS | 4 | INFORMATION |
| 1 | ERROR | 8 | AUDIT_SUCCESS |
| 2 | WARNING | 16 | AUDIT_FAILURE |

Метод `LogEvent` имеет следующий синтаксис:

```
Wshell.LogEvent (intType, strMessage [,strTarget])
```

Первый параметр `intType` определяет тип сообщения, второй `strMessage` — его содержание. Необязательный параметр `strTarget` может быть задан только в Windows 2k. Им определяется название системы, в которой протоколируются события. По умолчанию это локальная система. Метод возвращает `True`, если событие успешно занесено в журнал, и `False` в противном случае.

В приведенном в листинге 3.17 примере выполняется функция `Function_1()`, которая возвращает 0 (`TRUE`) или 1 (`FALSE`). Если функция отработала успешно (`E_Code=0`), то в журнале событий появится запись `Status: OK`, в противном случае — `Status: ERROR`.

**Листинг 3.17. Протоколирование работы функции**

```
Set WshShell=CreateObject("Wscript.Shell")
E_Code=Function_1()
If E_Code="True" then
WshShell.LogEvent(0,"Status: OK","Event")
Else
WshShell.LogEvent(1, "Status: ERROR", "Event")
End If
```

## Метод *Popup*

Метод `Popup` используется для вывода сообщений на экран. По своей сути этот метод является аналогом функции `MsgBox()` в VBScript. Вызов метода осуществляется посредством одноименной функции. Приведем ее синтаксис:

```
Wshell.Popup(strText [,secWait] [,strTitle] [,nType] )
```

Переменная `strText` содержит текст, выводимый в диалоговом окне. Параметр `secWait` определяет количество секунд, по истечении которых окно будет закрыто; `strTitle` — заголовок окна; параметр `nType` (см. табл. 3.11 и 3.12) определяет конфигурацию окна (значок и кнопки). Значения переменной `nType` полностью совпадают со значениями функции `MsgBox()` из Microsoft Win32 API.

***Таблица 3.11.*** *Константы, определяющие вид иконки окна*

| Константа | | Описание |
|---|---|---|
| 0 | – | Нет значка (по умолчанию) |
| 16 | vbCritical | Знак "Стоп" |
| 32 | vbQuestion | Вопросительный знак |
| 48 | vbExclamation | Восклицательный знак |
| 64 | vbInformation | Знак "Информация" |

***Таблица 3.12.*** *Константы, определяющие набор кнопок окна*

| Константа | | Описание |
|---|---|---|
| 0 | vbOKOnly | Кнопка **OK** (по умолчанию) |
| 1 | vbOKCancel | Кнопки **OK** и **Cancel** |
| 2 | vbOKRetryIgnore | Кнопки **OK**, **Retry**, **Ignore** |
| 3 | vbYesNoCancel | Кнопки **Yes**, **No**, **Cancel** |
| 4 | vbYesNo | Кнопки **Yes**, **No** |
| 5 | vbRetryCancel | Кнопки **Retry**, **Cancel** |

В методе `Popup` можно комбинировать значения параметров, приведенные в табл. 3.11 и 3.12.

## Определение возвращаемого значения методом *Popup*

Значение, возвращаемое методом `Popup`, зависит от кнопки, нажатой пользователем. Таким образом, анализируя значение переменной `result`, по возвращаемому коду можно определить, какая из кнопок была нажата (табл. 3.13).

***Таблица 3.13.*** *Значения, возвращаемые методом* `Popup`

| Константа | Нажатая кнопка |
|---|---|
| -1 | Пользователь не нажал ни одну из кнопок в течение времени, заданного параметром `secWait` |
| 1 | **OK** |
| 2 | **Cancel** |
| 3 | **Abort** |
| 4 | **Retry** |
| 5 | **Ignore** |
| 6 | **Yes** |
| 7 | **No** |

**Листинг 3.18. Использование метода `Popup`**

```
Set WshShell=CreateObject("Wscript.Shell")
result=WshShell.Popup("Операция завершена", 5, "Сообщение", 65)
Wscript.Echo result
```

Сообщение, выводимое на экран (листинг 3.18, рис. 3.7), автоматически закроется через 5 секунд.

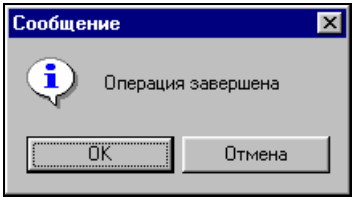



**Рис. 3.7.** Использование метода `Popup`

## Метод *Run*

С помощью метода `Run` осуществляется запуск различных приложений из сценария (листинг 3.19). Команда имеет следующий синтаксис:

```
Wshell.Run(strCommand [,WindowStyle], [,WaitOnReturn])
```

Где `strCommand` — обязательный параметр, значение которого содержит путь к файлу запускаемого приложения. Путь к файлу, содержащий пробелы, необходимо заключать в кавычки. В противном случае метод выдаст ошибку `The system cannot find the file specified` (Система не может найти указанный файл). Метод `Run` автоматически раскрывает переменные окружения.

`WindowStyle` — с помощью этого параметра контролируется размер окна, в котором выполняется приложение (табл. 3.14). Этот параметр необязательный.

***Таблица 3.14.*** *Значения параметра* `WindowStyle`

| Значение | Константа VBScript | Описание |
|---|---|---|
| 0 | vbHide | Скрывает окно и активизирует другое (показывает его и передает ему фокус) |
| 1 | vbNormalFocus | Активизирует и показывает окно. Если процесс был уже запущен, то восстанавливается его предыдущий размер |
| 2 | vbMinimizedFocus | Активизирует и сворачивает окно. Фокус переходит к кнопке на панели задач |
| 3 | vbMaximizedFocus | Активизирует окно, разворачивает его на весь экран и передает ему фокус |
| 4 | vbNormalNoFocus | Вызывает окно в его предыдущем положении и размере |
| 5 | – | Активизирует окно уже запущенного процесса |
| 6 | vbMinimizedNoFocus | Сворачивает окно и активизирует следующее окно |
| 7 | – | Выводит окно в виде значка. Фокус окна сохраняется |
| 8 | – | Выводит окно в текущем состоянии. Активное окно остается таковым |
| 9 | – | Активизирует окно и выводит его. Восстанавливает его первоначальный размер и положение |
| 10 | – | Устанавливает состояние окна в зависимости от состояния программы, запустившей это приложение |

Метод `Run` имеет существенный недостаток — им всегда порождается новый процесс, который необходимо каждый раз останавливать (касается утилит, запускаемых из командной строки).

WaitOnReturn — необязательный параметр, принимающий значение True или False. Этот параметр определяет, будет ли сценарий ожидать завершения существующего процесса. Если он не задан или его значение равно False, то метод Run возвращает значение 0. Код ошибки задается методом Quit, который был рассмотрен ранее.

**Листинг 3.19. Использование метода Run**

```
Set Wshell = CreateObject("Wscript.Shell")
intReturn = Wshell.Run ("cmd.exe dir *.* > c:\test.txt")
Response.Write(intReturn) ' This returns 0
Set Wshell = Nothing
```

## Метод *SendKeys*

Метод SendKeys используется для эмуляции нажатия различных клавиш и управления запущенными приложениями. Каждая клавиша задается одним или несколькими символами. Некоторые символы, а именно "+", "^", "%", "~", ",", "[", "]", "{", "}", необходимо заключать в фигурные скобки — {}.

Для задания неотображаемых символов, таких как Enter или Tab, используются следующие коды — табл. 3.15.

***Таблица 3.15.*** *Коды задания неотображаемых символов*

| Клавиша | Код | Клавиша | Код |
|---|---|---|---|
| <Alt> | % | <↑> | {Up} |
| <BackSpace> | {BackSpace}, {BS}, {BKSP} | <←> | {Left} |
| <Break> | {Break} | <↓> | {Down} |
| <Caps Lock> | {CapsLock} | <→> | {Right} |
| <Ctrl> | ^ | <F1> | {F1} |
| <Del>, <Delete> | {Del}, {Delete} | <F2> | {F2} |
| <End> | {End} | <F3> | {F3} |
| <Enter> | {Enter}, {~} | <F4> | {F4} |
| <Esc> | {Esc} | <F5> | {F5} |
| <Help> | {Help} | <F6> | {F6} |
| <Home> | {Home} | <F7> | {F7} |

***Таблица 3.15*** *(окончание)*

| Клавиша | Код | Клавиша | Код |
|---|---|---|---|
| <Ins>, <Insert> | {Ins}, {Insert} | <F8> | {F8} |
| <Num Lock> | {NumLock} | <F9> | {F9} |
| <Page Down> | {PGDN} | <F10> | {F10} |
| <Page Up> | {PGUP} | <F11> | {F11} |
| <Print Screen> | {PRTSC} | <F12> | {F12} |
| <Scroll Lock> | {ScrollLock} | | |
| <Shift> | + | | |
| <Tab> | {Tab} | | |

Чтобы передать приложению комбинацию клавиш, включающую <Ctrl>, <Alt> или <Shift>, необходимо добавить соответствующий код перед клавишей, например, комбинации <Ctrl>+<C> соответствует "^{C}". Если комбинация клавиш состоит из двух клавиш и более, то символы следует заключать в круглые скобки: комбинации клавиш <Shift>+<E>+<C> соответствует {+(EC)}. Для повторяющихся клавиш необходимо после клавиши указать количество раз повтора, например, если нужно нажать 10 раз кнопку <Tab>, то соответствующая команда будет выглядеть следующим образом: {Tab 10}.

Для эмуляции нажатия клавиши используется свойство SendKeys объекта WShell. Команда вызова имеет следующий синтаксис:

```
Wshell.SendKeys(string)
```

Пример использования метода SendKeys приведен в листинге 3.20.

**Листинг 3.20. Реализация функции закрывания активного окна (эмуляция нажатия комбинации клавиш <Alt>+<F4>)**

```
Set Wshell = CreateObject("Wscript.Shell")
Wshell.SendKeys(%{F4})
set wshell = nothing
```

## Метод *SpecialFolders*

SpecialFolders используется для получения доступа к спецпапкам Windows. Объект WShell содержит коллекцию путей SpecialFolders ко всем спецпапкам (табл. 3.16).

***Таблица 3.16.*** *Список спецпапок*

| Название папки | Описание |
|---|---|
| AllUsersDesktop | Хранят данные для всех пользователей |
| AllUsersStartMenu | |
| AllUsersPrograms | |
| AllUsersStartup | |
| Desktop | Рабочий стол |
| Favorites | Избранное |
| Fonts | Шрифты |
| MyDocuments | Мои документы |
| NetHood | Сетевое окружение |
| Programs | Программы |
| Recent | Корзина |
| SendTo | Отправить на |
| StartMenu | Меню пуск |
| Startup | Автозагрузка |
| Templates | Шаблоны |

Получение списка всех доступных спецпапок и соответствующих им путей показано в листинге 3.21 и на рис. 3.8. Члены набора `SpecialFolders` являются элементами массива.

**Листинг 3.21. Получение списка доступных спецпапок**

```
Set Wshell=CreateObject("Wscript.Shell")
For Each variables In Wshell.SpecialFolders
       temp=temp+ variables +chr(13)
Next
Msgbox temp
```

Объект `Wshell.SpecialFolders` имеет два аналогичных свойства — `Count` (для VBScript) и `Length` (для JScript).

```
Set Wshell=CreateObject("Wscript.Shell")
Wscript.Echo "Всего спецпапок: " &Wshell.SpecialFolders.Count
```

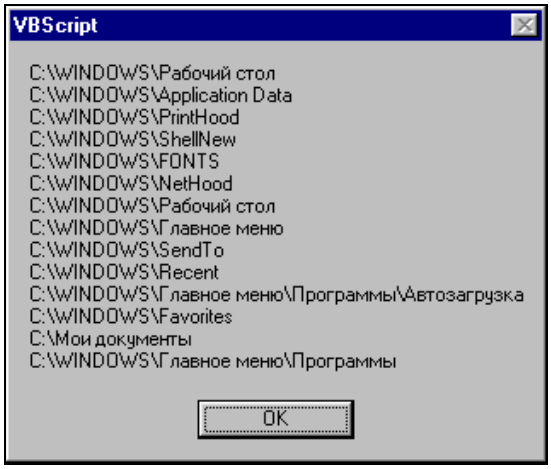



**Рис. 3.8.** Список доступных на рабочей станции спецпапок

Если необходимо получить путь к определенной спецпапке, то в запросе указывают ее конкретное название (см. табл. 3.16) — листинг 3.22.

**Листинг 3.22. Получение пути к определенной спецпапке**

```
Set Wshell=CreateObject("Wscript.Shell")
Wscript.Echo "Путь к папке с рабочим столом:"
&Wshell.SpecialFolders(Desktop)
```

# Объект *WshNetwork*

Свойства и методы объекта `WshNetwork` (табл. 3.17 и 3.18) позволяют получить доступ к различных объектам сети, управлять различными сетевыми ресурсами (дисками и принтерами), считывать имя пользователя, название домена (листинг 3.23, рис. 3.9).

***Таблица 3.17.*** *Свойства, поддерживаемые объектом* `WshNetwork`

| Свойство | Описание |
|---|---|
| `UserName` | Получение имени текущего пользователя |
| `ComputerName` | Считывание имени рабочей станции |
| `UserDomain` | Определение имени домена, в котором находится рабочая станция |

**Листинг 3.23. Использование свойств объекта `WshNetwork`**

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
Temp="Рабочая станция: "+WshNetwork.ComputerName+chr(13)
Temp=Temp+"Пользователь: "+WshNetwork.UserName+chr(13)
Temp=Temp+"Домен: "+WshNetwork.UserDomain
Wscript.Echo Temp
```

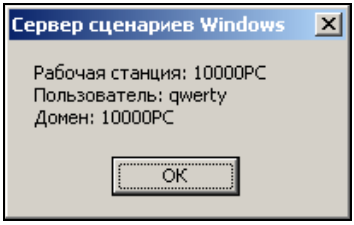



**Рис. 3.9.** Использование свойств объекта `WshNetwork`

***Таблица 3.18.*** *Методы, поддерживаемые объектом* `WshNetwork`

| Метод | Описание |
|---|---|
| `AddWindowsPrinterConnection` | Подключение сетевого принтера для Windows |
| `AddPrinterConnection` | Подключение сетевого принтера для Windows и DOS |
| `RemovePrinterConnection` | Удаление сетевого принтера |
| `EnumPrinterConnections` | Перечисление соединений с сетевыми принтерами |
| `SetDefaultPrinter` | Установка принтера по умолчанию |
| `MapNetworkDrive` | Подключение сетевого диска |
| `EnumNetworkDrives` | Перечисление подключенных сетевых дисков |
| `RemoveNetworkDrive` | Отключение сетевого диска |

## Методы *AddWindowsPrinterConnection* и *AddPrinterConnection*

Метод `AddWindowsPrinterConnection` используется для подключения сетевых принтеров в средах Windows 2k.

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.AddWindowsPrinterConnection(strPath)
```

Где `strPath` — путь к принтеру в формате UNC (`\\Server\ShareName`).

В том случае если предполагается печать в MS-DOS-эмуляции, подключите локальный порт принтера к сетевому, использовав метод `AddPrinterConnection` объекта `WshNetwork`:

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.AddPrinterConnection(strPort, strPath, [strConnect [, strUser-
Name [, strPWD]]])
```

Где `strPort` — локальный порт (`LPT1`).

`strPath` — путь к принтеру в формате UNC (`\\Server\ShareName`).

`strConnect` — необязательный параметр, характеризующий тип соединения: временный или постоянный; по умолчанию — `False` (временный).

`strUserName` — необязательный параметр, указывает имя пользователя, к которому монтируется принтер.

`strPWD` — необязательный параметр, пароль пользователя.

Не рекомендуется использовать четвертый и пятый параметры, т. к. публикация паролей наносит серьезный вред системе безопасности сети.

## Обработка ошибок

Подключая сетевой принтер, рекомендуется использовать обработчик ошибок для обеспечения корректной работы скрипта, создания условий для решения сложившихся проблем в кратчайшие сроки (листинг 3.24).

**Листинг 3.24. Обработчик ошибок при подключении сетевых принтеров**

```
On Error Resume Next
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.AddWindowsPrinterConnection(Printer)
 Select Case Err.Number
Case 0
Temp="Сетевой принтер успешно подключен"
Case -2147023688
Temp="Сетевой принтер невозможно подключить: ресурс "& Printer &" не най-
ден"
Case -2147024811
Temp="Сетевой принтер уже подключен"
Case Else
```

```
Temp="Ошибка:" &Err.Number&" "& Err.Descripion
End Select
MsgBox Temp
```

## Метод *RemovePrinterConnection*

Этот метод используется для удаления подключения сетевого принтера (листинг 3.25).

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.RemovePrinterConnection(strName, [bForce], [bUpdateProfile])
```

Где `strName` — имя удаляемого принтера (`ShareName`) или порта (`LPT1`, `LPT2`), к которому подключен принтер.

`bForce` — необязательный параметр; булево значение (`True` или `False`), определяющее, будет ли удалена связь, если принтер еще используется.

`bUpdateProfile` — необязательный параметр, управляющий профилем пользователя: `True` — удаление профиля пользователя, `False` — нет.

**Листинг 3.25. Удаление сетевого принтера с порта LPT1**

```
Port_Name="LPT1"
On Error Resume Next
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.RemovePrinterConnection(Port_Name)
Select Case Err.Number
Case 0
Temp="Сетевой принтер успешно подключен"
Case -2147023688
Temp="Сетевой принтер невозможно подключить: ресурс "& Printer &" не найден"
Case -2147024811
Temp="Сетевой принтер уже подключен"
Case Else
Temp="Ошибка:" &Err.Number&" "& Err.Descripion
End Select
MsgBox Temp
```

## Метод *EnumPrinterConnections*

Метод `EnumPrinterConnections` используется для определения подключенных сетевых принтеров к данной рабочей станции (листинг 3.26). Это свой-

ство возвращает массив, членами которого являются элементы, содержащие информацию о подключенных сетевых принтерах:

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
Set Property= WshNetwork.EnumPrinterConnections
```

Где `Property` — массив, содержащий информацию о подключенных принтерах. Четные элементы данного массива содержат имена портов, к которым подключены принтеры, нечетные — имя принтера в формате UNC (`\\Server\ShareName`). Нумерация элементов начинается с 0.

**Листинг 3.26. Определение всех принтеров, подключенных к рабочей станции**

```
Temp=""
Set WshNetwork=WScript.CreateObject("WScript.Network")
Set Property= WshNetwork.EnumPrinterConnections
For i=0 to Property.count-1 step 2
       Temp=Temp& Property(i) &" "& Property(i+1)& chr(13)+chr(10)
Next
MsgBox Temp
```

## Метод *SetDefaultPrinter*

С помощью метода `SetDefaultPrinter` на рабочей станции устанавливается принтер по умолчанию.

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.SetDefaultPrinter(strPath)
```

Где `strPath` — путь к принтеру в формате UNC (`\\Server\ShareName`).

## Метод *MapNetworkDrive*

С помощью метода `MapNetworkDrive` осуществляется подключение сетевых дисков к рабочей станции (листинг 3.27). Вызов метода производится следующим образом:

```
WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.MapNetworkDrive(strLetter,strPath)
```

Где `strLetter` — буква, на которую будет подключен сетевой ресурс.

`strPath` — путь к сетевому ресурсу в формате UNC (`\\Server\ShareName`).

**Листинг 3.27. Предоставление доступа к сетевому ресурсу с использованием обработчика ошибок**

```
On Error Resume Next
WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.MapNetworkDrive("Z:","\\Server\Multimedia")
ErrCheck Err.Number
…
Sub ErrCheck(Errors)
Select Case Errors
Case 0
       Wscript.Echo "Подключение диска выполнено успешно"
Case -2147024829
       Wscript.Echo "Ошибка: сетевой ресурс не существует"
Case -2147024811
       Wscript.Echo "Ошибка: диск уже существует"
Case Else
       Wscript.Echo "Ошибка: "& Cstr(Errors)
End Select
End Sub
```

## Метод *EnumNetworkDrives*

Для создания списка подключенных дисков используется метод `EnumNetworkDrives`, возвращающий массив, элементы которого содержат соответствующую информацию (листинг 3.28). Как и в методе `EnumPrinterConnections`, четные элементы массива содержат локальное имя диска, а нечетные — соответствующие UNC-пути. Нумерация элементов начинается с нуля.

**Листинг 3.28. Использование метода `EnumNetworkDrives`**

```
Temp=""
Set WshNetwork=WScript.CreateObject("WScript.Network")
Set Property= WshNetwork.EnumNetworkDrives
For i=0 to Property.count-1 step 2
       If Property(i)<>"" then
```

```
            Temp=Temp& Property(i) &" "& Property(i+1)& chr(13)+chr(10)
        End If
Next
MsgBox Temp
```

## Метод *RemoveNetworkDrive*

Как следует из названия, метод предназначен для отключения сетевого диска (листинг 3.29). Единственным параметром метода является буква, которой сопоставлен отключаемый ресурс.

**Листинг 3.29. Отключение сетевого диска**

```
Set WshNetwork=WScript.CreateObject("WScript.Network")
WshNetwork.RemoveNetworkDrive("Z:")
```



# Глава 4

# На пути от VBScript к .NET

## Переход от VBScript к ASP

Для программиста, создающего приложения для обслуживания серверов, поддержка OLE-объектов используемой им средой — основное требование, поскольку формирование отчетов, доступ к Active Directory и др. базируется на их использовании. DHTML и HTA основаны на HTML, который позволит делать вставки на VBScript или JScript, однако не поддерживает работу с OLE-объектами. ASP представляет собой решение, которое поддерживает HTML, OLE-объекты и позволяет делать вставки на скриптовых языках VBScript и JScript.

Переход от VBScript к ASP достаточно прост: исходный код на VBScript остается практически без изменений.

ASP-страница — это сценарий, программный код которого выполняется при запросе. Результатом действия скрипта является статическая HTML-страница, которая формируется на сервере, а затем отображается в браузере клиента. В первой строке ASP-документа (листинг 4.1) всегда указывается язык, с помощью которого созданы скриптовые вставки:

```
<%Script Language="VBScript"%>
```

или

```
<%Script Language="JScript"%>
```

Программный код, находящийся между `<%` и `%>`, выполняется на сервере и подчинен синтаксису одного из выбранных языков. Весь остальной код представляет собой HTML-страницу в явном виде.

**Листинг 4.1. Типовая ASP-страница**

```
<%@ Language=VBScript CODEPAGE=1251%>
<HTML>
<TITLE> Заголовок страницы </TITLE>
<HEAD>
<LINK href="../style.css" type=text/css rel=stylesheet>
<meta http-equiv="Content-Type" content="text/html" charset=windows-1251>
</HEAD> <BODY>
<FONT FACE="Arial">
…
<%
…
Response.write variable ' отображение на экране содержимого переменной
%>
```

После обработки интерпретатором IIS программного кода и преобразования результатов его работы в HTML/DHTML, необходимо дать команду на отображение страницы в браузере клиента. Такой командой является `Response.Write q`, где `q` — имя переменной, содержащей фрагмент HTML-кода.

## Настройка IIS для ASP

Создавая Web-приложения на основе ASP, следует учитывать некоторые особенности ASP.NET.

Код ASP-страниц исполняется на сервере, и только результат в виде HTML-страницы пересылается на клиентскую рабочую станцию. Поэтому для успешного запуска приложения на сервере, пользователь должен обладать соответствующими правами. IE, IIS и запускаемые им сервисы представляют собой трехзвенную систему (рис. 4.1).

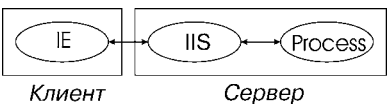



**Рис. 4.1.** Схема взаимодействия "клиент-сервер"

Пусть IIS имеет настройки по умолчанию. В этом случае, при загрузке любой ASP-страницы она стартует от имени встроенного пользователя (рис. 4.2).

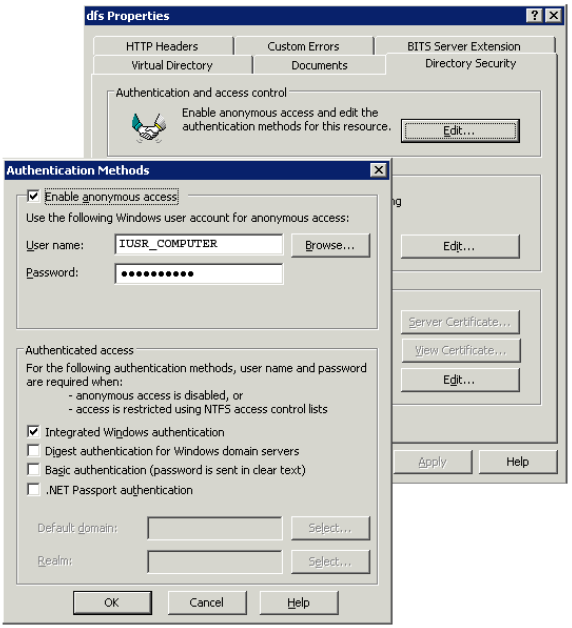



**Рис. 4.2.** Управление учетной записью запуска сервиса

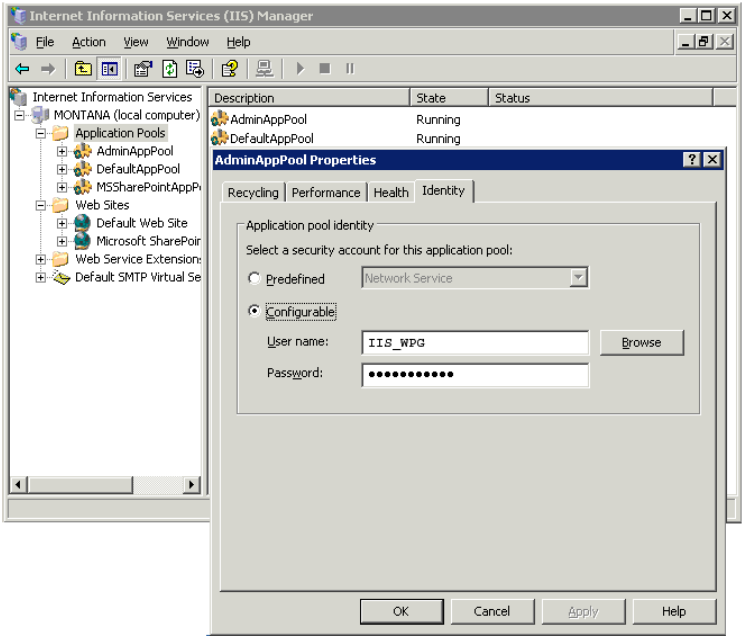



**Рис. 4.3.** Управление имперсонализацией сайта

Если страница работает с базами данных, например с Active Directory, то пользователь, запускающий данную страницу, должен обладать соответствующими правами системного администратора. Существует несколько способов соблюсти эти условия. Первый — учетную запись встроенного пользователя заменить учетной записью администратора сети. При таком решении любой пользователь в сети сможет посетить данную страницу, т. к. она будет запускаться от имени системного администратора. Этот способ предоставляет всем доступ к данной странице, что резко снижает безопасность всей системы. В случае ошибок на странице злоумышленник может легко запустить вредоносный код с правами администратора. Поэтому разумно использовать другой способ, с помощью которого можно ограничить доступ к ресурсам. В настройках IIS необходимо сбросить флажок (рис. 4.3) с `Enable anonymous access` и установить его напротив `Basic Authentication`. Также следует изменить права на файловую структуру используемого сайта, исключив оттуда группу `Everyone` и добавив соответствующие группы безопасности, которым назначить соответствующие права. При такой настройке IIS только системные администраторы получат доступ к данной странице. При попытке любого пользователя, не являющегося администратором сети, получить доступ к странице, интерпретатором IIS будут запрошены имя и пароль пользователя.

Если необходимо расширить круг лиц, которым должен быть доступен данный сайт и при этом пользователи не являются системными администраторами, то можно воспользоваться вариантом, являющимся синтезом двух ранее изложенных решений. Ограничить доступ на сайт с помощью **Basic Authentication** и правами на файловую структуру и запускать скрипт, интергированный в страницу с правами администратора.

Первым звеном трехзвенной системы является рабочая станция пользователя, вторым — сервер, на котором установлен IIS. Взаимосвязь этих звеньев осуществляется с помощью одного пользователя. Между вторым и третьим звеном — сервером IIS-процессов, порождаемых из ASP-процесса, — взаимодействие осуществляется с помощью другого пользователя. Рассмотрим взаимодействие второго и третьего звена подробнее.

При запуске из кода ASP-страницы какого-либо приложения осуществляется взаимодействие между вторым и третьим звеном. IIS порождает процесс, запускаемый от имени другого встроенного пользователя. Поскольку ASP-страница выполняется на сервере, то для запуска приложения необходимы соответствующие права. Управление этой учетной записью пользователя осуществляется в **Application Pools** (рис. 4.3).

# Переход от VBScript и ASP к ASP.NET

В построенной трехзвенной системе большая брешь в безопасности, поскольку для доступа к AD в ASP-файле в явном виде нужно указать имя и пароль системного администратора. Другим недостатком является публикация пароля администратора на вкладке **Anonymous Access** вместо встроенной учетной записи. Конечно, системный администратор постарается защитить файловую систему соответствующим распределением прав, однако принятых мер не достаточно. Необходимо сделать так, чтобы система сама определяла и подставляла имя и пароль пользователя между вторым и третьим звеном, т. е. запускала сервисы от имени пользователя, который вошел на сайт. Поставленная задача успешно решается переходом на ASP.NET и включением режима имперсонализации (Windows Authentication).

## Установка Visual Studio

Перед установкой Visual Studio .NET должен быть предварительно инсталлирован пакет программ (табл. 4.1):

- Microsoft IIS 5/6;
- Microsoft .NET Framework 1.1/2.0.3.0;
- Microsoft FrontPage Server Extensions 2000/2002;
- Microsoft Visual J# .Net Redistributable Package 1.1/2;
- Microsoft Windows Installer 2/3(.1).

В полном дистрибутиве перечисленные компоненты находятся в папке WCU, в которой присутствуют соответствующие им подпапки. Если же папки WCU нет, то компоненты необходимо загрузить с сайта Microsoft или установить с дистрибутива соответствующей версии операционной системы.

***Таблица 4.1.*** *Необходимые приложения для инсталляции Visual Studio .NET*

| Продукт | Источник |
|---|---|
| Microsoft IIS 5 | Входит в состав Windows 2000 — ver 5.0, XP — ver 5.1 |
| Microsoft IIS 6 | Входит в состав Windows 2003 Server |
| Microsoft .NET Framework 1.1 | **http://www.microsoft.com/downloads/details.aspx?FamilyID=262d25e3-f589-4842-8157-034d1e7cf3a3&displaylang=en** |
| Microsoft .NET Framework 2.0 | **http://www.microsoft.com/downloads/details.aspx?familyid=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=en** |
| Microsoft .NET Framework 3.0 | **http://www.microsoft.com/downloads/details.aspx?FamilyID=10cc340b-f857-4a14-83f5-25634c3bf043&DisplayLang=en** |

***Таблица 4.1*** *(окончание)*

| Продукт | Источник |
|---|---|
| Microsoft FrontPage Server Extensions 2000 | Входит в состав Windows 2000, XP |
| Microsoft FrontPage Server Extensions 2002 | Входит в состав Windows 2003 Server; KB13380<br>**http://www.microsoft.com/downloads/details.aspx?FamilyId=3E8A21D9-708E-4E69-8299-86C49321EE25&displaylang=en** |
| Microsoft Visual J# .Net Redistributable Package 1.1 | **http://www.microsoft.com/downloads/details.aspx?FamilyID=e3cf70a9-84ca-4fea-9e7d-7d674d2c7ca1&DisplayLang=en** |
| Microsoft Visual J# .Net Redistributable Package 2 | **http://www.microsoft.com/downloads/details.aspx?FamilyID=90cef3f3-9eaf-4d41-bad3-9f44fe8e5e81&DisplayLang=en** |
| Microsoft Windows Installer 2 | **http://www.microsoft.com/downloads/details.aspx?FamilyID=4b6140f9-2d36-4977-8fa1-6f8a0f5dca8f&DisplayLang=en** |
| Microsoft Windows Installer 3.1 | **http://www.microsoft.com/downloads/details.aspx?FamilyID=889482fc-5f56-4a38-b838-de776fd4138c&DisplayLang=en** |

После завершения установки необходимого пакета программ запустите процесс установки студии. Затем перезагрузите рабочую станцию и проверьте работоспособность Visual Studio .NET, создав проект.

Для инсталляции Visual Studio .NET без проверки установленных компонентов, выполните команду:

```
X:\SETUP\SETUP.EXE /NO_BSLN_CHECK
```

# Microsoft FRAMEWORK

Компания Microsoft поддерживает одновременно несколько языков программирования: Visual Basic, C++, C#. Большая часть функциональных возможностей этих языков совпадает: в каждом из них реализована работа с файловой системой, с базами данных, обработка строк, математические функции. Более того, некоторые из них поддерживают идентичные функции, операторы цикла и условные операторы. Наконец, многие из них имеют похожие типы данных.

Поддержка одних и тех же функциональных возможностей для нескольких языков программирования требует немалых усилий от разработчика, поэтому программисты компании Microsoft задумались над тем, как уменьшить трудозатраты. Результатом их деятельности стала библиотека классов .Net Framework Class Library, которая состоит из множества классов, предназначенных для решения различных задач. В настоящее время активно используется .Net Framework 3.0.

Пространство имен платформы .Net содержит более 3400 классов, которые организованы в иерархию имен. Например, пространство имен, связанных с работой файловой системы, называется `System.IO`.

Существует два способа импорта пространств имен в зависимости от метода программирования. Оба они будут рассмотрены далее на примере импорта пространства имен `System.Security.Principal` для проверки режима имперсонализации.

## Управление доверительными отношениями в ASP.NET

Для конфиденциальной работы приложения IIS всегда важно идентифицировать процесс, в котором оно выполняется. По умолчанию в IIS 5.0 процессы запускаются утилитой ASPEN_WP.EXE от имени встроенной учетной записи `PC_NAME\ASPNET`, а в IIS 6.0 — от имени `NT AUTHORITY\NETWORK SERVICE`. При этом режим имперсонализации выключен. Для его включения выполните описанные далее изменения в конфигурационном файле:

C:\Windows\Microsoft.NET\FrameWork\V номер версии\Config\Machine.Config

Присвойте следующие значения:

```
comAuthenthicationLevel= PktPrivacy;
comImpersonationLevel= Impersonate;
```

Список возможных значений этих параметров приведен далее:

```
comAuthenticationLevel="Default|None|Connect|Call|Pkt|PktIntegrity|
PktPrivacy"; comImpersonationLevel="Default|Anonymous|Identity|
Impersonate|Delegate";
```

Для IIS6 сделайте изменения в реестре. В ветви реестра `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters` — добавьте четыре параметра (рис. 4.4):

- `AuthentionLevel`;
- `CoInitializeSecurityParam`;
- `ImpersonationLevel`;
- `AuthenticationCapabilities`.

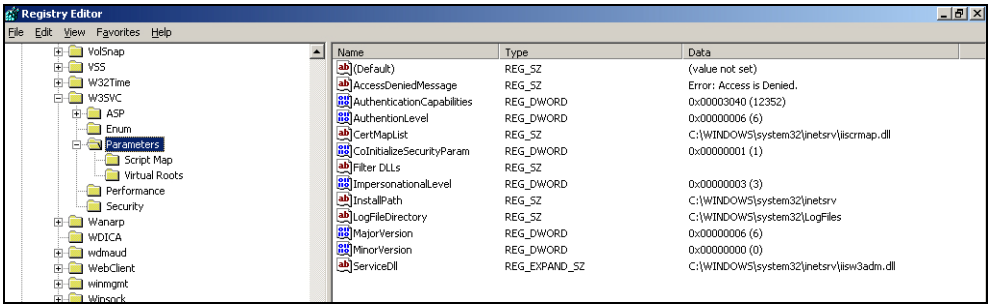



**Рис. 4.4.** Настройка имперсонализации в IIS

Параметр `CoInitializeSecurityParam` принимает значения 0 или 1 и отвечает за включение/отключение использования остальных трех. Все параметры имеют тип `REG_DWORD` (табл. 4.2).

***Таблица 4.2.*** *Описание параметров аутентификации IIS*

| **Имя параметра** | **Значение** | **Имя параметра** | **Значение** |
|---|---|---|---|
| **`AuthentionLevel`** | | **`AuthenticationCapabilities`** | |
| `Default` | 0 | `None` | 0x0 |
| `None` | 1 | `Mastual_Auth` | 0x1 |
| `Connect` | 2 | `Secure_Refs` | 0x2 |
| `Call` | 3 | `Access_Control` | 0x4 |
| `Pkt` | 4 | `APPID` | 0x8 |
| `PktIntegrity` | 5 | `Dynamic` | 0x10 |
| `PktPrivacy` | 6 | `Static_Cloaking` | 0x20 |
| **`ImpersonationLevel`** | | `Dynamic_Cloaking` | 0x40 |
| `Default` | 0 | `ANY_Authority` | 0x80 |
| `Anonymous` | 1 | `Make_FullSic` | 0x100 |
| `Identity` | 2 | `Require_FullSic` | 0x200 |
| `Impersonate` | 3 | `Auto_Impersonate` | 0x400 |
| `Delegate` | 4 | `Default` | 0x800 |
| | | `Disable_Aaa` | 0x1000 |
| | | `No-Custom_Marshal` | 0x1200 |

Для включения имперсонализации рекомендуется присвоить перечисленным параметрам следующие значения:

```
CoInitializeSecurityParam=1;
AuthentionLevel=6;
ImpersonationLevel=3;
AuthenticationCapabilities=12352;
```

В каждом создаваемом проекте автоматически генерируется файл Web.Config, в который также необходимо внести изменения, чтобы включить режим имперсонализации (листинг 4.2).

**Листинг 4.2. Файл Web.Config**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<system.web>
<identity impersonate ="true"/>
<authentication mode="Windows" />
</system.web>
</configuration>
```

Для того чтобы механизм имперсонализации работал от имени одного и того же пользователя, то нужно указать имя учетной записи и ее профиль в поле `<identity>`:

```
<identity impersonate ="true" username="user/domain" password="" />
```

Проверка имперсонализации осуществляется с помощью функции `WindowsIdentity.GetCurrent().Name`, возвращающей имя учетной записи пользователя, от которой будет запущен процесс.

Если программный код формируется непосредственно в ASPX-файла, то в его заголовок необходимо импортировать пространство имен `System.Security.Principal` (2-я строчка примера на рис. 4.5).

При обработке событий в виде функций следует перед описанием класса импортировать пространство имен, о котором уже шла речь, и привязать код к событию, например, к нажатию кнопки. Для этого сначала создайте в ASPX-файле кнопку (`Button1`) и в теле соответствующей функции, по умолчанию `Button1_Click()`, добавится функция `WindowsIdentity.GetCurrent().Name` (рис. 4.6).

С включенной имперсонализацией на странице должно отобразиться текущее доменное имя пользователя в формате `Domain/LogonUser`.

```
<%@ Page Language="vb" AutoEventWireup="false" Codebehind="impersonate.aspx.vb" Inherits="fs
<%@ import namespace="System.Security.Principal"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <title>WebForm2</title>
        <meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" content="Visual Basic .NET 7.1">
        <meta name="vs_defaultClientScript" content="VBScript">
        <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5"
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
            <%
                response.write( WindowsIdentity.GetCurrent().Name)
            %>
        </form>
    </body>
</HTML>
```

**Рис. 4.5.** Импорт пространства имен `System.Security.Principal` в ASPX-файле

```
Imports System.Security.Principal
Public Class WebForm2
    Inherits System.Web.UI.Page

    Web Form Designer Generated Code

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Console.WriteLine(WindowsIdentity.GetCurrent().Name)
    End Sub
End Class
```

**Рис. 4.6.** Импорт пространства имен `System.Security.Principal` в VB-файле

# Изменения в синтаксисе

По своей сути ASP представляет собой HTML со вставками VBScript или JScript (в общем случае), при этом ASP, в отличие от HTML, работает с OLE-объектами. Это принципиальная разница между ASP и DHTML. Вставки на VBScript практически не претерпели изменений за некоторым исключением. Переход от ASP к ASP.NET также достаточно прост, однако есть ряд принципиальных отличий.

## Файловая структура

Сопоставление вызываемых файлов соответствующим приложениям приведено в табл. 4.3.

***Таблица 4.3.*** *Расширения файлов, используемых в VBscript, ASP и ASP.NET*

| | **VBScript** | **ASP** | **ASP.NET** |
|---|---|---|---|
| **Расширение** | vbs | asp | aspx |
| **Обработчик** | cscript.exe, wscript.exe | Asp.dll | Aspnet_isapi.dll |

## Исполнение ASP-файлов в ASP.NET

Для страниц ASP.NET фактически приемлемы любые расширения, в том числе и ASP. Для того чтобы ASP-страницы распознавались файлами платформы ASP.NET, необходимо проделать следующее:

- ❒ Изменить ассоциацию ASP-файлов:
  - запустить Internet Services Manager (**Start | Programs | Administrative Tools | Internet Services Manager**);
  - открыть страницу свойств нужного Web-узла, щелкнув правой кнопкой мыши по нужному узлу и выбрав пункт **Properties**;
  - в появившемся диалоговом окне открыть вкладку **Home Directory**;
  - в разделе **Application Settings** нажать кнопку **Configuration**;
  - в появившемся диалоговом окне **Application Configuration** выбрать вкладку **App Mapping**;
  - ассоциировать файлы с расширением asp с библиотекой aspnet_isapi.dll. При этом предварительно необходимо удалить предыдущую ассоциацию asp.
- ❒ Внести изменения в конфигурационный файл .NET Framework:
  - открыть файл machine.config (листинг 4.3) с установленным на компьютере IIS, который находится в папке C:\Windows\Microsoft.NET\Framework\[version]\config;
  - в разделе `httpHandlers` приведены обработчики для файлов с различными расширениями. В нем должны присутствовать `*.asp`, `*.aspx`;
  - добавьте обработчик `System.Web.Ui.PageHandlerFactory`.

**Листинг 4.3. Фрагмент файла machine.config**

```
<add
verb="*.*"
path="*.asp"
type=" Web.Ui.PageHandlerFactory, System.Web, Version=1.0.2411.0, Cul-
ture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
```

# Оператор *Option Explicit*

Рассмотрим использование оператора `Option Explicit` для VBScript, ASP и ASP.NET.

## VBScript, ASP

По умолчанию в VBScript допускается неявное объявление переменных. При этом переменная создается без ее предварительного объявления операторами `Dim`, `Private`, `Public` или `ReDim`. Однако, разрешив неявное объявление переменных в сценариях, велик риск пропустить допущенную синтаксическую ошибку в имени переменной во время программирования.

В том случае, если допущена ошибка, VBScript просто объявит новую переменную и создаст ее, вследствие чего программа будет работать некорректно. Для обнаружения ошибок-опечаток такого рода в первую строку программы помещают оператор `Option Explicit`, который сигнализирует оператору о переменных, которые не были объявлены явно.

## ASP.NET

Опция `Explicit`, выключенная по умолчанию в предыдущих версиях ASP, на платформе ASP.NET включена. При этом перед использованием переменной ее нужно объявить. Например, приведенный в листинге 4.4, *а* сценарий успешно работает на ASP, но на ASP.NET он вызовет ошибку — `The name 'strMsgBox' is not declared`.

**Листинг 4.4, *а***

```
strMsgBox="Привет"
response.write(strMsgBox)
%>
```

Исправить ошибку можно двумя способами. Первый способ — объявить переменную `strMsgBox` явным образом — листинг 4.4, *б*.

**Листинг 4.4, *б***

```
<%
Dim strMsgBox As String
strMsgBox="Привет"
response.write(strMsgBox)
%>
```

Второй способ — отключить на всех страницах ASP.NET в файле machine.config опцию `Explicit` — листинг 4.4, *в*.

**Листинг 4.4, *в***

```
<Compilation>
Explicit = "False"
</Compilation>
```

## Преобразование типов данных

В языке VBScript отсутствует требование жесткой привязки переменной к определенному типу. В нем нельзя объявить тип переменной, в результате чего все создаваемые переменные принадлежат к типу `Variant`. В ASP.NET этот тип данных не поддерживается. Своеобразным его эквивалентом является тип `Object`. Рассмотрим пример сценария, в котором объявим переменную, и присвоим ей значение, после чего выведем его на экран (листинг 4.5). В качестве шаблона воспользуемся примером из предыдущего параграфа.

**Листинг 4.5. Автоматическое преобразование типа переменной**

```
<%
Dim strMsgBox
strMsgBox="Привет"
response.write(strMsgBox)
%>
```

При объявлении переменной `strMsgBox` не указан ее тип явным образом, поэтому считается, что она принадлежит к типу `Object`. При присвоении значения переменной осуществляется автоматическое преобразование к типу `String`.

Автоматическое преобразование к нужному типу данных очень удобно, однако оно негативно сказывается на скорости работы сценария.

Для определения типа переменной используйте функцию `VarType()`, возвращающую число, которому соответствует подтип (листинг 4.6).

При работе с массивами функция `VarType()` никогда не возвращает значение 8192. Это связано с тем, что элементы массива содержат данные, которые также соответствуют одному из типов данных. Например, если элементами массива являются числа типа `Integer`, то функция будет возвращать значения 2+8192, т. е. 8194.

**Листинг 4.6, *а*. Определение типа данных. VBScript**

```
Dim ArrayName(100)
T=""
T=VarType(ArrayName)
MsgBox T
```

**Листинг 4.6, *б*. Определение типа данных. ASP, ASP.NET**

```
<%
Dim ArrayName() as String ' Динамический массив
                          ' Элементы массива — строки
Response.Write(VartType(Parametr))
%>
```

Приведенный пример возвращает значение 8204=8192+12. Элементы массива не определены, поэтому они имеют тип данных `Variant (12)`. Значения, возвращаемые функцией `VarType`, рассмотрены в *главе 2*.

Процесс автоматического преобразования типов данных называется динамическим связыванием (late binding). Избежать его можно с помощью опции `Strict`:

```
<%@ Strict="True" %>
```

При включении опции `Strict` автоматически включается `Explicit`. Кроме того, опцию `Strict` можно включить для всех страниц ASP.NET в файле `machine.config` (листинг 4.7).

**Листинг 4.7. Раздел `<Compilation>` файла Machine.Config**

```
<Compilation>
        Strict = "True"
</Compilation>
```

## Использование скобок при передаче параметров подпрограммам и методам

В отличие от VBScript-сценариев и классических ASP-страниц в ASP.NET при передаче параметров подпрограммам или методам всегда должны использоваться скобки. Например, приведенный в листинге 4.8, *а* сценарий без ошибок будет выполняться под ASP.

**Листинг 4.8, *а*. Передача параметров. ASP**

```
<% Response.Write "Привет" %>
```

Но в ASP.NET он вызовет ошибку, поскольку метод `Write` необходимо использовать со скобками — листинг 4.8, *б*.

**Листинг 4.8, *б*. Передача параметров. ASP.NET**

```
<% Response.Write ("Привет")%>
```

## Операторы *SET* и *LET*

В VBScript и ASP для присвоения переменных необходимо было использовать переменную `SET` — листинги 4.9, *а* и *б*.

**Листинг 4.9, *а*. Присвоение переменных. VBScript**

```
Set obj=CreateObject("Adodb.Connection")
```

**Листинг 4.9, *б*. Присвоение переменных. ASP**

```
<% Set obj=Server.CreateObject("Adodb.Connection") %>
```

На страницах ASP.NET он больше не используется. Синтаксис присвоения переменной объекта выглядит следующим образом — листинг 4.9, *в*.

**Листинг 4.9, *в*. Присвоение переменных. ASP.NET**

```
<% obj=Server.CreateObject("Adodb.Connection") %>
```

Также не поддерживается оператор `LET`.

## Поддержка многопоточных компонентов

По умолчанию в ASP.NET не поддерживаются многопоточные методы, такие как ADO Connection, Scripting Dictionary. Для включения их поддержки в заголовке файла используют директиву:

```
<%@ PageASPCompat="True" %>
```

# Обработка ошибок в ASP.NET

Для обработки ошибок предназначен класс `System.Exception`. Анализируемый на ошибки код заключается внутрь конструкции `Try` (листинг 4.10).

**Листинг 4.10. Обработка ошибок в ASP.NET**

```
Tru
       <Sample Code>
       Cath err as Exception
       Response.write(err)
End Try
```

Для конкретизации ошибки, рекомендуется использовать встроенные свойства, например `Err.Message` (табл. 4.4).

***Таблица 4.4.*** *Свойства обработчика ошибок в ASP.NET*

| Свойство | Описание |
|---|---|
| `Err` | Краткое описание ошибки |
| `Err.Message` | Полное описание ошибки |
| `Err.GetType.ToString` | Тип ошибки |
| `Err.Source` | Источник ошибки (имя библиотеки) |



# Глава 5

# Программное управление реестром

*Реестр* — это иерархическая база данных, содержащая настройки аппаратного и программного обеспечения компьютера. Для обеспечения высокой скорости доступа к записям реестра, информация в нем хранится в двоичном формате, а сам реестр состоит из нескольких файлов.

В Microsoft Windows 3.x все настройки программного обеспечения располагались в файлах инициализации, которые имели расширение INI. Вся конфигурационная информация находится в двух файлах: SYSTEM.INI и WIN.INI. При установке любого приложения все его настройки сохранялись в одном из этих файлов. Приложения пользовались ограниченным количеством параметров. Оно ограничивалось размером INI-файла, который не должен превышать 64К. Чтобы обойти это ограничение, для каждой программы создавался свой INI-файл. Со временем, из-за большого количества конфигурационных файлов, производительность операционной системы значительно понижалась. В 1993 году была создана операционная система Microsoft Windows NT, в которой множество INI-файлов было заменено единой базой данных — реестром.

С точки зрения файловой системы реестр представляет собой файл с расширением DAT. Реестр Windows NT/200x хранится в файле NTUSER.DAT, который находится в каталоге %Windir%\Profiles. В Microsoft Windows принято использовать переменные среды. %WinDir% содержит полный путь к каталогу, в котором установлена ОС, например, C:\Windows.

В Windows 9x реестр состоит из двух файлов: USER.DAT и SYSTEM.DAT, которые хранятся в каталоге %Windir%. USER.DAT содержит настройки индивидуального пользователя, а SYSTEM.DAT — настройки компьютера.

Реестры Windows 9x, NT, 2000 несовместимы друг с другом, поэтому импорт ветвей из одной ОС в другую невозможен, однако идея построения реестров едина.

# Основы построения реестра

Реестр состоит из разделов верхнего уровня, называемых кустами (hives):

- HKEY_CLASSES_ROOT (HKCR);
- HKEY_CURRENT_USER (HKCU);
- HKEY_LOCAL_MACHINE (HKLM);
- HKEY_USER (HKU);
- HKEY_CURRENT_CONFIG (HKCC).

Структура реестра такова: в каждом из кустов находятся ключи, в которых содержатся параметры. Рассмотрим подробнее назначение кустов реестра.

В разделе HKLM находится информация об аппаратном и программном обеспечении, сведения о системе безопасности. Этот раздел — один из самых больших.

Раздел HKCR является виртуальной ссылкой на раздел HKLM\Software\Classes. В нем содержатся сведения обо всех расширениях файлов, определениях типов, ярлыках, привязке, классах идентификаторов и т. д.

Раздел HKU содержит настройки пользователя по умолчанию, в которые входят описания переменных среды, цветовых схем, шрифтов, сетевых настроек и т. д. Во время регистрации нового пользователя на рабочей станции, на жестком диске для него создается новый профиль. Настройки, содержащиеся в профиле, копируются из куста HKU.

Изменения в разделах HKU и HKLM можно сделать только с помощью утилиты Regedt32.exe в том случае, если у вас ОС Windows 2000, Regedit.exe — если Windows XP. Пользователь, от имени которого запускаются эти утилиты, должен обладать правами системного администратора.

Раздел HKCU содержит сведения о текущем пользователе и имеет название, соответствующее значению идентификатора безопасности (SID) данного пользователя. Каждый раз при перезагрузке компьютера этот раздел создается заново.

Раздел HKCC является ссылкой на текущий профиль оборудования, хранящийся в HKLM. С помощью профиля оборудования определяют список устройств, драйвера которых будут подгружены в данном сеансе работы пользователя. Профили изначально предназначены для переносных компьютеров.

Раздел HKDD не хранится в реестре, а динамически создается при загрузке операционной системы. Раздел содержит сведения о самонастраивающихся устройствах (Plug-and-Play).

Как и любая база данных, реестр поддерживает несколько типов данных (табл. 5.1).

***Таблица 5.1.*** *Типы данных, поддерживаемые реестром*

| Наименование | Тип данных | Описание |
|---|---|---|
| REG_NONE | Не определен | Зашифрованные данные |
| REG_SZ | Строка | Символьный текст |
| REG_EXPAND_SZ | Строка | Текст с переменными |
| REG_BINARY | Двоичный | Двоичные данные |
| REG_DWORD | Число | Цифровые данные |
| REG_DWORD_BIG_ENDIAN | Число | Данные с "не-интеловским" порядком байт |
| REG_LINK | Строка | Путь к файлу |
| REG_MULTI_SZ | Массив | Элементы массива — строки |
| REG_RESOURCE_LIST | Строка | Список оборудования |
| REG_FULL_RESOURCE_DESCRIPTION | Строка | Идентификатор оборудования |
| REG_FULL_RESOURCE_LIST | Строка | Идентификатор оборудования |

Групповые политики, описанные в ADM-файлах, могут производить операции только со следующими типами данных: REG_SZ, REG_EXPAND_SZ, REG_DWORD.

# Редакторы реестра

Используя редактор реестра, можно выполнить дополнительную настройку Windows. Перед тем как внести изменения в реестр, рекомендуется сделать резервные копии изменяемых ветвей.

В Windows 2k существует несколько средств, предназначенных для работы с реестром: Regedit и Regedt32. Каждое из них обладает своим набором функций. Regedit — это средство редактирования реестра, входящее в состав всех версий Microsoft Windows. С его помощью можно выполнять удаление, сохранение, восстановление, поиск параметров и разделов реестра на локальном и удаленном компьютере. Отличительной чертой редактора является то, что изменения, сделанные в реестре, начинают действовать моментально.

Regedt32 — редактор, появившийся в Windows 2k (рис. 5.1). Каждый раздел в данном редакторе располагается в отдельном окне. При удаленном доступе к реестру для редактирования доступны только два куста: `HKLM` и `HKU`. Редактор позволяет менять права доступа для ветви реестра, что может использоваться, например, при перемещении пользователя из домена в домен. Для этого копируют профиль и изменяют права доступа на ветви. Regedt32 реализует изменения только после выгрузки редактируемого куста и закрытия редактора.

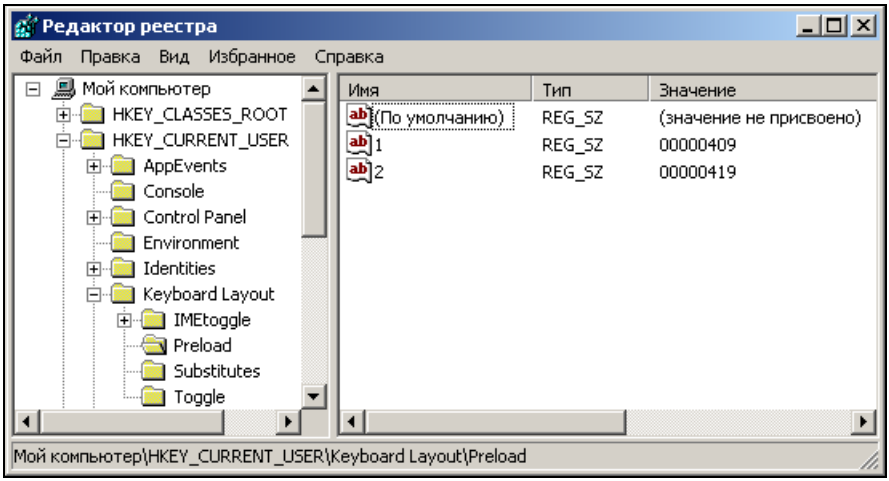



**Рис. 5.1.** Редактор реестра Regedit

Запуск редакторов осуществляется с помощью утилит Regedit.exe и Regedt32.exe, входящих в Windows 2000 по умолчанию. В Windows XP оба редактора объединены в Regedit.exe.

# Программное управление реестром

## Редактирование реестра сценарием на базе командной строки

Файлы, предназначенные для экспорта данных в реестр, имеют расширения reg и представляют собой текстовые файлы. С помощью REG-файла можно как добавлять, так и удалять данные. Для экспорта содержимого REG-файла в реестре используют команду `regedit.exe /s filename.reg`. Ключ `/s` предназначен для подавления подтверждения на вносимые изменения.

В начале любого REG-файла находится идентификатор. Для Windows 2k идентификатором является метка `Windows Registry Editor Version 5.00`, для Windows NT и 9x — `REGEDIT4`.

Общий вид REG-файла для добавления ключа (листинг 5.1):

```
Windows Registry Editor Version 5.00
[куст\путь к разделу]
"ключ"="тип значения:значение"
```

Если переменная имеет тип `REG_SZ`, то тип значения не указывается.

Для удаления раздела, синтаксис несколько другой:

```
Windows Registry Editor Version 5.00
[-куст\путь к разделу]
```

Для удаления какого-либо параметра:

```
Windows Registry Editor Version 5.00
[куст\путь к разделу]
"Параметр"=-
```

**Листинг 5.1. Добавление ключей в ветвь реестра Windows 2k**

```
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\Identities]
"Migrated5"=dword:00000001
"Last Username"="Main Identity"
"Last User ID"="{DBE58550-FA28-4338-8C4C-26F838B95AFD}"
"Default User ID"="{DBE58550-FA28-4338-8C4C-26F838B95AFD}"
```

## Редактирование реестра с помощью WSH

Доступ к реестру из WSH-скриптов обеспечивается с помощью объекта `WshShell`: `RegRead`, `RegWrite` и `RegDelete`. Синтаксис доступа к реестру следующий:

```
WshShell.RegWrite "ключ реестра\ключ", "значение", "тип"
WshShell.RegRead("ключ реестра\ключ")
WshShell.RegDelete "ключ реестра\ключ"
```

Подразделы разделяются, как видно, из синтаксиса знаком `"\"`. Поскольку в первом аргументе команды `RegWrite` можно передать как имя раздела, так и название самого параметра, то признаком раздела считается `"\"`(обратный слэш) в конце пути ветви реестра, например:

```
WshShell.RegWrite "HKCU\Test\", "";
```

В приведенном примере в кусте HKCU создается пустой раздел Test. Если же команда такая:

```
WshShell.RegWrite "HKCU\Test\Message", "Hello", "REG_SZ",
```

то создастся параметр Message со значением Hello с типом данных REG_SZ. При этом команда:

```
WshShell.RegWrite "HKCU\Test\Message\", "Hello", "REG_SZ"
```

будет эквивалентна команде:

```
WshShell.RegWrite "HKCU\Test\Message\", "".
```

WSH-скрипт имеет расширение vbs и может использоваться как сценарий загрузки в Windows 2k (листинг 5.2).

**Листинг 5.2. Запись данных в реестр с помощью WSH**

```
Set WshShell=WScript.CreateObject("WScript.Shell")
WshShell.RegWrite "HKCU\Test\Message", "Hello", "REG_SZ"
WScript.Echo "Реестр изменен!"
```

WSH непосредственно не позволяет выполнить перечисление параметров и подразделов внутри раздела. Однако нужную функцию позволяют получить API-функции. Для этого используется динамическая библиотека Regobj.dll, предоставляющая сценарию доступ к API-функциям через объектную модель. Библиотека поставляется с Microsoft Visual Basic, также может быть скопирована с Web-узла Microsoft: в поле "поиск" следует указать Regobj.dll.

Доступ к библиотеке осуществляется с помощью команды:

```
Set RegObject=WScript.CreateObject("RegObj.Registry")
```

Прежде чем использовать библиотеку, ее надо зарегистрировать командой:

```
"regsvr32 /s regobj.dll"
```

После ее создания используют метод RegKeyFromString, чтобы получить объекты выбранного раздела. Полученные данные обрабатываются циклом For Each … In (листинг 5.3).

**Листинг 5.3. Чтение названий и значений подключей в указанной папке реестра**

```
Set RegObject=WScript.CreateObject("RegObj.Registry")
Set Rootkey=RegObject.RegKeyFromString(key1)
Temp=""
For Each oVal In RootKey.SubKeys
```

```
        Temp=Temp & oVal.Name & vbTab & oVal.Value & vbCrlf
Next
Wscript.Echo Temp
```

`SubKeys` содержит все разделы родительского раздела. Свойство `Name` включает в себя название раздела (`vbCrlf` эквивалентно нажатию клавиши <Enter>). Свойство `Values` возвращает набор, содержащий все параметры данного раздела. Переменная `vbTab` эквивалентна <Tab>.

Библиотека Regobj.dll не поддерживает новый тип параметров `REG_FULL_RESOURCE_DESCRIPTOR`, применяемый в Windows 2k. При попытке получить такой параметр возникнет ошибка периода выполнения.

С помощью библиотеки можно получить доступ к удаленному компьютеру, используя объект `RemoteRegistry` (листинг 5.4).

**Листинг 5.4. Доступ к реестру удаленного компьютера**

```
Set RegRemote=WScript.CreateObject("RegObj.Registry")
Set RegObject=RegRemote.RemoteRegistry(host)
Set Rootkey=RegObject.RegKeyFromString(key1)
```

Переменная `host` имеет вид:

```
\\имя_компьютера
```

## Редактирование реестра с помощью KIXTart

KIXTart имеет богатый набор инструментов для работы с реестром по сравнению с VBS и WSH (см. табл. 5.2).

Все функции используют следующий формат обращения к реестру:

```
[\\Remote_computer_name\][Key\]Subkey
```

`Remote_computer_name` — имя удаленного компьютера в UNC-формате. Если переменная не указана, то осуществляется доступ к локальному реестру.

KIXTart работает со всеми ветвями реестра: `HKLM`, `HKU`, `HKCR`, `HKCU`, `HKCC`. Если не указывается ключ, то по умолчанию считается, что это `HKCU`. KIXTart поддерживает как полное (`HKEY_LOCAL_MACHINE`), так и сокращенное имя ветви (`HKLM`). При доступе к удаленному реестру вы получаете доступ только к `HKLM`, `HKU`. Доступ к удаленному реестру разрешен только клиентам Windows 2k (табл. 5.2).

***Таблица 5.2.*** *Функции KIXTart для работы с реестром*

| Функция | Описание |
|---|---|
| ADDKEY | Добавление ключа в реестр |
| DELKEY | Удаление ключа реестра |
| DELTREE | Удаление ветви реестра |
| DELVALUE | Удаление значения из реестра |
| ENUMKEY | Показывает список подключей, содержащихся в ветви реестра |
| ENUMVALUE | Показывает список значений, содержащихся в подключе реестра |
| EXISTKEY | Проверяет, существует ли в реестре данный ключ |
| EXPANDENVIRONMENTVARS | Преобразует переменный параметр, содержащийся в строке, текст, например, %windir% в Windows |
| KEYEXIST | Аналогично EXISTKEY |
| LOADHIVE | Загружает куст из файла. Справедливо только для HKLM и HKU. Функционал аналогичен действию Regedt32.exe. Для использования команды пользователь должен входить в группу BACKUP and RESTORE OPERATORS |
| LOADKEY | Загружает ключи вместе с подветвями и значениями из REG-файла |
| READTYPE | Возвращает тип данных заданного значения в формате строки ASCII |
| READVALUE | Возвращает значение в формате строки ASCII |
| SAVEKEY | Сохраняет ветвь и подветви со значениями в текстовый файл. Для использования команды пользователь должен входить в группу BACKUP and RESTORE OPERATORS |
| UNLOADHIVE | Загружает куст из реестра |
| WRITEVALUE | Создает новый ключ, если он не существует, записывает значение параметра |

Сравнивая между собой сценарии на базе командной строки (файлы с расширением bat), WSH, KIXTart, можно сделать вывод: KIXTart представляет собой унифицированный язык, по своим возможностям перекрывающий и сценарии на основе командной строки, и WSH. Отдельные функции, которые задействованы в сценарии загрузки, рассмотренном в данной работе, просто невозможно реализовать с помощью WSH. Выбор KIXTart как средства раз-

работки сценария загрузки обусловлен еще и тем, что он поддерживает OLE-объекты, а значит, любая функция, написанная на WSH, VBScript, JScript, может быть адаптирована под KIX.

Приведем несколько типовых примеров использования возможностей KIX-Tart в работе с реестром Windows.

Получение списка всех подпапок в указанной папке (листинг 5.5). Сценарий в указанной папке (`HKEY_CURRENT_USER\Printers\Connections`) в цикле `Do…Until` получает список всех имеющихся в ней подпапок с помощью функции `EnumKey()`, затем выводит список в виде сообщения.

**Листинг 5.5. Чтение названий подключей в указанной папке реестра**

```
$string_key="HKEY_CURRENT_USER\Printers\Connections"
$Index = 0
$t=""

Do
        $KeyName = EnumKey($string_key, $Index)
        $Index = $Index + 1
        $t=$t+ $KeyName+Chr(13)
until len($KeyName)=0
MessageBox($t,"",0,0)
```

Другим примером является чтение всех параметров и их значений в подразделе (листинг 5.6). Методика определения данных аналогична.

**Листинг 5.6. Чтение всех параметров и их значений в указанном разделе реестра**

```
$string_key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"
$Index = 0

:Loop1
$KeyName = EnumValue($string_key, $Index)
If @ERROR = 0
$string_value=readvalue($string_key, $KeyName)
$t=$t+ "Name found: $KeyName ,  $string_value"
$Index = $Index + 1
goto Loop1
Endif
MessageBox($t,"",0,0)
```

# Групповые политики

Групповые политики (Group Policy, GP) представляют собой средство, обеспечивающее централизованное управление настройками рабочих станций, профилями пользователей. С их помощью определяют поведение операционной системы, рабочего стола, системы безопасности ОС, выключения компьютера, приложений и т. д. Групповые политики включают несколько компонентов (табл. 5.3).

***Таблица 5.3.*** *Компоненты групповых политик*

| Компонент | Описание |
|---|---|
| Administrative Templates | Политики, действие которых основано на изменении ветвей `HKLM`, `HCU` реестра |
| Security Settings | Настройка безопасности для следующих объектов: домена, компьютеров, пользователей |
| Software Installation | Управление централизованной установкой программ из MSI-архивов |
| Internet Explorer Malignance | Настройка браузера IE |
| Scripts | Настройка параметров при вкл/выкл компьютера |
| Folder Redirection | Управление перенаправлением файлов и папок в сети |

С помощью политик можно изменить только ветви реестра `HKLM` и `HKU`. Консоль управления групповыми политиками (см. рис. 5.2) вызывают с помощью команды `GPEDIT.MSC`. Групповые политики логически делятся на две части: `Computer Configuration` и `User Configuration`. В разделе `Computer Configuration` находятся политики, посредством которых изменяется ветвь реестра `HKLM`, в разделе `User Configuration` — соответственно, `HKU`. Информация о параметрах и конфигурации групповых политик сосредоточена в `HKLM\Software\Policies` (предпочтительное расположение) или `HKLM\Software\Microsoft\Windows\ CurrentVersion\Policies` для раздела `Computer Configration`; в `HKU\Software\Policies` (предпочтительное расположение) или `HKU\Software\ Microsoft\Windows\CurrentVersion\Policies` для раздела `User Configuration`.

## Административные шаблоны. Синтаксис

Административные шаблоны представляют собой текстовые файлы с расширением adm. По умолчанию ADM-файлы находятся в каталоге %WinDir%\INF.

Рассмотрим возможности языка, с помощью которого создаются политики безопасности.

Синтаксис языка включает следующие ключевые компоненты:

- комментарии;
- строки;
- `CLASS`;
- `CATEGORY`;
- `POLICY`.

## Комментарии

Комментарии полезно использовать для документирования содержимого создаваемого шаблона (листинг 5.7). Комментарии предваряют точкой с запятой (`;`) или двумя прямыми слэшами (`//`). Их также можно помещать в конце строки.

**Листинг 5.7. Комментарий в ADM-файлах**

```
; Это комментарий
// И это комментарий
CLASS USER // Определение класса USER,
       // отвечающего за пользовательские настройки
CLASS MACHINE // Определение класса Machine,
       // отвечающего за общекомпьютерные настройки
```

## Строки

Все возможные описания (описание политики, названия разделов, параметров и т. д.) рекомендуется располагать в специально предназначенном разделе `[strings]`. В тексте политики перед переменной, ссылающейся на текст в разделе `[strings]`, ставят два восклицательных знака "`!!`". Каждая строка в разделе `[strings]` имеет формат `name="строка"`. Приведем два равнозначных примера (листинги 5.8 и 5.9). В первом примере не будем использовать преимущества строк.

**Листинг 5.8. Явное присвоение значения переменной**

```
CLASS Machine
POLICY "Пример политики"
…
END POLICY
```

**Листинг 5.9. Неявное присвоение значения переменной. Рекомендуемый метод**

```
CLASS Machine
POLICY !!Pname
…
END POLICY
[Strings]
Pname="Пример политики"
```

Использование строк позволяет создавать шаблоны групповых политик, что снижает трудозатраты и ускоряет процесс их создания.

## CLASS

Первым элементом в файле, содержащем шаблон групповой политики, является ключевое слово `CLASS`, определяющее, к какому типу относится описываемая ниже политика: компьютерная (`Computer Configuration`) или пользовательская (`User Configuration`). В одном файле допускается многократное использование ключевого слова `CLASS`. Синтаксис элемента `CLASS`:

```
CLASS Name
```

Где `Name` — параметр, принимает одно из двух значений: `USER` или `MACHINE`. Значение `USER` определяет, что политика пользовательская. Изменения будут вноситься в реестр ветви `HKU`, настройку политики следует осуществлять в разделе `User Configuration\Administrative Templates`. Значение `MACHINE`, соответственно, определяет, что политика компьютерная. Изменения будут вноситься в реестр ветви `HKLM`, настройку политики следует осуществлять в разделе `Computer Configuration\Administrative Templates`.

## CATEGORY

После определения класса политики необходимо обозначить местоположение в подпапке `Administrative Templates`. Политика по ключевому слову `CATEGORY` создает подпапку. После описания категории ее необходимо закрыть с помощью `END CATEGORY`. Категории поддерживают вложенность. Синтаксис элемента `CATEGORY`:

```
CATEGORY Name
KEYMAME SubKey
…
END CATEGORY
```

Где `Name` — имя папки, которое будет отображаться в редакторе Group Policy. Используйте строковую переменную или строку, заключенную в кавычках.

`SubKey` — необязательный параметр. Информация о параметрах и конфигурации групповых политик сосредоточена в:

- `HKLM\Software\Policies` (предпочтительное расположение) или `HKLM\Software\Microsoft\Windows\CurrentVersion\Policies` для раздела `Computer Configuration`;
- `HKU\Software\Policies` (предпочтительное расположение) или `HKU\Software\Microsoft\Windows\CurrentVersion\Policies` для раздела `User Configuration`.

Не рекомендуется использовать в пути корневой ключ (`HKLM`, `HKU`), поскольку он уже описан ключевым словом `CLASS`. Если путь содержит пробелы, заключайте его в кавычки.

В разделе `CATEGORY` могут быть использованы следующие ключевые слова: `CATEGORY`, `END`, `KEYNAME`, `POLICY`.

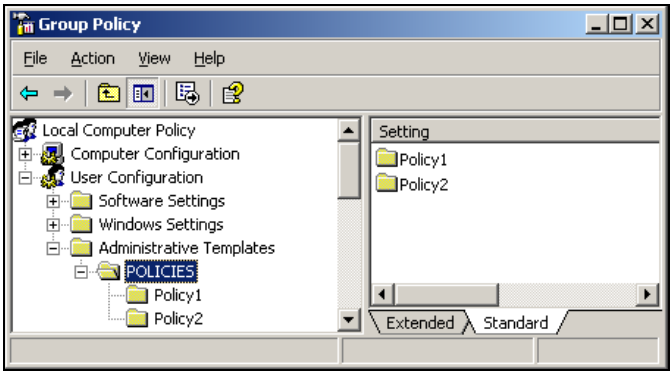



**Рис. 5.2.** Созданная групповая политика

**Листинг 5.10. Создание подпапок в групповой политике**

```
CLASS USER
CATEGORY "POLICIES"
        CATEGORY !!SubPol1
        KEYNAME "Software\Policies\SubPol1"
        …
        END CATEGORY

        CATEGORY !!SubPol2
```

```
        KEYNAME "Software\Policies\SubPol1"
        …
        END CATEGORY
END CATEGORY

[strings]
SubPol1="Policy1"
SubPol2="Policy2"
```

В приведенном примере (листинг 5.10) показано, как создать в указанной папке POLICIES две подпапки — Policy1 и Police 2 (рис. 5.2).

## POLICY

Ключевое слово `POLICY` используют для определения политики. В одной категории может быть включено несколько разделов `POLICY`, каждый из которых должен заканчиваться инструкцией `END POLICY`. Синтаксис элемента `POLICY`:

```
POLICY Name
[KEYNAME SubKey]
EXPLAIN Help
VALUENAME Value
[PARTS]
…
END POLICY
```

Где `Name` — название политики, отображаемое в редакторе Group Policy. Используйте строковую переменную или строку, заключенную в кавычки.

`SubKey` — необязательный параметр. Информация о параметрах и конфигурации групповых политик сосредоточена в:

- `HKLM\Software\Policies` (предпочтительное расположение) или `HKLM\Software\Microsoft\Windows\CurrentVersion\Policies` для раздела `Computer Configuration`;
- `HKU\Software\Policies` (предпочтительное расположение) или `HKU\Software\Microsoft\Windows\CurrentVersion\Policies` для раздела `User Configuration`.

Не используйте в пути корневой ключ (`HKLM`, `HKU`), поскольку он уже описан ключевым словом `CLASS`. Если путь содержит пробелы, заключайте его в кавычки. Ко всем политикам применяется последнее ключевое слово `KEYNAME`.

`Help` — значение этого параметра отображается в Group Policy в разделе `EXTENDED`. Для перевода каретки используйте `\n`.

`Value` — каждая политика содержит ключевое слово `VALUENAME`, которое связывает с ней значение реестра. По умолчанию редактор политик предполагает, что это значение типа `REG_DWORD` и имеет значение `1 (0x01)`, когда политика включена. Редактор политики удаляет значение, если политика выключена. В том случае, если необходимо сохранить это значение в реестре после удаления политики — используйте ключевые слова `VALUEON` и `VALUEOFF`, которые следуют непосредственно за словом `VALUENAME`:

```
VALUEON [NUMERIC] VValue
VALUEOFF [NUMERIC] VValue
```

По умолчанию тип данных значения `VValue` — `REG_SZ`. Если после ключевого слова указано `NUMERIC`, то тип данных значения `VValue` — `REG_DWORD`.

## Программирование интерфейса политик безопасности

Программирование интерфейса групповых политик сводится к созданию раскрывающихся списков, флажков, текстовых полей и т. д. Рассмотрим по порядку процедуры создания всех элементов интерфейса.

В первом варианте — самом простом, нет никаких управляющих элементов, кроме опции Вкл/Выкл политику. В файле политики записан ключ реестра и значение, имеющее тип `REG_SZ` или `REG_DWORD`, которое может меняться в зависимости от того, включена политика или нет. Приведем пример, в котором при включенной политике переключение раскладки клавиатуры в диалоговом окне регистрации пользователя в сети устанавливает <Ctrl>+<Shift>, при выключенной — <Alt>+<Shift>.

За переключение раскладки языка отвечает параметр `Hotkey`, принимающий значение 2 (<Ctrl>+<Shift>) или 1 (<Alt>+<Shift>). Он расположен в разделе `HKEY_USER\Keyboard Layout\Toggle`. Параметр `Hotkey` имеет тип данных `REG_SZ`. Приступим к созданию политики (листинг 5.11). Поскольку необходимо редактировать ветвь реестра `HKU`, то политика является пользовательской и будет создана в разделе `User Configuration\Administrative Templates`. Параметр `KEYNAME` раздела `CATEGORY` принимает значение `Keyboard Layout\Toggle`. В разделе `POLICY` параметр `VALUENAME` — `Hotkey`. `VALUEON` — `2`, а `VALUEOFF` — `1`.

**Листинг 5.11. ADM-файл политики**

```
CLASS USER
CATEGORY "Admin Policies"
CATEGORY "Keyboard Toggle "
KEYNAME ".Default\Keyboard Layout\Toggle"
              POLICY "Toggle Policy"
                     EXPLAIN !!help
VALUENAME "Hotkey"
VALUEON 2
VALUEOFF 1
END POLICY
END CATEGORY
END CATEGORY
[strings]
help="Управление переключением раскладки клавиатуры при регистрации пользователя в сети \n\n. При включенной политике переключение раскладки клавиатуры осуществляется с помощью комбинации клавиш CTRL+SHIFT, при выключенной – ALT+SHIFT."
```

Все остальные элементы интерфейса (флажки, выпадающие меню и т. д.) можно реализовать только в разделе `PART` — члене раздела `POLICY`. Синтаксис раздела `PART`:

```
PART Name Type
Keywords
[KEYNAME Subkey]
[DEFAULT Default]
VALUENAME Name
END PART
```

Где `Name` — название раздела, которое будет отображено в консоли во время редактирования политики.

`Type` — соответствует одному из типов данных (табл. 5.4).

***Таблица 5.4.*** *Доступные элементы управления в ADM-файлах*

| Тип | Описание | Тип данных |
|---|---|---|
| CHECKBOX | Отображает флажок. При установленном флажке принимает значение 1, при снятом — 0 | REG_DWORD |

***Таблица 5.4*** *(окончание)*

| Тип | Описание | Тип данных |
|---|---|---|
| COMBOBOX | Отображает список с полем ввода | REG_SZ (по умолчанию), REG_EXPAND_SZ (при наличии метки EXPANDABLETEXT) |
| DROPDOWNLIXT | Отображает раскрывающийся список с полем ввода. Пользователь может выбрать только одно из заданных значений | |
| EDITTEXT | Отображает текстовое поле, в котором можно вводить буквы и цифры | |
| LISTBOX | Отображает список с кнопками **Add** и **Remove** | – |
| NUMERIC | Отображает текстовое поле, позволяющее вводить только число | REG_DWORD |
| TEXT | Отображает текстовую строку (метку). Не сохраняет в реестре никаких значений. Отображается только в виде подсказок в диалоговом окне настройки групповой политики | |

Keywords — эта информация специфична для каждого из типов; дополнительная информация приведена ниже.

Subkey — необязательный подключ HKLM или HKU. Не используйте в пути корневой ключ (HKLM, HKU), поскольку он уже описан ключевым словом CLASS. Если путь содержит пробелы, заключайте его в кавычки.

Defaults — значение параметра по умолчанию. Когда администратор включает политику, то осуществляется считывание параметров по умолчанию.

Value — модифицируемое значение реестра; тип и данные значения полностью зависят от типа параметра.

## *CHECKBOX*

С помощью ключевого слова CHECKBOX в политике безопасности отображается флажок. По умолчанию флажок сброшен. Если требуется, чтобы по умолчанию флажок был установлен, необходимо в теле раздела PART указать ключевое слово DEFCHECKED. При установленном флажке в реестр записывается значение 1, если он сброшен — 0.

В реестре по умолчанию значение соответствует типу данных REG_SZ. Если необходимо записать данные в реестр в формате REG_DWORD, то необходимо после значений ключевых слов VALUEON и VALUEOFF добавить NUMERIC.

Синтаксис элемента CHECKBOX:

```
PART Name CHECKBOX
[DEFCHEKED]
```

```
VALUENAME Value
VALUEON [NUMERIC] Value1
VALUEOFF [NUMERIC] Value2
END PART
```

Где `Name` — название раздела, которое будет отображено в консоли во время редактирования политики. Если в названии раздела встречаются пробелы, заключите его в кавычки, в противном случае значение будет считано до первого пробела.

`Value` — название параметра, которое необходимо изменить.

`Value1` — значение, задаваемое при установленном флажке и включенной политике.

`Value2` — значение, задаваемое при снятом флажке и включенной политике.

## COMBOBOX

С помощью ключевого слова `COMBOBOX` в диалоговое окно политики добавляют список с текстовым полем. Внутри `COMBOBOX` включен обязательный блок `SUGGESTIONS`, заканчивающийся инструкцией `END SUGGESTIONS`. Внутри этого блока перечислены элементы, заключенные в кавычки и разделяющиеся пробелами.

```
PART Name COMBOBOX
SUGGESTIONS
"Suggestion1"  "Suggestion2" …  "Suggestionm"
END SUGGESTIONS
[DEFAULT Default]
[EXPANDABLETEXT]
[MAXLENGHT] Max
[NOSORT]
[REQUIRED]
VALUENAME Value
END PART
```

Где `DEFAULT` — указывает значение списка по умолчанию.

`EXPANDABLETEXT` — создает значение типа `REG_EXPAND_SZ`.

`MAXLENGHT` — указывает максимальную длину строки.

`NOSORT` — отключает сортировку списка редактором политик.

`REQUIRED` — указывает обязательное значение.

`Name` — название раздела, которое будет отображено в консоли во время редактирования политики.

SUGGESTIONS — список элементов, помещающихся в раскрывающийся список. Все элементы, содержащие пробелы, помещаются в кавычки. Элементы списка разделяются пробелами.

Default — значение по умолчанию. Считывается при включении политики.

Max — максимальная длина данных значения.

Value — модифицируемое значение реестра. По умолчанию имеет тип данных REG_SZ.

## *DROPDOWNLIST*

С помощью ключевого слова DROPDOWNLIST в политику добавляют диалоговое окно — раскрывающийся список. Внутри DROPDOWNLIST включен обязательный блок ITEMLIST, заканчивающийся инструкцией END ITEMLIST. Внутри него перечислены элементы раскрывающегося списка.

Синтаксис элемента DROPDOWNLIST:

```
PART Name DROPDOWNLIST
ITEMLIST
NAME Item VALUE Data
END ITEMLIST
[DEFAULT Default]
[EXPANDABLETEXT]
[NOSORT]
[REQUIRED]
VALUENAME Value
END PART
```

Где DEFAULT — указывает на значение раскрывающегося списка по умолчанию.

EXPANDABLETEXT — создает значение типа REG_EXPAND_SZ.

NOSORT — отключает сортировку списка редактором политик.

REQUIRED — указывает обязательное значение.

Name — название раздела, которое будет отображено в консоли во время редактирования политики.

Item — имя каждого из элементов раскрывающегося списка.

Data — значение, соответствующее отображаемому элементу Item.

Default — значение по умолчанию. Считывается при включении политики.

Value — модифицируемое значение реестра. По умолчанию имеет тип данных REG_SZ.

## EDITTEXT

Ключевое слово EDITTEXT позволяет вводить в диалоговом окне политики текст, состоящий из букв и цифр. Синтаксис элемента EDITTEXT:

```
PART Name EDITTEXT
[DEFAULT Default]
[EXPANDABLETEXT]
[MAXLENGHT] Max
[NOSORT]
[REQUIRED]
VALUENAME Value
END PART
```

Где DEFAULT — указывает значение списка по умолчанию.

EXPANDABLETEXT — создает значение типа REG_EXPAND_SZ.

MAXLENGHT — указывает максимальную длину строки.

REQUIRED — указывает обязательное значение.

Name — название раздела, которое будет отображено в консоли во время редактирования политики.

Default — значение по умолчанию; считывается при включении политики.

Max — максимальная длина данных значения.

Value — модифицируемое значение реестра; по умолчанию имеет тип данных REG_SZ.

## LISTBOX

Ключевое слово LISTBOX добавляет в диалоговое окно политик список с кнопками **Add** и **Remove**. Это единственный элемент, который может быть использован системным администратором для управления несколькими значениями, содержащимися в одном ключе. В разделе LISTBOX невозможно использовать опцию VALUENAME, поскольку он не связан с каким-либо конкретным значением.

Синтаксис элемента LISTBOX:

```
PART Name LISTBOX
[EXPANDABLETEXT]
[NOSORT]
[ADDITIVE]
[EXPLICITVALUE| VALUEPREFIX Prefix]
END PART
```

Где EXPANDABLETEXT — создает значение типа REG_EXPAND_SZ.

NOSORT — отключает сортировку списка редактором политик.

ADDITIVE — считывает значения, уже хранящиеся в реестре.

EXPLICITVALUE — ключевое слово позволяет указать имя и данные значения. Отображаемый список имеет два столбца: один для имени, другой для данных. Невозможно использовать эту инструкцию совместно с VALUEPREFIX.

VALUEPREFIX — указанный префикс определяет имена значений. Если указывается префикс, то редактор групповых политик добавляет к нему возрастающий номер. В том случае, если префикс пустой, то генерируются только возрастающие номера.

Name — название раздела, которое будет отображено в консоли во время редактирования политики.

Prefix — используется для генерации последовательности имен. Если указан префикс, например Sample, то генерируются следующие имена значений: Sample1, Sample2, …, Samplen. Если префикс не указан, то генерируются только порядковые номера: 1, 2, 3,…, n.

## NUMERIC

Ключевое слово NUMERIC позволяет вводить буквенно-цифровой текст, используя элемент управления "счетчик", который увеличивает или уменьшает число. По умолчанию переменные имеют тип данных REG_DWORD, однако, используя инструкцию TXTCONVERT, данные могут быть сохранены в формате REG_SZ.

Синтаксис элемента NUMERIC:

```
PART Name NUMERIC
[DEFAULT Default]
[MAX Max]
[MIN Min]
[REQUIRED]
[SPIN]
[TXTCONVERT]
VALUENAME Value
END PART
```

Где DEFAULT — указывает на значение списка по умолчанию.

REQUIRED — указывает обязательное значение.

SPIN — указывается шаг арифметической прогрессии; по умолчанию равен 1. Указанное значение 0 делает счетчик невидимым.

TXTCONVERT — по умолчанию значения имеют тип данных REG_DWORD. Используйте эту инструкция, чтобы сохранить данные в формате REG_SZ.

Name — название раздела, которое будет отображено в консоли во время редактирования политики.

Default — значение по умолчанию; считывается при включении политики.

Max — максимальное значение; по умолчанию равно 9999.

Min — минимальное значение; по умолчанию равно 0.

Value — модифицируемое значение реестра; по умолчанию имеет тип данных REG_DWORD.

### *TEXT*

Ключевое слово TEXT добавляет в диалоговое окно статический текст. Синтаксис элемента TEXT:

```
PART Text TEXT
END PART
```

Где Text — статический текст, добавляемый в диалоговое окно; если текст имеет пробелы, то его следует заключать в кавычки.

## Практика использования административных шаблонов

В корпоративной сети, как правило, существует один или несколько файловых серверов, в которых хранятся дистрибутивы программного обеспечения, в том числе Microsoft Office. Если пользователю по какой-либо причине необходимо перейти с одной рабочей станции на другую, то при первой загрузке для его учетной записи создается новый профиль. При первом запуске одного из компонентов MS Office, программа обращается к дистрибутиву, который расположен на файловом сервере. При изменении местоположения дистрибутива на сервере, пользователю выдается сообщение о невозможности завершить процесс конфигурации программы и просьба обратиться к системному администратору. Во избежание этой неприятной ситуации рекомендуется использовать групповую политику, которая меняет в реестре пути к дистрибутиву MS Office (листинг 5.12, рис. 5.3).

**Листинг 5.12. Политика изменения пути к дистрибутиву Microsoft Office**

```
LASS Machine
CATEGORY "Office 2000"
KEYNAME "Software\Policies"
POLICY "Office2000"
EXPLAIN !!help

       PART "Parametr1" EDITTEXT
       KEYNAME
"SOFTWARE\Classes\Installer\Products\914000001E872D116BF00006799C897E\
SourceList"
       VALUENAME LastUsedSource
       MAXLEN 100
       EXPANDABLETEXT
       END PART

PART !!help1 TEXT
END PART

       PART "Parametr2" EDITTEXT
       KEYNAME
"SOFTWARE\Classes\Installer\Products\914000001E872D116BF00006799C897E\
SourceList\Net"
       VALUENAME 1
       MAXLEN 100
       EXPANDABLETEXT
       END PART

PART !!help2 TEXT
END PART
END POLICY
END CATEGORY

[strings]
help1="Exapmle1: n;3;\\Server\software\Office2000\"
help2="Exapmle2: \\Server\software\Office2000\"
help="\n\nCorrecting path to Microsoft Office 2000"
```

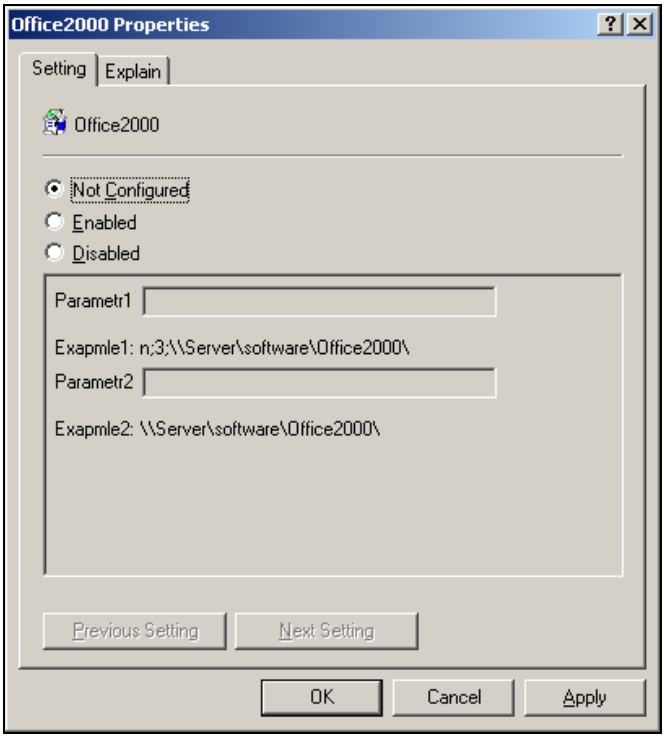



**Рис. 5.3.** Окно конфигурации политики

## Внедрение административных шаблонов

Для загрузки ADM-файла, содержащего политику, необходимо выполнить следующие действия:

1. Загрузить консоль групповых политик, выполнив команду `GPEDIT.MSC`.
2. В любом из разделов (`Computer Configuration\Administrative Templates` или `User Configuration\Administrative Templates`) вызвать контекстное меню, используя однократное нажатие на правую кнопку мыши.
3. В меню выбрать **Add/Remove Templates**.
4. В диалоговом окне нажать кнопку **Add**. В появившемся окне, указав путь к файлу с расширением adm, нажать кнопку **Open**.
5. Нажать кнопку **Close**, после этого программа проверит синтаксис файла; в случае синтаксической ошибки будет указана строка и предпочтительный вариант исправления этой ошибки.

## Удаление административного шаблона

Для удаления политики, подгруженной с помощью ADM-файла, необходимо выполнить действия в следующей последовательности:

1. Загрузить консоль групповых политик, выполнив команду `GPEDIT.MSC`.
2. В любом из разделов (`Computer Configuration\Administrative Templates` или `User Configuration\Administrative Templates`) вызвать контекстное меню, используя однократное нажатие на правую кнопку мыши.
3. В появившемся меню выбрать **Add/Remove Templates**.
4. В загрузившемся диалоговом окне выбрать файл описания политики, которую необходимо удалить. Нажать кнопку **Remove**, затем **Close**.



# Глава 6

# Управление файловой системой

## Команды ввода/вывода (*FSO*)

Для работы с файловой системой компьютера с помощью WSH разработан объект `FileSystemObject`, поддерживающий набор методов, позволяющий выполнять различные манипуляции с файловой системой (табл. 6.1).

***Таблица 6.1.*** *Объекты FSO*

| Объект или набор | Описание |
|---|---|
| `Drives` | Набор всех логических, физических и съемных дисков |
| `Drive` | Объект, методы и свойства которого позволяют обращаться к диску, получать о нем различную информацию |
| `Folders` | Набор всех вложенных папок в данную папку |
| `Folder` | Объект, методы которого используются для создания, перемещения, переименования и удаления папок, а свойства — для получения имен папок, путей и т. д. |
| `Files` | Набор всех файлов в папке |
| `File` | Объект, методы которого используются для создания, перемещения, переименования и удаления файлов, а свойства — для получения имен файлов и путей |
| `FileSystemObject` | Главный объект объектной модели. Поддерживает все методы и свойства для доступа к файловой системе |
| `TextStream`[1] | Методы этого объекта позволяют осуществлять различные манипуляции с текстовыми файлами |

[1] Объект `FileSystemObject` не поддерживает методов для доступа к двоичным файлам. Коллекция описана в библиотеке Scrrun.dll.

Получение доступа к файловой системе обеспечивается с помощью объекта `FileSystemObject` (`FSO`). Этот объект не является частью WSH, поэтому ссылку на него позволяют получить встроенные методы VBScript:

```
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
```

После создания объекта становятся доступны его методы, свойства и дочерние объекты.

## Работа с дисками. Набор *Drive*

Объект `FSO` позволяет обращаться к любому диску компьютера: локальным и сетевым; к CD-ROM, дисководам, жестким дискам. Для получения списков всех доступных дисков необходимо воспользоваться свойством `Drives`, которое возвращает набор объектов `Drive`. Он представляет собой набор массивов, элементы которого содержат информацию о подключенных к рабочей станции дисках (табл. 6.2).

***Таблица 6.2.*** *Набор объектов* `Drive`

| Свойство | Описание |
|---|---|
| `.DriveLetter` | Буква, на которую смонтировано устройство |
| `.VolumeName` | Метка тома |
| `.ShareName` | Для сетевых дисков позволяет определить UNC подключенного ресурса |
| `.FreeSpace` | Возвращает в байтах размер свободного места на логическом диске |
| `.DriveType` | Тип диска. Возвращает число (см. табл. 6.3) |
| `.IsReady` | Проверяет наличие носителя. Принимает значения `True` и `False` |
| `.TotalSize` | Возвращает в байтах размер всего логического диска |
| `.AvalibleSpace` | Возвращает в байтах размер свободного места на логическом диске. В случае назначения квот на диске, выдает свободное место, доступное текущему пользователю |

Тип диска закодирован целочисленным значением от 0 до n (табл. 6.3).

***Таблица 6.3.*** *Коды типов дисков*

| Константа | Значение | Описание |
|---|---|---|
| Unknown | 0 | Неизвестный тип диска |
| Removable | 1 | Устройство со сменным носителем: Floppy, ZIP, USB Flash |
| Fixed | 2 | Логический раздел жесткого диска |
| Remote | 3 | Сетевой диск |
| CD-ROM | 4 | CD-привод. CD-ROM, CD-RW, DVD-RW не различаются |
| RAMDisk | 5 | Виртуальный диск |

**Листинг 6.1. Статистика по подключенным дискам**

```
Dim i
Temp=""
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
Set oDrive=fso.Drives
For each i In oDrive
Temp=Temp+"Drive "+i.DriveLetter+": "
oType=i.DriveType

select case oType
case 0 'Unknown
Temp=Temp+"Unkbown Device"+chr(13)

case 1 'Removable
Temp=Temp+" Removable Device; "
       if i.IsReady then
              Temp=Temp+sizer(i)
       else
              Temp=Temp+"Drive Not Ready"+chr(13)

       end if

case 2 'Fixed
       Temp=Temp+" Hard Drive; "
       Temp=Temp+" Volume Name: "+i.VolumeName+"; "
```

```
        Temp=Temp+sizer(i)

case 3 'Remote
        Temp=Temp+" Network Disk; "
        Temp=Temp+" ShareName: "+i.ShareName+"; "
        Temp=Temp+sizer(i)

case 4 'CD-ROM
        Temp=Temp+" CD Drive; "
        if i.IsReady then
                Temp=Temp+" Volume Name: "+i.VolumeName+"; "
                Temp=Temp+sizer(i)
        else
                Temp=Temp+"Drive Not Ready"+chr(13)
        end if

case 5 'RAMDisk
        Temp=Temp+" RamDrive; "
        Temp=Temp+sizer(i)
end select
Next
MsgBox Temp

function sizer(i)
sizer=formatnumber(cstr(i.FreeSpace/(1024*1024)),2)+" /
"+formatnumber(cstr(i.TotalSize/(1024*1024)),2)+" Mb"    +chr(13)
end function
```

В примере из листинга 6.1 используется функция `FormatNumber()`, позволяющая округлить выводимое значение.

## Работа с папками и файлами. Наборы *Folders* и *Files*

С помощью набора `Folders` осуществляется формирование списка вложенных файлов и папок в указанном каталоге; просмотр атрибутов каталога; создание, перемещение, переименование и удаление папок.

Доступ к набору `Folders` можно получить, воспользовавшись следующим шаблоном — листинг 6.2.

**Листинг 6.2. Получение доступа к указанной папке**

```
path="…"
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
Set oFolders=fso.GetFolder(path)
```

Где `path` — путь к каталогу, например `C:\Folder` или `\\Server\ShareName`.

В качестве обработчика ошибок рекомендуется использовать функцию `FolderExists()`, с помощью которой можно определить, существует ли каталог, с которым в дальнейшем будет работать сценарий — листинг 6.3.

**Листинг 6.3. Использование встроенного в FSO обработчика ошибок**

```
path="…"
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
If fso.FolderExists(path) then
Set oFolders=fso.GetFolder(path)
…
Else
MsgBox "Folder " + path + " Not Exists"
End If
```

## Формирование списка вложенных объектов в папке

Формирование списка вложенных объектов в папке осуществляется с помощью свойства `Name` набора `SubFolders` и `Files`. С помощью набора `SubFolders` определяются вложенные папки в указанный каталог, с помощью `Files`, соответственно, файлы.

### Определение списка подпапок в указанной директории

Итак, определение списка вложенных папок осуществляется с помощью свойства `Name` (листинг 6.4).

**Листинг 6.4. Определение вложенных каталогов в указанной папке**

```
path="C:\Windows"
Temp=""
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
If fso.FolderExists(path) then
Set oFolders=fso.GetFolder(path)
```

```
Set OSubFolders=oFolders.Subfolders
       For each i In OSubFolders
       Temp=Temp+i.Name+chr(13)
Next
Else
Temp= "Folder " + path + " Not Exists"
End If
MsgBox Temp
```

На первом этапе необходимо получить доступ к папке Windows, затем к ее дочерним папкам, и только после этого становятся доступны свойства набора.

### Определение списка всех вложенных подпапок в каталоге

Для чтения всей структуры подкаталогов, описанный ранее механизм размещают в рекурсивной[2] функции, имеющей два параметра — путь к абсолютному подкаталогу (`path`) и уровень вложенности подкаталогов (`idx`) (листинг 6.6).

Данные, содержащие абсолютные пути к подкаталогам, разумно записывать в динамический одномерный массив — листинг 6.5.

**Листинг 6.5. Получение списка доступных подкаталогов в указанном родительском каталоге**

```
Path="C:\RootFolder"
i=0
Dim Array()  'Объявление динамического массива
Set fso=Wscript.CreateObject("Scripting.FileSystemObject")
Set oFolder=fso.GetFolder(path)
Set oFolders=oFolder.SubFolders
Redim Preserve Array(oFolders.count)       ' Изменение размера
                                           ' динамического массива
               For Each of In oFolders
                     Array(i)=cstr(oF.Path)  ' Запись элементов в массив
                     i=i+1  Next
'Чтение элементов массива
Temp=""
```

[2] Рекурсивная функция — это функция, вызывающая саму себя, при этом она обязательно передает параметры. Как правило, она используются для чтения иерархической структуры.

```
For i=Lbound(Array) to Ubound(Array)
Temp=Temp+cstr(Array(i))+chr(13)+chr(10)
Next
MsgBox Temp
```

**Листинг 6.6. Шаблон рекурсивной функции.**

```
…
RecFolder index, path        ' Вызов функции RecFolder в теле сценария
…
Function Recfolder (idx, path)
…
Call Recfolder (idx+1, path)
…
End Function
```

Исходя из ранее изложенного, можно создать скрипт, который определяет названия всех подкаталогов, вложенных в указанную директорию — листинг 6.7.

**Листинг 6.7. Определение всех вложенных подкаталогов в указанном каталоге**

```
dim path_array()
path="c:\windows"
i=0
temp=""
detect(q)
MsgBox "oFolders_count: "+cstr(detect(q))
For j=0 to cstr(ubound(path_array)-1)
path=cstr(path_array(j))
temp=temp+cstr(j)+": "+path+chr(13)
Next
MsgBox temp

Function detect(c)
i=0
Set fso=Wscript.CreateObject("Scripting.FileSystemObject")
Set oFolder=fso.GetFolder(path)
Set oFolders=oFolder.SubFolders
```

```
Redim Preserve path_array(oFolders.count)

detect=oFolders.count
       For Each oF In oFolders
       path_array(i)=path+"\"+cstr(oF.Name)
       i=i+1
       Next
End Function
```

### Определение списка файлов, находящихся в директории

Методика определения списка файлов, содержащихся в директории, та же, что и в случае определения списка подкаталогов. Разница заключается в том, что вызывается набор `Files`, а не `SubFolder` — листинг 6.8.

**Листинг 6.8. Определение списка файлов в корневой директории и подпапках**

```
path="c:\windows"
Set WSHShell = WScript.CreateObject("WScript.Shell")
Set fso=createobject("Scripting.filesystemobject")
Set oFolder=fso.GetFolder(path)
Set oFiles=oFolder.files
MsgBox "oFiles_count: "+cstr(oFiles.count)

temp=""
For each oFile in oFiles
temp=temp & oFile.name & chr(13)
Next
              MsgBox text
```

## Чтение характеристик файлов

Каждый файл имеет набор параметров, которые его характеризуют: название, расширение, размер, даты создания и последнего изменения файла, его атрибуты. О том, как считывать названия файлов и их расширения, было рассказано ранее. Пришло время рассказать о методике чтения дат и атрибутов файлов.

Начнем с определения атрибутов файла. Получив доступ к файлу с помощью объекта `FSO` (листинг 6.9), используя свойство `Attributs`, возвращающее двоичное значение, определяют его атрибуты (табл. 6.4).

***Таблица 6.4.*** *Атрибуты файлов*

| Константа | Значение | Описание |
|---|---|---|
| `Normal` | 0 | Файл без установленных атрибутов |
| `ReadOnly` | 1 | Файл с атрибутом "только для чтения" |
| `Hidden` | 2 | Скрытый файл |
| `System` | 4 | Системный файл |
| `Directory` | 16 | Папка |
| `Archive` | 32 | Файл с атрибутом "архивный", модифицированный со времени последнего резервного копирования |
| `Alias` | 1024 | Ярлык (файл с расширением LNK) |
| `Compressed` | 2048 | Сжатый файл (только для Windows 2k) |

**Листинг 6.9. Чтение атрибута файла boot.ini**

```
path="c:\boot.ini"
Set WSHShell = WScript.CreateObject("WScript.Shell")
Set fso=createobject("Scripting.filesystemobject")
Set oFile=fso.GetFile(path)
MsgBox oFile.Name&": "& oFile.Attributes
```

Если атрибуты этого файла никто не изменял, то значение свойства `oFile.Attributes` равно 39 (см. рис. 6.1), поэтому (см. табл. 6.4) файл boot.ini имеет следующие атрибуты: "только для чтения", "скрытый", "системный", "архивный".

Чтобы пользователь каждый раз не занимался определением атрибутов файлов, рекомендуется использовать функцию сопоставления кода атрибутам, осуществлять проверку на существование файла с помощью `FileExists()` — листинг 6.10.

**Листинг 6.10. Чтение атрибутов файлов. Расшифровка значений**

```
path="c:\boot.ini"
Set WSHShell = WScript.CreateObject("WScript.Shell")
Set fso=createobject("Scripting.filesystemobject")
if fso.FileExists(path) then

Set oFile=fso.GetFile(path)
```

```
Attr_value=oFile.Attributes
temp=A_detect()
MsgBox oFile.Name&": "& temp
else
MggBox "Файл " &path& " не существует"
end if

function A_detect()
t=""
if (Attr_value and &H01)> 0 Then t=t+"Только для чтения; "
if (Attr_value and &H02)> 0 Then t=t+"Скрытый; "
if (Attr_value and &H04)> 0 Then t=t+"Системный; "
if (Attr_value and &H20)> 0 Then t=t+"Архивный; "
if (Attr_value and &H800)> 0 Then t=t+"Сжатый; "
A_detect=t
end function
```

## Изменение атрибутов файлов

Изменение атрибутов файлов в операционной системе с помощью `FSO` осуществляется присвоением нового значения переменной `oFile.Name`, которое может быть представлено в шестнадцатеричном или десятеричном виде — листинг 6.11.

**Листинг 6.11. Изменение атрибутов файла**

```
path="c:\readme.txt"
Set WSHShell = WScript.CreateObject("WScript.Shell")
Set fso=createobject("Scripting.filesystemobject")
Set oFile=fso.GetFile(path)
MsgBox oFile.Name&": "& oFile.Attributes
oFile.Attributes=39
MsgBox oFile.Name&": "& oFile.Attributes
```

По умолчанию, вновь созданный текстовый файл readme.txt имеет атрибут "архивный" (32). Присвоив параметру `oFile.Attributes` значение 39, файлу назначаются следующие атрибуты: "только чтение", "скрытый", "системный", "архивный".

## Операции над файлами и папками

Над файлами и папками можно выполнять следующие операции: создание, копирование, перемещение, удаление.

### Создание и удаление папок

Создание и удаление каталогов реализуют с помощью метода `CreateFolder(path)` и `DeleteFolder(path)` объекта `FSO`. Параметром каждого из этих методов является путь к каталогу — `path`. Применяя эти методы, рекомендуется использовать обработчик ошибок, который проверяет наличие или отсутствие папки перед выполнением операции. Приведем два примера: первым (листинг 6.12) из них проиллюстрируем процесс создания новой папки C:\TempFolder, используя обработчик ошибок; во втором (листинг 6.13) — процесс удаления папки.

**Листинг 6.12. Создание папки**

```
path="C:\TempFolder"
Temp=""
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
If fso.FolderExists(path)=0 then
fso.CreateFolder(path)
Temp= "Folder " + path + " Created "
Else
Temp= "Folder " + path + " Already Exists"
End If
MsgBox Temp
```

**Листинг 6.13. Удаление папки**

```
path="C:\TempFolder"
Temp=""
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
If fso.FolderExists(path)<>0 then
fso.DeleteFolder(path)
Temp= "Folder " + path + " Deleted"
Else
Temp= "Folder " + path + " is Absent "
End If
MsgBox Temp
```

### Переименование, копирование и перемещение папок

Копирование папок осуществляется с помощью метода:

```
CopyFolder source, destination, overwrite
```

Перемещение и переименование — методом:

```
MoveFolder source, destination, overwrite.
```

Каждый их этих методов имеет три параметра:

- `Source` — обязательный параметр, в котором передается строка с именем исходной папки (путем); путь может включать знаки подстановки, такие как "`?`" или "`*`". Их рекомендуется использовать для копирования или перемещения нескольких папок одновременно, удовлетворяющих заданному шаблону;
- `Destination` — обязательный параметр, который задает папку назначения в виде строки. Значение параметра не может содержать символы подстановки. В том случае, если путь заканчивается "`\`" и папка-приемник не существует, то она будет создана;
- `Overwrite` — необязательный параметр. Если он равен `True` или 1 (по умолчанию), то целевая папка будет перезаписана.

Обратите внимание, что параметры указываются не в скобках (листинг 6.14).

**Листинг 6.14. Копирование/удаление папки**

```
' Копирование папки
OldPath="C:\TempFolderOld"
NewPath="C:\TempFolderNew"
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
fso.CopyFolder OldPath, NewPath, 1
MsgBox "Папка " + OldPath + " скопирована в " + NewPath

' Удаление папки
OldPath="C:\TempFolderOld"
NewPath="C:\TempFolderNew"
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
       fso.MoveFolder OldPath, NewPath, 1
MsgBox "Папка " + OldPath + " перемещена в " + NewPath
```

### Копирование, перемещение файла

Копирование и перемещение файлов осуществляются с помощью методов `Copy` и `Move`, соответственно. Поскольку методика использования этих методов одна и та же — рассмотрим их вместе (листинг 6.15).

**Листинг 6.15. Копирование/перемещение файла**

```
PathOld="C:\Folder\FileNameOld"
PathNew="C:\Folder\FileNameNew"

Set fso=WScript.CreateObject("Scripting.FileSystemObject")
If (fso.FileExistes(PathOld)) Then
       Set oFile=fso.GetFile(PathOld)
       oFile.**** PathNew
End If
```

Вместо "****" указывается имя метода. При вызове метода `Move` после копирования файла осуществляется удаление первоисточника — это единственное отличие этих методов.

### Удаление файла

Удаление файлов осуществляется с помощью метода `Delete`. Различается лишь синтаксис метода: метод `Delete`, в отличие от `Copy` или `Move`, не содержит параметров (листинг 6.16).

**Листинг 6.16. Удаление файла**

```
PathDel="C:\Folder\1.txt"
Set fso=WScript.CreateObject("Scripting.FileSystemObject")
If (fso.FileExistes(PathDel)) Then
       Set oFile=fso.GetFile(PathDel)
       oFile.Delete
End If
```

## Управление текстовыми файлами

С текстовым файлом можно выполнять следующие операции: чтение, запись данных в новый файл, добавление текста в существующий файл, замена текста в файле. Метод `OpenTextFile`, поддерживаемый объектом `FileSystemObject` (`FSO`), служит для получения доступа к существующему файлу, созданию новых файлов и осуществления в них записи данных.

```
fso.OpenTextFile(filename [,mode [, create [, format]]])
```

Где `Filename` — имя существующего тестового файла.

`Mode` — необязательный параметр, задает режим ввода/вывода для операций с файлом. Параметру присваивается одно из значений:

- 1 (`ForReading`) — с уровнем доступа только для чтения;
- 2 (`ForWriting`) — доступ только для записи;
- 8 (`ForAppending`) — для добавления данных в файл.

`Create` — необязательный параметр, содержащий булево значение (`True` — если файл с указанным именем не существует, то он будет создан, или `False` — файл не будет создан в любом случае).

`Format` — необязательный параметр, определяющий кодировку открываемого файла. Если параметр не задан, то предполагается, что файл формата ADCII. Принимает одно из трех значений:

- 0 (`TristateFalse`) — файл будет открыт в формате ASCII;
- –1 (`TristateTrue`) — файл будет открыт в формате Unicode;
- –2 (`TristateUseDefault`) — файл будет открыт с параметрами, заданными по умолчанию.

### Чтение данных

Чтение данных осуществляется с помощью функции `ReadLine()`, которая применяется к объекту "файл". Существует два варианта использования этой функции. Первый вариант — чтение содержимого всего файла, второй — чтение конкретной строки. Рассмотрим оба варианта. Для чтения всего содержимого файла необходимо использовать метод `atEndOfStream`, который осуществляет проверку конца файла (листинг 6.17).

**Листинг 6.17. Чтение текстового файла**

```
Path="c:\1.txt"
Temp=""
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(Path) Then
       Set Fline = fso.OpenTextFile(FileName)
              Do While Not (Fline.atEndOfStream)
                     Temp = Temp + Fline.ReadLine
              Loop
Else
Temp="File Not Found"
End If
MsgBox Temp
```

Содержимое файла накапливается в переменной `Temp`, затем выводится на экран пользователя.

Чтение конкретной строки осуществляется также с помощью функции `ReadLine()`. В качестве аргумента функции указывается номер строки, которую необходимо прочитать из файла. Нумерация строк начинается с нуля (листинг 6.18).

**Листинг 6.18. Чтение N (N=4) строки текстового файла**

```
FileName ="c:\1.txt"
N=3
Temp=""
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(FileName) Then
       Set Fline = fso.OpenTextFile(FileName)
                     Temp= Fline.ReadLine(N)
Else
Temp="File Not Found"
End If
MsgBox Temp
```

### Создание нового файла и запись данных

Запись данных в файл осуществляется с помощью метода `OpenTextFile`, при этом параметр `Mode` принимает значение 2 (`ForWriting`) в режиме; и функции `WriteLine()`, аргументом которой является переменная, содержащая текст (листинг 6.19).

**Листинг 6.19. Процедура записи данных в файл**

```
FileName ="c:\1.txt"
Array_Data[]
……
Set fso = CreateObject("Scripting.FileSystemObject")
       Set Fline = fso.OpenTextFile(FileName, 2, True)

For i=0 to Ubound(Array_Data)
WriteLine Array_Data( i)
Next
```

Данные, содержащиеся в массиве `Array_Data[]`, записываются в файл. Во время каждого обращения для записи метод `WriteLine` автоматически вставляет новую строку.

#### Добавление данных к существующему файлу

Для добавления данных в существующий файл методом `OpenTextFile` параметру `Mode` присваивают значение 8 (`ForAppending`), третьему параметру (`Create`) — `False`. Алгоритм работы сценария по добавлению строк аналогичен алгоритму, в соответствии с которым осуществляется запись данных в новый файл. Меняется только режим — значение параметра `Mode`.

# Управление правами доступа на файлы и папки

Создавая новую папку или файл на жестком диске с файловой системой NTFS, новому объекту присваиваются права доступа по определенным правилам. Если разговор идет о создании общих папок на сервере, то системного администратора, как правило, не устроят права доступа, назначаемые по умолчанию: они должны быть изменены в соответствии с определенными правилами, описываемыми шаблоном. О том, как программно изменить права доступа на папки и файлы, и пойдет разговор в следующем разделе.

## Библиотека ADsSequrity.dll

Программное управление правами доступа к файлам и папкам можно реализовать с помощью библиотеки ADsSequrity.dll, входящей в комплект поставки ADSI Resource Kit (**http://www.microsoft.com/ntserver/nts/downloads/other/ADSI25/default.asp**).

Для использования библиотеки ее необходимо зарегистрировать на компьютере, где будет запускаться сценарий. Команда регистрации библиотеки выглядит следующим образом:

```
regsvr32.exe /s ADsSequrity.dll
```

## Организация доступа к библиотеке

Для обеспечения доступа к объектам библиотеки в VBScript используют функцию `CreateObject("")`, создают два объекта: `ADsSecurity` и `AccessControlEntry`.

`ADsSecurity` отвечает за предоставление доступа к объекту, которым необходимо управлять: папка или файл. Путь к объекту указывается с помощью функции `GetSecurityDescriptor()`.

С помощью `AccessControlEntry` обеспечивается управление безопасностью объектами. Доступ к объектам библиотеки ADsSequrity.dll в VBScript выглядит следующим образом — листинг 6.20.

**Листинг 6.20. Шаблон доступа к NTFS (VBScript)**

```
Set sec = CreateObject("ADsSecurity")
Set sd = sec.GetSecurityDescriptor("FILE://c:\Folder")
Set dacl = sd.DiscretionaryAcl
Set ace = CreateObject("AccessControlEntry")
…
sd.DiscretionaryAcl = dacl
sec.SetSecurityDescriptor sd
set dacl=nothing
set sec=nothing
```

Для VB.NET все выглядит по-другому. Сначала необходимо подключить эти библиотеки к проекту. Для этого нужно выбрать **Active DS Type Library** и **ADSSequrity 2.5 Type Library**, используя свойство проекта **Add Reference** (рис. 6.1).

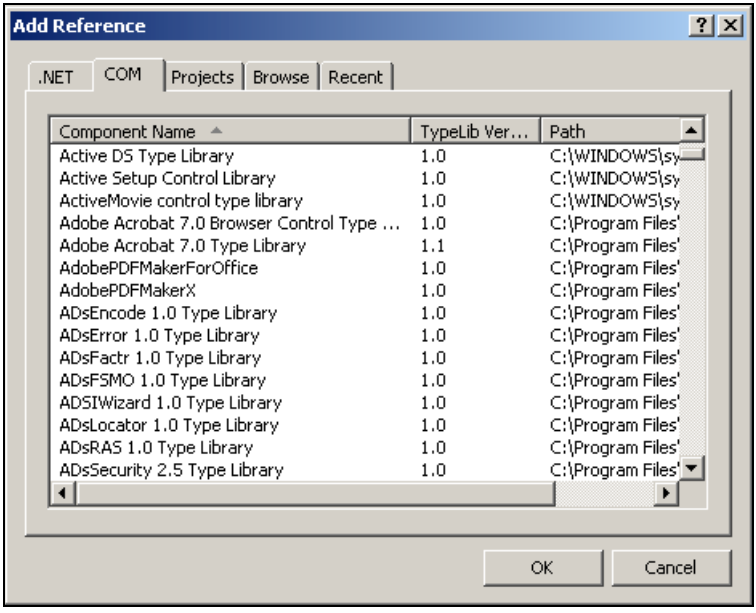



**Рис. 6.1.** Подключение COM-библиотеки в Visual Studio

После подключения библиотек их пространства имен необходимо импортировать в проект:

```
Imports ADSSECURITYLib
Imports ActiveDs
```

Библиотеке **ADSSequrity 2.5 Type Library** (файл ADSSequrity.dll) соответствует пространство имен `SDSSECURITYLib`, а **Active DS Type Library** (файл ActiveDS.dll) — `ActiveDS`. Доступ к объектам можно получить с помощью следующего шаблона — листинг 6.21.

**Листинг 6.21. Шаблон доступа к NTFS (VB.NET)**

```
vpath = "c:\path_to"
username = "domain\username"
Dim Sec As ADsSecurity
Dim SD As SecurityDescriptor
Dim DAcl As AccessControlList
Dim Ace As AccessControlEntry
Sec = New ADsSecurity
SD = CType(Sec.GetSecurityDescriptor("FILE://" & vpath),
SecurityDescriptor)
DAcl = CType(SD.DiscretionaryAcl, AccessControlList)
Ace = New AccessControlEntry
…
SD.DiscretionaryAcl = DAcl
Sec.SetSecurityDescriptor(SD)
```

Значением переменной `vpath` является путь к файлу или каталогу, безопасностью которого требуется управлять, а `username` — имя объекта (пользователь или группа), права которого необходимо изменить.

## Управление правами на папку

Для управления правами необходимо создать новый экземпляр объекта `ADsSecurity` и указать к нему путь, который является параметром функции `GetSecurityDescriptor()`. Таким образом, можно получить доступ к коллекции `AccessControlEntry` (рис. 6.2), членами которой являются свойства `Trustee`, `AccessMask`, `AceType`, `AceCount` и не фигурирующее в данном шаблоне `RemoveAcl`.

VB.NET снабжен всплывающими подсказками, показывающими как коллекцию доступных для выбранного объекта свойств, так и тип данных, соответствующий выбранному свойству (табл. 6.5—6.8).

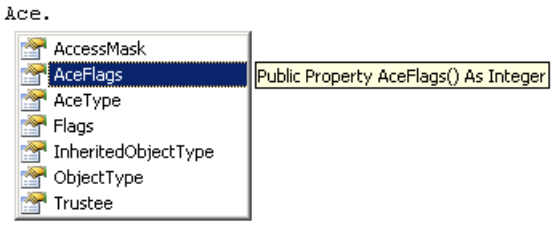



**Рис. 6.2.** Доступные свойства коллекции `AccessControlEntry`

***Таблица 6.5.*** *Члены коллекции* `AccessControlEntry`

| Параметр | Описание | Параметр | Описание |
|---|---|---|---|
| `.Trustee` | Объект (пользователь или группа), которому определяется уровень доступа | `AceType` | Определяет тип доступа (`Allow`, `Deny`, `Audit`) |
| `.AccessMask` | Определяет уровень доступа к ресурсу (`Full Control`, `Modify`, `Read`, `Execute` и т. д.) | `AceCount` | Количество объектов, которым назначены права |
| `.RemoveAcl` | Удаление существующего объекта (группы или пользователя) | | |

Поскольку на сайте компании Microsoft не удалось найти информации о том, какие флаги соответствуют всем возможным значениям параметров безопасности (`Full Control`, `Read`, `Modify` и т. д.), то пришлось выполнять ряд экспериментов. Результатом эксперимента являются составленные табл. 6.6—6.8.

***Таблица 6.6.*** *Параметр* `AccessMask`

| | &h1 | &h2 | &h4 | &h8 | &h10 | &h20 | &h40 | &h80 |
|---|---|---|---|---|---|---|---|---|
| `Full Control` | | | | | | | | |
| `Traverse Folder/ Execute Folder` | | | | | | + | | |
| `List Folder/ ReadData` | + | | | | | | | |
| `Read Attributes` | | | | | | | | + |
| `Read Extended Attributes` | | | | + | | | | |

***Таблица 6.6*** *(продолжение)*

| | &h1 | &h2 | &h4 | &h8 | &h10 | &h20 | &h40 | &h80 |
|---|---|---|---|---|---|---|---|---|
| Create Files/ Write Data | | + | | | | | | |
| Create Folder/ Append Data | | | + | | | | | |
| Write Attributes | | | | | | | | |
| Write Extended Attributes | | | | | + | | | |
| Delete Subfolders and Files | | | | | | | + | |
| Delete | | | | | | | | |
| Read Permissions | | | | | | | | |
| Change Permissions | | | | | | | | |
| Take Ownership | | | | | | | | |

***Таблица 6.6*** *(продолжение)*

| | &h100 | &h10000 | &h20000 | &h40000 | &h80000 | &h100000 | &h10000000 | &h20000000 | &h40000000 | &h80000000 |
|---|---|---|---|---|---|---|---|---|---|---|
| Full Control | | | | | | + | | | | |
| Traverse Folder/ Execute Folder | | | | | | + | | + | | |
| List Folder/ReadData | | | | | | + | | | | + |
| Read Attributes | | | | | | + | | + | | + |
| Read Extended Attributes | | | | | | + | | | | + |
| Create Files/ Write Data | | | | | | + | | | + | |
| Create Folder/ Append Data | | | | | | + | | | + | |
| Write Attributes | + | | | | | + | | | + | |
| Write Extended Attributes | | | | | | + | | | + | |

***Таблица 6.6*** *(окончание)*

| | &h100 | &h10000 | &h20000 | &h40000 | &h80000 | &h100000 | &h10000000 | &h20000000 | &h40000000 | &h80000000 |
|---|---|---|---|---|---|---|---|---|---|---|
| Delete Subfolders and Files | | | | | | + | | | | |
| Delete | | + | | | | + | | | | |
| Read Permissions | | | + | | | + | + | + | + | + |
| Change Permissions | | | | + | | + | | | | |
| Take Ownership | | | | | | + | | | | |

***Таблица 6.7.*** *Параметр* `AccessFlags`

| | **Inheritance** | |
|---|---|---|
| | **Not Inheritance** | **Parent Object** |
| This folder only | &h0 (&h4) | &h10 (&h14) |
| This folder, subfolder and files | &h3 (&h7) | &h13 (&h17) |
| This folder and subfolder | &h2 (&h6) | &h12 (&h16) |
| This folder and files | &h1 (&h5) | &h11 (&h15) |
| Subfolders and files only | &h0b (&h0f) | &h1b (&h1f) |
| Subfolder only | &h0a (&h0e) | &h1a (&h1e) |
| Files only | &h0d (&h9) | &h1d (&h19) |
| Nothing | &h0c (&h8) | &h1c (&h18) |

***Таблица 6.8.*** *Параметр* `AceType`

| **Type** | **Flag** |
|---|---|
| Deny | &h1 |
| Allow (Full Control) | &h0 |
| Audit | &h2 |

Для удобства дополнительно составлена табл. 6.9, в которой указаны все параметры стандартных атрибутов, которые видит пользователь, открывая окно свойств безопасности папки или файла, такие как `Full Control`.

***Таблица 6.9.*** *Набор стандартных атрибутов*

| Установка атрибута | Фактические атрибуты | .AccessMask | .AceFlags |
|---|---|---|---|
| `Full Control` | `Full Control`<br>`Modify`<br>`Read & Execute`<br>`List Folder Contents`<br>`Read`<br>`Write` | &h10000000 | &h3 |
| `Modify` | `Modify`<br>`Read & Execute`<br>`List Folder Contents`<br>`Read`<br>`Write` | &h20, &h10000, &h80000000, &h40000000 | &h3 |
| `Read & Execute` | `Read & Execute`<br>`List Folder Contents`<br>`Read` | &h20000000+&h80000000 | &h2 |
| `List Folder Contents` | `List Folder Contents` | &h20000000+&h80000000 | &h3 |
| `Read` | `Read` | &h80000000 | &h3 |
| `Write` | `Write` | &h2, &h4, &h10, &h100 | &h3 |

## Основные операции: просмотр, добавление, удаление

Операции добавления, удаления и просмотра для VBScript и VB.NET выглядят одинаково. Различны схемы получения доступа к объектам.

По умолчанию в VBScript допускается неявное объявление переменных. При этом переменная создается без ее предварительного объявления операторами `Dim`, `Private`, `Public` или `ReDim`. Однако, разрешив неявное объявление переменных в сценариях, велик риск пропустить допущенную синтаксическую ошибку в имени переменной во время программирования.

В том случае, если допущена ошибка, VBScript просто объявит новую переменную и создаст ее, вследствие чего программа будет работать некорректно. Для обнаружения ошибок-опечаток такого рода в первую строку программы необходимо поместить оператор `Option Explicit`, который сигнализирует о переменных, если они не были объявлены явно.

Опция `Explicit` на платформе VB.NET включена. При ее включении перед использованием переменной ее необходимо объявить. Опцию `Explicit` можно отключить на всех страницах VB.NET в файле machine.config:

```
<Compilation>
Explicit = "False"
</Compilation>
```

## Просмотр списка доступных объектов

Просмотр списка безопасности объекта (рис. 6.3) осуществляется чтением соответствующих данных из массива (листинги 6.22 и 6.23).

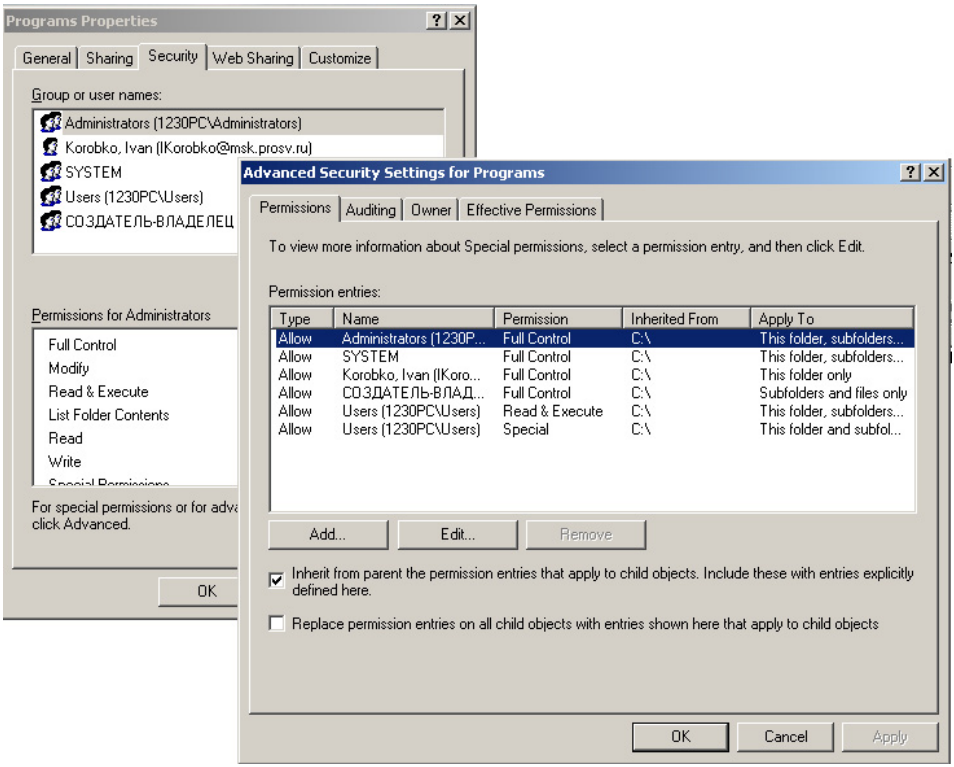



**Рис. 6.3.** Вкладка безопасности объекта (файла или папки)

**Листинг 6.22. Чтение параметров безопасности объекта. VBScript**

```
vpath = "c:\Programs"
Set sec = CreateObject("ADsSecurity")
Set sd = sec.GetSecurityDescriptor(FILE://+ vpath)
Set dacl = sd.DiscretionaryAcl
For Each ace In dacl
a = Ace.Trustee
b = Ace.AccessMask
c = Ace.AceType
temp = temp + a + " " + b + " " + c + Chr(13)
Next
Wscript.Echo temp
```

**Листинг 6.23. Чтение параметров безопасности объекта. VB.NET**

```
Imports ADSSECURITYLib
Imports ActiveDs

vpath = "c:\Programs"

Dim Sec As ADsSecurity
Dim SD As SecurityDescriptor
Dim DAcl As AccessControlList
Dim Ace As AccessControlEntry

Sec = New ADsSecurity
SD = CType(Sec.GetSecurityDescriptor("FILE://" & vpath), SecurityDescriptor)
DAcl = CType(SD.DiscretionaryAcl, AccessControlList)
Ace = New AccessControlEntry

Dim temp As String
 For Each Ace In DAcl
 Dim a, f, g, b, c, d, ee As String
 a = Ace.Trustee
 b = Ace.AccessMask.ToString
 c = Ace.AceType.ToString
 temp = temp + a + " " + b + " " + c + Chr(13)
 Next
 MsgBox(h)

SD.DiscretionaryAcl = DAcl
Sec.SetSecurityDescriptor(SD)
```

## Добавление объекта

Добавление объектов осуществляется по алгоритму, работающему в скриптах, созданных и на VBScript и на VB.NET. Сначала получают доступ к объекту, свойства которого корректируют в библиотеке (ADsSecurity). Затем осуществляется присвоение приведенных в табл. 6.5—6.8 значений для получения требуемого эффекта. На последнем этапе осуществляется запись сделанных изменений в файловую систему NTFS (листинг 6.24).

**Листинг 6.24. Назначение прав на один объект NTFS наVBScript**

```
vpath = "c:\Programs"
username = "domain\username"

Set sec = CreateObject("ADsSecurity")
Set sd = sec.GetSecurityDescriptor(FILE://+ vpath)
Set dacl = sd.DiscretionaryAcl

ace.Trustee = username
ace.AccessMask = &h20000000
ace.AceType = &h0
ace.AceFlags = &h3
dacl.AddAce ace1

sd.DiscretionaryAcl = dacl
sec.SetSecurityDescriptor sd
```

Если необходимо добавить сразу несколько новых объектов (листинги 6.25 и 6.26), то необходимо учесть следующее:

- для каждого нового добавляемого объекта нужно создать свой собственный объект AccessControlEntry, при этом описание объекта GetSecurityDescriptor и ADsSecurity у них может быть общим;
- после добавления нового объекта должно пройти некоторое время, прежде чем он станет доступен для последующих действий. Для этого необходимо сделать в сценарии паузу. Для этих целей используется функция Timer():

```
X=1' x — количество секунд задержки
Q = Timer + X
 Do
Loop Until Timer >= Q
```

Определение времени "простоя" — эвристическая операция, поскольку время зависит одновременно от нескольких факторов: скорости сети, мощности сервера и других факторов. После многочисленных экспериментов был определен диапазон временных задержек: 0,3—0,5 с.

**Листинг 6.25. Назначение прав на несколько объектов NTFS наVBScript**

```
vpath = "c:\Programs"
username1 = "domain\username1"
username2 = "domain\username2"

Set sec = CreateObject("ADsSecurity")
Set sd = sec.GetSecurityDescriptor(FILE://+ vpath)
Set dacl = sd.DiscretionaryAcl

Set ace1 = CreateObject("AccessControlEntry")
ace1.Trustee = username1
ace1.AccessMask = &h20000000
ace1.AceType = &h0
ace1.AceFlags = &h3
dacl.AddAce ace1

Q = Timer + 1
Do
Loop Until Timer >= Q

Set ace2 = CreateObject("AccessControlEntry")
ace2.Trustee = username2
ace2.AccessMask = &h20000000
ace2.AceType = &h0
ace2.AceFlags = &h3
dacl.AddAce ace2

sd.DiscretionaryAcl = dacl
sec.SetSecurityDescriptor sd

set dacl=nothing
set sec=nothing
```

**Листинг 6.26. Назначение прав на один объект NTFS на VB.NET**

```
Imports ADSSECURITYLib
Imports ActiveDs

vpath = "c:\Programs"
```

```
username = "username\domain"
Dim Sec As ADsSecurity
Dim SD As SecurityDescriptor
Dim DAcl As AccessControlList
Dim Ace As AccessControlEntry

Sec = New ADsSecurity
SD = CType(Sec.GetSecurityDescriptor("FILE://" & vpath),
SecurityDescriptor)
DAcl = CType(SD.DiscretionaryAcl, AccessControlList)
Ace = New AccessControlEntry

 Ace.Trustee = CStr(username)
 Ace.AccessMask = &H10000000
 Ace.AceType = &H0
 Ace.AceFlags = &H3
 DAcl.AddAce(Ace)

SD.DiscretionaryAcl = DAcl
Sec.SetSecurityDescriptor(SD)
```

## Удаление существующих объектов

Удаление существующих групп или пользователей выполняется с помощью свойства `RemoveAcl` коллекции объектов `DiscretionaryAcl` (листинги 6.27 и 6.28).

**Листинг 6.27. Удаление прав выбранного объекта на NTFS наVBScript**

```
vpath = "c:\Programs"
username = "domain\username"

Set sec = CreateObject("ADsSecurity")
Set sd = sec.GetSecurityDescriptor(FILE://+ vpath)
Set dacl = sd.DiscretionaryAcl

For Each ace In dacl
       If (ace.Trustee=username)
dacl.RemoveAce ace
end if
```

```
Nex
t
sd.DiscretionaryAcl = dacl
set dacl=nothing
set sec=nothing
```

**Листинг 6.28. Удаление прав выбранного объекта на NTFS на VB.NET**

```
Dim vpath, username As String
vpath = "c:\Programs"
username = "username\domain"

Dim Sec As ADsSecurity
Dim SD As SecurityDescriptor
Dim DAcl As AccessControlList
Dim Ace As AccessControlEntry

Sec = New ADsSecurity
SD = CType(Sec.GetSecurityDescriptor("FILE://" & vpath),
SecurityDescriptor)
DAcl = CType(SD.DiscretionaryAcl, AccessControlList)
Ace = New AccessControlEntry

 For Each Ace In DAcl
 If StrComp(UCase(Ace.Trustee), UCase(username)) = 0 Then
 DAcl.RemoveAce(Ace)
 End If
 Next

SD.DiscretionaryAcl = DAcl
Sec.SetSecurityDescriptor(SD)
```



# Глава 7

# Основы программирования Active Directory

## Active Directory изнутри

Active Directory (AD) — это служба каталогов, обеспечивающая централизованное управление сетью. Active Directory содержит информацию об объектах сети и обеспечивает доступ к этой информации пользователям, компьютерам и приложениям.

Active Directory обладает следующими особенностями.

- *Масштабируемость.* В отличие от большинства других баз данных, которые являются реляционными, база данных Active Directory является иерархической. В базах данных взаимосвязи между записями определяются с помощью ключей, которые хранятся совместно с данными. В иерархической базе данных взаимосвязи между записями имеют характер "родитель-потомок": каждая запись, за исключением корневой, обладает родительской записью. У каждой родительской записи может быть один или несколько потомков. Иерархическая база данных позволяет хранить большое количество объектов, при этом быстро получать доступ к необходимым объектам.
- *Поддержка открытых стандартов.* Active Directory объединяет в себе концепцию пространства имен Интернета со службой каталогов Windows NT, что позволяет объединить различные пространства имен в разноразрядных аппаратных и программных средах и управлять ими. Для управления пространством имен Active Directory используется библиотека интерфейса службы активного каталога (Active Directory Service Interface — ADSI).
- *Поддержка стандартных форматов имен.* Active Directory поддерживает несколько общих форматов имен, позволяет приложениям и пользователям получать доступ к каталогу, применяя наиболее удобный для них формат (табл. 7.1).

***Таблица 7.1.*** *Форматы имен, используемые в Active Directory*

| Формат | Описание |
|---|---|
| UPN | Формат основного имени пользователя (User Principial Name, UPN) описан в RFC 822[1]. UPN известны как адреса электронной почты. AD обеспечивает "дружественные" имена в этом формате. В качестве имени для регистрации в сети пользователь может использовать как имя учетной записи SAM, так и имя в формате RFC 822. Пример: NIvanov@company.com |
| LDAP URL | Имена LDAP (см. RFC 1779, 2247), имеющие более сложную структуру по сравнению с URL-именами, которые построены на основе протокола X.500. Имена URL LDAP состоят из следующих частей: CN, OU, DN.<br><br>CN расшифровывается как Common Name (общее имя), OU означает организационную единицу (Organization Unit), а DN — контроллер домена (Distinguished Name — отличительное имя). Часть имени "DC=" делает возможным подключения каталогов X.500 к пространству имен DNS.<br><br>Пример: LDAP://www.domain.ru/CN=NIvanov, OU=Support, OU=Developers или LDAP:// CN=NIvanov, OU=Support, OU=Developers, DN=ru, DN=domain, DN=www |
| UNC | UNC или канонический вид (Active Directory Canonical Name, ADCN), который имеет вид computer.domain.com/folder1/subfolder2. В AD данный формат является путем в иерархической структуре к объекту (см. RFC 1123) |

# DNS и Active Directory

Для иерархического именования доменов и компьютеров в Active Directory используется система DNS, поэтому объекты доменов и компьютеров являются частью иерархии доменов DNS и частью иерархии доменов Active Directory. Несмотря на то, что имена в обоих системах идентичны, они относятся к различным пространствам имен. Взаимодействие DNS-имен доменов и их IP-адресов в Active Directory реализовано в соответствии с общепринятыми соглашениями об именовании в DNS.

Доменная система именования (Domain Name System, DNS) — иерархическая система именования для идентификации хостов. Основная функция DNS заключается в прямом и обратном разрешении имен компьютеров в IP-адреса.

[1] Ознакомиться с полными версиями RFC-документов можно на сайте **http://www.rfc-archive.org/**.

База данных DNS — это древовидная структура, называемая пространством имен доменов (domain space name).

В Windows 2000 полное доменное имя (Fully Qualified Domain Name, FQDN) компьютера состоит из двух частей (рис. 7.1):

- имени DNS-узла. Крайняя левая метка — это полноценное имя DNS-узла, идентифицирующее учетную запись компьютера, хранящуюся в Active Directory. Кроме того, это имя локальной учетной записи компьютера в диспетчере безопасности учетных записей (Security Account Manager, SAM) на рабочей станции или рядовом сервере (не контроллер домена). По умолчанию имя DNS-узла также используется в качестве NetBIOS-имени. Это делается для совместимости с доменами на основе Windows NT 3.51 и Windows NT 4 и с рабочими станциями под управлением Windows 9x;
- основного суффикса имени DNS-имени домена. По-умолчанию это домен Windows, к которому относится данный компьютер.




**Рис. 7.1.** Схема образования полного доменного имени (FQDN)

Кроме DNS-имен компьютеров, контроллеры домена Active Directory идентифицируются по видам предоставляемых ими служб: серверы протокола LDAP (Lightweight Directory Access Protocol); контроллеры доменов; сервер глобального каталога GC (Global Catalog). Получив указание на имя домена и службу, DNS-сервер ищет контроллер со службой искомого типа в данном домене.

Глобальный каталог (Global Catalog, GC) — это контроллер домена, в котором существуют три доступных для записи каталога: домена, схемы и конфигурации (листинг 7.1). Каталог автоматически создается при репликации Active Directory. Все разделы каталогов на сервере глобального каталога хранятся в одной базе данных каталога (Ntds.dit). Глобальный каталог хранит сведения обо всех лесах, поэтому его можно использовать для поиска любых объектов в лесу без переадресации на другие серверы. Если запрос осуществляется по порту 389 (стандартный порт протокола LDAP), то в случае неудачного поиска, запрос будет последовательно передаваться другим контроллерам домена. Если обращение идет по стандартному порту глобального каталога (GC) 3268, то поиск осуществляется по всем разделам леса.

Для безопасного доступа к службам следует использовать порты, применяющие SSL (табл. 7.2).

***Таблица 7.2.*** *Описание портов, поддерживающихся ADSI*

| Порт | Описание |
|---|---|
| 389 | Порт для открытых запросов LDAP |
| 636 | Порт для запросов LDAP с использованием протокола SSL |
| 3268 | Порт для открытых запросов GC |
| 3269 | Порт для запросов GC с использованием протокола SSL |

**Листинг 7.1. Получение имени глобального каталога**

```
temp=""
Set gc = GetObject("GC:")
For each child in gc
       temp=temp+child.name
Next
msgbox temp
```

# Архитектура службы каталогов Active Directory

Чтобы понять, как хранятся и обрабатываются данные в Active Directory, необходимо представлять себе, как взаимодействуют отдельные компоненты этой службы каталогов. Службу каталогов можно представить в виде многоуровневой структуры (рис. 7.2). Существует три уровня служб и несколько интерфейсов и протоколов, которые образуют полный спектр служб каталогов. Три уровня служб содержат все данные для нахождения записей в базе данных каталога. Выше уровня служб находятся протоколы и API-интерфейсы, которые обеспечивают взаимодействие между клиентами и службами каталогов, или при репликации — между службами каталогов.

Основные службы-компоненты Active Directory:

- агент системы каталогов (directory system agent, DSA) формирует иерархию каталога на основе отношений "родитель-потомок" и обеспечивает интерфейс прикладного программирования (Application Programming Interfece, API) для запросов на доступ к каталогу;

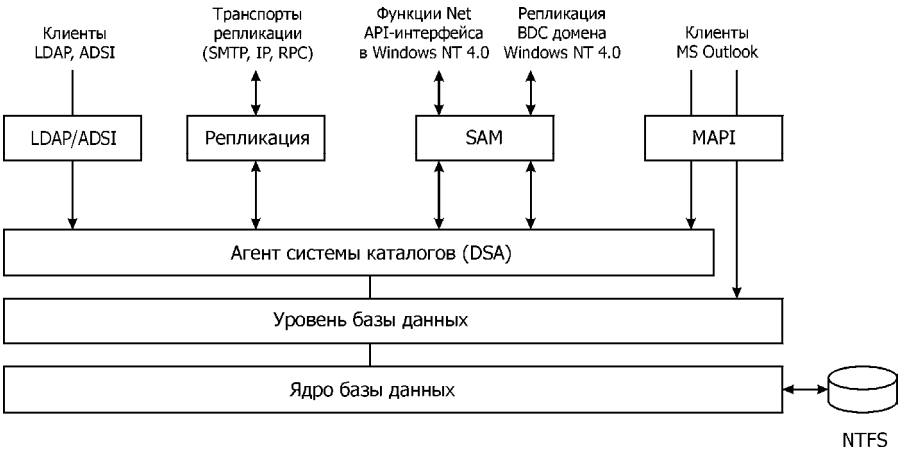



**Рис. 7.2.** Многоуровневая структура службы каталогов

- уровень базы данных — промежуточный уровень абстракций между базой данных и приложениями;
- ядро базы данных (Extensible Storage Engine, ESE), работающее непосредственно с записями хранилища каталогов, различает объекты по атрибуту относительно составного каталога;
- хранилище данных (файл базы данных Ntds.dit). С этим файлом работает только ядро базы данных. Обращаться напрямую к нему можно с помощью программы Ntdsutil, которая находится в папке Support/Tools на диске с операционной системой Windows 2000/2003 Server.

Клиенты могут получить доступ к Active Directory по одному из перечисленных механизмов:

- LDAP/ADSI — клиенты, поддерживающие протокол LDAP, используют его для доступа к агенту системы каталогов. Интерфейсы службы каталогов Active Directory (Active Directory Service Interface — ADSI) служат для абстрагирования от LDAP интерфейса прикладного программирования (API), представляя COM-интерфейсы для взаимодействия с Active Directory. Однако нужно помнить, что в Active Directory используется только LDAP;
- MAPI — при обмене сообщениями и коллективной работе клиенты MS Outlook подключаются к агенту системы каталогов по механизму вызова удаленных процедур MAPI через интерфейс средства доступа к адресной книге;
- SAM — клиенты MS Windows NT 4.0 и ранее, Windows 9x подключаются к агенту системы каталогов (DSA) через SAM;

- REPL — в процессе репликации каталогов агенты системы каталогов (DSA) Active Directory взаимодействуют через RPC-интерфейс.

Существует четыре способа доступа к Active Directory (см. рис. 1.3). С точки зрения программного управления Active Directory системного администратора интересует только ADSI. ADSI поддерживает такие языки программирования, как C/C++, VB/VBScript, Jscript, WSH, и представляет объекты Active Directory в виде COM-объектов, а управление осуществляется с помощью COM-интерфейсов. Провайдеры ADSI (ADSI providers) представляют объекты ADSI в соответствующие пространства имен, т. е. они преобразуют вызовы COM-интерфейсов к запросам API конкретной службы каталогов.

# Объектная модель ADSI

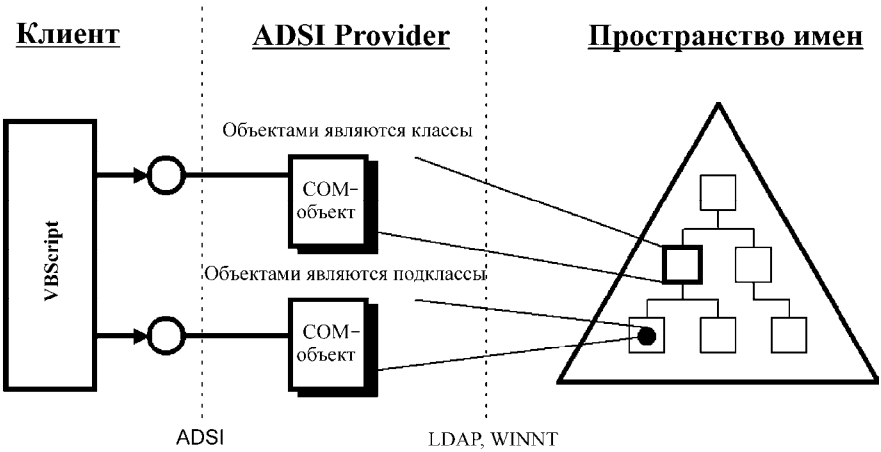



**Рис. 7.3.** Схема объектной модели ADSI

Схема объектной модели ADSI состоит из трех частей (рис. 7.3). Используя клиента — язык программирования, скрипт получает доступ к COM-объектам. Объекты могут быть двух видов — классами и подклассами. Обращение к подклассам осуществляется через классы, однако на схеме это не показано, чтобы не загромождать рисунок. С помощью COM-объекта через протокол LDAP или WinNT сценарий загрузки получает доступ к выбранному элементу объектной модели. Пространство имен условно обозначено треугольником; квадратами обозначены классы, а кружками подклассы. Методы доступа к каждому из перечисленных протоколов отличаются, однако можно

привести пример, который наглядно иллюстрирует приведенную схему — листинг 7.2.

**Листинг 7.2. Доступ к коллекции объектов**

```
Set obj=GetObject("Protocol://ClassName")
For each SubClass in ClassName
       temp=SubClass.Property
Next
```

# Провайдеры ADSI

Интерфейс ADSI поддерживает провайдеры (табл. 7.3), с помощью которых осуществляется программное администрирование.

***Таблица 7.3.*** *Поддерживаемые ADSI-провайдеры*

| Название | Протокол | Описание |
|---|---|---|
| LDAP Provider | "LDAP:" | Администрирование Active Directory, Microsoft Exchange Server |
| WinNT Provide | "WinNT:" | Администрирование Windows NT, Windows 200x, Windows XP |
| NDS Provider | "NDS:" | Администрирование Novell NetWare Directory Service |
| NVCOMPAT | "NVCOMPAT:" | Администрирование Novell NetWare 3.1 |
| IIS | "IIS: | Протокол IIS предназначен для управления WWW- и FTP-узлами по протоколу HTTP |

Для определения всех доступных протоколов в сети используют службу ADs. С помощью функции `GetObject` получают доступ к коллекции провайдеров (листинг 7.3, рис. 7.4).

**Листинг 7.3. Определение доступных в домене служб ADs**

```
Set obj=GetObject("ADs:")
       For Each provider IN obj
       temp = temp + provider.name + chr(13)
       Next
MsgBox temp
```




**Рис. 7.4.** Доступные службы ADs в домене

Из всех перечисленных провайдеров будут рассмотрены только: LDAP и WINNT.

Провайдер ADSI LDAP выполняется на клиенте ADSI и обеспечивает доступ к Active Directory. Кроме служб каталогов Active Directory Windows 2000, LDAP-провайдер обеспечивает доступ к:

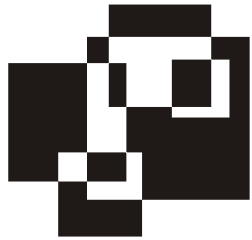
- Netscape Directory Server;
- Microsoft Exchange Server 5.x и выше;
- Microsoft Commercial Internet System (MCIS) Address Book Server;
- и т. д.

При программном управлении Active Directory часто используется именно этот провайдер. Запрос провайдеру LDAP составляется в формате LDAP URL:

```
LDAP://HostName[:PortNumber][/DistinguishedName]
```

Провайдер WinNT ADSI поддерживает доступ к каталогам Microsoft Windows 4.0/3.x, обеспечивает связь с PDC и BDC. Провайдер WinNT в основном используется для работы с принтерами. В отличие от провайдера LDAP, провайдер WinNT рассматривает принтер не как сетевое, а как локальное устройство. Только с помощью провайдера WinNT можно управлять состоянием и очередями принтеров. Совместное использование обоих провайдеров позволит осуществлять мониторинг и управление сетевыми принтерами домена в полном объеме. Порядок построения запроса для провайдера WinNT в формате UNC следующий:

```
WinNT:[//DomainName[/ComputerName[/ObjectName[,className]]]]
```



# Глава 8

# Программное управление ADSI: WinNT

## Объектная модель провайдера WinNT

Рассмотрим программное управление ADSI с помощью провайдера WinNT. Доступ к классам и подклассам осуществляется в соответствии с объектной моделью, приведенной на рис. 8.1. Каждый из классов может содержать несколько подклассов (табл. 8.1 и 8.2).

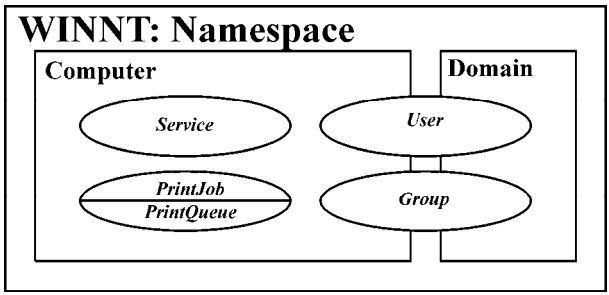



**Рис. 8.1.** Объектная модель протокола WinNT

Доступ к объектам по протоколу WinNT описывается запросом, имеющим вид:

```
WinNT:[//DomainName[/ComputerName[/ObjectName[,ClassName]]]]
```

Доступ к провайдеру WinNT осуществляется по одному из приведенных далее шаблонов.

- Название класса содержится в запросе. С помощью функции `GetObject()` формируется запрос, который включает следующие параметры: название протокола — WinNT, имя домена — `DomainName`, имя рабочей станции — `ComputerName`, название объекта — `ObjectName`, название класса — `ClassName`. В этом случае, доступ к подклассам осуществляется с помощью

цикла `For…Next`. На языке VBScript запрос в общем виде выглядит следующим образом — листинг 8.1. (Параметры `ComputerName` и `ObjectName` могут отсутствовать, если осуществляется поиск объектов.)

- Подключение к классу с помощью фильтра. Метод по своей сути аналогичен предыдущему. Такой способ доступа к данным позволяет значительно увеличить скорость исполнения скрипта (листинг 8.2).

**Листинг 8.1. Пример запроса к AD с помощью провайдера WinNT**

```
Set obj=GetObject("WinNT://" & DomainName & "/" & ComputerName & "/" &
ObjectName & "," &  ClassName).
       For Each element In obj
       element.value
       Next
```

**Листинг 8.2. Пример запроса к AD с помощью провайдера WinNT с использованием фильтра**

```
Set obj = GetObject("WinNT://" & DomainName)
obj.Filter = Array("user")
       For Each element In obj
       element.value
       Next
```

Приведем два примера: в первом примере будет осуществляться поиск объектов класса и вывод свойств этих объектов (листинги 8.3 и 8.4), во втором — чтение и вывод свойств заданного объекта (листинги 8.5 и 8.6). В первом примере с помощью скрипта на VBScript будут определены учетные записи пользователей домена и прочитаны их имена.

**Листинг 8.3. Поиск объектов класса и вывод свойств этих объектов. Способ 1**

```
strDomain="MyDomain"
    Set Computer =GetObject("WinNT://" & strDomain & ",user")
         For Each User in Computer
           users_d=users_d & "  "& User.Name  & chr(13)
         Next
Wsh.Echo users_d
```

**Листинг 8.4. Поиск объектов класса и вывод свойств этих объектов. Способ 2**

```
strDomain="MyDomain"
Set Computer =GetObject("WinNT://" & strDomain)
Computer.Filter = Array("user")
           For Each User in Computer
             users_d=users_d & "  "& User.Name  & chr(13)
           Next
Wsh.Echo users_d
```

Во втором примере явным образом задано имя пользователя и осуществляется чтение его свойств.

**Листинг 8.5. Чтение и вывод свойств заданного объекта. Способ 1**

```
strDomain="MyDomain"
strUser="MyUserName"
Set Computer =GetObject("WinNT://" & strDomain & "/" & strUser & ",user")
users_d= User.Name & chr(13)
Wsh.Echo users_d
```

**Листинг 8.6. Чтение и вывод свойств заданного объекта. Способ 2**

```
strDomain="MyDomain"
Set Computer =GetObject("WinNT://" & strDomain)
Computer.Filter = Array("user")
         For Each User in Computer
           users_d= User.Name  & chr(13)
         Next
Wsh.Echo users_d
```

Для эффективного пользования объектной моделью, необходимо знать назначение классов, какие они включают в себя подклассы, и какие параметры имеют подклассы (табл. 8.1).

***Таблица 8.1.*** *Классы, поддерживаемые WinNT*

| objectClass | Описание класса |
|---|---|
| Domain | Класс предназначен для подключения к домену. Явное использование домена экономит сетевой трафик |
| Computer | Класс предназначен для определения роли компьютера в домене, для управления компьютером |
| User | Осуществляется управление пользователями: создание, удаление. Управление свойствами пользователя: активизация пользователя, подключение домашнего каталога или сценария загрузки; управление параметрами, касающимися пароля. Чтение характеристик пользователя: описание, расположение, телефон и др. параметры |
| Group | Управление группами: создание локальных/глобальных групп, преобразование типов групп, определение членства в группах и т. д. |
| FileShare | Управление файловыми ресурсами: предоставление ресурсов пользователям |
| PrintQueue | Управление локальным принтером: перезагрузка принтера, его приостановка, возобновление работы; чтение полей принтера |
| PrintJobs | Подкласс класса PrintQueue, с помощью которого осуществляется управление очередью печати: установка приоритетов заданий, управление заданиями |

Соответствие классов и подклассов приведено в табл. 8.2.

***Таблица 8.2.*** *Классы и подклассы, поддерживаемые WinNT*

| Название класса | Описание класса |
|---|---|
| Namespace | Контейнер верхнего уровня |
| Domain | Доступные домены Windows |
| User | Управление пользователем |
| Group | |
| UserGroupCollection | Управление группами пользователей |
| GroupCollection | Управление другими группами |
| Computer | Определение параметров компьютера |
| PrintJob | |
| PrintJobsCollection | Управление заданием |
| PrintQueue | Управление принтером |

***Таблица 8.2*** *(окончание)*

| Название класса | Описание класса |
|---|---|
| Service | |
| FileService | |
| FileShare | Управление предоставлением объектов в общее пользование |
| Resource | Управление сервисами |
| Session | Управление сессиями |
| User | Управление локальными учетными записями пользователей |
| Group | |
| UserCollection | Свойства локальных пользователей |
| GroupCollection | Свойства локальных групп |

Из объектной модели протокола WinNT видно (см. рис. 8.1), что в пространстве имен `NameSpace` существует всего два класса: `Domain` и `Computer`. Рассмотрим каждый из классов в отдельности.

# Класс *Domain*

## Определение доступных доменов

Класс `Domain` является верхним уровнем пространства имен, поэтому для определения доступных доменов в функции `GetObject()` ограничиваются названием протокола.

В названии протокола должно быть написано именно `WinNT` — в противном случае сценарий выдаст ошибку (листинг 8.7, рис. 8.2).

**Листинг 8.7. Определение списка доступных доменов с помощью протокола WinNT**

```
Set obj=GetObject("WinNT:")
       For Each element In obj
              temp=temp+element.Name
       Next
MsgBox temp
```




**Рис. 8.2.** Список доступных доменов через провайдер WinNT

## Чтение параметров класса *Domain*

Класс `Domain` содержит восемь параметров (листинг 8.8). Все эти параметры задаются в групповых политиках, за исключением параметра `Name`. Описание параметров см. в *приложении 2.*

Создавая скрипт на VBScript, помните, что VBScript не преобразует типы данных автоматически, поэтому числовые данные необходимо преобразовывать в строковые с помощью функции `cstr()`.

**Листинг 8.8. Определение свойств класса `Domain`**

```
Set obj=GetObject("WinNT:")
       For Each element In obj
t1="Name: " + cstr(element.Name)+chr(13)
t2="MinPasswordLength: "+ cstr(element.MinPasswordLength)+chr(13)
t3="MinPasswordAge: " + cstr(element.MinPasswordAge)+chr(13)
t4="MaxBadPasswordsAllowed: "+
cstr(element.MaxBadPasswordsAllowed)+chr(13)
t5="AutoUnlockInterval: " + cstr(element.AutoUnlockInterval)+chr(13)
t6="LockoutObservationInterval: " +
cstr(element.LockoutObservationInterval)
temp=temp+t1+t2+t3+t4+t5+t6+chr(13)+chr(13)
       Next
MsgBox temp
```

По окончанию работы скрипта выдается соответствующее сообщение (рис. 8.3).

Значения `MinPasswordAge` и `MaxPasswordAge` указываются в групповых политиках в днях, поэтому необходим перевод в дни, для чего полученное число нужно разделить на 86400. Значения `MinPasswordAge` и `LockoutObservationInterval` указываются в групповых политиках в минутах, поэтому полученные значения необходимо разделить на 60.

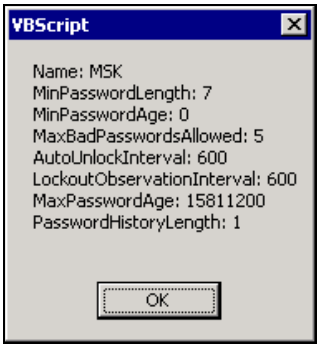



**Рис. 8.3.** Свойства класса `Domain`

## Обновление параметров класса *Domain*

Установка новых параметров перечисленных значений осуществляется с помощью метода `SetInfo` (листинг 8.9).

**Листинг 8.9. Сохранение изменений в AD с помощью метода `SetInfo`**

```
       NewLenght=10
Set obj=GetObject("WinNT:")
       For Each element In obj
              element.MinPasswordLength= NewLenght
              obj.SetInfo
       temp ="NEW: MinPasswordLength: "+
cstr(element.MinPasswordLength)+chr(13)
       Next
MsgBox temp
```

## Перечисление объектов класса *Domain*

Просмотр содержимого контейнера осуществляется с помощью конструкции `For…Next` в соответствии с приведенным в листинге 8.10 шаблоном. В качестве значения переменной `Container` может быть имя домена или компьютера.

**Листинг 8.10. Просмотр содержимого контейнера**

```
Container="Value"
Set obj=GetObject("WinNT://"& Container)
       For Each element In obj
```

```
        temp = temp + element.name + ";     "
        Next
MsgBox temp
```

Такой метод перечисления объектов будет возвращать все содержимое любой базы SAM — контроллера домена, сервера или рабочей станции.

В приведенном примере возвращались объекты всех классов, содержащиеся в домене (или в локальной базе SAM), поскольку тип объектов не определялся. В домене огромное количество объектов, для увеличения скорости работы скрипта целесообразно использовать фильтр (листинг 8.11), который включает в себя объекты, перечисленные в табл. 8.3.

***Таблица 8.3.*** *Фильтры провайдера WinNT*

| Объект | Описание | Объект | Описание |
|---|---|---|---|
| `User` | Учетная запись пользователя | `Group` | Все группы |
| `Computer` | Учетная запись компьютера | `LocalGroup` | Локальная группа |
| `Service` | Сервис | `GlobalGroup` | Глобальная группа |

**Листинг 8.11. Получение списка всех учетных записей пользователей в домене**

```
Set objDomain=GetObject("WinNT:")
        For Each domain_element In objDomain
        Domain_Name= domain_element.Name
        Next
Set obj=GetObject("WinNT://" & Domain_Name)
obj.filter=array("user")
        For Each element In obj
        temp =temp+element.name+";     "
        Next
MsgBox temp
```

## Создание, переименование и удаление объектов в домене

В Active Directory существует три типа встроенных объектов: учетная запись пользователя; группа, которая может быть локальной или глобальной; учетная запись компьютера. С учетной записью пользователя можно проделывать следующие операции: создавать, удалять, переименовывать; группу можно создавать и удалять; учетную запись компьютера можно создавать и удалять.

## Создание объектов

Создание учетной записи пользователя осуществляется с помощью функции `Create()`. Пример создания учетной записи пользователя в AD с именем `UserName` приведен в листинге 8.12. Все манипуляции с паролем и другими параметрами учетной записи должны производиться после ее создания, т. е. после применения метода `SetInfo`.

**Листинг 8.12. Создание учетной записи пользователя**

```
Set objDomain=GetObject("WinNT:")
       For Each domain_element In objDomain
       Domain_Name= domain_element.Name
       Next
              Set obj=GetObject("WinNT://" & Domain_Name)
              NewUser="UserName"
              Set CU=obj.Create("User",NewUser)
              CU.SetInfo
```

Создание учетной записи группы отличается тем, что с помощью функции `Put()` указывают тип создаваемой группы (листинг 8.13). Локальной группе соответствует значение 4, глобальной — 2.

**Листинг 8.13. Создание учетной записи группы**

```
Set objDomain=GetObject("WinNT:")
       For Each domain_element In objDomain
       Domain_Name= domain_element.Name
       Next
              Set obj=GetObject("WinNT://" & Domain_Name)
              NewGroup="GroupName"
              Set CG=obj.Create("Group",NewGroup)
              CG.Put "groupType", 4
              CG.SetInfo
```

Создание учетной записи компьютера (листинг 8.14) аналогично созданию учетной записи пользователя за исключением следующего:

- объект должен быть создан с использованием класса `Computer`;
- у объекта должен быть установлен пользовательский флаг `ADS_UF_WORKSTATION_TRUST_ACCOUNT (0x1000)`. Подробную информацию о флагах

см. в *разд. "Манипулирование пользовательскими флагами функцией UserFlags()"*;

- начальный пароль учетной записи должен соответствовать имени компьютера, введенного строчными буквами. Результата добиваются с помощью функции `LCase()`.

**Листинг 8.14. Создание учетной записи компьютера**

```
Set objDomain=GetObject("WinNT: ")
       For Each domain_element In objDomain
       Domain_Name= domain_element.Name
       Next
              Set obj=GetObject("WinNT://" & Domain_Name)
              NewComputer="ComputerName"
              Set CC=obj.Create("Computer",UCase(NewComputer))
              CC.SetInfo
Set CAccount= GetObject("WinNT://" & Domain_Name&"/
"& NewComputer&"$,user")
CAccount.Put "UserFlags", (CAccount.Get("UserFlags") Or &H1000)
CAccount.SetPassword(LCase(NewComputer))
CAccount.SetInfo
```

## Удаление объектов

Все три типа объектов могут быть удалены с помощью метода `Delete` (листинг 8.15).

**Листинг 8.15. Удаление учетной записи объекта**

```
Set objDomain=GetObject("WinNT: ")
       For Each domain_element In objDomain
       Domain_Name= domain_element.Name
       Next
              Set obj=GetObject("WinNT://" & Domain_Name)
              ClassName="____"
              NameOfObject="____"
              Set CC=obj.Delete (ClassName, NameOfObject)
```

Переменная `ClassName` может принимать одно из значений: `computer`, `user`, `group`. `NameOfObject` — имя удаляемого объекта. Объект удаляется немед-

ленно после вызова функции `delete()`, и использовать метод `SetInfo` не нужно.

### Переименование объектов

Среди ранее перечисленных объектов переименовать можно только имя учетной записи пользователя с помощью функции `MoveHere()` (листинг 8.16).

**Листинг 8.16. Перемещение объекта в AD**

```
Set objDomain=GetObject("WinNT: ")
       For Each domain_element In objDomain
       Domain_Name= domain_element.Name
       Next
              OldUserName="____"
              NewUserName="____"
              Set obj=GetObject("WinNT://" & Domain_Name&"/"& OldUser-
Name&",user")
obj.MoveHere(User.AdsPath, NewUserName)
obj.Nothing
```

# Подкласс *User*

Учетные записи пользователей домена содержатся в подклассе `User` (листинг 8.17). Он включает в себя более 20 параметров, некоторые из них не поддерживаются Windows 2k. Описание параметров см. в *разд. "Объектная модель провайдера WinNT"*. Параметры подкласса изначально задаются в групповых политиках.

**Листинг 8.17. Чтение свойств учетной записи пользователя**

```
Set obj=GetObject("WinNT:")
       For Each str In obj
       DomainName=str.Name
       Next
Set UserName="Value"
Set element=GetObject("WinNT://" & DomainName & "/"& UserName)
u1="FullName: "+ cstr(element.FullName)+chr(13)
u2="UserFlags: "+ cstr(element.UserFlags)+chr(13)
u3="LoginScript: "+ cstr(element.LoginScript)+chr(13)
```

```
u4="MaxBadPasswordsAllowed: "+
cstr(element.MaxBadPasswordsAllowed)+chr(13)
u5="PasswordHistoryLength: "+ cstr(element.PasswordHistoryLength)+chr(13)
u6="AutoUnlockInterval: "+ cstr(element.AutoUnlockInterval)+chr(13)
u7="PasswordAge: "+ cstr(element.PasswordAge)+chr(13)
u8="PasswordExpired: "+ cstr(element.PasswordExpired)+chr(13)
temp=""
temp=u1+u2+u3+u4+u5+u6+u7+u8
MsgBox temp
```

## Манипулирование пользовательскими флагами функцией *UserFlags()*

Для просмотра и изменения состояния пользовательских флагов в базе SAM используются методы `Get()` и `Put()`. В табл. 8.4 приведены константы и их описание. Константы представляют собой шестнадцатеричные значения флагов.

***Таблица 8.4.*** *Пользовательские флаги функции* `UserFlags()`

| Название | Значение | Описание |
|---|---|---|
| ADS_UF_SCRIPT | 0X0001 | Управление исполнением сценария загрузки |
| ADS_UF_ACCOUNTDIABLE | 0X0002 | Управление флагом "Account is Disable" |
| ADS_UF_HOMEDIR_REQUIRED | 0X0003 | Требуется домашний каталог |
| ADS_UF_LOCKOUT | 0X0010 | Управление блокировкой учетной записи |
| ADS_UF_PASSW_NOTREQUIRED | 0X0020 | Не требуется пароль для регистрации в сети и на локальном компьютере |
| ADS_UF_PASSWORD_CANNOT_CHANGE | 0X0040 | Управление флагом "User Cannot Change Password" |
| ADS_UF_ENCRYPTED_PASSWORD | 0X0080 | Разрешить пользователю отправлять зашифрованный пароль |
| ADS_UF_TEMP_ACCOUNT | 0X0100 | Этот флаг обеспечивает доступ пользователя в текущем домене, но не дает им права доступа в домены, которые имеют доверительные отношения с текущим доменом. Для данного домена учетная запись становится локальной |

***Таблица 8.4*** *(окончание)*

| Название | Значение | Описание |
|---|---|---|
| ADS_UF_NORMAL_ACCOUNT | 0X0200 | Типичная учетная запись пользователя |
| ADS_UF_TRUST_ACCOUNT | 0X0800 | Флаг, противоположный флагу ADS_UF_TEMP_ACCOUNT.<br>Делает учетную запись глобальной |
| ADS_UF_WORKSTATION_TRUST_ACCOUNT | 0X1000 | Управление членством в домене рабочей станции |
| ADS_UF_SERVER_TRUST_ACCOUNT | 0X2000 | Учетная запись компьютера BDC |
| ADS_UF_PASSWORD_NEVER_EXPIRED | 0X10000 | Управление флагом "Password Never Expires" |
| ADS_UF_MSN_LOGON_ACCOUNT | 0X20000 | Учетная запись MNS Logon |
| ADS_UF_SAMRTCARD_REQUIRED | 0X40000 | Необходимо использовать Smart-карту для регистрации пользователя в сети |
| ADS_UF_TRUSTED_FOR_DELEGATION | 0X80000 | Установка делегирования |
| ADS_UF_NOT_DELEGATION | 0X100000 | Флаг, противоположный ADS_UF_TRUSTED_FOR_DELEGATION |

В таблице нет флага для параметра "User Must Change Password at Next Logon". Для установки флага следует изменить значение свойства `PasswordExpired`. О том, как это сделать, речь пойдет позже. Для просмотра и изменения шестнадцатеричных флагов используют операторы `Or`, `Xor` и `And` следующим образом:

- оператор `Or` — для начальной установки бита. На практике используется в случае создания новой учетной записи пользователя;
- оператор `Xor` — для переключения статуса флага. Флаг может быть активизирован и дезактивирован;
- оператор `And` — для просмотра значения, хранящегося в базе SAM.

Для просмотра значения флага используется функция `Get()`. Рассмотрим пример, в котором прочитаем значение параметра `ADS_UF_PASSWORD_NEVER_EXPIRED` — управление флагом "Password Never Expires" (листинг 8.18).

**Листинг 8.18. Чтение флагов учетной записи пользователя**

```
Set obj=GetObject("WinNT:")
        For Each str In obj
        DomainName=str.Name
        Next
Set UserName="Value"
temp=""
        Set element=GetObject("WinNT://" & DomainName & "/"& UserName)
        flag=element.Get("UserFlags")
                if (flag AnD &H10000)<>0 then
                        temp="Флаг установлен"
                else
                        temp="Флаг не установлен"
                end if
MsgBox temp
```

Для изменения значения флага используется функция `Put()`, которая имеет следующий формат: `value.Put UserFlags String`. Рассмотрим использование данной функции на примере, в котором изменим значение параметра на противоположное `ADS_UF_DONTEXPIREPASSWD` (листинг 8.19). Для того чтобы изменения вступили в силу, необходимо использовать метод `value.SetInfo`.

**Листинг 8.19. Изменение флага учетной записи пользователя**

```
Set obj=GetObject("WinNT:")
        For Each str In obj
        DomainName=str.Name
        Next
ADS_UF_DONTEXPIREPASSWD=&H0040
Set UserName="Value"
        Set element=GetObject("WinNT://" & DomainName & "/"& UserName)
element.put "userFlags", element.Get("UserFlags") Xor ADS_UF_
DONTEXPIREPASSWD
element.setinfo
        MsgBox element.get("UserFlags")
```

# Подкласс *Group*

Подкласс `Group` включает в себя два параметра: описание и SID группы. Чтение этих параметров происходит аналогичным описанному в предыдущем разделе способом. Изменение описания группы осуществляется с помощью метода `SetInfo`.

## Взаимосвязь учетных записей пользователей и групп

### Добавление и удаление учетной записи пользователя в группу

Добавление пользователя в группу осуществляется с помощью функции `Add()`, для удаления пользователя — функция `Remove`(). При использовании метода `Add()`, чтобы изменения вступили в силу, необходимо использовать метод `SetInfo` (листинг 8.20). Удаление учетной записи пользователя из группы происходит сразу после вызова метода `Remove`.

**Листинг 8.20. Добавление учетной записи пользователя в группу**

```
Set obj=GetObject("WinNT:")
       For Each str In obj
       DomainName=str.Name
       Next
Set UserName="Value_Name"
Set GroupName="Value_Group"


Set element_user=GetObject("WinNT://" & DomainName & "/"& UserName & ",
user")
Set element_group=GetObject("WinNT://" & DomainName & "/"& GroupName & ",
group")
element_group.Add(element_user.ADsPath)
element_group.SetInfo
```

Для удаления учетной записи из группы, в приведенном примере последние две строки необходимо заменить на строку:

```
element_group.Remove(element_user.ADsPath).
```

## Получение списка всех учетных записей, входящих в группу

Для перечисления всех пользователей группы, например, группы `GroupName`, используют свойство `Members` (листинг 8.21).

**Листинг 8.21. Получение списка всех учетных записей, входящих в группу**

```
Set obj=GetObject("WinNT:")
        For Each str In obj
        DomainName=str.Name
        Next
Set GroupName="Value_Group"
Set element_group=GetObject("WinNT://" & DomainName & "/"& GroupName & ",
group")
        For each obj inGroup.Members
temp=temp+Member.Name
Next
MsgBox temp
```

## Просмотр списка групп

Просмотр списка групп, к которым принадлежит пользователь, показан в листинге 8.22. Использующийся в данном примере метод `ISMember` возвращает значение типа `Boolean`, т. е. `True/False`. Сценарий можно разделить на три части: определение текущего домена, определение списка групп в домене и проверка членства в группе.

**Листинг 8.22. Просмотр списка групп, в которые входит учетная запись пользователя**

```
        Set objDomain=GetObject("WinNT: ")
        For Each domain_element In objDomain
        Domain_Name= domain_element.Name
        Next
UserName="Administrator"
Set element_user=GetObject("WinNT://" & DomainName & "/"& UserName  )
Set obj=GetObject("WinNT://" & DomainName)
obj.filter=array("group")
        For Each element In obj
Set element_group=GetObject("WinNT://" & DomainName& "/"& element.name )
```

```
                if element_group.IsMember(element_user.ADsPath)="True"
then
                        temp=temp+ element.name & chr(13)
                end if
        Next
MsgBox temp
```

Убрав из данного примера конструкцию `For Each element In obj`, можно, задав имя группы пользователя в явном виде, определять, является ли конкретный пользователь членом конкретной группы. Задачу, которая была решена в примере, можно решить иным способом: необходимо получить доступ к учетной записи пользователя и в цикле `For…Next` в качестве имени массива указать свойство `ArrayName.Groups` (листинг 8.23).

**Листинг 8.23. Быстрый способ определения членства пользователя в группе**

```
Set obj=GetObject("WinNT://" & DomainName & "/"& UserName  )
For Each element In obj.Groups
        temp = temp + element.name + chr(13)
Next
```

# Класс *Computer*

Свойства компьютера могут быть получены тем же способом, что и свойства полей для пользователя или домена.

# Подклассы PrintQueue, PrintJob, PrintJobsCollection

## Управление принтерами и очередями принтеров

Провайдер WinNT создавался для доступа к объектам для семейства Microsoft Windows NT 4.0, поэтому принтеры рассматриваются как локальные устройства, что позволяет осуществлять программное управление их очередями печати. В классе `Computer` существует два подкласса: `PrintQueue` и `PrintJob`. Для доступа к принтеру необходимо, чтобы устройство было предоставлено в общее пользование.

Список папок и устройств, предоставленных в общее пользование в домене, включая скрытые папки, можно получить, используя следующий код — листинг 8.24.

**Листинг 8.24. Определение списка устройств, предоставленных в общее пользование**

```
On error Resume Next
Set objDomain=GetObject("WinNT:")
For Each domain_element In objDomain
Domain_Name= domain_element.Name
Next
Set obj=GetObject("WinNT://" Domain_Name &" /LanmanServer,fileservice")
For Each element In obj
temp=temp+element.Name+chr(13)
Next
msgbox temp
```

Подключение к принтеру осуществляется с помощью протокола `GetObject()` по доступному сетевому имени. Хотя подключение к принтеру осуществляется как к сетевому устройству, он рассматривается как локальное устройство.

Описание объектной модели подклассов `PrintQueue` и `PrintJob` см. далее.

## Управление принтером

Управление принтером осуществляется с помощью элементов массива объекта, вызванного с помощью функции `GetObject()`. Элементы массива содержат не только строки и массивы, но и команды (табл. 8.5). Приведем пример использования свойства `Purge`, с помощью которого удаляются все задания из очереди принтера (листинг 8.25).

***Таблица 8.5.*** *Описание доступных свойств сетевых ресурсов*

| Элемент | Тип данных | Описание элемента |
|---|---|---|
| `.Pause` | Команда | Приостановить работу принтера |
| `.Purge` | Команда | Удалить все задания из очереди печати и переинициализировать принтер |
| `.Resume` | Команда | Восстановить работу принтера |

**Листинг 8.25. Удаление всех заданий из очереди указанного принтера**

```
Set pq = GetObject("WinNT://" & server_name & "/" & shares_enum)
pq.purge
```

## Просмотр состояния принтера

Чтение состояния принтера осуществляется с помощью функции `Status()` (листинг 8.26). После получения значения функцией `Status()` требуется его расшифровка (табл. 8.6). Следует отметить, алгоритм просмотра состояния пользователя очень похож на алгоритм определения состояния пользователя.

***Таблица 8.6.*** *Константы состояния очереди печати*

| Название | Значение | Описание |
|---|---|---|
| ADS_PRINTER_READY | 0x0 | Устройство готово к печати |
| ADS_PRINTER_PAUSED | 0x1 | Пауза |
| ADS_PRINTER_PENDING_DELETION | 0x2 | Удаление задания |
| ADS_PRINTER_ERROR | 0x3 | Ошибка печати |
| ADS_PRINTER_PAPER_JAM | 0x4 | Замятие бумаги |
| ADS_PRINTER_PAPER_OUT | 0x5 | Отсутствие бумаги |
| ADS_PRINTER_MANUAL_FEED | 0x6 | Ручная подача бумаги |
| ADS_PRINTER_PAPER_PROBLEM | 0x7 | Проблема с бумагой |
| ADS_PRINTER_OFFLINE | 0x8 | Устройство выключено |
| ADS_PRINTER_IO_ACTIVE | 0x100 | Загрузка задания в очередь печати |
| ADS_PRINTER_BUSY | 0x200 | Устройство занято |
| ADS_PRINTER_PRINTING | 0x400 | Идет печать |
| ADS_PRINTER_OUTPUT_BIN_FULL | 0x800 | Приемный лоток полон |
| ADS_PRINTER_NOT_AVAILABLE | 0x1000 | Устройство недоступно |
| ADS_PRINTER_WAITING | 0x2000 | Устройство в состоянии ожидания |
| ADS_PRINTER_PROCESSING | 0x4000 | Просчет задания |
| ADS_PRINTER_INITIALIZING | 0x8000 | Инициализация устройства |
| ADS_PRINTER_WARMING_UP | 0x10000 | Загрузка устройства после включения |
| ADS_PRINTER_TONER_LOW | 0x20000 | Мало тонера |
| ADS_PRINTER_NO_TONER | 0x40000 | Отсутствует тонер |
| ADS_PRINTER_PAGE_PUNT | 0x80000 | Проблемы с бумагой |
| ADS_PRINTER_USER_INTERVENTION | 0x100000 | Требуется вмешательство пользователя |

***Таблица 8.6*** *(окончание)*

| Название | Значение | Описание |
|---|---|---|
| ADS_PRINTER_OUT_OF_MEMORY | 0x200000 | Переполнение памяти |
| ADS_PRINTER_DOOR_OPEN | 0x400000 | Крышка устройства открыта |
| ADS_PRINTER_SERVER_UNKNOWN | 0x800000 | Неизвестный сервер |
| ADS_PRINTER_POWER_SAVE | 0x1000000 | Устройство в состоянии энергосбережения |

**Листинг 8.26. Определение статуса принтера**

```
Set objDomain=GetObject("WinNT:")
For Each domain_element In objDomain
       Domain_Name= domain_element.Name
Next
Set PC_Name="_______"
Set Share_Name="_______"
Set element=GetObject("WinNT://" & Domain_Name &"/" & PC_Name &"/" &
Share_Name)
flag=element.status
MsgBox flag
```

Значение переменной `flag` будет выдано в десятичном виде, поэтому его необходимо перевести в шестнадцатеричную систему.

## Чтение свойств заданий в очереди принтера

Очередь печати представляет собой массив, содержащий задания, которые находятся в очереди печати на момент извлечения из нее данных с помощью протокола WinNT (листинг 8.27). Список и описание свойств параметров очереди см. в *разд. "Объектная модель провайдера WinNT", табл. 8.1 и 8.2*.

**Листинг 8.27. Чтение свойств заданий в очереди принтера**

```
Set objDomain=GetObject("WinNT: ")
       For Each domain_element In objDomain
       Domain_Name= domain_element.Name
       Next
Shares_Name="Value"
```

```
        Set pq = GetObject("WinNT://" & Domain_Name & "/" & Shares_Name)
        For Each printJob In pq.PrintJobs
status_pre=printJob.status
        select case status_pre
                case "0" status_="Нормально"
                case "1" status_="Пауза"
                case "18" status_="Ошибка"
        end select
summary = summary & "Номер докумета: " & number_docum & chr(13) & chr(13)
& "Статус: " & status_pre & chr(13) & "Приоритет: " & print-
Job.Description & chr(13) & "Пользователь: " & printJob.User & chr(13) &
"Всего стр. " & printJob.TotalPages & chr(13) & "Размер, (Mb) " &
round(printJob.Size/1000000,2) & chr(13) & "Статус: " & status_pre &
chr(13) & chr(13)
        Next
Wscript.Echo summary
```

## Управление очередью печати

Подкласс `PrintJobsCollection` — управление очередью осуществляется тем же способом, что и управление принтером. Существует три команды, которые могут быть использованы для управления документом, находящимся в очереди печати (табл. 8.7).

***Таблица 8.7.*** *Описание доступных свойств очереди печати*

| Элемент | Тип данных | Описание элемента |
|---|---|---|
| `Pause` | Команда | Приостановить печать задания |
| `Remove` | Команда | Удалить задание из очереди печати |
| `Resume` | Команда | Восстановить печать задания |

Пример, в котором удаляется второе задание из очереди печати, показан в листинге 8.28. Если задание № 2 отсутствует, то ошибка обрабатывается с помощью выражения `On Error Resume Next`.

**Листинг 8.28. Удаление задания из очереди принтера**

```
On Error Resume Next
Set objDomain=GetObject("WinNT: ")
        For Each domain_element In objDomain
                Domain_Name= domain_element.Name
```

```
        Next
Shares_Name="Value"
Set pq = GetObject("WinNT://" & Domain_Name & "/" & Shares_Name)
For Each printJob In pq.PrintJobs
        If (number_docum=2) then
                printJob.remove
        end if
Next
```

# Класс *Service*

## Подкласс *FileService* и *FileShare*

Подклассы `FileService` и `FileShare` являются дочерними для класса `Service`, причем `FileShare` является дочерним для `FileService`. Поскольку два подкласса тесно связаны между собой, то их необходимо рассматривать вместе. Используя эти классы, программно управляют безопасностью и предоставляют доступ к файлам и каталогам.

Для управления совместно используемыми ресурсами применяется контейнер `LanmanServer`.

Рассмотрим чтение свойств ресурсов, создание и удаление совместно используемых ресурсов.

## Чтение свойств совместно используемых ресурсов

Чтение свойств и назначение новых значений параметров осуществляется ранее описанным методом. Пример, в котором считывается и выводится на экран описание ресурса, затем осуществляется смена описания ресурса, приведен в листинге 8.29.

**Листинг 8.29. Изменение поля `Description` сетевой папки**

```
Set objDomain=GetObject("WinNT:")
        For Each domain_element In objDomain
                Domain_Name= domain_element.Name
        Next
Set PC_Name="_______"
Set Share_Name="_______"
Set New_Description_Name="_______"
```

```
Set element=GetObject("WinNT://" & Domain_Name &"/" & PC_Name
&"/LanmanServer/" & Share_Name)
temp="Old Description: " + Element.Description+chr(13)
Element.Description = New_Description_Name
Element.SetInfo
temp="New Description " + Element.Description
msgbox temp
```

## Программное создание и удаление совместно используемого ресурса

Создание совместно используемого ресурса реализуется с помощью метода `Create`. В его свойствах указывается тип создаваемого ресурса, в данном случае `fileshare`, и название ресурса (`ShareName`). Метод `Create` обязательно сопровождается методом `Path`, с помощью которого задается путь к ресурсу, и методом `SetInfo`. В листинге 8.30 показан пример предоставления в общее пользование папки, локальный путь к которой `c:\folder001`. Сетевой путь папки должен быть `\\1000pc\Share1`. Описание папки — `Shared Folder #1`.

**Листинг 8.30. Создание сетевой папки**

```
Set objDomain=GetObject("WinNT:")
       For Each domain_element In objDomain
              Domain_Name= domain_element.Name
       Next
Set PC_Name="1000pc"
Set Share_Name="Share1"
Set Folder_Path="c:\Folder1"
Set Description_Name="Shared Folder #1"
Set object=GetObject("WinNT://" & Domain_Name &"/" & PC_Name
&"/LanmanServer")
Set element=object.Create("fileshare", Share_Name)
element.Path= Folder_Path
element.Description= Description_Name
element.MaxUserCount =10
element.SetInfo
```

Для удаления используемого ресурса вместо метода `Create` используют метод `Delete`. Изменения вступают в силу немедленно (листинг 8.31).

**Листинг 8.31. Удаление сетевого доступа к папке**

```
Set objDomain=GetObject("WinNT:")
       For Each domain_element In objDomain
              Domain_Name= domain_element.Name
       Next
Set PC_Name="1000pc"
Set Share_Name="Share1"
Set Folder_Path="c:\Folder1"
Set Description_Name="Shared Folder #1"

Set object=GetObject("WinNT://" & Domain_Name &"/" & PC_Name
&"/LanmanServer")
Call object.Delete("fileshare", Share_Name)
```

# Подкласс *Service*

С помощью данного подкласса осуществляется управление различными службами. С помощью данного подкласс, могут быть осуществлены следующие действия, касающиеся служб: перечисление служб, установленных на локальном или удаленном компьютере; чтение свойств выбранной службы; управление службой.

## Перечисление и чтение свойств служб на выбранном компьютере

Перечисление служб на рабочей станции осуществляется с помощью фильтра `Service` (листинг 8.32). В примере осуществляется чтение свойства `Name`.

**Листинг 8.32. Формирование списка служб на выбранном компьютере**

```
On Error Resume Next
Set objDomain=GetObject("WinNT:")
       For Each domain_element In objDomain
              Domain_Name= domain_element.Name
       Next
Set Service_Name="______"
Set PC_Name="______"
Set object=GetObject("WinNT://" & Domain_Name &"/" & PC_Name &",Computer")
```

```
Object.Filter=Array("Service")
For Each Serv in Object
Wscript.Echo Serv.Name
Next
```

## Связывание служб на выбранном компьютере

Понятие "связывание служб" лучше всего продемонстрировать на реальном примере. Представьте, что служба 1 связана, т. е. является зависимой от службы 2. Это обозначает, что при остановке службы 1 появится сообщение о необходимости остановки службы 2. Связанность служб характеризуется свойством `Dependencies`. Свойство `Dependencies` является массивом (листинг 8.33).

**Листинг 8.33. Определение "связанной" службы**

```
On Error Resume Next
Set objDomain=GetObject("WinNT:")
       For Each domain_element In objDomain
              Domain_Name= domain_element.Name
       Next
Set Service_Name="______"
Set PC_Name="______"
Set object=GetObject("WinNT://" & Domain_Name &"/" & PC_Name
&",Computer")
Set Service=object.GetObject("service", Service_Name)
```

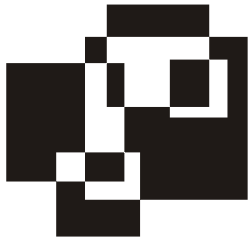
Вторым этапом является установка новой зависимой службы (листинг 8.34).

**Листинг 8.34. Установка новой взаимосвязи между службами**

```
Flag1=0
Flag2=0
Dim New_Array()       ' Объявление пустого массива
Set Dependency_Name="_______"
If IsArray(Service.Dependencies)=True Then
For Each obj in Service.Dependencies
i=Ubound(New_Array)+1 ' Определение верхней границы массива Dependencies
ReDim Preserve New_Array (i)        ' Переопределение размера массива
New_Arrray
New_Array(i)=obj
```

```
       If obj="" then
              Flag1=1
       end if
       If  obj= Dependency_Name then
              Flag2=1
       end if

if Flag1=1 then
                              Service.dependencies=Array(Dependency_Name)
                              Service.SetInfo
Else
       If  Flag2<>1
                              i=Ubound(New_Array)+1
                              ReDim Preserve New_Array (i)
                              New_Array(i)= Dependency_Name
                              Service.dependencies= New_Array
                              Service.SetInfo
       End if
End if
Else
       If Service.dependencies <> Dependency_Name then
              Service.dependencies =array(Service.dependencies,
Dependency_Name)
              Service.SetInfo
       End if
End if
```



# Глава 9

# Программное управление ADSI: LDAP

## Структура объектной модели провайдера LDAP

Для программного управления Active Directory с помощью провайдера LDAP необходимо использовать его объектную модель. Она представляет собой совокупность объектов, которые взаимосвязаны друг с другом и образуют между собой иерархическую структуру. Каждый из этих объектов имеет набор свойств, характерный исключительно для объектов данного типа. Существует несколько типов (идентификаторов) объектов: `CN`, `DC`, `OU` (табл. 9.1).

***Таблица 9.1.*** *Типы идентификаторов объектов провайдера LDAP*

| Сокращение | Описание |
|---|---|
| DC (Domain Component) | Метка доменного имени |
| OU (Organization Unit) | Подразделение — организационная единица |
| CN (Соёmmon Name) | Идентификатор пользователя |

## Имена LDAP URL

Имена LDAP URL (см. RFC 1779, RFC 2247) построены на основе протокола X.500 и используются для связывания с объектами.

Идентификаторы объектов `DC`, `OU`, `CN` образуют полное составное имя — `DN` (Distinguished Name), а имя самого объекта — относительное составное имя — `RDN` (Relative Distinguished Name).

Полное составное имя объекта включает в себя имя объекта и всех его родителей, начиная с корня домена.

Существует две формы доступа к ADSI: развернутая и сокращенная. Рассмотрим принципы построения путей к ресурсу обоими способами на примере домена `domain.com` (рис. 9.1).

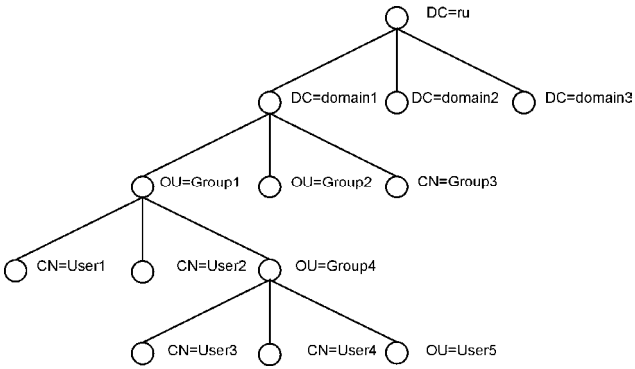



**Рис. 9.1.** Древовидная структура AD

# Развернутая форма записи

При развернутой форме записи строка связывания начинается с описания верхнего элемента структуры. Затем происходит переход вниз по иерархии (листинг 9.1). Следует помнить, что при написании пути к объекту необходимо исключать пробелы.

**Листинг 9.1. Шаблон доступа к объектам AD с помощью провайдера LDAP. Полная форма записи**

```
Set obj = GetObject ("LDAP://DC=Domain_name1/DC=Domain_name2/
DC=Domain_name3/ OU= OU_Name_Level1/OU=OU_Name_Level2…/
OU=OU_Name_Levelμ /CN=CN_Name")
```

`DC=Domain_name1/DC=Domain_name2/DC=Domain_name3` образуют полное имя контроллера домена, элементы `OU=OU_Name_Level1/OU=OU_Name_Level2…/OU=OU_Name_Levelμ` представляют собой вложенные друг в друга элементы. В развернутой форме доступа объект `CN` является "дном колодца" в иерархии.

В листинге 9.2 показан пример запроса к объекту `CN=User3` с помощью провайдера LDAP в развернутой форме доступа (см. рис. 9.1).

**Листинг 9.2. Запрос к объекту `CN=User3`. Полная форма записи**

```
Set obj = GetObject ("LDAP://DC=RU/DC=Domain1/OU=
Group1/OU=Group4/CN=User3")
```

# Сокращенная форма записи

Сокращенная форма записи характеризуется тем, что строка запроса строится в соответствии с обратной иерархией структуры организации (листинг 9.3).

**Листинг 9.3. Шаблон доступа к объектам AD с помощью провайдера LDAP. Сокращенная форма записи**

```
Set obj=GetObject("LDAP://CN=CN_Name,OU=OU_Name_Level μ …,
OU=OU_Name_Level2,OU=OU_Name_Level1/ DC=…")
```

Запрос к объекту `CN=User3` с помощью сокращенной формы доступа показан в листинге 9.4.

**Листинг 9.4. Запрос к объекту `CN=User3`. Сокращенная форма записи**

```
Set obj=GetObject("LDAP://CN=User3,OU=Group4, OU=Group3,DC=Domain1, dc=RU")
```

# Определение имени домена. Обзор способов

С помощью провайдера WinNT можно определить список доступных доменов, в то время как с помощью LDAP — текущего. Поскольку провайдеры WinNT и LDAP используются в связке, то необходимо создать механизм вычленения короткого имени домена из длинного. Основываясь на описанной ранее информации, приведем пример определения длинного имени домена (LDAP) и преобразования его в короткое имя (совместимое с WinNT) — листинг 9.5.

**Листинг 9.5. Определение короткого и длинного доменного имени с помощью провайдера LDAP**

```
Set rootDSE_ = GetObject("LDAP://RootDSE")
D_def=rootDSE_.Get("defaultNamingContext")
Long_Ldap_name = "LDAP://" + d_def
Short_winnt_name= Mid(d_def, instr(d_def,"=")+
1,instr(d_def,",")-instr(d_def,"=")-1)
Wscript.Echo Long_Ldap_name
Wscript.Echo Short_Wnnt_name
```

Используя встроенный в ОС Windows 2k объект `ADSystemInfo`, можно определить не только короткое имя и DNS-имя домена с помощью вызова свойств `DomainShortName` и `DomainNDSName`, но и другие важные характеристики. Среди таких свойств имя компьютера, имя текущего сайта, имя текущего пользователя. Все имена приведены в LDAP-формате (листинг 9.6).

**Листинг 9.6. Определение короткого и длинного доменного имени с помощью встроенной библиотеки `ADSystemInfo`**

```
Set objSysInfo = CreateObject("ADSystemInfo")
WScript.Echo objSysInfo.ComputerName        ' имя компьютера
wscript.Echo objSysInfo.DomainShortName     ' короткое имя домена
wscript.Echo objSysInfo.DomainDNSName       ' DNS-имя компьютера
wscript.Echo objSysInfo.ForestDNSName       ' DNS-имя леса
wscript.Echo objSysInfo.SiteName            ' имя сайта
wscript.Echo objSysInfo.UserName            ' имя компьютера
```

# Объекты Active Directory

Active Directory поддерживает несколько типов объектов, каждый из которых относится как правило к набору классов. Например, объект `Computer` относится к `objectclass=User` и `objectclass=Computer`. Типы объектов и соответствующие им классы приведены в табл. 9.2.

***Таблица 9.2.*** *Соответствия объектов классам AD*

| Тип объекта | Значение objectclass | Тип объекта | Значение objectclass |
|---|---|---|---|
| Computer | Top | OU | Top |
| | Person | | Organizational Person |
| | Organizational Person | Printer | Top |
| | User | | Leaf |
| | Computer | | ConnectionPoint |
| Contact | Top | | PrintQueue |
| | Person | SharedFolder | Top |
| | OrganizationalPerson | | Leaf |
| | Contact | | ConnectionPoint |

***Таблица 9.2*** *(окончание)*

| Тип объекта | Значение objectclass | Тип объекта | Значение objectclass |
|---|---|---|---|
| Group | Top | | Volume |
| | Group | User | Top |
| InetOrgPerson | Top | | Person |
| | Person | | Organizational-Person |
| | OrganizationalPerson | | User |
| | User | | |
| | InetOrgPerson | | |

## Поиск объектов в AD

При создании фильтра SQL-запроса, осуществляющего поиск тех или иных объектов в AD, необходимо следить за совпадением классов. Например, если вы в поиске укажете `objectClass='User'`, то результатом поиска будет не список пользователей домена, а список объектов, относящихся к типу: `Computer`, `InetOrgPerson`, `User`. Приведем несколько часто используемых шаблонов поиска.

## Поиск всех учетных записей пользователей в домене

Сценарий работает по следующему алгоритму (листинг 9.7). На первом этапе осуществляется определение имени текущего домена. Затем соединение с AD с помощью ADODB-соединения. Далее формируется SQL-запрос, с помощью которого получают значение поля `samaccounname`. В фильтре необходимо указать принадлежность искомого объекта к классу `User`, но не к классу `Computer`. После считывания данных в переменную `temp`, осуществляется их вывод на экран.

**Листинг 9.7. Формирование списка учетных записей пользователей в домене**

```
' определение текущего домена
domain="LDAP://" + GetObject("LDAP://RootDSE").rootDSE_
.Get("defaultNamingContext")


' соединение с AD через ADODB
```

```
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.CommandTimeout = 120
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection
' SQL-запрос

set sql=objconnection.execute("SELECT samaccountname FROM '" & Domain & "
' WHERE objectClass='user' and NOT objectClass='computer'" )

objCommand.properties("Page size)=10000
objCommand.properties("Timeout")=300
objCommand.properties("Cache Results")=false

sql.Movefirst
temp=""
' Чтение данных
Do Until st.EOF
       temp=temp+St.Fields(0).Value ' или cn=St.Fields("").Value+chr(13)
sql.MoveNext
Loop
MsgBox temp
```

Рассмотрим подробнее каждый из типов объектов.

# Объектная модель провайдера LDAP

Описывать очень подробно объектную модель провайдера LDAP не имеет смысла, поскольку существует несколько очень хороших программ, предназначенных для просмотра объектной модели каталога: Active Directory Viewer (Microsoft) и LDAP Browser 2.6 (Softerra).

# Active Directory Viewer (Microsoft)

Active Directory Viewer (ADV) — графическая утилита, позволяющая выполнять операции чтения, модифицирования, осуществлять поиск в любых совместимых каталогах, таких как Active Directory, Exchange Server, Netscape Directory, Netware Directory.

Active Directory Viewer входит в состав SDK для Active Directory Services Interface, который можно бесплатно загрузить с сайта Microsoft: **http://www.microsoft.com/ntserver/nts/downloads/other/adsi25**.

После установки SDK for ADSI утилита Active Directory Viewer (AdsVw.exe) будет находиться в c:\Program Files\Microsoft\ADSI Resource Kit, Samples and Utilities\ADsVw.

Программа работает в двух режимах: **ObjectViewer** и **Query** (рис. 9.2). Для просмотра объектной модели какого-либо провайдера необходимо использовать режим **ObjectViewer**. Режим **Query** используется для осуществления поиска объектов в выбранной объектной модели. В данной книге режим **Query** рассматриваться не будет.

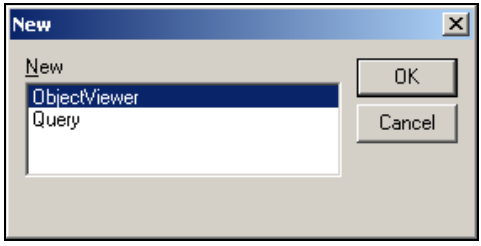



**Рис. 9.2.** Доступные режимы работы Active Directory Viewer

## Просмотр и редактирование объектной модели программой ADV в режиме *ObjectViewer*

После выбора режима работы **ObjectViewer**, появится диалоговое окно (рис. 9.3). Для получения доступа к каталогу необходимо указать путь к каталогу и параметры учетной записи, обладающей правами администратора (имя и пароль). Путь к каталогу должен быть построен в соответствии со следующим шаблоном:

```
<Provider_Name:>//<Server_Name>/<Full_Domain_Name>
```

Обратите внимание на две особенности получения доступа к AD:

- при вводе пути в поле **Enter ADs path** не должно быть пробелов;
- "слэши" должны быть прямыми — "/".

Невыполнение хотя бы одного из перечисленных условий приведет к ошибке в соединении с каталогом. Для доступа к серверу `server` домена `domain.ru` с помощью протокола LDAP используется следующий запрос: `LDAP://server/DC=domain,DC=ru`. После соединения с каталогом на экране будет отображена его иерархическая структура (рис. 9.4). В левой части экрана отображается

иерархическая структура каталога, в правой части — характеристики объекта, на котором установлен курсор. Список свойств объекта и соответствующих им значений приведен в полях **Properties** и **Property Value**.

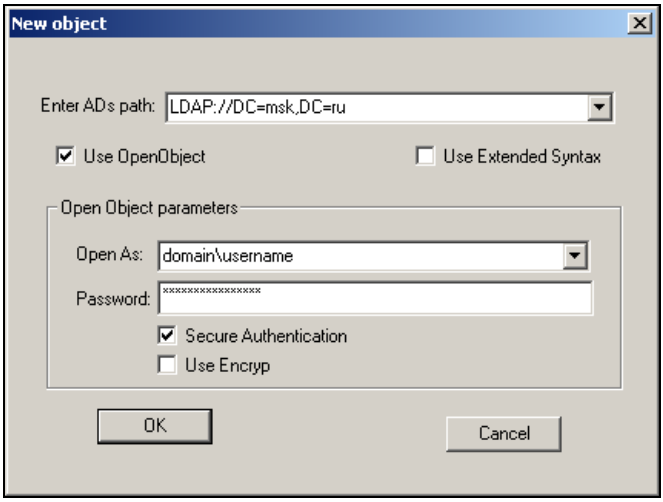



**Рис. 9.3.** Получение доступа к каталогу

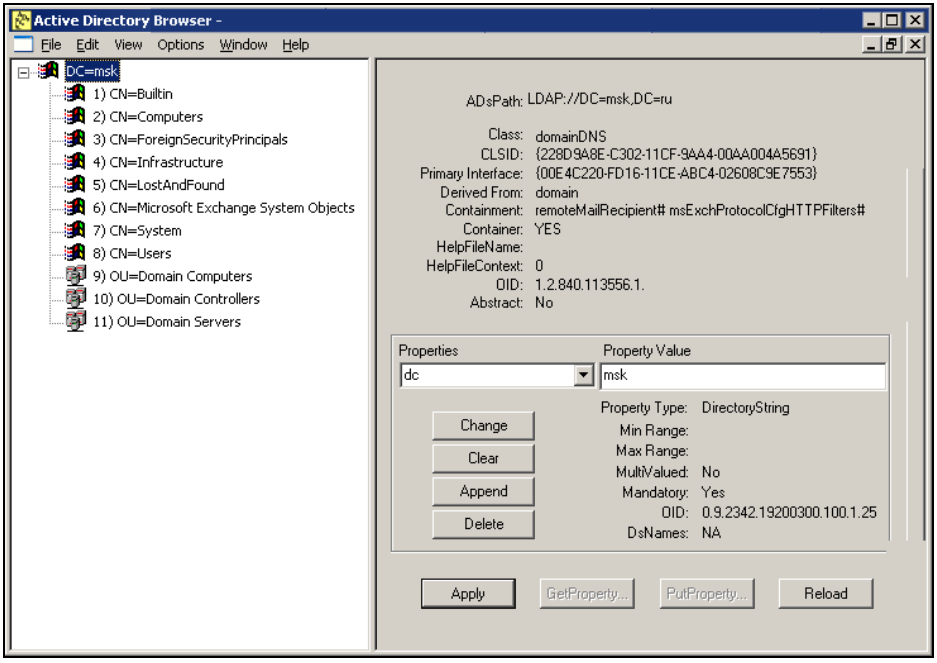



**Рис. 9.4.** Общий вид программы Active Directory Viewer

С помощью кнопок **Change**, **Clear**, **Append**, **Delete** можно изменять объектную модель каталога — изменять, удалять, добавлять поля в свойствах объектов.

# LDAP Browser 2.6. (Softerra)

LDAP Browser 2.6 является бесплатной программой (**http://www.ldapadministrator.com**). По своим возможностям программа превосходит Active Directory Viewer, в использовании LDAP Browser гораздо удобнее. В процессе создания соединения с каталогом могут быть заданы фильтры, параметры административной учетной записи, номер TCP-порта и др. Настройка программы нетривиальна, поэтому остановимся на данном вопросе подробнее. После запуска программы в ней уже присутствуют три тестовых домена, которые вряд ли пригодятся, поэтому их можно удалить. Для получения доступа к интересующему домену необходимо запустить соответствующий мастер, нажав <Ctrl>+<N> или выбрав в контекстном меню **Browse Root | New Profile** (рис. 9.5). Далее пошагово отвечайте на вопросы, задаваемые мастером настройки соединения.

Ввести название нового профиля. Рекомендуется для удобства сделать его совпадающим с названием домена. По умолчанию предлагается имя `New Profile` (рис. 9.6).

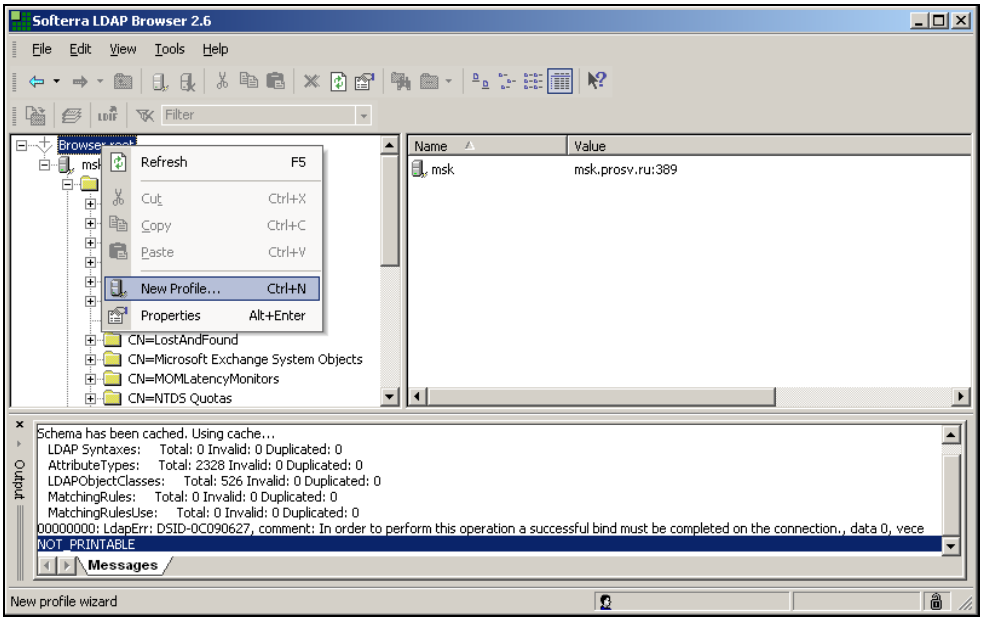



**Рис. 9.5.** Создание нового профиля




**Рис. 9.6.** Шаг 1

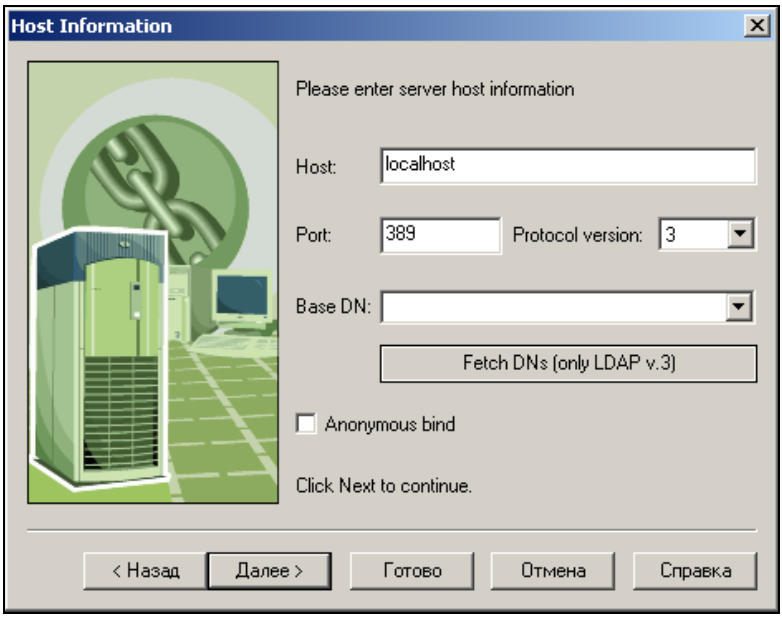



**Рис. 9.7.** Шаг 2

Если программа устанавливается на контроллере домена, то значение параметра **Host** не надо изменять (по умолчанию `localhost`). Если же программа установлена на локальном компьютере, то значение параметра **Host** — имя домена в соответствии с RFC 1123, например, `www.domain.ru`. Остальные параметры можно оставить без изменений (рис. 9.7).

После нажатия кнопки **Далее**, мастер выведет сообщение о том, что параметр **Base DN** не задан (см. рис. 9.7). Необходимо продолжить работу мастера, нажав кнопку **Yes** (рис. 9.8).

Мастер предлагает ввести имя и пароль учетной записи, от имени которой будет осуществляться доступ к AD. В поле **User DN** введите учетное имя пользователя в сети (login без указания домена, например, `User101`), в поле **Password** — пароль. Рекомендуется включить опцию **Save password** (рис. 9.9).

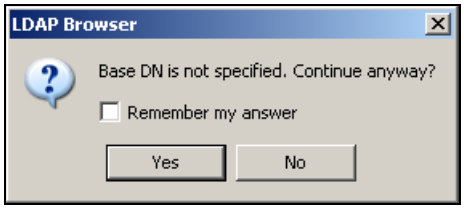



**Рис. 9.8.** Шаг 3

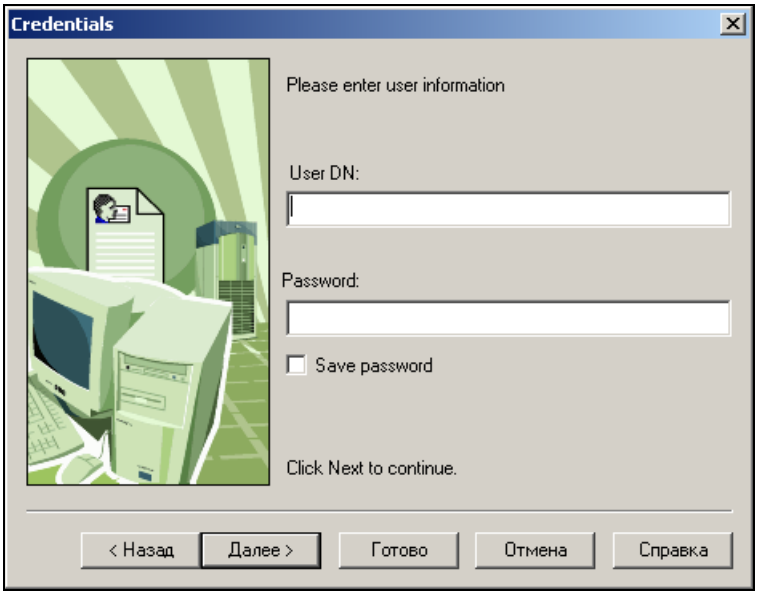



**Рис. 9.9.** Шаг 4

Если была установлена опция **Save password**, то на следующем шаге мастер запросит подтверждение пароля (рис. 9.10).

Последний шаг — настройка фильтра и временных характеристик. Не рекомендуется изменять значения настроек соединения, за исключением поля **Entry count limit** (рис. 9.11). Практика показывает, что количество объектов часто превышает 1000, поэтому рекомендуется ставить значение больше в несколько раз. Например — 10 000.

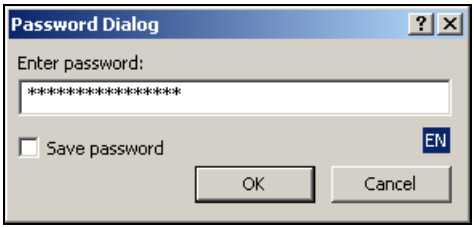



**Рис. 9.10.** Шаг 5

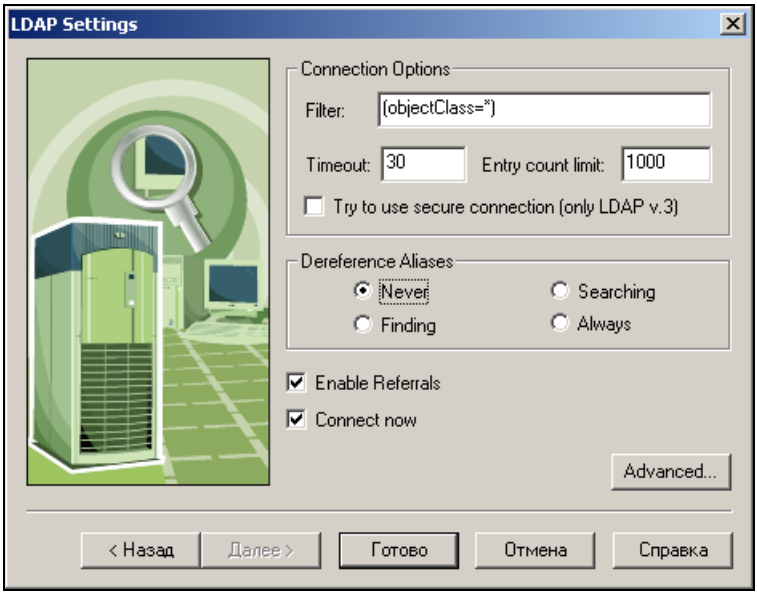



**Рис. 9.11.** Шаг 6

После завершения работы мастера (рис. 9.12) будет предпринята попытка подключения к домену. В случае успеха — программа отобразит структуру каталога подключенного домена.

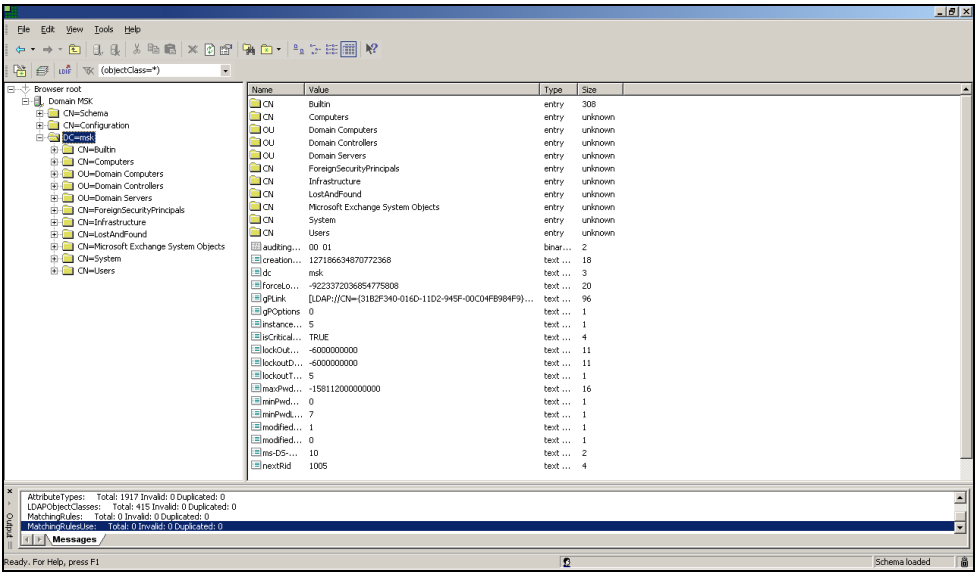



**Рис. 9.12.** Общий вид программы LDAP Browser

# Различия провайдеров LDAP и WinNT

Провайдер LDAP (Lightweight Directory Access Protocol) рассматривает принтер как сетевое устройство, в то время как провайдер WinNT рассматривает его исключительно как локальное устройство. Использование обоих провайдеров при работе с принтерами позволяет полностью управлять принтерами.

Вторым принципиальным отличием провайдеров являются расширенные возможности поиска провайдера LDAP. Используя провайдер WinNT, можно было осуществлять поиск, пользуясь фильтром, принадлежащим к одному из классов — `computer`, `user`, `service` и др. Провайдер позволяет искать объект, при этом не обязательно указывать класс, к которому относится объект. По нахождению объекта осуществляется чтение его свойств, включая местоположение объекта в AD, класс, к которому относится объект, и другие параметры.

# Выполнение команд и программ от имени определенного пользователя

Для управления Active Directory необходимо обладать административными привилегиями.

Для полноценного управления AD, например, с помощью сценария, необходимо войти в систему с привилегиями системного администратора или, используя административные средства, задать альтернативные имя и пароль пользователя. Еще один вариант, о котором речь пойдет немного позже, — задать параметры (имя и пароль) системного администратора непосредственно в скрипте, однако это небезопасно и требует шифрования скрипта.

В качестве альтернативного способа доступа к AD можно использовать утилиту RunAs.exe, которая устанавливается вместе с операционной системой. Одним из рекомендуемых решений является создание командного файла с расширением bat, который будет запускать утилиту Cmd.exe от имени системного администратора (листинг 9.8).

**Листинг 9.8. Запуск приложения с привилегиями администратора**

```
[cmd_admin.bat]
@echo=off
runas /user:administrator@domain.com cmd.exe
```

После запуска файла cmd.exe появится приглашение ввести пароль системного администратора и, в том случае, если пароль будет введен правильно, в появившемся окне можно будет вводить различные команды, которые будут выполняться с правами администратора.

Альтернативной возможностью является запуск приложения с помощью выбора **Run As** в контекстном меню. Щелкните правой кнопкой мыши по ярлыку запускаемого приложения, удерживая клавишу <Shift>. В появившемся контекстном меню выберите **Запуск от имени**, или **Run As**.

## Выполнение сценариев от имени конкретного пользователя

Нередко возникает необходимость выполнить сценарий от имени другой учетной записи, которая имеет расширенный набор прав. Некоторые из возможных способов только что были описаны, однако их использование не всегда удобно. Существует еще одна возможность — жестко задать имя и пароль пользователя в сценарии. Очевидно, что это не самое удачное решение, поскольку файл сценария могут просмотреть другие пользователи, однако использовать такое решение на стадии отладки скрипта очень удобно. В листинге 9.9 показано, как это сделать с помощью ADSI и ADO.

**Листинг 9.9. Пример публикации пароля системного администратора в теле скрипта**

```
' Определение имени домена
Set domain = "LDAP://" + GetObject("LDAP://RootDSE").
Get("defaultNamingContext")

'AdoDB Connection
Set objConnection = CreateObject("ADODB.Connection")
objConnection.Provider = "ADsDSOObject"
objConnection.Properties("ADSI Flag")=1+2

'Имя администратора
objConnection.Properties("User ID")="domain\admin_name"

'Пароль администратора
objConnection.Properties("Password")="password"

objConnection.Properties("Encrypt Password")=TRUE
objConnection.Open "Active Directory Provider"

objConnection.CommandTimeout = 120
Set objCommand = CreateObject("ADODB.Command")
Set objCommand.ActiveConnection = objConnection

set rootDSE_ = GetObject("LDAP://RootDSE")
domain_ldap = "LDAP://" + rootDSE_.Get("defaultNamingContext")

set zero_point=objconnection.execute("SELECT * FROM '" & Domain & " '  WHERE …
```

# Импорт и экспорт данных из AD

До появления Windows Server 2003 поддержка изменений в AD с помощью командной строки была очень слаба. Существовавшая утилита dsmod.exe справлялась со своей задачей, поскольку набор модифицируемых объектов был ограничен. В настоящее время существует две утилиты: ldifde.exe и csvde.exe. С помощью первой можно пересылать данные из одного каталога

в другой и изменять информацию внутри AD. С помощью другой — извлекать данные, для использования в других приложениях.

В некоторых ситуациях возникает потребность описать изменения, которые необходимо внести в AD в удобном текстовом формате, который можно отредактировать, а затем передать его AD для внесения изменений в каталог. Для реализации такой возможности был специально разработан стандарт LDIF (RFC 2849) — формат обмена данными LDAP. Он позволяет описывать операции добавления, изменения и удаления данных из AD в текстовый файл, откуда ее можно импортировать обратно в AD с помощью специальных средств. Поддержка функций импорта и экспорта реализована в утилите ldifde.exe, которая появилась еще в Windows 2000.

## LDIF-файлы

LDIF-файл состоит из блоков записей, каждая из которых определяет одну из операций — удаления, обновления или добавления. В первой строке записи указано имя объекта, во второй тип — `add`, `modify`, `delete`. Блоки отделяются строкой, содержащей единственный символ — знак минуса (-).

Приведем примеры использования LDIP-файлов (листинги 9.10—9.12).

**Листинг 9.10. Создание учетной записи пользователя с помощью LDIF-файла**

```
[user_add.ldif]

dn: cn=AIvanov,cn=users,dc=domain,dc=ru
changetype: add
objectClass: user
samaccountname: AIvanov
```

**Листинг 9.11. Изменение свойств учетной записи пользователя с помощью LDIF-файла**

```
[user_change.ldif]

dn: cn=AIvanov,cn=users,dc=domain,dc=ru
changetype: modify
add: givenName
givenName: Artem
```

**Листинг 9.12. Удаление учетной записи пользователя с помощью LDIF-файла**

```
[user_delete.ldif]

dn: cn=AIvanov,cn=users,dc=domain,dc=ru
changetype: delete
objectClass: user
samaccountname: AIvanov
```

## CSVDE-файлы

CSVDE-файлы представляют собой текстовые файлы, содержимое которых пригодно для экспорта в различные приложения, такие как Excel, поскольку значения параметров разделены символом точка с запятой (`;`).

В Active Directory существует несколько типов объектов. В скриптах чаще всего осуществляются различные операции с такими объектами, как `User` и `Group`. С объектами можно производить следующие манипуляции: создавать, удалять, считывать свойства, изменять их.

# Действия над объектами

## Создание объекта

Для создания объекта в Active Directory следует указать путь, имя и тип создаваемого объекта. Кроме того, сценарий должен быть выполнен от имени системного администратора.

Рассмотрим создание объекта на примере учетной записи пользователя. Сценарий работает по следующему алгоритму. С помощью функции `GetObject()` получают доступ к заданному контейнеру в AD, в котором будет создан объект. Затем, используя функцию `Create()`, одним из аргументов которой является класс объекта (в данном случае `User`), создают объект. Задав дополнительные свойства методом `Put`, сохраняют сделанные изменения в каталоге с помощью метода `SetInfo`.

**Листинг 9.13, *а*. Создание объекта `OU` (папка)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain = GetObject("LDAP://" & oRoot.Get("defaultNamingContext"))
Set oOU=oDomain.Create("organizationalUnit", "ou=Дирекция")
oOU.Put "Description", "Описание папки"
oOU.SetInfo
```

Сценарием в листинге 9.13, *а* создается папка `Дирекция` с описанием в корне каталога.

**Листинг 9.13, *б*. Создание объекта `User` (учетная запись пользователя)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain=GetObject("LDAP://ou=Дирекция,"&
oRoot.Get("defaultNamingContext"))
Set oUser = oDomain.Create("User", "cn=Ivanov,Ivan")
oUser.Put "sAMAccountName", "IIvanov"
oUser.Put "Description", "Иванов Иван Иванович"
oUser.SetInfo
oUser.SetPassword "1234567890"
oUser.AccountDisabled = True
oUser.SetInfo
```

В папке `Дирекция` создается учетная запись пользователя `Иванов Иван Иванович` (листинг 9.13, *б*). Учетная запись выключена, ей назначен пароль `1234567890`.

В Active Directory Windows 2000 для совместимости с доменами Windows NT должен быть обязательно определен атрибут `sAMAccountName`. Если в Windows Server 2003 не задано значение `sAMAccountName`, оно будет заполнено автоматически.

В Windows Server 2003 можно также создавать учетные записи пользователей, используя класс `InetOrgPerson` (листинг 10.13, *в*), который применяется во многих LDAP-каталогах для представления пользователей (см. RFC 2798).

**Листинг 9.13, *в*. Создание объекта `InetOrgPerson` (учетная запись пользователя)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain=GetObject("LDAP://ou=Дирекция,"&
oRoot.Get("defaultNamingContext"))
Set objUser =oDomain.Create("InetOrgPerson","CN= Ivanov, Ivan")
ClassArray=Array("InetOrgPerson","person","top","organizationPerson")
objUser.Put "objectClass", ClassArray
objUser.Put "description", "Иванов Иван Иванович"
objUser.SetInfo
```

**Листинг 9.13, *г*. Создание объекта `Group` (группа)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain=GetObject("LDAP://ou=Дирекция,"&
oRoot.Get("defaultNamingContext"))
Set oGroup = oDomain.Create("Group", "cn=AdminDepartment")
oGroup.Put "sAMAccountName", "AdminDepartment"
oGroup.Put "Description", "Описание группы"
oGroup.SetInfo
```

В папке `Дирекция` создается группа с описанием (листинг 9.13, *г*).

В Active Directory Windows 2000 для совместимости с доменами Windows NT должен быть обязательно определен атрибут `sAMAccountName`. Если в Windows Server 2003 не задано значение `sAMAccountName`, оно будет заполнено автоматически.

**Листинг 9.13, *д*. Создание объекта `volume` (опубликованная папка)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain=GetObject("LDAP://ou=Дирекция,"&
oRoot.Get("defaultNamingContext"))
Set oVol = oDomain.Create("volume", "cn=DiskX")
oVol.Put "uncName", "\\server\share"
oVol.Put "Description", "Описание сетевой папки"
oVol.SetInfo
```

В папке `Дирекция` создается опубликованная папка с описанием, которой соответствует UNC-путь `\\server\share` (листинг 9.13, *д*).

## Удаление объекта

Для удаления объекта, например папки (`OU`), используется метод `Delete`, первым атрибутом которого является название класса (табл. 9.3), к которому относится удаляемый объект; вторым — имя и значение одного из полей.

***Таблица 9.3.*** *Соответствия аргумента функции `Create()` типу создаваемого объекта*

| **Тип объекта** | **Первый аргумент функции `Create()`** |
|---|---|
| `Computer` | `Person` |
| `Contact` | `Contact` |

***Таблица 9.3*** *(окончание)*

| Тип объекта | Первый аргумент функции `Create()` |
|---|---|
| Group | Group |
| InetOrgPerson | InetOrgPerson |
| OU | OrganizationalUnit |
| Printer | PrintQueue |
| SharedFolder | Volume |
| User | User |

**Листинг 9.14. Удаление объекта (на примере OU)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain=GetObject("LDAP://ou=Дирекция,"&
oRoot.Get("defaultNamingContext"))
Set objGroup =oDomain.delete("organizationalUnit","ou=temp")
```

В приведенном в листинге 9.14 примере осуществляется удаление папки `TEMP`, созданной в папке `Дирекция`. После применения метода `delete` метод `SetInfo` не используют. Удаление объектов других типов осуществляется аналогично.

## Перемещение объектов

Перемещение объекта в AD осуществляется с помощью метода `MoveHere` с предварительным получением доступа к папке (`OU`), куда нужно переместить объект, и перемещаемому объекту (листинг 9.15).

**Листинг 9.15. Перемещение объекта (на примере `User`)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oUser = GetObject("LDAP://cn=Ivanov,Ivan,"&
oRoot.Get("defaultNamingContext"))
Set oOu=GetObject("LDAP://ou=Дирекция,"&
oRoot.Get("defaultNamingContext"))
oOu.MoveHere oUser.ADsPath, vbNillStaring
```

Если объект нужно переименовать и перенести, то в качестве второго аргумента функции `MoveHere` используйте новое имя объекта:

```
oOu.MoveHere oUser.ADsPath, "Petrov, Vasily"
```

## Чтение атрибутов объектов

Атрибуты объектов могут принадлежать к одному из типов данных: число, строка, массив. Для чтения какого-либо свойства объекта используют следующий алгоритм. Сначала получают доступ к объекту с помощью функции `GetObject()` или с помощью поиска объекта по заданному критерию; определяют тип данных, к которому принадлежит читаемое значение, с помощью функции `VarType()` и, в зависимости от типа данных, осуществляют чтение значения по алгоритму. Приведем универсальный пример, в котором будут считаны значения переменных, относящихся к разному типу данных (листинг 9.16).

**Листинг 9.16. Чтение свойств объектов (на примере `User`)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oUser = GetObject("LDAP://cn=Ivanov,Ivan,"&
oRoot.Get("defaultNamingContext"))
ReadValue= oUser.Fields("description").Value
If VarType(ReadValue)>8192 then
       For Each ResultValue in ReadValue
              Wscript.Echo ResultValue
       Next
Else
       Wscript.Echo ReadValue
End If
```

Если возвращаемое значение функции `VarType()` больше, чем 8192, то данные представляют собой массив, элементы которого являются строками. Удобнее всего читать значения элементов массива с помощью цикла `For Each…Next`. Существует другой вариант чтения данного массива, однако он менее компактен:

```
For i=0 to ReadValue.Count
       Wscript.Echo ReadValue (i)
Next
```

Если с помощью программы LDAP Browser или другого источника информации определен тип данных читаемого значения, то нет необходимости использовать функцию `VarType()`. Однако необходимо учесть, что некоторые параметры, например `MemberOf()`, имея тип данных "массив", могут изменить его на "строку", если в массиве один элемент. В такой ситуации без функции `VarType()` не обойтись.

## Изменение атрибутов объекта

Изменение атрибутов любого объекта осуществляется с помощью метода `Put()` с предварительным получением доступа к нему (листинг 9.17).

**Листинг 9.17. Изменение свойств объектов (на примере `User`)**

```
Set oRoot = GetObject("LDAP://rootDSE")
Set oUser = GetObject("LDAP://cn=Ivanov,Ivan,"&
oRoot.Get("defaultNamingContext"))
oUser.Put "Description", "Иванов Иван Петрович"
oUser.SetInfo
```

Первый аргумент метода `Put` — изменяемое поле, второй — его значение. Список полей, характерных для объекта, можно найти в программе LDAP Browser, которая была описана в предыдущей главе.

## Поиск объекта

Поиск объекта осуществляется с помощью ADODB-соединения, провайдером которого является `ADsDSOObject`. Приведем пример поиска всех пользователей в домене (листинг 9.18).

**Листинг 9.18. Поиск объектов в домене**

```
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.CommandTimeout = 120
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection

Set oRoot = GetObject("LDAP://rootDSE")
Set oDomain = GetObject("LDAP://" & oRoot.Get("defaultNamingContext"))

Set search=objconnection.execute("SELECT * FROM '" & oDomain & " '  WHERE
objectClass='user' and NOT objectClass='computer'" )

search.Movefirst
Do Until search.EOF
       Wscript.Echo search.Fields("SamAccounName").Value
       search.MoveNext
Loop
```



# Глава 10

# Microsoft Windows Management Instrument

Решение задачи накопления и сохранения информации об аппаратной конфигурации рабочей станции обеспечивается стандартными средствами Microsoft Windows. Одним из таких средств является Microsoft Windows Management Instrument (WMI).

WMI, разработанный в 1998 году, предоставляет разработчикам программного обеспечения и администраторам сети стандартизированные способы наблюдения и управления локальными и сетевыми ресурсами.

По своей сути WMI — это расширенная и адаптированная компанией Microsoft реализация стандарта WBEM (Web-Based Enterprise Management) компании DMTF Inc. В основе WBEM лежит идея создания универсального интерфейса мониторинга и управления различными системами.

В основе структуры представления данных в стандарте WBEM лежит CIM (Common Information Model — модель информации общего типа), с помощью которой реализован объектно-ориентированный подход к представлению компонентов систем как классов со своим набором свойств и методов, а также принципов наследования.

Поскольку WMI построена по объектно-ориентированному принципу, то все данные об операционной системе, ее свойствах, управляемых приложениях и обнаруженном оборудовании, представлены в виде объектов. Каждый тип объекта описан классом, в состав которого входят свойства и методы. Определения классов описаны в MOF-файлах, а объекты этих классов с заполненными свойствами и доступными методами при их вызове возвращаются WMI-провайдерами. Управляет созданием и удалением объектов, а также вызовом их методов служба CIM Object Manager.

Для управления настройками сетевого адаптера нужно запросить у CIM Object Manager экземпляр объекта `Win32_NetworkAdapterConfiguration` и вызвать нужные методы. В частности для того, чтобы обновить аренду адреса

на DHCP–сервере, достаточно вызвать метод `RenewDHCPLease` экземпляра объекта `Win32_NetworkAdapterConfiguration`.

# Средства управления WMI

Инструменты управления WMI можно условно разделить на утилиты, поставляемые с операционной системой по умолчанию и сторонние.

В поставки Windows 2k входят следующие утилиты:

- Wmimgmt.msc — оснастка консоли MMC, позволяющая в целом управлять системой WMI на указанном компьютере;
- Winmgmt.exe — консольная утилита управления WMI. Выполняет аналогичные действия, что и консоль MMC Wmimgmt.msc. Кроме того, является исполняемым файлом сервиса WMI в системе;
- Wbemtest.exe — графическая утилита для интерактивной работы с WMI. Удобна для тестирования классов и методов, просмотра свойств и т. п.;
- Wmic.exe — консольная утилита для вызова объектов и методов WMI (WMI Console). Присутствует только в Windows XP и Windows Server 2003.

Ко второй категории средств для работы с WMI, которые требуется дополнительно устанавливать, относятся:

- WMI Code Creator 1.0 (**http://download.microsoft.com/download/0/c/a/0ca7691c-6335-4143-8f9f-6708969f8212/WMICodeCreator.zip**) — утилита для создания готовых сценариев WMI. Поддерживает языки Visual Basic Script, C# и Visual Basic .NET;
- WMI Administrative Tools **http://download.microsoft.com/download/.NetStandardServer/Install/V1.1/NT5XP/EN-US/WMITools.exe**) — удобная среда разработки и тестирования WMI-классов и методов; комплект средств в составе: WMI CIM Studio, WMI Event Registration, WMI Event Viewer и WMI Object Browser;
- Scriptomatic 2.0 (**http://www.microsoft.com/downloads/details.aspx?FamilyID=09dfc342-648b-4119-b7eb-783b0f7d1178&DisplayLang=en**) — мастер, созданный на основе Hyper Text Application (HTA). Удобна для создания готовых сценариев на различных скриптовых языках, в том числе Visual Basic Script, Perl, Java Script, Python и т. д.;
- Tweakomatic Utility (**http://www.microsoft.com/downloads/details.aspx?FamilyID=bd328d1e-6c01-4447-bd7c-c09646d722c8&DisplayLang=en**) — утилита в формате Hyper Text Application (HTA). Содержит множество

настроек системы, обычно доступных через такие утилиты, как Windows XP Power Toys TweakUI, для которых позволяет сгенерировать соответствующие WMI-скрипты.

# Внутреннее устройство WMI

Исполняемым файлом, обеспечивающим функционирование WMI, является c:\WINNT\system32\wbem\winmgmt.exe. Механизм работы WMI следующий: на первом этапе осуществляется подключение к службе WMI локального или уделенного компьютера. При обращении приложения к WMI запросы приложения пересылаются диспетчеру объектов CIM (Common Information Model Object Manager). CIMOM обеспечивает первоначальное создание объектов и единообразный способ доступа к управляемым объектам. С помощью функции `GetObject()` осуществляется наиболее простое подключение к удаленному компьютеру. На втором этапе делается проверка хранилища объектов. Затем запрос передается провайдеру объекта (табл. 10.1). *Провайдер* (provider) — это интерфейс между управляемым устройством и диспетчером CIMOM. Он собирает информацию об устройствах и делает их доступными для диспетчера. На третьем этапе, после окончания обработки запроса, провайдер пересылает результаты исходному сценарию или приложению.

***Таблица 10.1.*** *Провайдеры WMI*

| Провайдер | Файл | Пространство имен | Описание |
|---|---|---|---|
| Active Directory Provider | dsprov.dll | `Root\Directory\Ldap` | Доступ к объектам AD через WMI |
| Event Log Provider | Ntevt.dll | `Root\Cimv2` | Управление протоколами событий Windows. Например, чтение, копирование, удаление, сжатие, переименование и т. д. |
| Registry Provider | Stdprov.dll | `Root\Default` | Чтение, запись, создание и удаление ключей и значений реестра. |
| SNMP Provider | Snmpincl.dll | `Root\Snmp` | Доступ к SNMP |
| WDM Provider | Wmiprov.dll | `Root\Wmi` | Доступ к информации о драйверах WDM-устройств |

***Таблица 10.1*** *(окончание)*

| Провайдер | Файл | Пространство имен | Описание |
|---|---|---|---|
| Win32 Provider | cimwin32.dll | `Root\Cimv2` | Доступ к информации на компьютере, дискам, периферийным устройствам, файлам, папкам, файловой системе, сетевым компонентам, операционной системе, принтерам, сервисам и т. д. |
| Windows Installer Provider | Msiprov.dll | `Root\Cimv2` | Доступ к информации об установленном ПО на рабочей станции |

# Язык запросов WQL

WMI Query Language (WQL) — это SQL-язык запросов, разработанный для WMI, который является подмножеством ANSI SQL. Основное отличие WQL от ANSI SQL состоит в том, что WQL не позволяет производить изменения в данных WMI, т. е. фактически в WQL поддерживается лишь один оператор SQL `select`. Кроме того, в операторе `select` языка WQL не поддерживаются следующие ключевые слова: `DISTINCT`, `COUNT`, `JOIN`, `SUBSTRING`, `ORDER BY`, `UPPER`, `LOWER` и `DATEPART`. Также не поддерживаются арифметические операторы. Кроме того, конструкции `IS` и `IS NOT` могут применяться только в сочетании с константой `NULL`. Языком WQL поддерживается оператор `LIKE`.

Далее приведен пример некоторых типичных WQL-запросов:

```
SELECT * FROM Win32_LogicalDisk WHERE FileSystem IS NULL

SELECT * FROM Win32_LogicalDisk WHERE FileSystem IS NOT NULL

SELECT * FROM Win32_LogicalDisk WHERE FileSystem = "NTFS"

SELECT * FROM Win32_DiskDrive WHERE Partitions < 2 OR
SectorsPerTrack > 100

SELECT * FROM Win32_LogicalDisk WHERE (Name = "C:" OR Name = "D:")
AND FreeSpace > 2000000 AND FileSystem = "NTFS"

SELECT * FROM Win32_NTLogEvent WHERE Logfile = 'Application'

SELECT * FROM Meta_Class WHERE __Class LIKE %Win32%

SELECT * FROM __InstanceCreationEvent WHERE TargetInstance ISA
"Win32_NTLogEvent" GROUP WITHIN 600 BY TargetInstance.SourceName
HAVING NumberOfEvents > 25
```

В формировании и тестировании SQL-запросов неоценимую помощь может оказать утилита `WBEMTEST` (см. подробнее *разд. "Средства управления WMI"*).

# Безопасность и WMI

Все взаимодействие с ядром WMI происходит с использованием интерфейсов COM+/DCOM. В свою очередь COM+ и DCOM в качестве транспортного протокола используют RPC. Эта архитектурная особенность накладывает определенный отпечаток на идеологию системы безопасности WMI.

Для того чтобы некая учетная запись имела возможность подключаться к репозиторию WMI, необходимо дать ей соответствующие права. Права нужно дать как на пространство имен — WMI namespace (воспользовавшись оснасткой wmimgmt.msc), так и на DCOM-приложения диспетчера WMI — CIM Object Manager (воспользовавшись оснасткой управления COM+ comexp.msc или утилитой dcomcnfg.exe). Минимальный список приложений DCOM, права на которые необходимы для удаленной работы с WMI: Windows Management and Instrumentation, WMI Event Viewer. Некоторые сведения по вопросам настройки прав доступа к WMI и сетевой безопасности можно найти в статье Microsoft Knowledge Base KB875605. Права на другие DCOM-приложения могут понадобиться в зависимости от используемого режима имперсонации.

## Имперсонация

*Имперсонация* — это метод, в котором для подключения к ресурсу система будет использовать не свой контекст безопасности, а учетные данные другого субъекта безопасности.

Расширенный вариант имперсонации — делегирование (табл. 10.2), когда подключение к конечному ресурсу выполняется не самим субъектом безопасности, а через посредника (например, промежуточный сервер).

В случае с WMI делегирование может выглядеть так: работая на рабочей станции, сценарий по WMI обращается к серверу и запускает на нем процесс с помощью метода `Execute` класса `Win32_Process`. Если в данной ситуации не воспользоваться делегированием, то на конечной машине скрипт будет запущен от имени учетной записи промежуточного сервера, что не желательно.

***Таблица 10.2.*** *Уровни имперсонации*

| Уровень имперсонации | Описание |
|---|---|
| `Anonymous`<br>`1` | Анонимный уровень имперсонации COM, маскирующий учетную запись |

***Таблица 10.2*** *(окончание)*

| Уровень имперсонации | Описание |
|---|---|
| `Default`<br>`0` | Уровень имперсонации по умолчанию |
| `Delegate`<br>`4` | Уровень имперсонации COM — делегирование. Разрешает использовать другим объектам учетные данные вызывающего субъекта для обращения к третьим объектам. Этот уровень может дать неоправданно высокие привилегии промежуточному объекту |
| `Identify`<br>`2` | Уровень имперсонации COM — идентификация. Позволяет объектам вызова запрашивать учетные данные у вызывающего субъекта |
| `Impersonate`<br>`3` | Уровень имперсонации COM — обычная имперсонация (рекомендуемый уровень имперсонации). Позволяет вызываемому объекту использовать учетные данные вызывающего субъекта для совершения только своих действий |

## Аутентификация

Аутентификация, целостность и конфиденциальность являются неотъемлемыми характеристиками безопасного взаимодействия систем по сети. При использовании WMI поддерживаются несколько уровней аутентификации (табл. 10.3). Наиболее часто используемый уровень — `Connect` (аутентификация и авторизация при вызове). Если нужно предотвратить возможное изменение передаваемых данных или их перехват, то рекомендуется выбрать режимы `Pkt` (проверка аутентичности клиента), `PktIntegrity` (проверка аутентичности клиента и целостности передаваемых данных) или `PktPrivacy` (проверка аутентичности клиента и шифрование передаваемых данных с проверкой целостности).

***Таблица 10.3.*** *Уровни аутентификации и проверки целостности*

| Уровень аутентификации | Описание |
|---|---|
| `Call`<br>`Call`<br>`3` | Call-level COM authentication.<br>Аутентификация в начале каждого вызова объекта WMI |
| `Connect`<br>`Connect`<br>`2` | Connect-level COM authentication.<br>Аутентификация только при установлении соединения с сервером WMI. Одни учетные данные используются для всего сеанса взаимодействия |

***Таблица 10.3*** *(окончание)*

| Уровень аутентификации | Описание |
|---|---|
| `Default`<br>`Default`<br>`0` | WMI использует настройки аутентификации COM по умолчанию |
| `None`<br>`None`<br>`1` | Аутентификация COM не используется |
| `Packet`<br>`Pkt`<br>`4` | Packet-level COM authentication.<br>Аутентификация всех данных, получаемых от клиента, с подтверждением подлинности отправителя для каждого RPC-пакета |
| `PacketIntegrity`<br>`PktIntegrity`<br>`5` | Packet Integrity-level COM authentication.<br>Аутентификация и проверка целостности передаваемых данных для каждого RPC-пакета |
| `PacketPrivacy`<br>`PktPrivacy`<br>`6` | Packet Privacy-level COM authentication.<br>Аутентификация, проверка целостности и шифрование данных каждого передаваемого RPC-пакета |

## Привилегии

Администраторам Windows хорошо известны настройки безопасности системы и их раздел `User Right Assignments` (Привилегии пользователей), доступные в консоли безопасности системы и групповых политиках домена. Ряд действий с операционной системой можно проделать только при наличии у пользователя или группы соответствующих прав. К таким действиям относится: перезагрузка системы (завершение ее работы), восстановление состояния системы из резервной копии, смена системного времени и т. д.

Поскольку с использованием WMI можно выполнить все эти действия, разработчики WMI заложили дополнительный механизм защиты. Суть его в следующем: если учетная запись пользователя обладает необходимыми привилегиями (табл. 10.4), он не сможет выполнить это действие, пока явно будет активирована привилегия непосредственно перед выполнением действия. В частности, если администратор запустит скрипт WMI, запрашивающий перезагрузку системы, этого все равно не произойдет, пока в скрипте не будет явно активирована эта привилегия (листинг 10.1).

**Листинг 10.1. Скрипт перезагрузки операционной системы**

```
strComputer = "server01"

Set objLocator = CreateObject("WbemScripting.SWbemLocator")
objLocator.Security_.AuthenticationLevel = 3
objLocator.Security_.Privileges.Add(18)

Set objWMIService = objLocator.ConnectServer(_
strComputer, "root\cimv2", "mydomain\administrator", "password")

objWMIService.Security_.ImpersonationLevel = 3

Set colItems = objWMIService.ExecQuery(_
 "SELECT * FROM Win32_OperatingSystem",,48)

For Each objItem in colItems
Wscript.Echo "-----------------------------------"
Wscript.Echo "Win32_OperatingSystem instance"
Wscript.Echo "-----------------------------------"
Wscript.Echo "Caption: " & objItem.Caption
Wscript.Echo "Name: " & objItem.Name

Rem Первый вариант вызова метода Reboot()
Set         objOutParams         =         objWMIService.ExecMethod(_
"Win32_OperatingSystem.Name='" & CStr(objItem.Name) & "'", "Reboot")

Rem Второй вариант вызова метода Reboot()
objItem.Reboot()
Next
```

***Таблица 10.4.*** *Привилегии и их числовые эквиваленты*

| Привилегии | Описание |
|---|---|
| wbemPrivilegeCreateToken<br>SeCreateTokenPrivilege<br>CreateToken<br>1<br>0x1 | Привилегия требуется для создания основного токена безопасности процесса |

***Таблица 10.4*** *(продолжение)*

| Привилегии | Описание |
|---|---|
| wbemPrivilegePrimaryToken<br>SeAssignPrimaryTokenPrivilege<br>AssignPrimaryToken<br>2<br>0x2 | Привилегия требуется для замены (назначения нового) основного токена безопасности процесса |
| wbemPrivilegeLockMemory<br>SeLockMemoryPrivilege<br>3<br>0x3 | Привилегия требуется для закрепления соответствия между страницами физической памяти и логического адресного пространства |
| wbemPrivilegeIncreaseQuota<br>SeIncreaseQuotaPrivilege<br>IncreaseQuotaPrivilege<br>4<br>0x4 | Привилегия требуется для назначения квот процессу |
| wbemPrivilegeMachineAccount<br>SeMachineAccountPrivilege<br>MachineAccount<br>5<br>0x5 | Привилегия требуется для создания учетной записи компьютера |
| wbemPrivilegeTcb<br>SeTcbPrivilege<br>Tcb<br>6<br>0x6 | Привилегия обозначает ее владельца как часть `Trusted Computer Base` |
| wbemPrivilegeSecurity<br>SeSecurityPrivilege<br>Security<br>7<br>0x7 | Привилегия требуется для выполнения ряда функций, связанных с безопасностью, например просмотр журналов аудита. Привилегия определяет ее владельца как `Security Operator` |
| wbemPrivilegeTakeOwnership<br>SeTakeOwnershipPrivilege<br>TakeOwnership<br>8<br>0x8 | Привилегия требуется для получения права владельца объекта на объекты безопасности, в отсутствии явных на то разрешений |
| wbemPrivilegeLoadDriver<br>SeLoadDriverPrivilege<br>LoadDriver<br>9<br>0x9 | Привилегия требуется для загрузки и выгрузки драйверов устройств |
| wbemPrivilegeSystemProfile<br>SeSystemProfilePrivilege<br>SystemProfile<br>10<br>0xA | Привилегия требуется для сбора профилирующей информации всей системы |

***Таблица 10.4*** *(продолжение)*

| Привилегии | Описание |
|---|---|
| wbemPrivilegeSystemtime<br>SeSystemtimePrivilege<br>Systemtime<br>11<br>0xB | Привилегия требуется для изменения системного времени |
| wbemPrivilegeProfileSingleProcess<br>SeProfileSingleProcessPrivilege<br>ProfileSingleProcess<br>12<br>0xC | Привилегия требуется для сбора профилирующей информации для одного процесса |
| wbemPrivilegeIncreaseBasePriority<br>SeIncreaseBasePriorityPrivilege<br>IncreaseBasePriority<br>13<br>0xD | Привилегия требуется для увеличения базового приоритета процесса |
| wbemPrivilegeCreatePagefile<br>SeCreatePagefilePrivilege<br>CreatePagefile<br>14<br>0xE | Привилегия требуется для создания и/или изменения файла подкачки |
| wbemPrivilegeCreatePermanent<br>SeCreatePermanentPrivilege<br>CreatePermanent<br>15<br>0xF | Привилегия требуется для создания постоянного общего объекта |
| wbemPrivilegeBackup<br>SeBackupPrivilege<br>Backup<br>16<br>0x10 | Привилегия требуется для выполнения резервного копирования |
| wbemPrivilegeRestore<br>SeRestorePrivilege<br>Restore<br>17<br>0x11 | Привилегия требуется для выполнения операции восстановления. Эта привилегия позволяет ее владельцу устанавливать для любого объекта произвольный существующий SID в качестве владельца объекта |
| wbemPrivilegeShutdown<br>SeShutdownPrivilege<br>Shutdown<br>18<br>0x12 | Привилегия требуется для перезагрузки и завершения работы ОС |
| wbemPrivilegeDebug<br>SeDebugPrivilege<br>Debug<br>19<br>0x13 | Привилегия требуется для отладки процессов |

***Таблица 10.4*** *(окончание)*

| Привилегии | Описание |
|---|---|
| wbemPrivilegeAudit<br>SeAuditPrivilege<br>Audit<br>20<br>0x14 | Привилегия требуется для записи в журналы аудита |
| wbemPrivilegeSystemEnvironment<br>SeSystemEnvironmentPrivilege<br>SystemEnvironment<br>21<br>0x15 | Привилегия требуется для модификации энергонезависимой памяти в тех системах, которые используют ее для хранения своей конфигурации |
| wbemPrivilegeChangeNotify<br>SeChangeNotifyPrivilege<br>ChangeNotify<br>22<br>0x16 | Привилегия требуется для получения нотификаций об изменении файлов и директорий. Также эта привилегия отменяет перекрестную проверку доступа к файлам и папкам. Эта привилегия по умолчанию дана всем пользователям |
| wbemPrivilegeRemoteShutdown<br>SeRemoteShutdownPrivilege<br>RemoteShutdown<br>23<br>0x17 | Привилегия требуется для завершения работы ОС по сети |
| wbemPrivilegeUndock<br>SeUndockPrivilege<br>Undock<br>24<br>0x18 | Привилегия требуется для снятия компьютера с док-станции |
| wbemPrivilegeSyncAgent<br>SeSyncAgentPrivilege<br>SyncAgent<br>25<br>0x19 | Привилегия требуется для вызова процедуры синхронизации службы каталога |
| wbemPrivilegeEnableDelegation<br>SeEnableDelegationPrivilege<br>EnableDelegation<br>26<br>0x1A | Привилегия требуется для доверия пользователям или группам при делегировании |
| wbemPrivilegeManageVolume<br>SeManageVolumePrivilege<br>ManageVolume<br>27<br>0x1B | Привилегия требуется для операций обслуживания дисковых томов |

Для каждой привилегии в табл. 10.4 указано три параметра: первый — это константа для использования в скриптах VBScript, второй — символические константы языка C++, третий — имя, использующееся при составлении `moniker string`.

Активировать привилегии нужно до подключения к репозиторию WMI, а не после.

# Способы доступа к объектам WMI (VBScript)

На практике для получения доступа к объектам WMI используют один из следующих шаблонов (листинги 10.2—10.4).

**Листинг 10.2. Доступ к объектам WMI. Шаблон 1. Вариант 1**

```
strComputer=""
strNameSpace=""
strClass=""
Set objElements = GetObject(
"winmgmts:{ImpersonationLevel=Impersonate}!//" & strComputer & "/ " &
strNameSpace & ": " & strClass)
       For each Element in objElements
              Temp=Element.Value
       Next
```

**Листинг 10.3. Доступ к объектам WMI. Шаблон 1. Вариант 2**

```
strComputer=""
strNameSpace=""
strClass=""
Set objWMIService = GetObject(
"winmgmts:{ImpersonationLevel=Impersonate}!//" & strComputer & "/ " &
strNameSpace)
       Set colItems = objWMIService.InstancesOf(strClass )
              For Each objItem in colItems
                     Temp=Element.Value
              Next
```

**Листинг 10.4. Доступ к объектам WMI. Шаблон 1. Вариант 3**

```
strComputer=""
strNameSpace=""
strClass=""
Set objWMIService = GetObject("winmgmts:{ImpersonationLevel=Impersonate
}!// " & strComputer & "/ " & strNameSpace)
```

```
Set colItems = objWMIService.ExecQuery("SELECT поле_1, поле_2, …, поле_n
FROM" & strClass )
       For Each objItem in colItems
              Temp=Element.Value
       Next
```

В приведенных примерах переменная `strComputer` содержит имя удаленного компьютера. Если компьютер локальный, то вместо имени указывается символ ".": `strComputer="."`. Переменная `strNameSpace` содержит пространство имен, переменная `strClass` — название класса. Например, для `Win32 Provider` одним из классов является `Win32_BIOS`, соответственно, переменная `strNameSpace` принимает значение `Root\Cimv2`. Инструкция `{ImpersonationLevel=Impersonate}!` заставляет выполнить сценарий с привилегиями пользователя, вызывающего сценарий, а не с привилегиями пользователя, который в настоящий момент работает на рабочей станции. Таким образом, осуществляется подстановка имени и пароля другого пользователя. Эта часть применяется для удаленного управления рабочими станциями в скриптах регистрации пользователей в сети. Инструкция `{ImpersonationLevel=Impersonate}!` в Microsoft Windows 200x является выражением подстановки по умолчанию.

Варианты 1 и 2 шаблона 1 очень похожи, отличаются только формой записи. В варианте 3 обращение к классу строится с помощью SQL-запроса, что позволяет экономить трафик.

В общем случае запрос SQL выглядит следующим образом:

```
SELECT поле_1, поле_2, …, поле_n FROM strClass
```

В поле `SELECT` указываются поля, по которым идет выборка. Поля перечисляются через запятую, пробелы после запятой обязательны. Если выборка осуществляется по всем полям, то вместо названий полей указывается символ "`*`". В поле `FROM` указывается один из ранее перечисленных поставщиков: `SELECT * FROM strClass`.

В зависимости от способа доступа размер пересылаемых данных разный (табл. 10.5).

***Таблица 10.5.*** *Распределение трафика при использовании разных методов доступа*

| **Метод** | **Трафик (Кбайт)** |
|---|---|
| `objSWbemServices.InstanceOf("Win32_Service")` | 157 |
| `objSWbemServices.ExecQuery("Select * From Win32_Service")` | 156 |
| `objSWbemServices.ExecQuery("Select field1 From Win32_Service")` | 86 |

Из данных табл. 10.5 видно, что наименьший трафик наблюдается при использовании шаблона 3. Несмотря на то, что код получится несколько громоздким, уменьшится межсетевой трафик, сократится время работы сценария. Рекомендуется использовать именно этот шаблон, но в несколько модифицированном виде (листинг 10.5).

**Листинг 10.5. Доступ к объектам WMI. Шаблон 1. Обобщенный вариант 3**

```
strComputer=""
strNameSpace=" Root\Cimv2"
strClass="Win32_Value"
Set objWMIService = GetObject( " winmgmts: // " & strComputer & "/ " &
strNameSpace)
Set colItems = objWMIService.ExecQuery("SELECT поле_1, поле_2, …, поле_n
FROM" & strClass )
       For Each objItem in colItems
              Temp=Element.Value
       Next
```

# Продукты, использующие WMI

На основе WMI построены продукты Microsoft SMS 2003 (**http://www.microsoft.com/smserver**) и Microsoft Operations Manager 2005 (**http://www.microsoft.com/mom**).

Коллекцию различных скриптов можно найти на TechNet Script Center:

**http://www.microsoft.com/technet/scriptcenter/default.mspx**

и Portable Script Center:

**http://www.microsoft.com/downloads/details.aspx?FamilyID=b4cb2678-dafb-4e30-b2da-b8814fe2da5a&DisplayLang=en**

## SMS 2003 SP1

Система управления изменениями и конфигурациями, построенная на базе Microsoft Systems Management Server 2003 (SMS 2003), представляет собой клиент-серверное распределенное приложение. Клиентская часть устанавливается на управляемые рабочие станции и периодически запрашивает с сервера, входящего в состав системы SMS 2003, данные о конфигурации рабочей станции, установленном программном обеспечении. Основными задачами в сетях

на базе продуктов Microsoft, решаемыми системой управления на базе SMS 2003, являются:

- ❒ автоматизация процесса установки клиентской части системы SMS 2003 на рабочие станции;
- ❒ инвентаризация аппаратного обеспечения рабочих станций;
- ❒ инвентаризация программного обеспечения рабочих станций;
- ❒ инвентаризация установленных и пропущенных обновлений системного ПО;
- ❒ группировка рабочих станций на основании данных инвентаризации в коллекции;
- ❒ установка программного обеспечения на группы компьютеров (коллекции);
- ❒ установка операционных систем из образов на новые компьютеры и переустановка с сохранением профилей пользователей на эксплуатируемых компьютерах;
- ❒ установка необходимых обновлений безопасности;
- ❒ выполнение на клиентских рабочих станциях задач обслуживания, оформленных в виде скриптов и/или исполняемых файлов;
- ❒ управление мобильными устройствами PocketPC;
- ❒ определение частоты запуска указанных приложений;
- ❒ отслеживание соответствия набора и версий программного обеспечения на рабочих станциях заданным параметрам;
- ❒ формирование отчетов о рабочих станциях и серверах;
- ❒ формирование отчетов о состоянии и работе самой системы;
- ❒ оптимизация нагрузки на сеть при передаче дистрибутивов и установке ПО на рабочие станции и серверы;
- ❒ удаленная поддержка пользователей с использованием Remote Tools, в состав которых входят удаленное управление рабочим столом АРМ, передача файлов, удаленная перезагрузка, интерактивное общение администратора с пользователем и удаленная командная консоль;
- ❒ обеспечение контроля доступа ко всем функциям и объектам системы управления на базе SMS 2003 для авторизованных пользователей.

Все функции SMS 2003 реализованы с использованием технологии WMI. В частности весь процесс инвентаризации, который производит агент SMS, представляет собой набор WMI-запросов. Все свои настройки и задания

(advertisements), а также некоторые промежуточные результаты инвентаризации агент SMS хранит в пространстве имен `root\CCM`.

## MOM 2005 SP1

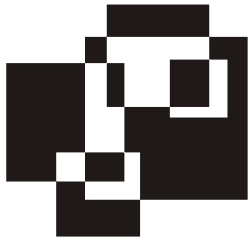
Система мониторинга на базе Microsoft Operations Manager 2005 (MOM 2005) позволяет решать следующие задачи:

- обеспечение непрерывного наблюдения в реальном времени за состоянием ИС компании и автоматическая регистрация инцидентов;
- снижение времени, требуемого для оповещения дежурной смены группы мониторинга об изменениях и сбоях, происходящих в ИС компании;
- снижение количества незарегистрированных инцидентов;
- обеспечение мониторинга всех компонентов ИС компании и распределенное по всем сегментам сети наблюдение за их состоянием;
- предоставление кратких рекомендаций администраторам системы по большинству событий, регистрируемых системой мониторинга;
- обеспечение отказоустойчивости системы мониторинга;
- обеспечение возможности разделения административных полномочий на участки системы мониторинга;
- возможность ведения корпоративной базы знаний по инцидентам;
- выявление взаимосвязей между возникающими инцидентами и источниками проблем.

Система мониторинга представляет собой клиент-серверное распределенное приложение. На целевые серверы (те серверы прикладных систем, на которых осуществляется мониторинг) устанавливается специальное программное обеспечение — агент системы мониторинга MOM 2005. Агенты системы мониторинга получают свои настройки и правила (политики) мониторинга с управляющих серверов.

Для каждого сервера, за которым ведет наблюдение его агент, в зависимости от установленного на него программного обеспечения, применяется свой набор правил мониторинга, которые объединяются в группы правил и применяются к группам компьютеров. Группы компьютеров содержат целевые системы мониторинга, которые туда добавляются либо автоматически на основании данных из реестра целевых систем, либо вручную.

Множество правил мониторинга, которые использует MOM 2005, представляют собой скрипты, использующие WMI для получения и обработки данных о системе и ее компонентах.



# Глава 11

# Сценарий регистрации пользователей в сети

Сценарий регистрации пользователей в сети предназначен для автоматизации процессов, связанных с подключением рабочих станций к сети, уменьшения временных затрат на их администрирование и конфигурирование.

Существует несколько языков, используемых для создания сценариев загрузки. Они были описаны в *главе 1*. Рассмотрим более подробно создание сценариев регистрации пользователей в сети на базе языка программирования KIXTart.

## KIXTart

### Системные требования

Язык KIXTart может быть установлен на рабочую станцию с процессором i486 и выше под управлением одной из операционных систем: MS-DOS, Windows 9x всех модификаций, Windows NT 4.0, Windows 2000/2003/XP/Vista. Таким образом, KIXTart успешно функционирует на любой современной рабочей станции, однако поддержка KIXTart Windows 9x ограничена из-за различий в идеологии построения систем печати, ядер операционных систем. В дальнейшем будет рассматриваться взаимодействие KIXTart с операционными системами Windows 2k.

Последнюю версию KIXTart можно загрузить с одного из следующих сайтов:

- **http://kixtart.org**
- **http://www.scriptlogic.com/kixtart**
- **http://kixhelp.com**

## Комплект поставки KIXTart

В дистрибутив KIXTart входит интерпретатор, документация, несколько примеров скриптов и комплект драйверов для обеспечения работы скриптов под управлением Windows 9x (табл. 11.1).

***Таблица 11.1.*** *Компоненты дистрибутива KIXTart*

| Файл | Описание |
|---|---|
| Kix2001.doc | Документация KIXTart (справочник по функциям) |
| Kix32.exe | Интерпретатор KIXTart (полная версия) |
| WKix32.exe | Интерпретатор KIXTart (сокращенная версия) |
| Kxrpc.exe | RPC-сервис, обеспечивающий работу KIXTart под Windows 9x |
| Kx95.dll | Динамическая библиотека для обеспечения работы KIXTart под Windows 9x |
| Kx16.dll, Kx32.dll | Динамические библиотеки для обеспечения подключения к NetApi.dll под Windows 9x |
| 11.kix | Примеры скриптов |
| 11.spk | Примеры звуковых файлов (специальный формат для работы с KIXTart) |
| Chimes.wav | Пример звукового файла |
| Kix2001.txt | Пресс-релиз, содержащий информацию об обновлениях, сделанных в последней версии KIXTart |

## Установка KIXTart

В доменах, построенных на основе Windows 2k, сценарии регистрации пользователей располагаются в папке Netlogon на контроллере домена, соответственно, интерпретатор — kix32.exe и сам сценарий рекомендуется располагать в этом же каталоге. Если в сети присутствуют несколько контроллеров домена, то оговоренный набор файлов достаточно расположить только на одном из них. С помощью системы репликации содержимое папки будет синхронизировано с другими папками Netlogon на серверах.

## Синтаксис KIX32

Синтаксис утилиты kix32.exe:

```
KIX32 [script1] [...] [$var=123] [/f[:yyyy/mm/dd]] [/r:irel] [/d] [/i] [/?]
```

Описание ключей:

- `/f[:yyyy/mm/dd]` — очистка кэша;
- `/r:irel` — поиск KXRPC-сервера;
- `/d` – включение режима редактирования;
- `/i` — включение режима сокрытия CMD-консоли;
- `/t` — компиляция сценария в бинарный файл;
- `/?` — вызов справки.

## Запуск KIX32

Запуск KIXTart может быть осуществлен одним из следующих способов:

- пользователем — осуществляется запуск файлов kix32.exe и script.kix автоматически при регистрации пользователя в сети; в свойствах пользователя в AD на вкладке **Profile** в поле **Logon Script Name** необходимо указать `kix32.exe script.kix`;
- из BAT-файла — для определения настоящего месторасположения файла используют следующую команду: `%0\..\Kix32.exe`. Подробнее о пути `%0\..\` см. в статье Q121387 Knowledge Base (**http://support.microsoft.com/kb/121387**).

## Режим отладки сценариев

KIXTart поддерживается режим отладки сценариев, вызываемый ключом `/d`: `kix32.exe scrip.kix /d`. В режиме отладки первая строка (строка на синем фоне) представляет собой управляющее меню, вторая (строка на зеленом фоне) — строка, которая выполняется в настоящее время. В строках, начиная с третьей, отображается результат работы скрипта.

Приведем расшифровку клавиш, поддерживаемых режимом редактирования:

- <F5> — запуск сценария с выключением режима редактирования;
- <F8>, <Space>, <Enter> — пошаговое выполнение сценария;
- <F10> — игнорирование строк, содержащих обращение к внешним сценариям и функциям;
- <Esc>, <Q> — выход.

## Синтаксис KIXTart

Полная версия online-документации, а также CHM-файл, приведены на сайте **http:\\kixtart.org** в разделах **Command Reference** и **Manual** соответственно.

# Адаптация листингов VBScript и WSH к KIXTart

Большинство листингов VBScript и WSH можно адаптировать под KIXTart. Оба языка интерпретируемые и поддерживают COM-объекты. Это позволяет легко адаптировать листинги VBScript к KIXTart. Рассмотрим сценарий определения текущего имени. Для определения короткого и DNS-имени домена используется объект `ACTIVEDIRECTORYSystemInfo`, доступный в любой версии Windows 2k. Для определения LDAP-имени домена необходимо использовать функцию `GetObject()`, с помощью которой можно получить доступ к пространству имен `defaultNamingContext`. Затем с помощью соответствующей функции отобразить определенные имена домена в разных форматах на экране. Для корректного преобразования листинга из VBScript в KIXTart необходимо помнить, что переменные в KIXTart предваряются знаком доллара (`$`), инструкция `SET` присутствует только в VBScript. В KIXTart вместо нее ничего не пишется. При формировании одной строки из нескольких подстрок на языке KIXTart вместо символа плюс (`+`) используют амперсанд (`&`). Функции вывода сообщений на экран применяются в соответствии с синтаксисом языка программирования (табл. 11.2).

***Таблица 11.2.*** *Преобразование листинга из VBScript в KIXTart*

| VBScript | KIXTart |
|---|---|
| `Set objSysInfo = CreateObject("ACTIVE DIRECTORYSystemInfo")` | `$objSysInfo = CreateObject("ACTIVE DIRECTORYSystemInfo")` |
| `ShortN=objSysInfo.ComputerName` | `$ShortN= $objSysInfo.ComputerName` |
| `DNSN=objSysInfo.DomainDNSName` | `$DNSN= $objSysInfo.DomainDNSName` |
| `Set RootTree= GetObject("LDAP://RootDSE")` | `$RootTree= GetObject("LDAP://RootDSE")` |
| `LDAPN= RootTree .Get("defaultNamingContext")` | `$LDAPN= $RootTree .Get("defaultNamingContext")` |
| `Domain= ShortN+chr(13)+ DNSN+chr(13)+ LDAPN` | `$Domain= ShortN&chr(13)&DNSN&chr(13)&LDAPN` |
| `MsgBox Domain` | `MessageBox("$Domain","",0,0)` |

# Задачи, решаемые сценарием

С помощью сценария регистрации пользователей в сети можно решить множество различных задач:

- инвентаризация — включает в себя сбор информации о регистрирующемся в сети пользователе, рабочей станции и экспорте данных в файл отчета или базу данных;
- автоматическое подключение сетевых ресурсов: принтеров и дисков;
- автоматическое конфигурирование рабочих станций;
- обеспечение интерактивности работы скрипта — в ходе выполнения скрипта на экране отображается различная информация.

# Решение задачи инвентаризации

Решение задачи инвентаризации состоит из нескольких частей: сбора, записи на носитель в определенном формате и обработки информации. Сбор и запись информации может быть реализована с помощью сценария загрузки; ее обработка — с помощью дополнительного программного обеспечения.

Собираемая информация касается программного и аппаратного обеспечения, характеристик учетной записи пользователя, регистрирующегося в сети, и выполняется с помощью различных средств: макросов, чтения реестра, службы WMI.

Несмотря на то, что KIXTart поддерживает работу с INI-файлами, не рекомендуется сохранять отчеты в этом формате, поскольку он неудобен для дальнейшей обработки данных. Рекомендуется сохранять данные в формате XML для последующего преобразования в таблицы Excel или передавать данные в базу данных, например, SQL.

Собираемые данные можно условно разделить на две части. К первой части относится информация об аппаратном и программном обеспечении. Данные об аппаратном обеспечении считываются с помощью WMI или соответствующих макросов KIXTart из ветвей реестра рабочей станции. Вторая часть — информация, характеризующая положение рабочей станции в сети и свойства учетной записи. Эта часть реализуется с помощью макросов, встроенных в KIXTart, и чтения полей Active Directory.

# Сбор информации об аппаратном обеспечении с помощью WMI

Быстро и надежно получить информацию об аппаратной конфигурации компьютера можно с помощью WMI (листинг 11.1). На практике для доступа к WMI-объектам используют Win32 Provider, которому соответствует пространство имен `Root\Cimv2`. Для получения информации используют различные классы: `Win32_BIOS`, `Win32_Processor`, `Win32_PhysicalMemory` и др. Перечень рекомендуемых классов и некоторых их свойств приведен в табл. 11.3, для просмотра объектной модели WMI воспользуйтесь утилитой WMI Object Browser (рис. 11.1), входящей в пакет WMI Tools (**http://msdn.microsoft.com/developer/sdk/wmisdk/default.asp**).

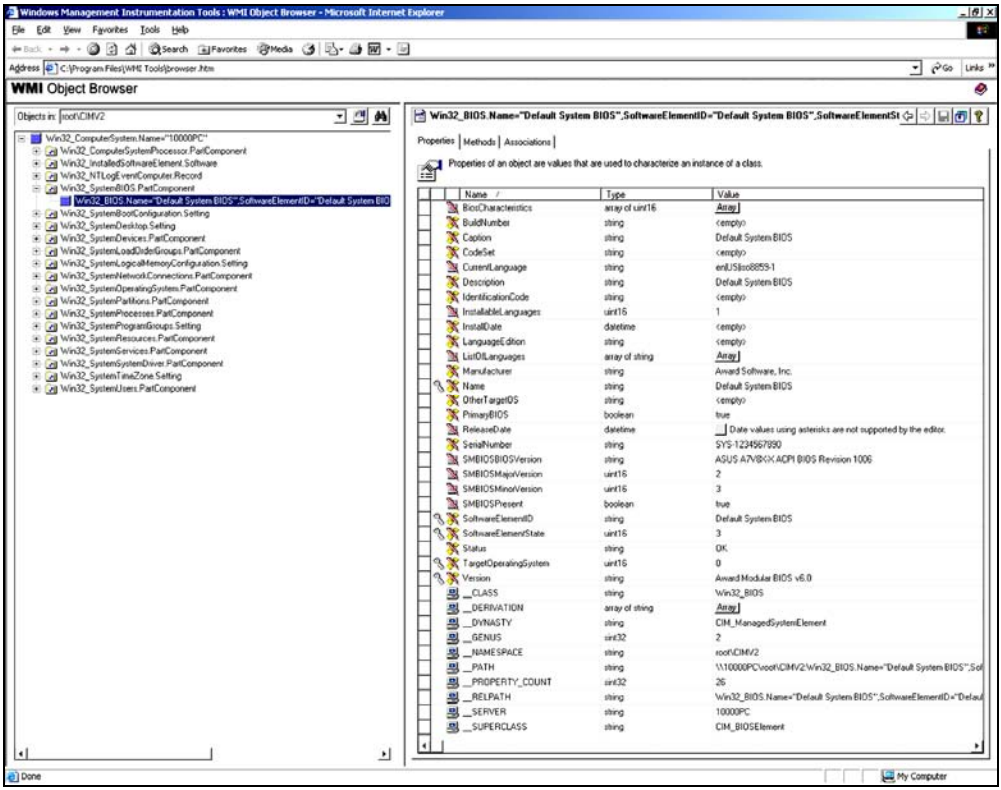



**Рис. 11.1.** Утилита WMI Object Browser

**Листинг 11.1. Шаблон доступа к классам WMI на языке KIXTart**

```
$strComputer=""
$strNameSpace=" Root\Cimv2"
$strClass="Win32_Value"
$objWMIService = GetObject( " winmgmts: // " & $strComputer & "/ " &
$strNameSpace)
$colItems = objWMIService.ExecQuery("SELECT поле_1, поле_2, …,
поле_n FROM" & $strClass )
      For Each $Element in $colItems
            $Temp=$Element.Value
      Next
```

***Таблица 11.3.*** *Перечень избранных свойств некоторых WMI-классов*

| Компонент | Класс | Свойство | Описание |
|---|---|---|---|
| Материнская плата | Win32_BIOS | Name | Название BIOS |
| | | Manufacturer | Производитель BIOS |
| | | SMBIOSBIOSVersion | Номер версии BIOS |
| | | Version | Код материнской платы |
| Процессор | Win32_Processor | UpgrActive DirectoryeMethod | Массив, элементами которого являются сокеты (socket) |
| | | Architecture | Архитектура процессора |
| | | Name | Полное название процессора |
| | | MaxClockSpeed | Максимальная частота в Гц |
| | | Version | Полная версия процессора |
| | | Level | |
| | | ExtClock | Внешняя частота в МГц |
| | | L2CacheSize | Размер кэша 2-го уровня в Кбайт |
| | | CurrentVoltage | Питание процессора |
| Оперативная память | Win32_ PhysicalMemory | MemoryType | Массив, элементами которого являются типы памяти |
| | | DeviceLocator | Номер слота |
| | | Capacity | Емкость в байтах |
| | | Speed | Скорость в МГц |

***Таблица 11.3*** *(окончание)*

| Компонент | Класс | Свойство | Описание |
|---|---|---|---|
| Жесткий диск | Win32_DiskDrive | InterfaceType | Интерфейс |
| | | Manufacturer | Производитель |
| | | Model | Модель |
| | | Size | Размер в байтах |
| Оптический накопитель | Win32_CDROMDrive | Name | Модель устройства |
| | | Drive | Буква, назначенная устройству в папке Мой компьютер |
| | | TransferRate | Скорость передачи данных в байт/с |
| Звук | Win32_SoundDevice | Manufacturer | Фирма-производитель |
| | | Name | Модель звуковой карты |
| | | ProductName | Модель звуковой карты |
| Видеоадаптер | Win32_VideoConfiguration | AdapterType | Тип разъема адаптера (PCI, AGP, PCI-E) |
| | | AdapterRAM | Размер видеопамяти в байтах |
| | | AdapterChipType | Чипсет видеокарты |
| | | AdapterCompatibility | Совместимость адаптера |
| | | HorizontalResolution | Разрешение по горизонтали в пикселах |
| | | VerticalResolution | Разрешение по вертикали в пикселах |
| | | BitsPerPixel | Глубина цвета в битах |
| | | RefreshRate | Частота обновления в Гц |

Приведем пример чтения информации BIOS материнской платы с помощью WMI на Kixtart (листинг 11.2).

**Листинг 11.2. Чтения информации BIOS**

```
$PC = @WKSTA
$en=chr(10)
$objWMIService = GetObject( "winmgmts://" + $pc+"/Root/Cimv2")
```

```
$colItems = $objWMIService.ExecQuery( "Select * from Win32_BIOS")
For Each $objItem in $colItems
       ? " BIOS Name : " + $objItem.Name
       ? " Version : " + $objItem.Version
       ? " Manufacturer : " + $objItem.Manufacturer
       ? " SMBIOS Version : " + $objItem.SMBIOSBIOSVersion
Next
```

Все переменные, использованные в примере, строковые. Если переменная — число, то необходимо выполнить ее преобразование к строковому типу данных с помощью функции `Cstr()`. Если же переменная — массив, то после считывания элементы массива также преобразуют к строковому типу данных.

Приведенный ранее механизм используется многократно, поэтому рекомендуется оптимизировать программный код, сформировав из имен классов массив данных (листинг 11.3).

**Листинг 11.3. Прием чтения информации для различных устройств**

```
$wmi_array="Win32_BIOS","Win32_Processor",……"
$ж=…
for $a=0 to $ж
$objWMIService = GetObject( "winmgmts://@wksta/root/cimv2" )
$colItems = $objWMIService.ExecQuery( "Select * from "+
$wmi_array[$ж])
For Each $objItem in $colItems
select
case $a=0
$mb1=$objItem.Name
$mb2=$objItem.Manufacturer
$mb3=$objItem.SMBIOSBIOSVersion
$mb4=$objItem.Version
$lx1="<mb> <bios> $mb1 </bios>"
$lx2="<manufacture> $mb2 </manufacture>"
$lx3="<version> $mb3 </version> "
$lx4="<data_release> $mb4 </data_release> </mb> "
$t="$t $lx220 $en $lx222 $en $lx224 $en $lx226 $en"
case $a=1
$cpu_arhitect="x86","MIPS","ALPHA", "Power PC"
$cpu_socket="","Other","Unknown","Daughter Board","ZIF Socket",
"Replacement/Piggy Back","None","LIF Socket","Slot 1",
```

```
 "Slot 2", "370 Pin Socket", "Slot A", "Slot M","","",
 "Socket 478"
$i=$objItem.Architecture
$ii=$cpu_arhitect[$i]
$cpu1=$objItem.Name + ", " + $objItem.MaxClockSpeed + " Mhz"
$cpu2=$objItem.Version + ", Level " + $objItem.Level
$i=$objItem.UpgrActive DirectoryeMethod
$ii=$cpu_socket[$i]
$cp1=$objItem.ExtClock+"Mhz"
$cp2=$objItem.L2CacheSize+"Kb"
$lx5="<cpu> <processor> $cpu1 </processor>"
$lx6="<version> $cpu2 </version> "
$lx7="<socket> $ii </socket>"
$lx8="<external_clock> $cp1 </external_clock>"
$lx9="<l2_cache> $cp2 </l2_cache> </cpu> "
$t="$t $lx230 $en $lx232 $en $lx234 $en $lx236 $en $lx238 $en"
case $a=2
.....
End Select
Next
```

Опираясь на опыт, отметим, что целесообразно получать следующую информацию об аппаратном обеспечении оборудования удаленной рабочей станции:

- BIOS — его версия однозначно определяет модель материнской платы;
- процессор — полное имя, архитектуру, типоразмер (сокет), частоты и питание;
- оперативная память — помодульно определить занимаемый слот, типоразмер, скорость передачи данных и размер;
- жесткий диск — производитель, модель, интерфейс, емкость;
- оптические накопители — производитель, модель, скорость передачи данных (если во время тестирования в приводе находится диск);
- звуковой адаптер — производитель, модель;
- видеокарта — производитель, модель, размер памяти, установленное разрешение экрана, глубина цвета, частота обновления экрана;
- сетевой адаптер — производитель, модель, MAC-адрес, назначенные IP-адрес, маска подсети, шлюз.

## Сбор информации об учетной записи пользователя с помощью Acctive Directory

Имя учетной записи пользователя определяется с помощью встроенного в KIXTart макроса `@USERID`. Полученное значение участвует в SQL-запросе ADODB-соединения для поиска учетной записи в Active Directory, поскольку оно совпадает с полем `SamAccountName`. Поиск осуществляется по следующему сценарию:

- устанавливается соединение с Active Directory Provider через ADODB;
- составляется запрос, на основе которого будет осуществляться поиск;
- осуществляется поиск по заданным критериям.

Из Active Directory о пользователе извлекают следующую информацию: имя, отчество, подразделение, должность и телефон (листинг 11.4). Набор этих параметров может меняться в зависимости от специфики компании, в которой функционирует скрипт.

**Листинг 11.4. Чтение информации учетной записи пользователя**

```
$objRoot = GetObject("LDAP://RootDSE")
$strDefaultDomainNC = $objRoot.Get("DefaultNamingContext")
$strGetArg=@userid ; определение имени пользователя.
$strADSQuery = "SELECT department, physicaldeliveryofficename,
telephonenumber, title FROM 'LDAP:// " + $strDefaultDomainNC +
"' WHERE samAccountName = '" + $strGetArg + "'"
$objACTIVE DIRECTORYOConn = createObject("ADODB.Connection")
$objACTIVE DIRECTORYOConn.Provider = "ACTIVE DIRECTORYsDSOObject"
$objACTIVE DIRECTORYoConn.Open ("Active Directory Provider")
$objACTIVE DIRECTORYOCommand = CreateObject("ADODB.Command")
$objACTIVE DIRECTORYOCommand.ActiveConnection = $objACTIVE DIRECTORYOConn
$objACTIVE DIRECTORYOCommand.CommandText = $strACTIVE DIRECTORYSQuery
$objQueryResultSet = $objACTIVE DIRECTORYOCommand.Execute
$objRoot_m1=$objQueryResultSet.Fields("department")
$objRoot_m2=$objQueryResultSet.Fields("physicaldeliveryofficename")
$objRoot_m3=$objQueryResultSet.Fields("telephonenumber")
$objRoot_m4=$objQueryResultSet.Fields("title")
```

Вместо названия свойства, которое необходимо прочитать, можно указать порядковый номер поля, под которым оно обозначено в запросе. Поля

отсчитываются с 0. Таким образом, основываясь на приведенном примере, вместо:

```
$objRecordSet.Fields("serverName").Value
```

можно записать:

```
$objRecordSet.Fields(1).Value
```

## Формирование файла отчета в формате XML

KIXTart обладает встроенной поддержкой INI-файлов, однако этот формат файлов устарел. Целесообразно использовать XML-файлы или сразу импортировать данные в SQL. Как и HTML, XML является независимым от платформы стандартом. Полная спецификация XML находится в сети Интернет по адресу **http://www.w3c.org/xml**.

Внешне XML-документ очень похож на HTML-документ, т. к. XML-элементы также описываются с помощью ключевых слов — тегов. В отличие от HTML, в XML пользователь может самостоятельно создавать собственные элементы, поэтому набор тегов не предопределен. Теги XML определяют структурированную информацию файла.

Синтаксис элементов (тегов), составляющих структуру XML-файла, в упрощенном виде можно представить следующим образом — листинг 11.5.

**Листинг 11.5. Оформление тега XML**

```
<Element>
[attr1="val1" [attr2="val2"…]]
</Element>
```

При создании XML-документов (листинг 11.6) необходимо руководствоваться следующими положениями:

- документ состоит из элементов разметки (`markup`) и данных (`content`);
- все элементы описываются с помощью тегов;
- в заголовке документа должны быть идентификационные данные: используемый язык разметки, его версия, кодировка страницы и т. д. (описывается с помощью тега `<% . . . . . %>`);
- XML чувствителен к регистру символов;
- значения атрибутов, используемых в определении тегов, должны быть заключены в кавычки.

**Листинг 11.6. Структура XML-файла в общем виде**

```
<?xml version="1.0" encoding="windows—1251" ?>
<main_element>
<Element1>
[attr1="val1" [attr2="val2"…]]
</Element1>
<Element2>
[attr1="val1" [attr2="val2"…]]
</Element2>
……….
</main_element>
```

Формирование отчета состоит из двух частей: накопления данных в переменную и запись значения переменной в XML-файл (листинг 11.7). Каждый XML-файл имеет заголовок, в котором содержится версия используемого языка и кодировка. Если кодировка не указана, то при наличии русских символов, созданный файл не будет открываться и обрабатываться ни одним из браузеров.

Название файла отчета может совпадать с именем рабочей станции или с логином пользователя. Из двух возможных решений рекомендуется использовать в качестве имени файла название рабочей станции, потому что с одной рабочей станции в сети может зарегистрироваться только один пользователь, в то время как один и тот же пользователь может зарегистрироваться на нескольких рабочих станциях. Таким образом, чтобы системный администратор получил исчерпывающую информацию о каждой из рабочих станций, о свойствах учетных записей пользователей и др., в имени файла и главного тега рекомендуется использовать имя рабочей станции, которое возвращает макрос `@wksta`. Ранее говорилось о том, что в синтаксисе XML-файла есть ограничение — тег не может начинаться с цифр, поэтому если в сети рабочие станции имеют цифровые наименования, например `1130pc`, `2310nb` (`pc` — Personal Computer, `nb` — Notebook), то главный тег рекомендуется предварять каким-нибудь символом, например символом подчеркивания — "_". Также в названии тегов необходимо учитывать регистр.

**Листинг 11.7. Создание XML-файла на KIXTart**

```
$en=chr(13)+chr(10) ' переход на новую строку
; накопление переменной
$t=""<?xml version=""1.0"" encoding=""windows—1251"" ?>"
```

```
$t=$t+"<_@wksta>"+$en
$t=$t+""<Parameter1> @ProductType </ Parameter1>"+$en
$t=$t+"< Parameter2> "+var1+ "</ Parameter2>"+$en
……
$t=$t+"</_@wksta>"
; формирование XML-файла
$filename=@wksta+".xml"
$fso = CreateObject("Scripting.FileSystemObject")
 $MyFile = $fso.CreateTextFile($filename, True, TRUE)
 $MyFile.WriteLine($t)
 $MyFile.Close
```

## Экспорт данных в SQL

После того как на SQL-сервере созданы соответствующие собираемой информации таблицы, настроены их связи и назначены необходимые права доступа, можно приступить к экспорту данных в SQL.

Запись собранной информации в SQL осуществляется с помощью ADODB-соединения *(см. главу 9)* .

Для экспорта данных в SQL-базу сначала необходимо установить соединение с базой данных, после этого идет проверка на существование записываемого параметра в базе. Если параметр существует, то его значение обновляется, если отсутствует, то в таблице создается новая запись.

Рассмотрим ключевые операции, чтобы создать сценарий по указанному алгоритму:

- подключение к базе данных;
- чтение и проверка значений в таблице базы данных;
- обновление записей в таблице базы данных;
- создание новой записи в таблице базы данных.

### Подключение к базе данных

Подключение к базе данных осуществляется с помощью ADODB-соединения (листинг 11.8), при этом необходимо знать имя сервера, на котором расположена SQL-база, ее название, имя таблицы. Если не включена Windows-идентификация, то следует указать имя и пароль пользователя, от имени которого будут осуществляться операции с таблицами. Выбор способа доступа к базе на SQL-сервере находится в ведении администратора баз данных.

**Листинг 11.8. Подключение к SQL-базе с помощью ADODB-соединения**

```
$obj=createobject("ADODB.connection")
$str="driver={sql serv-
er};server="+ServerName+";trusted_connected=yes;database=+"BaseName"
$obj.open($str)
$obj.cursorlocation=3
```

Где **ServerName** — имя SQL-сервера.

**BaseName** — имя SQL-базы, хранящейся на сервере.

## Операции с SQL-таблицей

После подключения к базе данных выполняется одно из следующих действий с таблицей: чтение, запись, удаление или обновление. Операция, которая выполняется с таблицей, зависит исключительно от SQL-запроса (табл. 11.4), остальная часть скрипта одинакова (листинг 11.9).

**Листинг 11.9. Чтение поля `fields` в таблице `users` из SQL-таблицы**

```
$rs=createobject("ADODB.recordset")
$query="select * from users"
$rs.open str_q, a, 1, 2, 1
$rs.movefirst
Do Until $rs.EOF
        $rs.Fields("fields").Value
        $rs.MoveNext
Loop
```

***Таблица 11.4.*** *Операции с SQL-таблицей*

| Операция | SQL-запрос | Комментарий |
|---|---|---|
| Чтение | `QueryString ="select * from "+TableName` | В таблице `TableName` осуществляется чтение всех колонок |
| Запись | `QueryString = "insert into " + TableName + " (" + SQLfield1 + "," + SQLfield2 + "," + SQLfield3 + ") values ('" + value1 + "','" + value2 + "','" + value3 + "')"` | В таблицу `TableName` добавляется три записи в колонки `SQLfield1`, `SQLfield2`, `SQLfield3`, куда, соответственно, записываются значения `value1`, `value2` и `value3` |

***Таблица 11.4*** *(окончание)*

| Операция | SQL-запрос | Комментарий |
|---|---|---|
| Удаление | `QueryString = "delete from " + TableName + " where " + SQLfield1 + "='" + persona + "' And " + SQLfield2 + " like '" + letters + "%'"` | Из таблицы `TableName` удаляются все строки, удовлетворяющие 2-м условиям: значение столбца `SQLfield1` совпадает со значением `persona` и значение в столбце `SQLfield2` совпадает с `letters` |
| Обновление | `QueryString = "update " + TableName + " (" + SQLfield1 + "," + SQLfield2 + "," + SQLfield3 + ") values ('" + value1 + "','" + value2 + "','" + value3 + "') where " + SQLfield4 + "=" + value4` | В таблице `TableName` обновляются поля `SQLfield1`, `SQLfield2`, `SQLfield3` значениями `value1`, `value2`, `value3`, где поле `SQLfield4` принимает значение `value4` |

# Решение задачи подключения сетевых принтеров

Идея подключения принтеров основана на членстве учетных записей пользователей в соответствующих группах безопасности. Если пользователь включен в группу, имя которой построено по определенным правилам, то при загрузке компьютера скрипт подключит ему необходимый сетевой принтер, при этом список локальных принтеров останется прежним. Если пользователь исключен из этой группы системным администратором, а принтер подключен, то скрипт отключит ему этот принтер. Основная задача скрипта управления сетевыми принтерами — поддерживать список подключенных принтеров актуальным.

## Соглашение об именах групп безопасности и принтеров

Имя должно содержать как можно больше информации о принтере, при этом оно должно быть удобным для использования. На рабочих станциях под управлением операционной системы Windows, пользователь имеет дело с двумя именами — локальным и сетевым именем принтера. Локальное имя принтера назначается ему во время установки. Его длина ограничивается 220 символами. Сетевое имя назначается принтеру для использования уст-

ройства в сети. Максимальная длина сетевого имени составляет 80 символов, хотя его не рекомендуется делать длиннее 8 символов для обеспечения совместимости с клиентами MS-DOS и Windows 3.x. Существует еще одно ограничение: некоторые приложения не могут работать с принтерами, у которых полное составное имя (имя компьютера, объединенное с сетевым именем принтера по шаблону `\\Server\ ShareName`) длиннее 31 символа.

Чаще всего, имя, назначаемое принтеру, представляет собой реальное имя принтера с порядковым номером, если есть несколько принтеров одинаковой модели, например: `HP LaserJet 1200 (1)`, `HP LaserJet 1200 (2)`. Такой способ именования рекомендуется использовать в небольших организациях. В крупных корпорациях принцип именования принтеров может быть другим. Например, названия могут быть даны по именам отделов, офисов или филиалов, где территориально находится принтер, например: `Краснодар ОКС` или `Ростов АТС-34`.

Сетевое имя, как было отмечено ранее, должно быть более коротким, но при этом не должно терять смысловой нагрузки. Оно чаще всего представляет собой общую характеристику принтера, например: `HP1200_1`, `HP1200_2`.

## Предварительная настройка принтера и Active Directory

Работа сценария строится на анализе и обработке данных, содержащихся в Active Directory и пользовательском реестре. На основе полученных данных осуществляется подключение и отключение принтера в зависимости от членства пользователя в соответствующих группах безопасности. Для того чтобы сценарий мог считывать и сопоставлять полученные данные из Active Directory, необходимо одновременное выполнение нескольких условий.

- Названия принтеров и групп безопасности должны удовлетворять соглашению об именах.
- Сетевые принтеры, подключением и отключением которых должен управлять скрипт, должны быть опубликованы в Active Directory.
- Названия групп безопасности должны строиться в соответствии со следующим шаблоном: название одной из групп, члены которой могут только выводить задания на печать, образуется добавлением к сетевому имени принтера через дефис слова `Print`, например: `HP2300_1 - Print`. Название другой группы строится аналогично, с той разницей, что слово `Print` заменяют на словосочетание `Print Managers`. Члены этой группы могут управлять очередью печати и принтером. Таким образом, принтеру с сетевым именем `HP1200_1` соответствуют следующие названия групп: имя

первой группы `HP1200_1 - Print`, второй `HP1200_1 - Print Managers` (рис. 11.2).

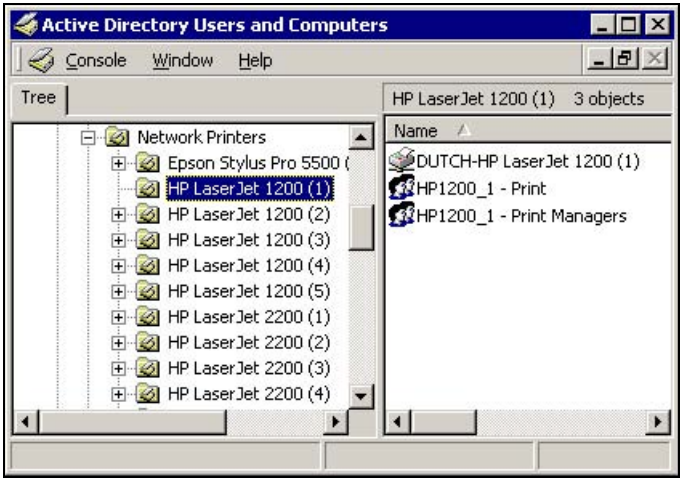



**Рис. 11.2.** Именование принтеров и групп в Active Directory

- Параметры безопасности принтера должны быть определены. В свойствах принтера (рис. 11.3) на сервере печати на вкладке **Security** (Безопасность) должна быть удалена группа `Everyone` и добавлены две группы безопасности, соответствующие данному принтеру: `HP1200_1 - Print` и `HP1200_1 - Print Managers`. В противном случае скрип будет подключать этот принтер всем пользователям сети. Для группы `HP1200_1 - Print` в разделе **Permissions** должен быть установлен флажок напротив свойства **Print** (рис. 11.3), а для группы `HP1200_1 - Print Managers` — флажки напротив **Print** и **Manage Documents**. Ставить флажок напротив **Manage Printers** не рекомендуется, поскольку управление принтерами подразумевает возможность изменять настройки принтера, удалять его.

Работа сценария строится на анализе данных, содержащихся в Active Directory и реестре пользователя. Его работу можно разбить на три этапа:

1. Формирование списка принтеров, которые должны быть подключены пользователю.
2. Формирование списка принтеров, установленных на рабочей станции пользователя.
3. Приведение списков в соответствие.

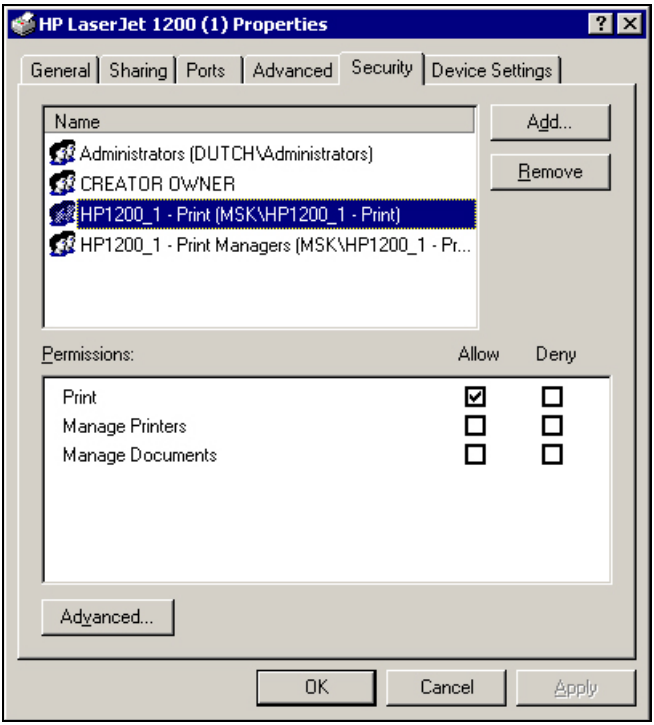



**Рис. 11.3.** Параметры безопасности принтера на сервере

## Формирование списка принтеров, которые необходимо подключить пользователю

Поиск объектов в Active Directory с помощью провайдера LDAP реализуется через ADODB-соединение (листинг 11.10). После создания соединения формируется SQL-запрос и осуществляется поиск по заданным критериям. Результат поиска — массив, содержащий значения полей, которые указаны в параметре `SELECT` SQL-запроса.

**Листинг 11.10. Поиск опубликованных принтеров в текущем домене**

```
$strADSQuery = "SELECT shortservername, portname,
servername, printername, printsharename, location, description FROM '"
+$domain_+"'  WHERE objectClass='printQueue'"
$objConnection = CreateObject("ADODB.Connection")
$objCommand = CreateObject("ADODB.Command")
$objConnection.CommandTimeout = 120
```

```
$objConnection.Provider = "ACTIVE DIRECTORYsDSOObject"

$objConnection.Open ("Active Directory Provider")
$objCommand.ActiveConnection = $objConnection
$objCommand.CommandText = $strACTIVE DIRECTORYSQuery
$st = $objCommand.Execute
$st.Movefirst
$i=0

Do
$server_enum=""
$name_enum=""
$shares_enum=""
$description_enum=""
       $server_enum = $St.Fields("shortservername").Value
       $name_enum = $St.Fields("printername").Value
       $shares=$St.Fields("printsharename").Value
              for each $share in $shares
              $shares_enum = $shares_enum + $share
              next
       $descrs=$St.Fields("description").Value
              for each $desc in $descrs
              $description_enum = $description_enum + $desc
              Next
$st.MoveNext
$temp="Название принтера: " & $name_enum & chr(13) & "Путь к принтеру: "
& "\\" & $server_enum & "\" & $shares_enum & chr(13) &  "Описание: " &
$description_enum.
MessageBox($temp,"Характеристики принтера",0,0)
$temp=""
Until $st.EOF
```

В Active Directory объектом класса `printQueue` является принтер (табл. 11.5). Этот объект имеет свойства, значения которых могут быть двух типов: строкой и массивом. В приведенном примере поле, содержащее название принтера, является строковой переменной, а сетевое имя принтера — массивом.

***Таблица 11.5.*** *Характеристики полей принтера класса* `printQueue`

| Поле | Описание | Тип | Пример |
|---|---|---|---|
| `Description` | Описание принтера | `Array` | `Принтер формата А4` |
| `Location` | Физическое место размещения принтера | `String` | `2 эт., 10 комн.` |
| `PrinterName` | То же, что и `Name` | `String` | `PrinterName` |
| `PrintShareName` | Имя принтера для подключения | `Array` | `Printer` |
| `ServerName` | Полное имя сервера, к которому подключен принтер | `String` | `Server.Domain.Ru` |
| `ShortServerName` | Краткое имя сервера | `String` | `Server` |

После того как в Active Directory найден очередной опубликованный принтер и прочитаны его свойства, для него формируется UNC-путь (`\\server\sharename`). Затем осуществляется попытка подключить пользователю принтер и считывать код функции, производящей подключение.

Если функция подключения возвращает код ошибки 0 (подключение к принтеру прошло успешно), то пользователь является членом одной из двух групп безопасности, перечисленных в свойствах принтера на сервере печати. Для тех принтеров, на которые пользователь имеет право печатать, формируется массив (листинг 11.11), например, `$access_array[$i]`. Формат элементов массива следующий: `",,server,printername"`, где `server` — короткое имя сервера, `printername` — локальное имя принтера.

**Листинг 11.11. Формирование массива подключенных принтеров**

```
$path_enum_connect = "\\" + $server_enum + "\" + $shares_enum
$connect_flag = Active Directorydprinterconnection( $path_enum_connect )
if $connect_flag=0
$path_full =",," + $server_enum + "," + $name_enum
$print_sysinfo=$print_sysinfo+$shares_enum+": "+$description_enum+chr(13)
$access_array[$i] = lcase($path_full)
$i=$i+1
Endif
```

## Формирование списка сетевых принтеров, подключенных пользователю

Процесс определения подключенных пользователю сетевых принтеров основан на анализе ветви HKCU локального реестра (рис. 11.4).

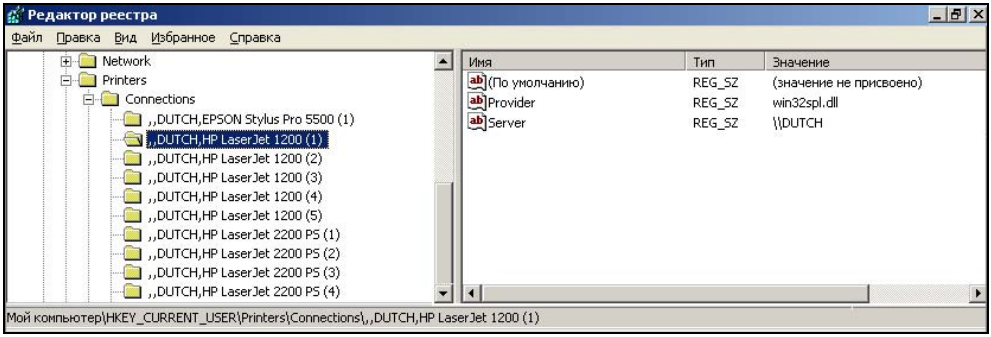



**Рис. 11.4.** Список установленных сетевых принтеров на рабочей станции

С помощью функции ENUM() осуществляется чтение названий папок, содержащих короткое имя сервера и полное имя принтера (листинг 11.12). На основе полученной информации формируется массив, элементами которого являются строки, имеющие следующий формат: ,,server,printername, где server — короткое имя сервера, printername — локальное имя принтера. Для удобства сравнения обоих массивов (подключенных принтеров и принтеров, на которые пользователь имеет права) необходимо, чтобы форматы элементов массивов совпадали. Формат элементов продиктован особенностью построения реестра Windows 2000.

**Листинг 11.12. Чтение параметров сетевых принтеров из реестра локальной рабочей станции**

```
$Index=0
DO
$connected_array[$index]= lcase(ENUMKEY("HKEY_CURRENT_USER\Printers\
Connections\", $Index))
$Index = $Index + 1
UNTIL Len($Group) =0
```

После формирования второго массива, между ними соблюдается следующее неравенство: M2≥M1, где M2 — массив, элементами которого являются назва-

ния подключенных принтеров, `M1` — массив, элементами которого являются названия принтеров, на которые пользователь имеет права. На третьем, заключительном этапе, добиваются выполнения следующего условия: `M1=M2`.

## Приведение списков принтеров в соответствие

Сопоставление массивов `M1` и `M2` осуществляется с помощью функции `ASCAN()` (листинг 11.13). Если функция возвращает значение `-1`, то элемент, найденный в одном массиве, не является элементом другого, поэтому принтер, соответствующий этому элементу, должен быть отключен.

Удаление принтера осуществляется с помощью соответствующей функции, параметром которой является UNC-путь принтера. Для того чтобы сформировать этот путь, осуществляется анализ ветви `HKLM` (рис. 11.5).

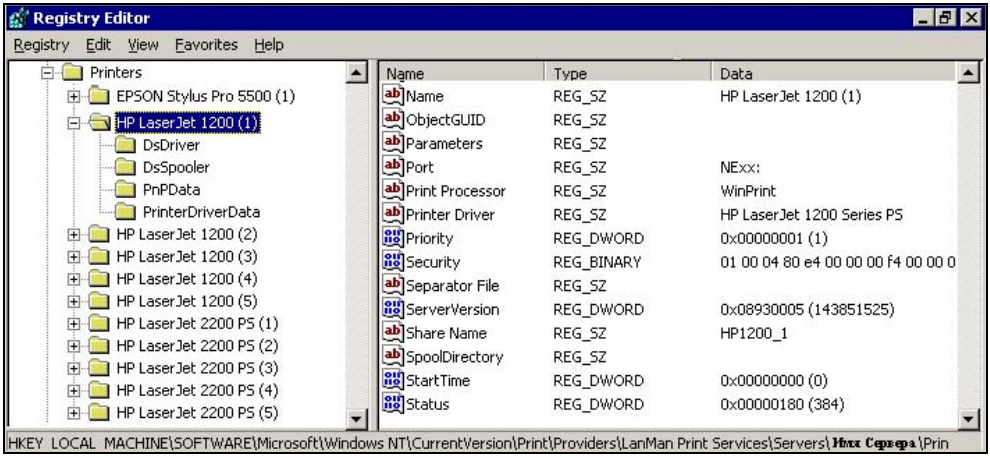



**Рис. 11.5.** Ветвь реестра, содержащая характеристики сетевых принтеров

**Листинг 11.13. Сопоставление списков принтеров**

```
for $i=0 to ubound($connected_array)
$flag_p=0
$flag_p=Ascan($access_array,$connected_array[$i])
       if $flag_p=-1
       …
       endif
```

```
      next

              if $flag_p=-1
$group=$connected_array[$i]
$name_=right($group, len($group)-instrrev($group,","))
$server_=right(left($group,len($group)-len($name_)-1),len(left($group,
len($group)-len($name_)-1))-2)
$share_=reActive Directoryvalue("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
Windows NT\CurrentVersion\Print\Providers\LanMan Print Services\Servers\
"+$server_+"\Printers\"+$name_,"Share Name")
$disconnect_ ="\\"+$server_+"\"+$share_
$r=DelPrinterConnection( $disconnect_ )
              endif
```

# Решение задачи подключения сетевых дисков

Для хранения информации в AD о подключаемом ресурсе был использован стандартный объект `SharedFolder`, содержащий информацию о букве, на которую подключается сетевой ресурс, UNC-путь к сетевому ресурсу, описание сетевого ресурса, которое фигурирует в папке Мой компьютер, имя группы безопасности, членам которой будет подключен ресурс.

Опишем механизм подключения одного из сетевых ресурсов, на примере общей папки подразделения Otdel1. Пусть папка имеет следующий сетевой путь — `\\Esmiralda\Work$\Otdel1`, подключается к диску K и имеет описание `Служебные файлы` (рис. 11.6). В AD в любой `OU`, например, `Shares` создайте объект `Share Folder` (рис. 11.7). В любой другой `OU` создайте группы безопасности, имена которых совпадают с URL сетевого ресурса после последней "точки". В приведенном примере, группа, соответствующая подключаемому диску, должна называться `Otdel1`. Членам этой группы будет подключен сетевой ресурс при условии, что она будет добавлена на вкладку безопасности папки Otdel1 с соответствующими правами на доступ к этой папке. Если необходимо сделать несколько различных уровней доступа, в названии группы можно использовать префикс.

Каждому подключаемому скриптом сетевому ресурсу соответствуют два объекта AD: опубликованная сетевая папка и соответствующая ей группа безопасности.

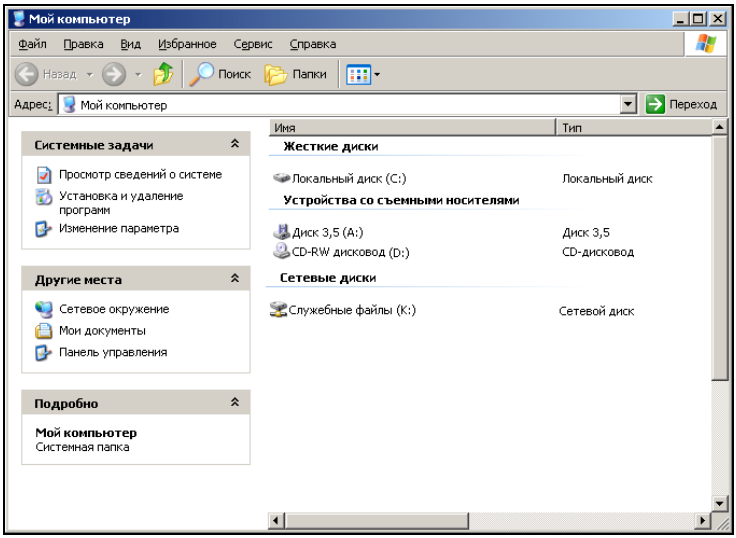



**Рис. 11.6.** Переименование диска в папке Мой компьютер

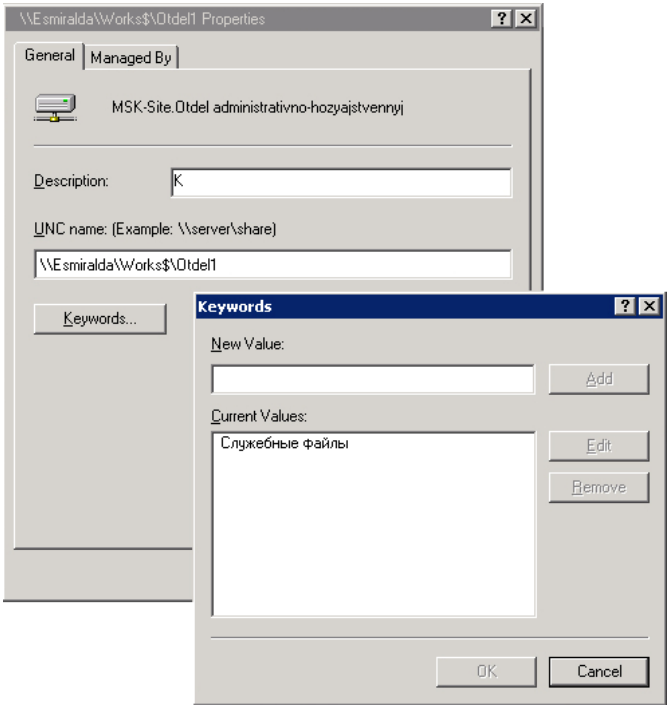



**Рис. 11.7.** Описание свойств подключаемого диска в AD

Если ресурс должен быть доступен всем пользователям сети, то необходимо в соответствующую ресурсу группу добавить группу `domain users` с надлежащими правами.

Сценарий можно условно разбить на несколько логических частей:

1. Определение списка групп, в которые входит текущий пользователь.
2. Подключение к AD и чтение значений соответствующих полей.
3. Отключение всех обрабатываемых скриптом сетевых дисков.
4. Подключение необходимых сетевых дисков.
5. Корректировка описания сетевых дисков в папке `Мой компьютер`.

## Определение членства в соответствующих группах безопасности

Определение членства в соответствующих группах безопасности осуществляется с помощью встроенной функции `EnumGroup()` (листинг 11.14). Список групп, возвращаемый этой функцией (переменная `$Group`), имеет следующий шаблон: `DOMAIN\GROUP`. Поскольку в AD группы хранятся без префикса (в данном случае префиксом является короткое имя домена), то полученный список групп необходимо преобразовать. Результат рекомендуется записать в ту же переменную, т. е. `GROUP`.

**Листинг 11.14. Определение списка групп, в которые входит пользователь**

```
$i=0
Do
        $Group=Right($Group, Len($Group)-InstrRev($Group,"\")
        $i=$i+1
Until Len($Group) = 0
```

## Чтение данных из AD

Подключение к AD реализовано с помощью ADODB-соединения. Поиск необходимых объектов осуществляется с помощью SQL-запроса. В качестве фильтра указывается `ObjectClass='volume'`:

```
"SELECT uname,  keywords, description, cn FROM '" +$domain+"'  WHERE
objectClass='volume'"
```

Где $domain — имя текущего домена, который определяется чтением глобального каталога RootDSE. Для его чтения достаточно прав обычного пользователя. Переменная $domain имеет вид:

```
DC=domain,DC=com
```

Чтобы правильно составить SQL-запрос, понадобятся теоретические знания объектной модели Shared Folder. Для удобства они сведены в табл. 11.6.

***Таблица 11.6.*** *Описание используемых полей объекта* `Shared Folder`

| Название поля | Тип данных | Значение |
|---|---|---|
| Description | Массив | Буква, на которую будет подключаться ресурс |
| Cn | Строка | Имя, отображаемое в AD. Используется для определения объекта подключения |
| UNCName | Строка | UNC-путь к сетевому ресурсу |
| Keywords | Строка | Имя диска, фигурирующее в папке Мой компьютер |

Чтение полей всех опубликованных сетевых ресурсов (листинг 11.15) осуществляется с помощью цикла Do…Until:

```
$st.Movefirst
Do
…
        $st.MoveNext
Until $st.EOF
```

Навигация по объектам, отобранным в результате SQL-запроса, осуществляется с помощью функций MoveFirst() и MoveNext(). Признаком конца списка является EOF.

**Листинг 11.15. Чтение полей `Shared Folder` в AD**

```
$st.Movefirst
Do
$cn=$St.Fields("cn").Value
$uncname=$St.Fields("uncname").Value
$ds_s=""
        $dss=$St.Fields("description").Value
                for each $ds in $dss
                $ds_s=$ds_s + $ds
```

```
                Next
$keyw_s=""
        $kss=$St.Fields("keywords").Value
                for each $ks in $kss
                $keyw_s=$keyw_s+cstr($ks)
                Next
…
$st.MoveNext
Until $st.EOF
```

## Отключение сетевых дисков

Отключение всех обрабатываемых скриптом сетевых дисков осуществляется с помощью команды `USE` (листинг 11.16).

**Листинг 11.16. Отключение всех сетевых дисков**

```
        $st.Movefirst
        Do
        $ds_s=""
        $dss=$St.Fields("description").Value
                for each $ds in $dss
        $ds_s=$ds_s + $ds
                Next

                use $ds_s+":" /delete /persistent
        $St.MoveNext
        Until $St.EOF
```

Переменная `$ds_s` содержит значение поля, `description` — буква, на которую монтируется диск. Благодаря такому механизму, скрипт управляет только теми сетевыми дисками, которые используются системным администратором в AD.

## Подключение необходимых сетевых дисков

Подключение дисков осуществляется по следующему алгоритму (листинг 11.17). Определяется необходимая для подключения сетевого диска информация, а именно буква, на которую монтируется диск (поле `description`); UNC-путь ресурса (поле `UNCName`); группа, членам которой будет подключен ресурс (поле `cn`).

Чтение поля `CN` обязательно, т. к. сетевой ресурс может быть двух видов: общим и индивидуальным. Приведем два примера. В первом случае осуществ-

ляется подключение общей папки обмена данными, назовем ее `Exchange`. Исходя из этого папка, к которой предоставлен доступ, называется `Exchange`, соответствующая ей группа безопасности — `Exchange`, и поскольку в эту группу входят все пользователи, то членом этой группы является группа `Domain Users`. Во втором случае, нам необходимо подключить каждому пользователю домашний каталог. Для этого необходимо создать папку, например, `HomeDirs` и создать в ней подкаталоги, соответствующие логинам пользователей. Ситуация изменилась — сетевой путь к папке использовать нельзя, однако есть поле `CN` — имя ресурса, где можно записать всю необходимую информацию: имя пользователя и соответствующую группу безопасности.

Теперь необходимо выяснить, входит ли пользователь в соответствующую ресурсу группу безопасности, и при положительном исходе проверки приступить к подключению ресурса по одному из двух алгоритмов. Выбор алгоритма зависит от наличия логина в считанном значении `CN`.

**Листинг 11.17. Подключение сетевого диска с помощью команды `Use`**

```
; элементами массива $usergroup_name[] являются группы,
; членами которых является текущий пользователь
for $t=0 to ubound($usergroup_name)
if instr($usergroup_name[$t],$group_b)<>0
; критерием персонализации является членство
; в перечисленных группах PersonalGroup1…3
if instr(ucase($group_b),ucase("PersonalGroup1"))<>0  or
instr(ucase($group_b),ucase("PersonalGroup2"))<>0  or
instr(ucase($group_b),ucase("PersonalGroup3"))<>0
       if instr($cn,@userid)<>0
use $ds_s+":" $uncname
? " !!!!" + $ds_s+":" $uncname
       if @error=5 ; ошибка доступа к ресурсу
$error_code="1"
              $error_message=$error_message+"Не удалось подключить диск
"+$ds_s+" к ресурсу "+ $cn+chr(13)
EndIf
       endif
else
use $ds_s+":" $uncname
? " !!!!" + $ds_s+":" $uncname
if @error=5 ; ошибка доступа к ресурсу
```

```
                $error_code="1"
                $error_message=$error_message+"Не удалось подключить диск
"+$ds_s+" к ресурсу "+ $cn+chr(13)
EndIf
endif
endif
next
```

## Корректировка описаний дисков в папке Мой компьютер

Механизм переименования сетевых дисков в папке Мой компьютер лучше всего запускать после завершения процесса подключения самих сетевых дисков. Это связано с тем, что на подключение диска требуется время, которое зависит от местоположения рабочей станции, загруженности сервера и других факторов. Точно его рассчитать не представляется возможным, поэтому для ускорения работы скрипта лучше разделить блоки подключения дисков и смены описания с помощью функции-задержки. Действие функции основано на вычислении промежутка времени. С помощью нового макроса `@Ticks`, который возвращает количество миллисекунд с момента загрузки компьютера, определяется разность времен. При достижении указанного интервала во времени — задержки (листинг 11.18), осуществляется выход из цикла.

**Листинг 11.18. Функция задержки времени**

```
$delay_size=3 ;3 - количество секунд задержки
       $Q = @Ticks + $delay_size*1000
Do
Until @Ticks >= $Q
```

Механизмы смены описания в Windows 2000 и Windows XP различны. Характеристики всех типов дисков (сетевых, локальных и съемных) в Windows 2000 хранятся в ветви реестра:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Explorer\MountPoints
```

в Windows XP:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Explorer\MountPoints2
```

Рассмотрим каждый механизм подключения сетевых дисков отдельно. Начнем с Windows XP — он наиболее понятен и прост.

## Переименование описаний сетевых дисков для Windows XP

Точки монтирования сетевых дисков организованы по следующему принципу. В подпапке `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2` для сетевых дисков создаются подразделы, имена которых организованы по принципу:

```
## имя сервера # имя папки, к которой предоставлен доступ #
подпапка1 #.... # подпапкаX
```

В каждой из этих подпапок присутствуют 3 обязательных параметра, четвертый (`_LabelFromReg`) также должен быть создан (рис. 11.8). Для смены имени сетевого диска в папке Мой компьютер необходимо изменить значение строковой переменной `_LabelFromReg`:

```
WriteValue(cstr($keyw_path+"\"+$temp1),"_
LabelFromReg",cstr($keyw_s),"REG_SZ")
```

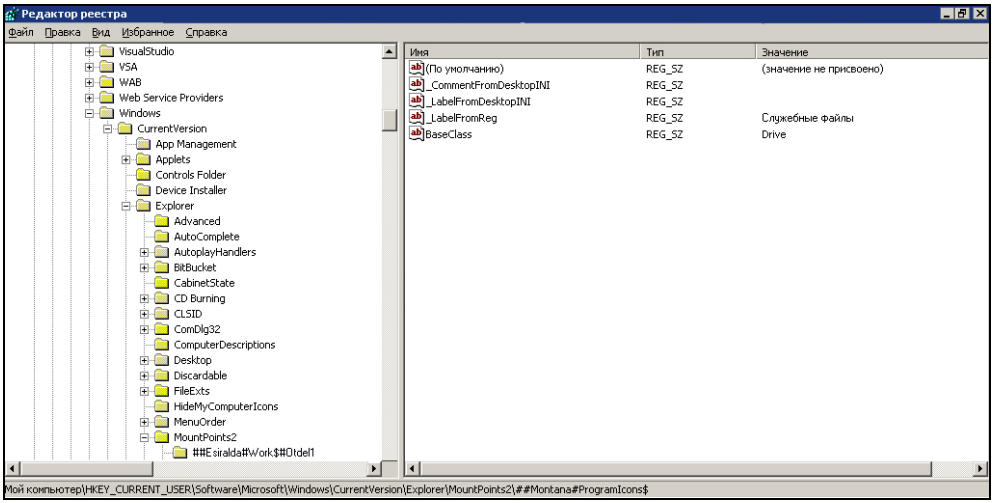



**Рис. 11.8.** Точка монтирования сетевых дисков в реестре Windows XP

Для определения переменной `$key_path` нужно вычислить названия папок, UNC-путь и заменить символ "\" на "#" (листинг 11.19).

**Листинг 11.19. Переименование описаний сетевых дисков**

```
$st.Movefirst
Do
$uncname=$St.Fields("uncname").Value
```

```
        …
$keyw_path="HKEY_CURRENT_USER\Software\Microsoft\Windows\
CurrentVersion\Explorer\MountPoints2"
        $uncname=Right($uncname,Len($uncname)-2)
        $keyw_array=split($uncname,"\")
        $temp1="#"
        For Each $ugu in $keyw_array
              $temp1=$temp1+"#"+$ugu
        Next
        WriteValue(cstr($keyw_path+"\"+$temp1),
"_LabelFromReg",cstr($keyw_s),"REG_SZ")

        $st.MoveNext
        Until $st.EOF
```

## Переименование описаний сетевых дисков для Windows 2000

Логика подключения сетевых дисков в Windows 2000 и XP сильно отличается. В ветви реестра `HKEY_CURRENT_USER\Software\ Microsoft\Windows\ CurrentVersion\Explorer\MountPoints` созданы подразделы (рис. 11.9), имена которых совпадают с буквой подключаемого сетевого диска. Внутри каждого из них существуют еще подразделы. Среди них обязательными являются `Autorun`, `_DIL`, `_LabelFromDesktopINI`. Для изменения описания диска в папке Мой компьютер в каждой из них необходимо создать два раздела, если они еще не существуют: `_GVI` и `_LabelFromReg`. В этих разделах при внесении изменений следует менять значение параметра `Version`, имеющего тип данных `REG_DWORD`. В подпапке `_LabelFromReg` дополнительно нужно создать ключ `Cache`, в котором в зашифрованном виде находится описание подключаемого сетевого диска. Тип данных параметра `Cache` — `REG_BINARY`. Опишем систему шифрования данного параметра на примере фразы "Служебные файлы", причем последняя буква "е" в первом слове пусть будет английская. Нам нужен пример любой английской буквы. Рассмотрим алгоритм обработки латинских и русских символов с преобразованием первой буквы выражения — русской заглавной буквы "С".

Процедура чтения кода символа осуществляется с помощью функции `ASC()`. Букве "С" соответствует номер 209 в таблице ASCII *(см. приложение 3. Таблица ASCII)*. Затем переведем полученное значение в шестнадцатеричное с помощью функции `DecToHex()`, параметром которой является

число — ASCII-код символа. Символу с ASCII с номером 209 соответствует шестнадцатеричное число D1. Полученное значение разбиваем на два символа. Буквенное значение первого символа преобразуют в цифру в соответствии с табл. 11.7, а для идентификации языка добавляют второй байт. Первым байтом являются два символа — HEX-код буквы.

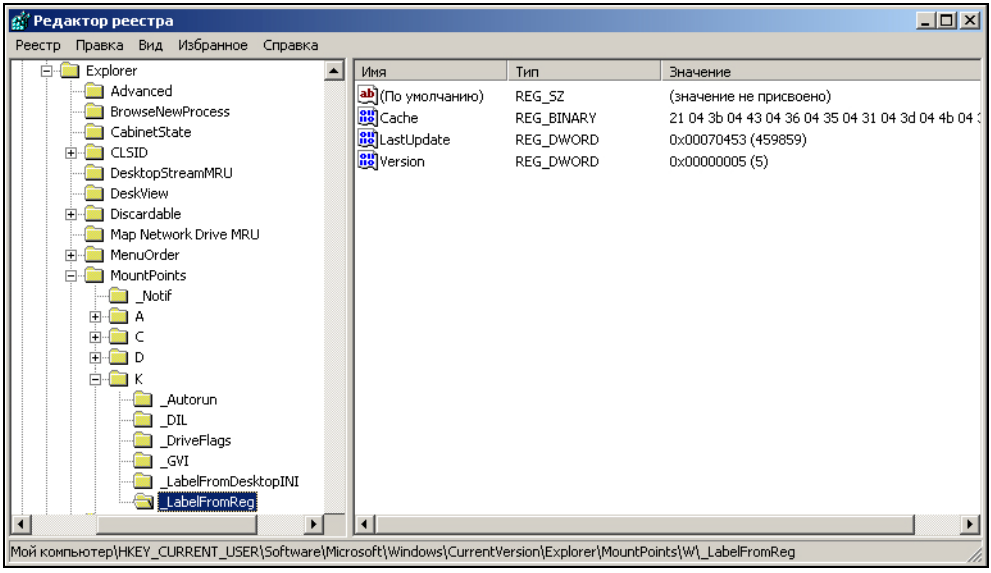



**Рис. 11.9.** Точка монтирования сетевых дисков в реестре Windows 2000

***Таблица 11.7.*** *Преобразование первого символа HEX-кода буквы*

| **Буква** | C | D | E | F | G |
|---|---|---|---|---|---|
| **Значение** | 1 | 2 | 3 | 4 | 5 |

Первому символу соответствует код "21" — это первый байт. Второй байт принимает значение в зависимости от языка: 04 — русский язык, 00 — английский. Он определяется по коду символа: если ASCII-код символа больше 128, то буква русская (см. таблицу ASCII в *приложении 3*), в противном случае — английская. Итак, второй байт символа равен "04", а сам символ — "21 04". В табл. 11.8 продемонстрировано преобразование фразы "Служебные файлы".

***Таблица 11.8.*** *Карта преобразования фразы "Служебные файлы"*

| Символ | Язык | ASCII | >128 | 2-й байт | HEX | Преобразование | 1-ый байт | 1+2 байт |
|---|---|---|---|---|---|---|---|---|
| С | Рус | 209 | + | 04 | D1 | D->2 | 21 | 21 04 |
| л | Рус | 235 | + | 04 | EB | E->3 | 3B | 3B 04 |
| у | Рус | 243 | + | 04 | F3 | E->3 | 33 | 33 04 |
| ж | Рус | 230 | + | 04 | E6 | E->3 | 36 | 36 04 |
| е | Рус | 229 | + | 04 | E5 | E->3 | 35 | 35 04 |
| б | Рус | 225 | + | 04 | E1 | E->3 | 31 | 31 04 |
| н | Рус | 237 | + | 04 | ED | E->3 | 3D | 3D 04 |
| ы | Рус | 251 | + | 04 | FB | F->4 | 4B | 4B 04 |
| е | Англ | 101 | – | 00 | 65 | 6->6 | 65 | 65 00 |
| _ | Пробел | 32 | – | 00 | 20 | 2->2 | 20 | 20 00 |
| ф | Рус | 244 | + | 04 | F4 | F->4 | 44 | 44 04 |
| а | Рус | 224 | + | 04 | E0 | E->3 | 30 | 30 04 |
| й | Рус | 233 | + | 04 | E9 | E->3 | 39 | 39 04 |
| л | Рус | 235 | + | 04 | EB | E->3 | 3B | 3B 04 |
| ы | Рус | 251 | + | 04 | FB | F->4 | 4B | 4B 04 |

Рассмотрим преобразование фразы с точки зрения программирования. Выражение состоит из множества символов, которые необходимо последовательно преобразовать (листинг 11.20).

**Листинг 11.20. Посимвольное преобразование фразы "Служебные файлы"**

```
$keyw_s="Служебные файлы"
For $c=1 to Len($keyw_s)
$symbol= cstr(dectohex(Asc(Right(Left($keyw_s,$c),1)))
Next
```

Переход от символа к символу осуществляется с помощью цикла `For…Next` и комбинации функций `Left()` и `Right()`. После преобразования переменная `$symbol` является шестнадцатеричным числом (HEX).

Отдельно получим первый и второй символы каждого HEX-числа и с первым символом, если его ASCII>128, выполним преобразование (см. табл. 11.8). В преобразовании первого символа шестнадцатеричного числа целесообразно использовать систему флагов и инструкцию `select…case`. Приведем пример преобразования первой буквы выражения — "C" (листинг 11.21). Как было определено ранее, этому символу соответствует 16-ричное значение D1, исходя из табл. 11.7, D должно преобразоваться в 2, а само число в 21. Флаговая система используется для определения алгоритма преобразования, в зависимости от HEX-кода символа.

**Листинг 11.21. Преобразование буквы "C" . Определение 1-го байта**

```
$string=ucase(left(cstr(dectohex(Asc(Right(Left($keyw_s,$c),1)))),1))
$string2=ucase(right(cstr(dectohex(Asc(Right(Left($keyw_s,$c),1)))),1))

If $flag_S=0

Select
Case instr(ucase($string),UCase("c"))<>0
$t=$t+cstr("1")+$string2
$flag_S=1
…
       EndSelect

       If $flag_S=0
              $t=$t+cstr(dectohex(Asc(Right(Left($keyw_s,$c),1))))
       endif
```

Определение 2-го байта символа осуществляется по ASCII-коду символа (листинг 11.22).

**Листинг 11.22. Преобразование буквы "C" . Определение 2-го байта**

```
If Asc(Right(Left($keyw_s,$c),1))>128
$t=$t+"04"
Else
$t=$t+"00"
Endif
```

Поскольку строка описания может иметь только 32 символа и обновляется исключительно записываемая часть, то во избежание оставления "хвостов" необходимо все остальные символы обнулить, затем записать полученные данные в реестр (листинг 11.23).

**Листинг 11.23. Запись преобразованных данных в реестр локальной машины**

```
For $r=1 to 32-len($keyw_s)
      $t=$t+cstr("00")
Next


WriteValue($keyw_path+"\"+$ds_s+"\_LabelFromReg", "Cache", $t, "REG_
BINARY")
$r_w1=ReadValue($keyw_path+"\"+$ds_s+"\_LabelFromReg","version")
WriteValue($keyw_path+"\"+$ds_s+"\_LabelFromReg", "Version", $r_
w1+1, "REG_DWORD")
$r_w2=ReadValue($keyw_path+"\"+$ds_s+"\_GVI","version")
WriteValue($keyw_path+"\"+$ds_s+"\_GVI", "Version", $r_w2+1, "REG_
DWORD")
```

В приведенном примере осуществляется наращивание версии в разделах `_LabelFromReg` и `_GVI`.

Тип определенной системы в KIXTart распознается с помощью макроса `@PRODUCTTYPE` (табл. 11.9).

***Таблица 11.9.*** *Описание возвращаемых значений макросом* `@PRODUCTTYPE`

| Название макроса | Возможные возвращаемые значения |
|---|---|
| `@PRODUCTTYPE` | "Windows 95" |
| | "Windows 98" |
| | "Windows Me" |
| | "Windows NT Workstation" |
| | "Windows NT Server" |
| | "Windows NT Domain Controller" |
| | "Windows 2000 Professional" |
| | "Windows 2000 Server" |
| | "Windows 2000 Domain Controller" |
| | "Windows XP Home Edition" |

***Таблица 11.9*** *(окончание)*

| Название макроса | Возможные возвращаемые значения |
|---|---|
| PRODUCTTYPE | "Windows XP Professional" |
| | "Windows XP Professional Tablet PC" |
| | "Windows Media Center Edition" |
| | "Windows Starter Edition" |
| | "Windows Server 2003" |
| | "Windows Server 2003 Domain Controller" |

Если в возвращаемом макросом значении присутствует комбинация символов 2000, то выполняется сценарий для Windows 2000, а для Windows XP должна быть комбинация XP (листинг 11.24).

**Листинг 11.24. Определение версии ОС**

```
If instr(ucase(@producttype), ucase("2000"))<>0
…; сценарий для Windows 2000
Else If  instr(ucase(@producttype), ucase("xp"))<>0
…; сценарий для Windows XP
EndIf EndIf
```

# Визуализация работы сценария KIXTart

Существует несколько способов визуализировать результат работы скрипта с помощью:

- стандартных диалоговых окон;
- сторонней надстройки KIXTart в виде DLL-библиотеки;
- сторонней утилиты, передающей параметры из KIXTart в HTML-файл.

## Визуализация работы скрипта с помощью стандартных диалоговых окон

Пользователю выводится информация на экран в виде сообщения. Этот метод рекомендуется использовать в начале сценария, чтобы уведомить пользователя о его начале работы. Основным недостатком этого метода визуализации является полное отсутствие интерактивности работы сценария, поэтому не рекомендуется использовать данный метод в каком-либо виде.

# Визуализация работы скрипта с помощью DLL-библиотеки

Визуализация и интерактивность работы скрипта реализуется с помощью специально созданной надстройки — KIXForms (**http://www.kixforms.org**). Программа существует в двух вариантах KIXForms Classic и .NET. Версия Classic была разработана одной из первых, и для ее успешного функционирования требуется DLL-файл, который должен быть зарегистрирован в реестре рабочей станции с помощью утилиты REGSVR32. Необходимость этой манипуляции сдерживала развитие этой утилиты. Несколько позже возможности KIXForms расширились, благодаря появлению .NET-версии, которая использует интегрированные в Windows библиотеки из Microsoft FrameWork. 1 сентября 2007 года проблема, связанная с необходимостью регистрации библиотеки, была решена утилитой KIXTart Script Packager(KIX2EXE) v 1.0.0.

## KIXForms

В настоящее время используется KIXForms Classic v2.46 и KiXforms.NET v3.01. Для регистрации библиотеки kixforms.dll необходимо с административными привилегиями выполнить команду `REGSVR32 /s kixforms.dll`. После регистрации библиотеки становится доступен объект `Kixtart.System`, вызываемый в листинге KIX-файла командой `$System = CreateObject("Kixtart.System")`. Объектная модель библиотеки описана в справочном файле, который можно загрузить с сайта **http://www.kixforms.org/assets/index.htm**.

Для разработки приложений на KIXForm удобно использовать KIXForms Designer .NET (**http://www.kixforms.org**).

## KIXTart Script Packager

KIX2EXE — бесплатная утилита, позволяющая создавать запускаемые файлы из KIXTart-скриптов. В настоящее время используется KIX2EXE v.1.2 (с сайта **http://kix2exe.ramonitor.nl**), обладающая следующими ключевыми возможностями:

- создание запускаемого EXE-файла;
- интеграция в EXE-файл утилиты KIX32.EXE или WKIX32.EXE;
- при использовании KIXForms интегрирование в архив Kixforms.dll с последующей временной регистрацией библиотеки в `HKCU` кусте реестра рабочей станции;
- подавление вывода окон, свидетельствующих о работе утилиты;
- поддержка листов заданий (job list);

- защита пакета файлов паролем;
- запуск файлов с имен разных учетных записей.

После загрузки и установки KIX2EXE на компьютер разработчика необходимо в папку C:\Program Files\Kix2Exe\Bin\Kix поместить файл KIX32.EXE и/или WKIX32.EXE. Версия KIXTart должна быть не ниже, чем KIXTart 2010 v4.51, поскольку, начиная с этой версии, осуществляется поддержка шифрования сценария паролем. Для использования KIXForms библиотеку kixforms.dll необходимо поместить в каталог C:\Program Files\Kix2Exe\Bin\KixForms, а при использовании утилиты KIX2EXE воспользоваться ключом `/kixforms`.

Утилита KIX2EXE может запускаться с параметрами командной строки. Приведем несколько примеров:

- компиляция KIX32EXE-сценария в исполняемый модуль:

```
c:\temp>kix2exe /script myscript.kix
```

- в компилируемый EXE-файл интегрируются файлы myscript.kix, file1.exe, file2.exe и kix32.exe:

```
c:\temp>kix2exe /script myscript.kix /include file1.exe,file2.exe
/kix kix32
```

Подробную информацию о всех возможностях утилиты KIX32EXE можно получить на сайте производителя по адресу **http://kix2exe.ramonitor.nl**.

## Визуализация работы скрипта с помощью DHTML

Этот способ наиболее прост для реализации визуализации и интерактивности работы скрипта в крупных сетях, поскольку утилита самодостаточна и представляет собой файл с расширением exe, который рекомендуется располагать в каталоге Netlogon, вместе со скриптом. Утилита KIXWin 1.1 (**http://www.kixtart.org**) вызывается из скрипта с набором параметров. Затем она передает эти параметры в DHTML-файл. KIXWin запускается от имени пользователя, регистрирующегося в сети, поэтому для формирования HTML-файла на основе DHTL-файла пользователям необходимо предоставить возможность чтения и записи в каталог, в котором находится DHTML-файл. Раздробление скрипта на две части является основным недостатком данного варианта. DHTML-файл вместе с сопутствующими ему файлами (GIF, JPEG, CSS) рекомендуется располагать в скрытой сетевой папке. DHTML-файл содержит сценарии, созданные с помощью VBScript или JScript.

## Синтаксис KIXWin

Приведем синтаксис утилиты и фрагмент DHTML-файла, отвечающий за чтение переведенных файлов. Синтаксис утилиты следующий:

```
kixwin "dialog" ["arguments"] ["options"]
```

Где `dialog` — строка, содержащая путь в формате URL к HTML-документу.

`arguments` — строка, содержащая параметры, передаваемые из KIX в HTML. Для разделения параметров в HTML-файле используют строку `window.dialogArguments.split(";")`. В данном примере разделителем параметров является ";".

`style` — строка, которая определяет оформление диалогового окна. Используются один или несколько из следующих параметров стиля:

```
dialogHeight:sHeight
dialogLeft:sXPos
dialogTop:sYPos
dialogWidth:sWidth
center:{ yes | no | 1 | 0 | on | off }
dialogHide:{ yes | no | 1 | 0 | on | off }
edge:{ sunken | raised }
help:{ yes | no | 1 | 0 | on | off }
resizable:{ yes | no | 1 | 0 | on | off }
scroll:{ yes | no | 1 | 0 | on | off }
status:{ yes | no | 1 | 0 | on | off }
unActive Directoryorned:{ yes | no | 1 | 0 | on | off }
```

Логическое разделение передаваемых параметров осуществляется с помощью заранее оговоренного символа. DHTML возвращает в KIX код ошибки (после обработки кода) в виде целого числа, макросу — `@ERROR`. В случае успешного завершения операции `@ERROR=0`.

Передача данных с помощью утилиты KIXWin осуществляется с помощью сценария загрузки следующим образом:

```
shell '%0/../kixwin.exe $html "$system_info ^ $hardware_info ^ Установленные программы: $en $prog ^ Подключенные сетевые диски:$en $n1 ^ Подключенные сетевые принтеры:$en $n2" "scroll:off;resizable:on"'
```

## Основы формирования файла DHTML

Параметры передаются DHTML-странице, содержащей вставки на VBScript. При загрузке страницы в теге `body` указывается функция (например, `onloActive Directory="startrun()"`), с помощью которой запускается таймер. В том случае, если таймер не прерван, то по его завершению осуществ-

ляется автоматическое закрытие окна и продолжение процесса загрузки компьютера (листинг 11.25).

**Листинг 11.25. Запуск функции таймера при загрузке DHTML-страницы**

```
<BODY onloActive Directory="startrun()" TEXT="BLACK" …>
…
<Script Language="JavaScript">
function startrun()
{
window.form1.textarea1.value="Процесс настройки компьютера завершен.\
nДля продолжения нажмите кнопку ПРОДОЛЖИТЬ или подождите 7сек. — окно
закроется само."
timer1=setTimeout('exit()', 20000);
}
…
</Script>
</BODY>
```

В текстовой области выводится сообщение о том, что настройка рабочей станции завершена и запускается таймер. Если ни одна кнопка на клавиатуре не нажималась в течение `20000 мс`, то осуществляется вызов функции `exit()`, которая закрывает окно браузера.

Кнопки, при нажатии которых отображается информация различного рода, представляют собой иллюстрацию с надписями. Она разделена на области. При нажатии на описанную область осуществляется запуск соответствующей функции, которая, в свою очередь, отображает переданную информацию из скрипта в текстовую область `window.form1.textarea1` (листинг 11.26).

**Листинг 11.26. Обработка фрагмента графической карты иллюстрации**

```
…
<img src="bar2.jpg" name="imgActive Directoryvert" usemap="#LogoTip"
border=0>
<map name="LogoTip">
<area onclick=system() alt="Сведения об операционной системе"
shape=circle coords="69,40,50">
<area onclick=hardware() alt="Сведения о компьютере" shape=circle
coords="166,40,50">
…
```

```
</map>
<Script Language="JavaScript">
…
function system()
{
clearTimeout(timer1);
var argv;
argv = window.dialogArguments.split("^");
document.all.parametr0 = argv[0];
window.form1.textarea1.value=''
window.form1.textarea1.value=document.all.parametr0
}
function hardware()
{
clearTimeout(timer1);
var argv;
argv = window.dialogArguments.split("^");
document.all.parametr1 = argv[1];
window.form1.textarea1.value=''
window.form1.textarea1.value=document.all.parametr1
}
…
```

При вызове любой функции осуществляется прерывание таймера функцией `clearTimeout(timer1)`, затем чтение соответствующего аргумента.

# Внедрение скрипта в эксплуатацию

Скрипт выполняется каждый раз при регистрации пользователя в сети, если он указан в разделе **Profile** свойств пользователя (рис. 11.10) службы Active Directory: Users and Computers.

Для автоматизации установки KIXTart на рабочих станциях домена следует в папку Netlogon поместить файлы KIX32.EXE, SCRIPT.KIX, START.BAT, KIXWIN.EXE.

Затем в скрытую сетевую папку скопировать DHTML-файл и все сопутствующие ему файлы.

В ходе выполнения файла START.BAT определяется тип операционной системы, установленной на рабочей станции, и запускается скрипт (листинг 11.27).

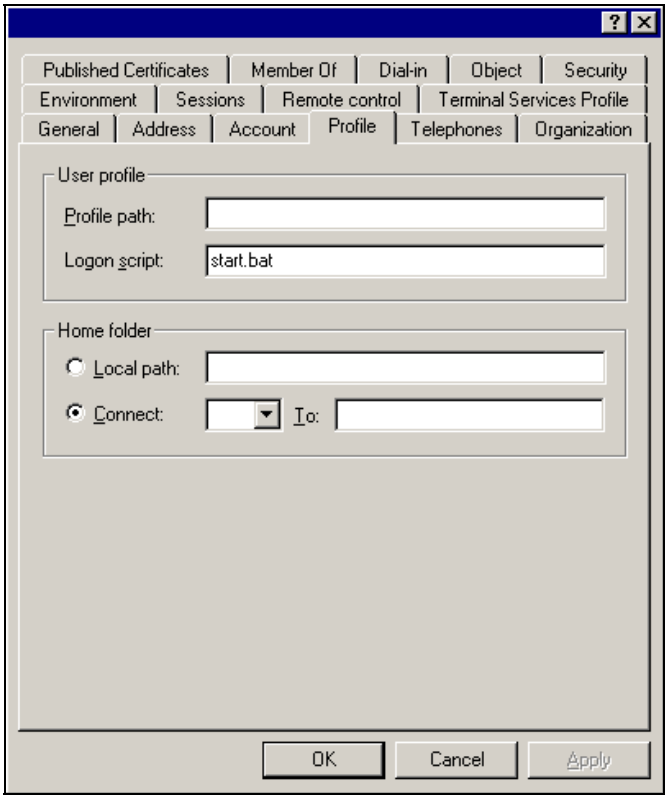



**Рис. 11.10.** Вкладка **Profile** в свойствах учетной записи пользователя в AD

**Листинг 11.27. Файл START.BAT**

```
@ECHO OFF
start /wait Kix32.exe Script.kix
@echo End Of Batch File
```

С помощью строки `start /wait Kix32.exe Script.kix` добиваются последовательной загрузки — сначала скрипта, затем рабочего стола и т. д., а не одновременно. Во время выполнения скрипта многозадачность "отключается". Это делается для того, чтобы до окончания действия скрипта дальнейшая загрузка операционной системы не производилась.

На время выполнения скрипта необходимо скрыть CMD-панель, в которой выполняется скрипт, и приостановить загрузку рабочего стола до окончания всего скрипта. Этого добиваются с помощью групповой политики, распространяющейся на домен (**Default Domain Controllers Policy**). В разделе

групповой политики **User Configuration** необходимо, соответственно, включить **Run legacy logon script synhronously** (запускать сценарий загрузки синхронно) и **Run legasy script hidden** (запускать сценарий скрыто). Для этого нужно выполнить следующие действия:

1. Зарегистрироваться на сервере с помощью учетной записи, имеющей административные права.
2. Загрузить в Active Directory: Users and Computers (**Start | Programs | Active Directory | Aministrative Tools**) и войти в свойства контроллера домена.
3. Перейти на вкладку **Group Policy** и загрузить **Default Domain Policy** (рис. 11.11).

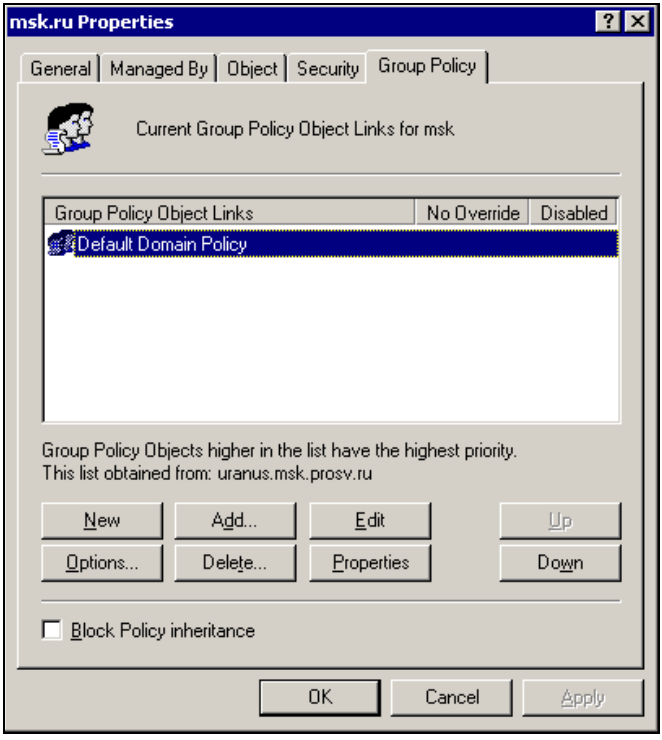



**Рис. 11.11.** Вызов консоли групповых политик домена

4. В загруженной групповой политике (**Default Domain Policy**) необходимо в настройках пользователя (**User Configuration**) войти в административные настройки (**Active Directoryministrative Templates**). Там выбрать

раздел **System** (система), вкладку **Logon/Logoff** (войти/выйти) и включить ранее оговоренные политики (рис. 11.12).

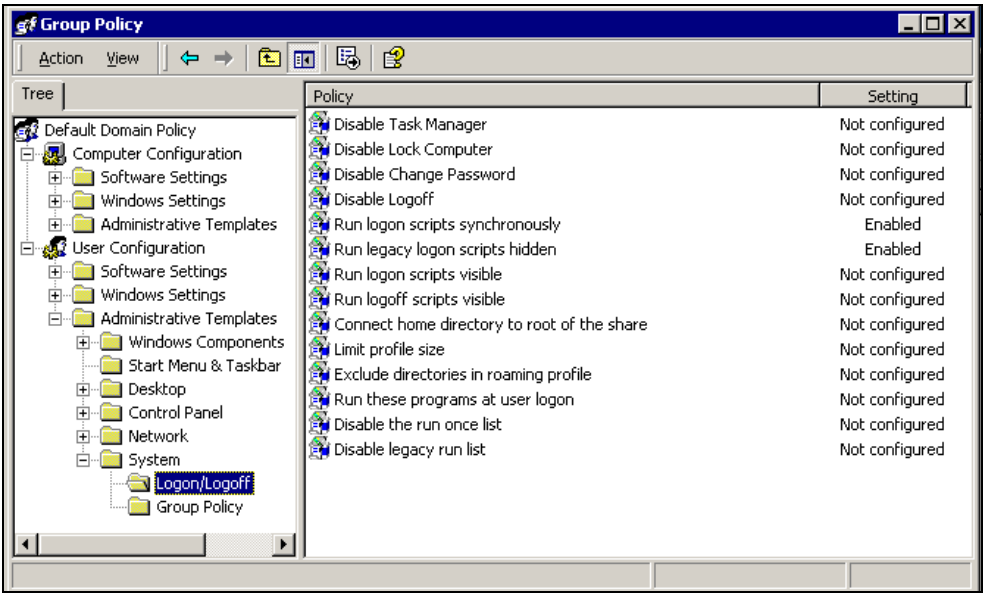
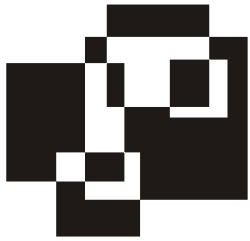



**Рис. 11.12.** Настройка групповых политик



# Глава 12

# Подготовка рабочей станции к функционированию в сети

Подготовка ПК к работе в сети заключается в установке операционной системы и необходимых программ, подключении рабочей станции к сети (введении в домен).

Указанный круг задач можно решить несколькими способами:

- ❒ клонирование разделов жесткого диска;
- ❒ создание сценария для автоматической установки ОС и необходимого ПО.

## Клонирование жестких дисков

Клонирование какого-либо объекта представляет собой процесс создания идентичного по своим свойствам и параметрам объекта. В компьютерной индустрии под термином "клонирование" понимают клонирование разделов жесткого диска рабочей станции.

Программы, предназначенные для клонирования жестких дисков, обладают следующими основными характеристиками:

- ❒ создание образа жесткого диска (монолитного или разделенного на части);
- ❒ поддержка технологии "клиент-сервер";
- ❒ программное создание загрузочного носителя (компакт-диска, дискеты) для извлечения из образа информации;
- ❒ работа с ранее созданным образом жесткого диска;
- ❒ поддержка утилиты sysprerp.exe.

Клонированию подвергают гомогенные рабочие станции, имеющие одинаковое аппаратное обеспечение. Процесс клонирования состоит из трех этапов: подготовительного, основного, заключительного.

На подготовительном этапе специалистом системной поддержки на рабочую станцию устанавливается операционная система; необходимые драйверы, обеспечивающие корректную работу устройств; сопутствующее программное обеспечение.

На основном этапе происходит клонирование рабочей станции. Слепок жесткого диска помещают на сервер. Клонирование осуществляется специально предназначенной для этого программой. Наиболее известными программами, предназначенными для клонирования жестких дисков, являются Symantec Ghost Corporate Edition (**http://www.symantec.com**) и PowerQuest Deploy Option (**http://www.powerquest.com**).

Заключительный этап включает в себя переименование рабочей станции и подключение ее к сети.

Процесс клонирования имеет особенности. Рабочая станция, подвергаемая клонированию, не должна быть членом домена. Это объясняется тем, что после окончания процесса клонирования в сети будет две идентичные рабочие станции с одинаковыми именами, что приведет к конфликту имен в сети (домене). Смена имени не сможет решить появившейся проблемы в том случае, если операционная система клонируемой рабочей станции принадлежит к семейству Windows 2k[1]. Все операционные системы семейства Windows 2k имеют идентификационный номер (SID) и серийный номер (serial number), который вводится пользователем при установке операционной системы на рабочую станцию[2].

Для успешной работы рабочей станции в сети необходимо сменить имя рабочей станции, идентификационный и серийный номера операционной системы. Смена идентификационного номера возможна только в случае нахождения рабочей станции в рабочей группе. Для смены SID используется утилита ghstwalk.exe из пакета программ Symantec Ghost, которая также позволяет изменить имя рабочей станции. Предоставление возможности смены серийного номера осуществляется с помощью утилиты sysprep.exe, которая входит в дистрибутив Microsoft Windows 2000 и выше.

Другая особенность клонирования возникает из-за неидентичности аппаратного обеспечения. Для каждого набора микросхем характерен свой уникальный набор драйверов. Из-за отсутствия взаимозаменяемости драйверов, клонирование рабочих станций без предварительной подготовки приводит

[1] К семейству Microsoft Windows 2k относятся следующие версии операционных систем: Windows 2000, Windows XP, Windows 2003.

[2] Проверка на повторяющиеся в локальной сети серийные номера осуществляется только в операционных системах, начиная с Microsoft Windows XP Service Pack 1.

к некорректной работе операционной системы. Исправить появившийся недостаток можно, только переустановив операционную систему. Очевидно, что данное решение проблемы неприемлемо.

Подготовка рабочей станции к клонированию осуществляется с помощью утилиты SysPrep, которая присваивает операционной системе новый идентификационный номер; удаляет информацию о старом серийном номере, установленном оборудовании (если это необходимо). При первом запуске рабочей станции после завершения процесса клонирования утилитой SysPrep запускается сокращенная версия установки операционной системы, в ходе которой осуществляется генерация нового идентификационного номера операционной системы, запрашиваются серийный номер операционной системы, имя пользователя, название организации, предлагается подключить рабочую станцию в домен.

Последовательность действий по клонированию рабочей станции.

1. На клонируемой рабочей станции.
   - Установка операционной системы, необходимых для корректной работы устройств драйверов, сопутствующего программного обеспечения. Рабочая станция находится в рабочей группе и подключена к сети.
   - Запуск с удаленного компьютера утилиты SysPrep.
   - Загрузка с альтернативного носителя информации утилиты Ghost, предоставляющей доступ к файловой системе NTFS и к локальной сети.
   - Создание клона жесткого диска на удаленном компьютере с помощью утилиты Ghost.
2. На целевой рабочей станции.
   - Загрузка с альтернативного носителя информации утилит Ghost и Ghstwalk, предоставляющих доступ к файловой системе NTFS и к локальной сети.
   - Запуск утилиты Ghost и извлечение содержимого слепка жесткого диска на раздел жесткого диска целевого компьютера.
   - Запуск утилиты Ghstwalk, с помощью которой происходит смена SID и имени рабочей станции.
   - Загрузка с жесткого диска клонированной ОС, в ходе которой запустится сокращенная версия установки ОС. Необходимо ввести серийный номер, название компьютера и организации.
   - Ввод рабочей станции в домен. После перезагрузки клонированная рабочая станция готова к работе в сети.

# Автоматизация процесса установки: ОС

Рассмотрим вариант установки Windows + MUI, как более сложный. Если читатель предпочитает не использовать MUI, то он может пропустить раздел, который ему посвящен. В типовой набор программ входят:

- проводник (Total Commander, FAR, DOS Navigator и др.);
- антивирус (DrWeb, Norton Antivirus и др.);
- архиватор (WinZip, WinRar, WinAce и др.);
- программа, позволяющая просматривать множество графических форматов (ACDSee, InfraView);
- Microsoft Office;
- программа, предназначенная для записи компакт-дисков (Nero, Easy CD Creator и др.), просмотра DVD-дисков (Power DVD, AsusDVD); кодек для просмотра видеоматериалов в формате MPEG4 (DivX).

# Обзор инсталляторов

На сегодняшний день существует множество инсталляторов. Рассмотрим наиболее часто используемые из них:

- Windows Installer;
- Inno Setup;
- Nullsoft Scriptable Install System (NSIS);
- Wise Installer.

## Пакетная установка ПО

Для установки нескольких программ в автоматическом режиме рекомендуется использовать пакетную установку, реализованную, например, с помощью BAT-файла. Пользоваться же параллельной установкой не рекомендуется, поскольку некоторые инсталляторы, например Windows Installer, не могут работать параллельно. Для того чтобы процесс установки программ шел последовательно, необходимо строку установки предварять командой `start /wait`:

```
start /wait %systemdrive%\install\setup.exe /s
```

Некоторые инсталляторы Inno Setup пытаются запустить программу по окончании процесса установки. Если вы устанавливаете несколько программ подряд в "тихом" режиме, то это неудобно. Решить проблему можно с помощью

утилиты Taskkill, поставляющейся с операционной системой, следующим образом:

```
[Setup.bat]
start /wait %systemdrive%\install\filename.exe /SILENT
/SP-taskkill.exe /F /IM filename.exe
```

## Windows Installer

Windows Installer — это сервис установки и конфигурирования программных продуктов, который входит в состав ОС (рис. 12.1). Также он может устанавливаться с пакетами обновлений операционной системы или в качестве отдельного дистрибутива.

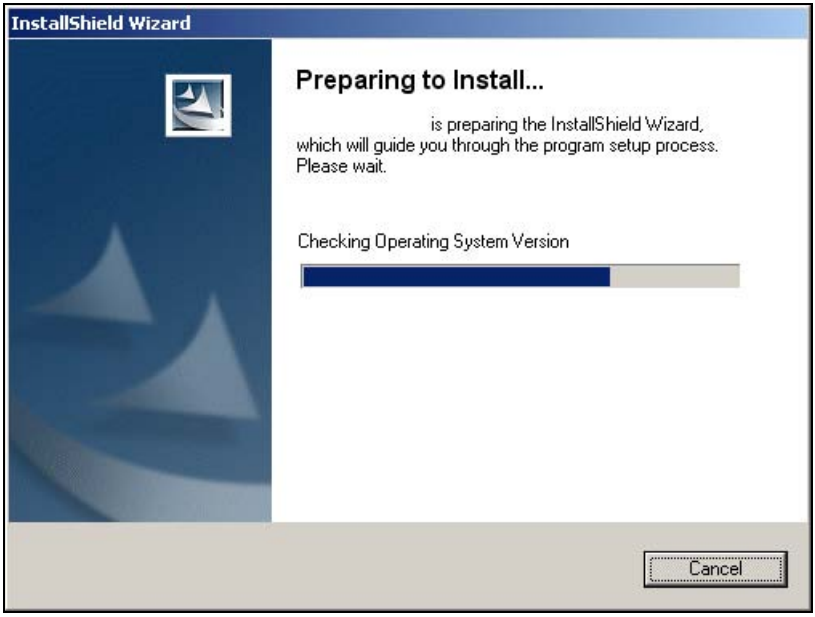



**Рис. 12.1.** Windows Installer

Некоторые версии Windows Installer и Windows несовместимы. Это касается Windows 9x и Windows 2k. С помощью табл. 12.1 можно определить, какую версию Windows Installer и где можно устанавливать.

В настоящее время используется Windows Installer 3.1, который можно загрузить с сайта разработчика как hotfix: **http://support.microsoft.com/?id=893803**.

***Таблица 12.1.*** *Версии Windows Installer* [3]

| Название продукта | Версия продукта | Описание |
|---|---|---|
| Windows Installer 1.0 | 1.00.5104.0 | Интегрирован в Office 2000, дистрибутив |
| Windows Installer 1.1 | 1.10.1029.0 | Интегрирован в Windows 2000 |
| | 1.10.1029.1 | Дистрибутив |
| Windows Installer 1.11 | 1.11.1314.0 | Интегрирован в Windows 2000 SP1 |
| | 1.11.2405.0 | Интегрирован в Windows 2000 SP2 |
| Windows Installer 1.2 | 1.20.1410.0 | Интегрирован в Windows Me |
| | 1.20.1827.1 | Дистрибутив |
| Windows Installer 2.0 | 2.0.2600.0 | Интегрирован в Windows XP |
| | 2.0.2600.1 | Интегрирован в Windows 2000 SP3 |
| | 2.0.2600.1183 | Интегрирован в Windows 2000 SP4 |
| | 2.0.2600.2 | Дистрибутив |
| | 2.0.2600.1106 | Интегрирован в Windows XP SP1 |
| | 2.0.3754.0, 2.0.3790.0 | Интегрирован с семейством Windows Server 2003 |
| Windows Installer 3.0 | 3.0.3790.2180 | Интегрирован в Windows XP SP2, дистрибутив |
| Windows Installer 3.1 | 3.1.4000.1823 | Дистрибутив |
| | 3.1.4000.1830 | Интегрирован с семейством Windows Server 2003 SP1 |
| | 3.1.4000.2435 | Дистрибутив |

Для инициализации процесса установки используют команду:

```
WindowsInstaller-   -x86.exe [<options>]
```

Параметры запуска приведены в табл. 12.2.

[3] **http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/windows_installer_redistributables.asp**

***Таблица 12.2.*** *Параметры запуска hotfix*

| Параметр | Описание |
|---|---|
| `/norestart` | Не перезагружать компьютер после установки обновления |
| `/quiet` | Включение "тихого" режима. Во время установки программа не задает никаких вопросов |
| `/help` | Вызов справки, содержащей ключи запуска и их описание |

На каждом компьютере, использующем Windows Installer, хранится единая база данных с информацией о каждом установленном с помощью этой технологии приложении. Она включает файлы, записи в реестре и компоненты. При удалении приложения с машины Installer проверяет базу данных, чтобы удостовериться в том, что не будут удалены файлы, ключи реестра и компоненты, от которых зависят другие приложения. Таким образом, удаление приложения становится практически безопасным для других программ, имеющихся на компьютере.

Использование технологии Windows Installer дает пользователям следующие преимущества:

- ❒ более простая и быстрая установка программного обеспечения;
- ❒ возможность установки по требованию. Информация о неустановленных компонентах приложения хранится в одном месте, и при обновлении конфигурации программы нет необходимости переустанавливать все компоненты;
- ❒ самовосстановление программ. Хранение инсталляционной информации в одном месте позволяет приложению самовосстанавливаться. Неправильно работающее приложение может проверить инсталляционные данные, определить, какие файлы повреждены или отсутствуют, а затем восстановить их;
- ❒ возможности отката. Windows Installer позволяет отменять любые изменения в конфигурации как устанавливаемого продукта, так и операционной системы. Это делает установку программ, поддерживающих технологию Windows Installer, намного более безопасным и предсказуемым занятием, чем каких-либо других.

Благодаря перечисленным преимуществам пользователи теряют меньше времени на удаление и переустановку приложений, а также избавляются от необходимости исправлять непонятные и трудно диагностируемые ошибки в конфигурации приложения.

## Взгляд изнутри: файл msiexec.exe

Windows Installer включает в себя множество файлов, среди которых присутствует msiexec.exe. Этот файл — не самый важный компонент в Windows Installer. Всю основную работу выполняет динамически подключаемая библиотека msi.dll. А msiexec.exe служит оболочкой, позволяющей:

- Windows Installer работать как службе ОС;
- разбирать параметры командной строки и выполнять соответствующие задачи;
- устанавливать при выходе уровень ошибки, соответствующий системным кодам ошибок (**http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/error_codes.asp**);
- связывать MSI-файлы с Windows Installer командой:

```
%SystemRoot%\System32\msiexec.exe" /i "your_filename.msi
```

## Параметры командной строки для Msiexec

Параметры командной строки (**http://www.microsoft.com/technet/prodtechnol/windowsserver2003/ru/library/ServerHelp/9361d377-9011-4e21-8011-db371fa220ba.mspx?mfr=true**), которые понимает msiexec.exe, приведены в табл. 12.3. Полный список параметров смотрите на сайте Microsoft.

***Таблица 12.3.*** *Параметры командной строки msiexec.exe*

| Опция | Параметры | Значение |
|---|---|---|
| /i | Код пакета / продукта | Установить или конфигурировать приложение |
| /f | [p\|o\|e\|d\|c\|a\|u\|m\|s\|v]Код пакета / продукта | Восстановить приложение. Если задана эта опция, Msiexec игнорирует значения свойств, заданные в командной строке. Список по умолчанию для данной опции pecms.<br>p — переустановить, только если файл отсутствует.<br>o — переустановить, если файл отсутствует или установлена более старая версия.<br>e — переустановить, если файл отсутствует или установлена такая же либо более старая версия.<br>d — переустановить, если файл отсутствует или установлена другая версия |

***Таблица 12.3*** *(окончание)*

| Опция | Параметры | Значение |
|---|---|---|
| /f | [p\|o\|e\|d\|c\|a\|u\|m\|s\|v]Код пакета / продукта | c — переустановить, если файл отсутствует или его сохраненная контрольная сумма не соответствует вычисленному значению.<br>a — переустановить все файлы.<br>u — перезаписать все необходимые специфичные для пользователя ключи реестра.<br>m — перезаписать все необходимые специфичные для машины ключи реестра.<br>s — перезаписать все существующие ярлыки.<br>v — запустить с исходного носителя и заново кэшировать локальный пакет |
| /a | Пакет | Опция Административной установки. Установить продукт в сети |
| /x | Код пакета / продукта | Деинсталлировать продукт |
| /j | [u\|m]Пакетили[u\|m]Пакет /g Идентификатор языка | Опубликовать продукт. Если задана эта опция, Msiexec игнорирует значения свойств, заданные в командной строке.<br>u — опубликовать для текущего пользователя.<br>m — опубликовать для всех пользователей компьютера.<br>g — идентификатор языка |
| /q | n\|b\|r\|f | Задать уровень пользовательского интерфейса /q.<br>qn — отсутствие интерфейса. В этом случае не показывается никаких окон и не задается никаких вопросов. Данный режим очень удобен для автоматической инсталляции ПО.<br>qb — уровень базового пользовательского интерфейса.<br>qr — уровень сокращенного пользовательского интерфейса.<br>qf — уровень полного пользовательского интерфейса |

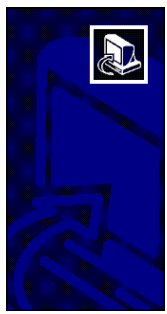
## Inno Setup

Inno Setup — бесплатно распространяемый инсталлятор для Windows-приложений, разрабатываемый с 1997 года по настоящее время. Последнюю версию можно загрузить из Интернета — **http://www.jrsoftware.org/isinfo.php**.

Узнать инсталлятор Inno Setup можно очень просто. Во-первых, в левой части окна мастера-инсталлятора присутствует одна из двух картинок, показанных на рис. 12.2, а во-вторых, на самом первом экране в меню присутствует команда **Файл | О программе** (рис. 12.3).



*а*



*б*

**Рис. 12.2.** Внешний вид инсталлятора Inno Setup

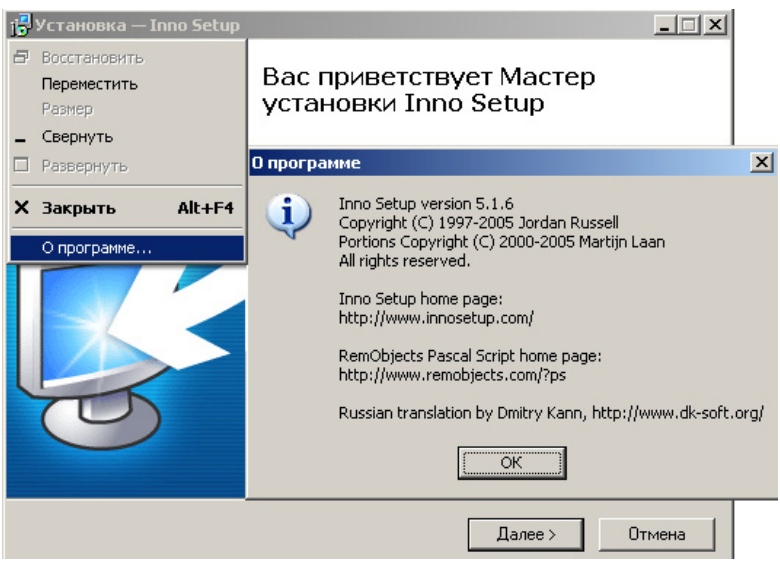



**Рис. 12.3.** Версия инсталлятора Inno Setup

Для автоматизации процесса установки ПО рекомендуется использовать команду `setup.exe /SILENT`. Полный список ключей приведен в табл. 12.4.

***Таблица 12.4.*** *Список параметров запуска инсталляторов Inno Setup*

| Параметр | Описание |
|---|---|
| `/SILENT`, `/VERYSILENT` | Подавить вывод диалоговых окон |
| `/NOCANCEL` | Скрыть кнопку отмены процесса установки |
| `/NORESTART` | Управлять перезагрузкой после завершения процесса установки |
| `/LOADINF="filename"` | Передать управление установкой конфигурационному файлу |
| `/SAVEINF="filename"` | Записать в конфигурационный файл сценарий установки |
| `/LANG=language` | Управлять языковыми настройками. Если параметр указан, сообщение о выборе языка не появляется |
| `/DIR="x:\dirname"` | Путь установки ПО |
| `/GROUP="folder name"` | Название группы, в которой будут размещены ярлыки в папке **Пуск \| Программы** |
| `/NOICONS` | Не создавать папку с ярлыками в **Пуск \| Программы** |
| `/COMPONENTS` | Использовать при установке набор по умолчанию |
| `/SN` | Указать серийный номер продукта |

Инсталлятор поддерживает файлы ответов, которые имеют расширение iss. После установки инсталлятора их можно найти в каталоге C:\Program Files\Inno Setup 5\Examples. Файл ответа можно сгенерировать к уже готовому продукту с помощью ключей, описанных в табл. 12.4. Синтаксис файла ответов подробно описан в справке C:\Program Files\Inno Setup 5\ISetup.hlp.

В комплекте установки инсталлятора в каталоге C:\Program Files\Inno Setup 5\Languages находятся файлы с расширением isl, позволяющие инсталлятору поддерживать несколько языков интерфейса.

## Nullsoft Scriptable Install System (NSIS)

Как и предыдущий инсталлятор, NSIS (рис. 12.4) является бесплатным. С его помощью используются такие приложения, как DIVX. Скачать NSIS 2.15 (4 марта 2006 г.) можно по адресу **http://nsis.sourceforge.net/Main_Page**.

Инсталлятор NSIS используют такие приложения, как WinAmp, SoundForge, Emule, DivX. Дополнительные скриншоты смотрите на **http://nsis.sourceforge.net/Screenshots**.

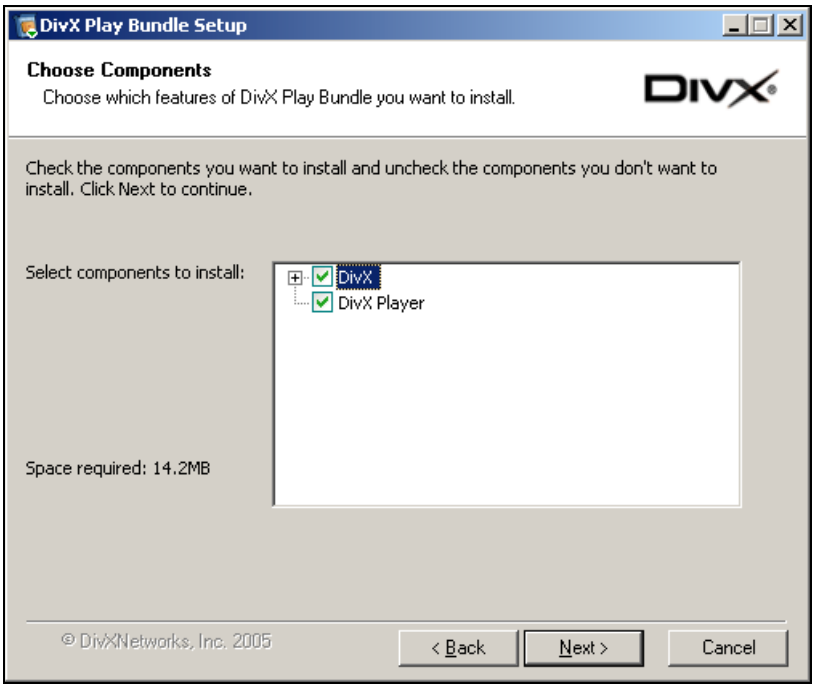



**Рис. 12.4.** Внешний вид инсталлятора NSIS

Инсталлятор поддерживает файлы ответов, которые имеют расширение nsi. После установки инсталлятора их можно найти в каталоге C:\Program Files\NSIS\Examples\. Подробное описание скриптового языка можно найти в файле C:\Program Files\NSIS\NSIS.chm.

Инсталлятор поддерживает параметры запуска, указанные в табл. 12.5.

***Таблица 12.5.*** *Параметры запуска инсталлятора NSIS*

| Параметр | Описание |
|---|---|
| `/NCRC` | Отключает функцию проверки контрольной суммы архива. Работает, если в сценарии установки не используется функция `CRCCheck()` |
| `/S` | Подавляет все диалоговые окна в процессе установки |
| `/D` | Задает каталог инсталляции продукта |

Приведем несколько примеров использования инсталлятора:

```
installer.exe /NCRC
installer.exe /S
installer.exe /D=C:\Program Files\NSIS
installer.exe /NCRC /S /D=C:\Program Files\NSIS
```

## Wise Installer

Этот инсталлятор является платным. Его бесплатную 30-дневную версию можно скачать по адресу **http://www.wise.com/index.asp**, зарегистрировавшись на сайте. Wize Installer используют такие программы, как AdWare, AGraber. Внешний вид программы показан на рис. 12.5. Автоматическая установка осуществляется с помощью ключа `/S`.

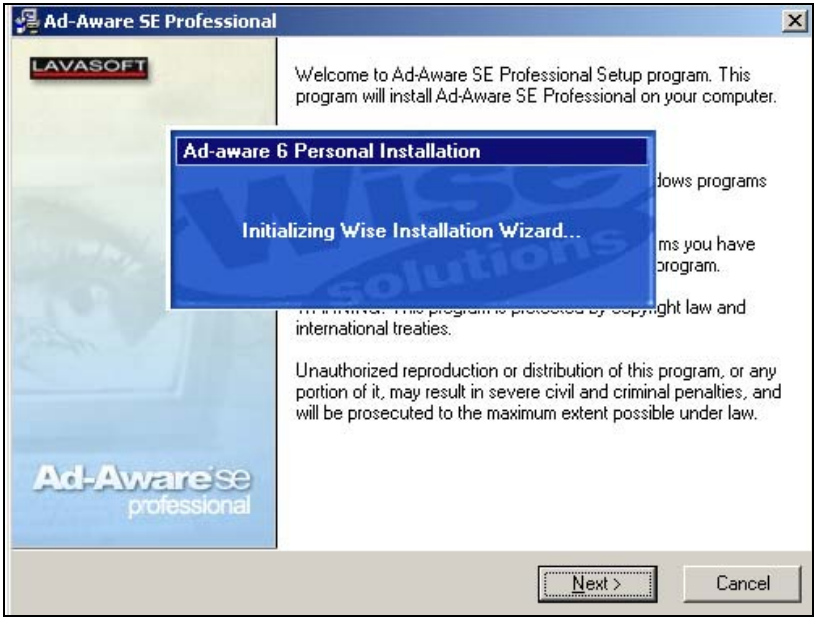



**Рис. 12.5.** Внешний вид инсталлятора Wise Installer

# Создание дистрибутива

## Подготовка файловой структуры будущего диска

Существует несколько способов автоматической установки Microsoft Windows. Самым распространенным из них является использование файла отве-

тов во время инсталляции. Автоматизация процесса установки ОС дает специалистам системной поддержки следующие преимущества:

- минимизировать влияние человеческого фактора (доверить специалисту системной поддержки решать вопросы, касающиеся введения регистрационного номера ОС, управления разделами жесткого диска);
- сократить время на установку ОС.

## Создание файла ответов для установки Windows

Существует три типа автоматической установки Windows. В каждом из трех случаев используется индивидуальный файл ответа (табл. 12.6).

***Таблица 12.6.*** *Типы автоматической установки Windows*

| Название файла | Случай использования файла |
|---|---|
| *.txt | Используется для автоматической установки Windows. Можно указать любое имя файла. Стандартное имя — unattend.txt |
| Winnt.sif | Используется при установке Windows 2000 с загрузочного CD |
| Sysprep.inf | Используется совместно с утилитой Sysprep.exe для создания образа установки Windows |

Файл ответа представляет собой текстовый файл со следующей структурой:

```
[разделM]
параметр1M=значение1M
параметр2M =значение2M
. . .
параметрNM=значениеNM
```

Файлы Winnt.sif, Sysprep.inf должны находиться в каталоге i386, в то время как файлы *.txt могут располагаться в любом месте. Чтобы при установке Windows воспользоваться файлом ответа с расширением txt, необходимо выполнить следующую команду:

```
winnt[32] /u:<файл ответов>
```

Оптимальным решением является создание базового шаблона с помощью Setup Manager, который затем нужно доработать вручную. Утилита Setup Manager запускается с помощью файла Setupmgr.exe, который находится в архиве Deploy.cab на диске с дистрибутивом Windows в папке \Support\Tools. Документация по файлам ответов находится в том же архиве — файл setupmrg.chm. Пример файла ответов winnt.sif смотрите в *приложении 4*.

Для получения полной информации о параметрах командной строки WinNT.exe и WinNT32.exe используйте параметр `/?`, например:

```
winnt.exe /?
```

## Файловая структура дистрибутивного диска Windows

Расположение папки $OEM$ зависит от используемого файла ответов. Если автоматическая установка Windows запускается автоматически с компакт-диска, т. е. в качестве файла ответов используется WinNT.sif, то папка $OEM$, содержащая все дополнительно устанавливаемые компоненты, должна располагаться в корневом каталоге загрузочного диска. Во всех остальных случаях папка $OEM$ должна располагаться в каталоге i386, а файлом ответов будет Unattended.txt. Root — корневая папка загрузочного компакт-диска, содержащего дистрибутив ОС.

i386 содержит установочные файлы Windows.

$OEM$ — папка, содержимое которой используется для установки дополнительного программного обеспечения. Опишем файловую структуру папки $OEM$ — рис. 12.6 и 12.7.

В разделе `[Unattended]` файла ответов параметру `OEMPreinstall` должно быть присвоено значение `Yes`: `OEMPreinstall=Yes`.

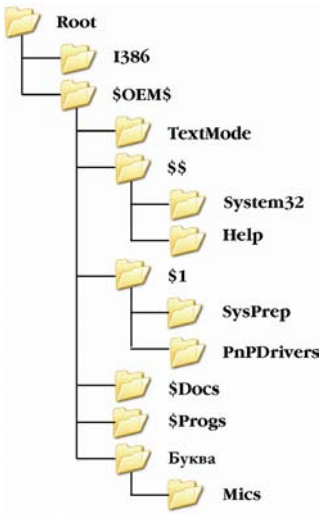



**Рис. 12.6.** Структура файлов дистрибутива. Файл ответов unattended.txt

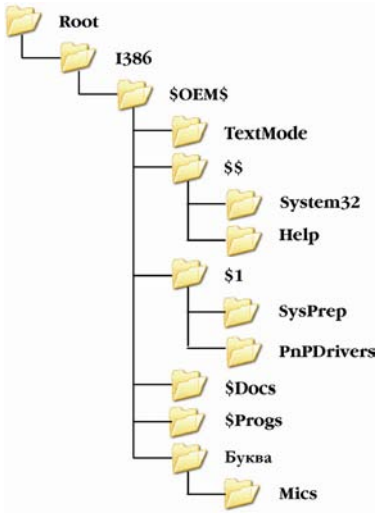



**Рис. 12.7.** Структура файлов дистрибутива. Файл ответов winnt.sif

Описание вспомогательных папок дистрибутива Windows приведено в табл. 12.7.

***Таблица 12.7.*** *Вспомогательные папки дистрибутива Windows*

| Папка | Описание возможного содержимого папки |
|---|---|
| TextMode | Содержит драйверы накопителей (IDE и SCSI), предназначенные для распознавания носителей из текстового режима установки ОС. В ней обязательно должен присутствовать файл TxtSetup.oem, содержащий информацию о драйверах устанавливаемых устройств. Раздел `[OEMBootFiles]` файла ответов также должен быть изменен соответствующим образом |
| $$ | Соответствует %WinDir% |
| System32 | Файлы, содержащиеся в этой папке, будут скопированы в %WinDir%\System32 |
| Help | Файлы, содержащие справочную информацию (*.chm, *.hlp) в этой папке, будут скопированы в %WinDir%\Help |
| $1 | Название директории соответствует букве раздела жесткого диска, содержащего MBR |
| SysPrep | Содержит файлы, необходимые для использования утилиты Sys-Prep |

***Таблица 12.7*** *(окончание)*

| Папка | Описание возможного содержимого папки |
|---|---|
| PnPDrivers | Содержит драйверы устройств Plag&Play, которые не включены в стандартный набор ОС |
| $Docs | Соответствует папке Document and Settings |
| $Progs | Соответствует папке Programs |
| Буква\Mics | Название папки совпадает с именем диска, на который будут скопированы данные, находящиеся в ней. Mics — произвольный каталог, созданный в ней |

## Механизм инсталляции быстрых исправлений

На первом этапе осуществляется копирование инсталляционных файлов различных приложений, в том числе и hotfix, в соответствующий подкаталог $OEM$. Копирование файлов и папок на жесткий диск компьютера осуществляется до запуска графического режима.

На втором этапе на основе информации, содержащейся в файле ответов, осуществляется запуск скопированных приложений с указанием явных путей к файлам, последовательно инициализирующих процессы установки. На завершающем этапе установки последнего приложения происходит удаление каталогов, созданных во время установки.

# Подготовка дистрибутива Windows

## Интеграция пакета исправлений Service Pack в Windows

Процесс интеграции Service Pack в Windows осуществляется следующим образом: на жестком диске создаются две папки. В одну из них копируется содержимое компакт-диска с дистрибутивом Microsoft Windows XP Professional (C:\BOOTCD\WINDOWS), в другую — дистрибутив Microsoft Service Pack (C:\BOOTCD\SERVICEPACK), который представляет собой архив. Его необходимо распаковать, выполнив следующую команду:

```
C:\BOOTCD\SERVICEPACK\XP-SP1.EXE /U /X:C:\BOOTCD\SERVICEPACK
```

Интегрировать Service Pack в дистрибутив Windows можно, выполнив следующую команду:

```
C:\BOOTCD\SERVICEPACK\UPDATE.EXE /S:C:\BOOTCD\WINDOWS
```

В папке Windows должен находиться дистрибутив Windows, а не содержимое папки i386.

# Установка пакетов быстрых исправлений в автоматическом режиме

## Принципы именования hotfix

Быстрые исправления (hotfix) представляют собой самораскрывающиеся архивы с расширением exe. Имена hotfix присваиваются в соответствии со следующим шаблоном:

```
Q######_XXX_YYY_ZZZ _LLL.exe
```

Где `Q######` — номер соответствующей статьи Microsoft Knowledge Base с описанием решения обнаруженной проблемы.

`XXX` — сокращенное название операционной системы, для которой предназначено данное исправление.

`YYY` — номер пакета исправления (Service Pack), к которому относится hotfix.

`ZZZ` — платформа рабочей станции.

Для Windows XP в соответствии с данным шаблоном вид имен следующий:

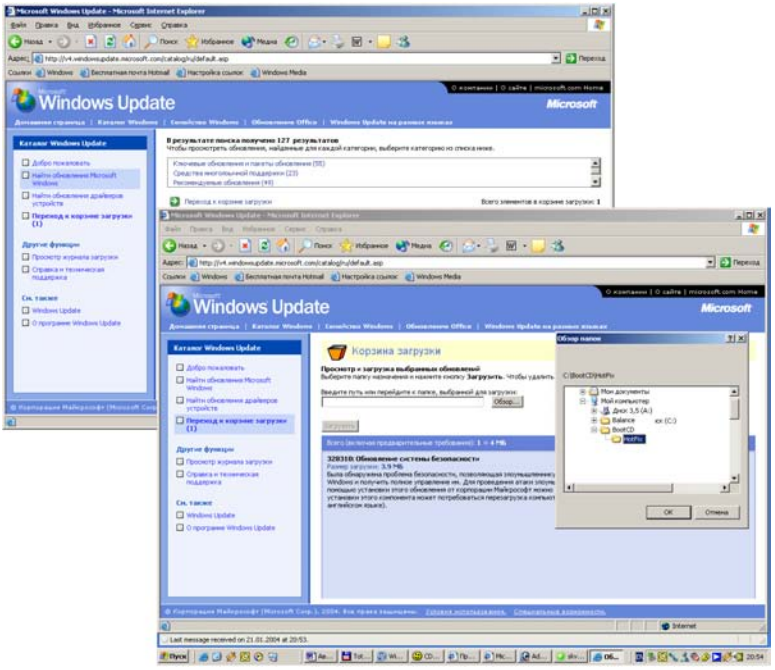
```
Q######_WXP_SP#_x86_EN.exe или Q######_WXP_SP#_x86_RU.exe
```

## Копирование hotfix из Интернета

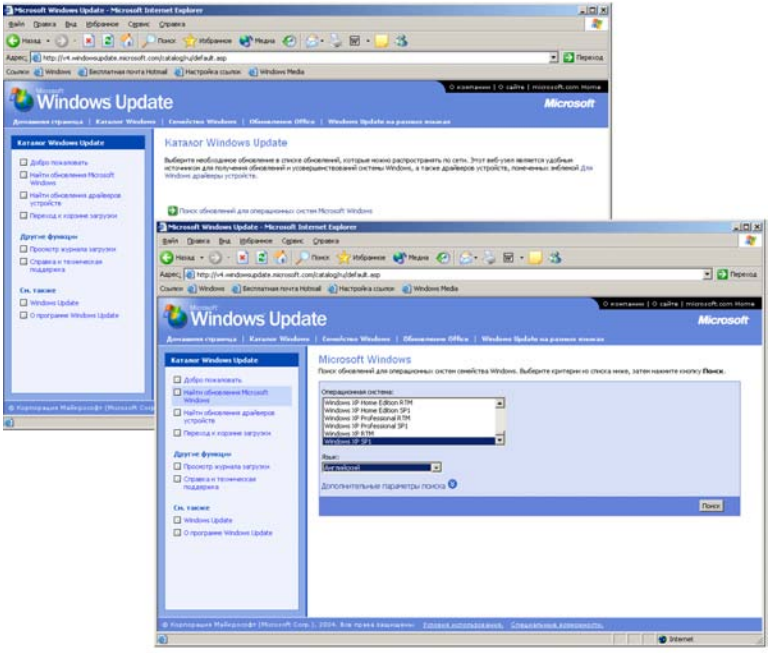
Копирование быстрых исправлений осуществляется с сайта компании Microsoft: **http://v4.windowsupdate.microsoft.com\catalog**.

После того как вход на сайт Windows Update успешно выполнен, для получения возможности копирования исправлений на жесткий диск вашей рабочей станции выберите из списка исправлений только необходимые из них, подходящие языковую версию и версию ОС (рис. 12.8).

Отметим, что не рекомендуется пользоваться общим адресом каталога Windows Update **http://windowsupdate.microsoft.com\catalog**, поскольку, если вы зайдете на него из операционной системы Windows XP, то будет выполнен редирект не на четвертую версию Windows Update, а на пятую: **http://v5.windowsupdate.microsoft.com\catalog**. К сожалению, в пятой версии еще не реализована возможность копирования hotfix на жесткий диск: их можно только установить.




*а*




*б*

**Рис. 12.8.** Сайт Microsoft Windows Update

## Интеграция hotfix в дистрибутив Windows

Дистрибутивы рекомендуется расположить в папке $OEM$\C\Install\Folder (C — буква, MICS — Install\Folder), где Folder — соответствующее назначению название папки. Например, для hotfix рекомендуется использовать следующий путь:

$OEM$\C\Install\HotFix

В файле ответов необходимо сделать следующие изменения. Для того чтобы установка Windows скопировала данные из папки $OEM$ в соответствующие места, в разделе `[Unattended]` следует написать:

```
[Unattended]
OEMPreinstall=Yes
```

Управление hotfix осуществляется таким образом:

```
[GuiRunOnce]
C\Install\HotFix\Q######.exe /n /q /z
%WinDir%\System32\Cmd.exe /c RmDir C\Install /s /q
```

Для простоты восприятия в данном примере приведены только две строки: в первой строке осуществляется запуск процесса установки управления в скрытом режиме. Во второй — удаление каталога C:\Install с жесткого диска. Инициализация процесса инсталляции приложений осуществляется после завершения процесса установки операционной системы.

## Ключи, используемые при установке hotfix

`/F` — закрыть все открытые приложения после установки исправления перед перезагрузкой ОС.

`/N` — не создавать файлы отката (backup files) для восстановления системы.

`/Z` — не перезапускать рабочую станцию после внедрения hotfix.

`/Q` — включить скрытый режим установки исправлений.

`/U` — использовать автоматический режим установки. Вывод сообщений только о возникающих критических ошибках.

`/L` — вывести список установленных исправлений на рабочей станции. Данная информация также находится в реестре по пути: `HKLM\Software\Microsoft\CurrentVersion\HotFix\{SP\}Q######`,

`HKLM\Software\Microsoft\CurrentVersion\Uninstall\Q######`.

Для установки hotfix рекомендуется использовать следующую команду:

```
Q######.exe /q /n /z
```

## Установка MUI в автоматическом режиме

Загруженный с сайта Microsoft MUI необходимо распаковать, используя тот же метод, что и при распаковке дистрибутива SP:

```
C:\BOOTCD\MUI\MUI_RUS.EXE /U /X:C:\BOOTCD\MUI
```

Версия MUI зависит от номера Service Pack. Будьте внимательны при копировании SP с сайта — не ошибитесь версией.

Затем необходимо скопировать его в папку $$\Install\MUI, а, например, в файл ответов внести следующие изменения:

```
[GuiRunOnce]
c:\MUIINST\MUISETUP.EXE /i 0419 /d 0419 /r /s
%windir%\system32\cmd.exe /c rmdir c:\MUIINST /s/q
```

Ключ `/i` указывает на кодовую страницу выбранного языка (`0419` — русский), ключ `/d` назначает язык интерфейса Windows по умолчанию. Полный список ключей, а также значений языков можно найти в файле muisetup.hlp, входящем в комплект поставки MUI.

### Интеграция драйверов в дистрибутив

Драйверы устройств, не вошедших в стандартный набор, могут быть в него интегрированы. Для этого необходимо скопировать дистрибутивы драйверов в папку $OEM$\$1\PnPDrivers, а соответствующие им INF-файлы — в каталог $OEM$\$1\INF. В папке PnPDrivers можно сделать соответствующие подпапки, например, Sound, Video.

В файле ответов в разделе `[Unattended]` следует прописать путь к этим драйверам. Каталоги разделяются точкой с запятой, пробелы в строке не допускаются. Рекомендуется также отключить проверку драйверов на "подпись", задав соответствующее значение параметру `DriverSigningPolicy`. Обязательным условием, как отмечалось ранее, является значение `Yes` параметра `OEMPreinstall`:

```
[Unattended]
OemPnPDriversPath=drivers\video;drivers\sound
DriverSigningPolicy=Ignore
OEMPreinstall=Yes
```

# Автоматизация процесса установки: ПО

## Установка антивируса: Norton Antivirus

Помня о том, что рабочая станция будет функционировать в сети, необходимо установить клиентскую часть пакета Norton Antivirus (находящегося

на сервере, где пакет установлен) по пути \\Server\VPHOME\CLT-INST\WIN32\ и сконфигурировать его для работы в сети. В этой папке находятся дистрибутив и конфигурационный файл GRC.DAT, представляющий собой текстовый файл. При установке программа ищет его сама в текущем каталоге и читает из него данные. Если же его нет, то антивирус устанавливается и работает в автономном режиме.

Команда автоматической установки NAV CE следующая:

```
c:\Install\NAV\setup.exe /qn
```

## Установка навигатора: WinCMD

Сегодня одним из популярных навигаторов является Total Commander. Его дистрибутив представляет собой самораскрывающийся ZIP-архив, в котором содержится файл ответов — install.inf. Распакуйте этот архив с помощью WinZip или WinRar, скорректируйте файл install.inf. Исправленный пример конфигурационного файла с соответствующими комментариями находится в *приложении 5*. Запуск автоматической установки осуществляется исполнением файла install.exe (рис. 12.9).

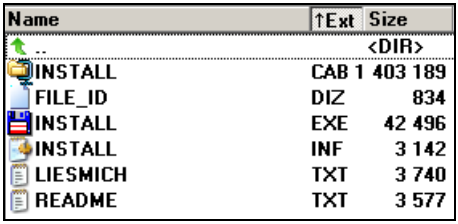



**Рис. 12.9.** Установка Windows Commander

## Автоматическая установка архиватора: WinRar 3.x

Для реализации автоматической установки WinRar в качестве параметра необходимо указать ключ `/s`. В этом случае WinRar будет ассоциирован со всеми типами архивов, а также будет создана соответствующая группа в меню **Пуск**:

```
C:\Install\Wrar\Wrar340ru.exe /s
```

Если же использовать ключ `/silent`, то пользователю будет предложено выполнить дополнительную настройку программы (рис. 12.10).

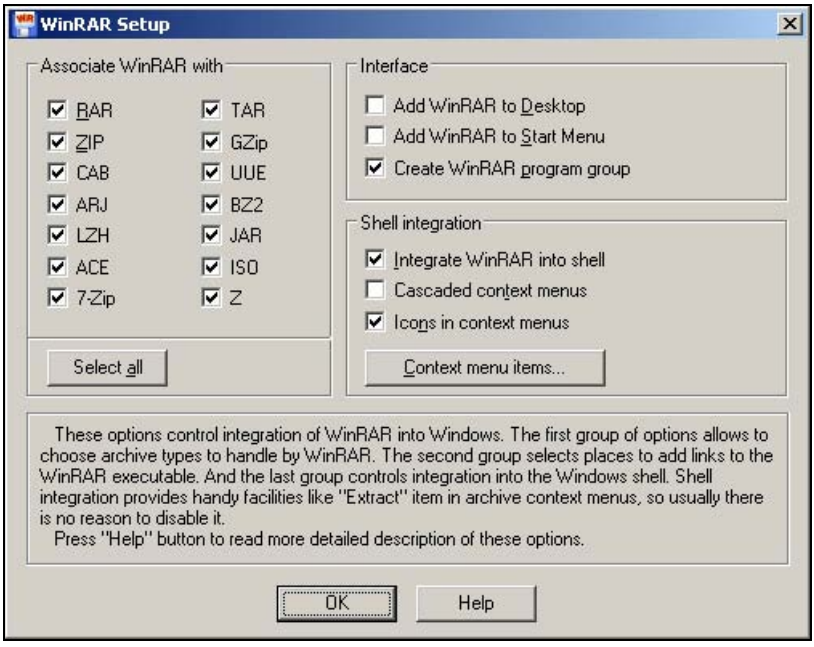



**Рис. 12.10.** Установка WinRAR

Регистрационный файл wrar.reg, созданный на основе данных из реестра, необходимо экспортировать после установки программы с помощью команды:

```
regedit /s c:\Install\WRar\wrar.reg.
```

## Установка программы записи CD/DVD: Nero Burning Rom 6.3.0.x

Полноценный режим работы Nero доступен только в том случае, если известен серийный номер продукта. Существует два способа регистрации программы.

- Указать серийный номер в ходе инсталляции программы в качестве одного из параметров командной строки:

  ```
  c:\Install\Nero\Nero551054.exe /silent /noreboot /sn=xxxx-xxxx-xxxx-xxxx-xxxx-xxxx /write_sn
  ```

  Где `/silent` обеспечивает автоматический режим установки программы.

  `/noreboot` — не перезагружает рабочую станцию после установки программы.

  `/sn=xxxx-xxxx-xxxx-xxxx-xxxx-xxxx` — вместо `xxxx-…` указывается серийный номер продукта. Используется вместе с ключом `/write_sn`.

- Создать REG-файл (листинг 12.1), который содержит данные о регистрации программного продукта. Для экспорта этих данных в реестр необходимо использовать следующую команду:

  ```
  Regedit /S c:\Install\Nero\nero.reg
  ```

  Тогда команда, обеспечивающая автоматическую установку Nero, будет выглядеть следующим образом:

  ```
  c:\Install\Nero\Nero551054.exe /silent /noreboot
  ```

**Листинг 12.1. Файл Nero.Reg**

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Ahead\Nero - Burning Rom\Info]
"User"="UserName"
"Company"="CompanyName"
"Serial6"=" xxxx-xxxx-xxxx-xxxx-xxxx-xxxx "
```

# Установка Adobe Acrobat Reader 6

Дистрибутив Adobe Acrobat Reader 6 представляет собой самораскрывающийся архив. Необходимо запустить установку Adobe Reader и дождаться, пока FEAD Optimizer распакует все файлы и предложит начать установку программы. В %WinDir%\Cache в папке Adobe Reader 6 находится содержимое архива. Все файлы из нее необходимо скопировать в $OEM$\C\Install\Areader6. Файл "Adobe Reader 6.0.1.msi" нужно переименовать в файл, удовлетворяющий формату имен 8.3, например Areader6.msi. Команда запуска автоматической установки будет выглядеть следующим образом:

```
C:\Install\Areader\Areader.msi /qb
```

# Автоматическая установка Microsoft Office 2003 и необходимых дополнений

## Способы автоматической установки Office

Существует как минимум два способа автоматической установки Microsoft Office.

- Автоматическая установка с настройками по умолчанию. Процесс установки инициализируется следующей командой:

  ```
  Pro11.msi /qb или Setup.exe /qb
  ```

- Автоматическая установка с использованием файла ответов с расширением mst, который создается с помощью соответствующего мастера из набора Resource Kit For Microsoft Office 2003. Установка осуществляется с помощью следующей команды:

  ```
  Pro11.msi Transforms=FileName.mst /qb
  ```

  или:

  ```
  Setup.exe Transforms=FileName.mst /qb
  ```

Рассмотрим второй способ установки более подробно.

Чтобы во время автоматической установки не запрашивался серийный номер, необходимо создать административную установку, которая содержит такие важные параметры, как серийный номер Office и название организации, на имя которой зарегистрирован продукт.

## Подготовка дистрибутива Office

### Создание административной установки

Создание административной установки необходимо для реализации автоматической установки Microsoft Office и возможности интегрировать пакеты исправлений (SP) и обновления (updates) в дистрибутив.

По своей сути, создание административной установки является установкой Microsoft Office в специальном режиме, который инициализируется командой `Setup /a`.

В процессе установки Office будет запрошен серийный номер и название организации, на которую должен быть зарегистрирован продукт. Впоследствии при установке Office с этого дистрибутива серийный номер и названия организации запрашиваться не будут.

### Интеграция пакета исправлений и обновлений в Office

Перед интеграцией в Office дистрибутив пакета исправлений, скопированный с сайта Microsoft, необходимо распаковать, выполнив команду:

```
Office2003SP1-KB842532-fullfile-enu.exe /q /c /t:C:\Office2003\SP1
```

Где `C:\Office2003\SP1` — путь, куда будет распаковано содержимое архива.

Интеграция в дистрибутив осуществляется с помощью 2-х команд:

```
MsiExec /p C:\Office2003\SP1\MainSp1f.msp /a C:\Office2003\Office\
Pro11.msi ShortFileNames=True /qb
```

```
MsiExec /p C:\Office2003\SP1\Owc11Sp1ff.msp /a C:\Office2003\Office\
OWC11.msi ShortFileNames=True /qb
```

Где `C:\Office2003\SP1\` — путь к распакованной версии SP, а `C:\Office2003\Office\` — путь к дистрибутиву Microsoft Office 2003.

Интеграция обновлений осуществляется по такому же сценарию: сначала необходимо скопировать обновления с сайта Microsoft на жесткий диск, затем распаковать их и интегрировать в дистрибутив Office 2003. Имена файлов обновлений строятся по следующему принципу:

```
XXX_KB######_YYY_ZZZ.exe
```

Где `XXX` — версия офиса, для которой предназначено обновление.

`KB######` — номер статьи Microsoft Knowledge Base, в которой приведен список исправлений.

`YYY` — тип версии.

`ZZZ` — языковая принадлежность.

Итак, для Office 2003 файлы обновлений строятся по следующему шаблону:

```
Office2003_KB######_FullFile_Enu.exe
```

### Создание файла ответов MST

Файл ответа для Microsoft Office можно создать с помощью мастера Custom Installation Wizard, входящего в набор Resource Kit соответствующей версии.

Запустив мастер Custom Installation Wizard, необходимо создать новый MST-файл. Команда для установки Office в автоматическом режиме будет следующей:

```
C:\Install\Office\setup.exe transforms= C:\Install\Office\answer.mst
/qb- /noreboot
```

### Установка INF Update от Intel и Via

Если используются материнские платы на микросхемах компании Intel или Via, необходимо установить соответствующие наборы драйверов. Можно предложить как минимум два способа интеграции этих продуктов в дистрибутив. Первый — рассматривать дистрибутив как набор INF-файлов и, пользуясь этим фактом, устанавливать эту программу как совокупность драйверов, интегрировав соответствующие файлы в папки PnPDrivers и INF. Второй вариант — рассматривать дистрибутив как программу и для автоматической установки использовать следующую команду: `setup.exe /s`. Оба этих способа равнозначны, следует лишь отметить, что, скопировав дистрибутив из Интернета, его необходимо разархивировать с помощью программы WinZip.

Можно сделать универсальный дистрибутив, содержащий набор драйверов для материнских плат на основе микросхем Intel (**http://www.intel.com**) и Via (**http://www.via.com.tw**). Для этого необходимо пометить в дистрибутив оба набора и устанавливать их драйверы как приложения, т. е. использовать второй вариант. Таким образом, при установке драйверов один из пакетов выдаст сообщение об ошибке, которое не будет отображено на экран, т. к. уста-

новка ведется в скрытом режиме, и установка данного пакета будет завершена, в то время как необходимый пакет драйверов будет установлен успешно.

## Создание загрузочного диска

В настоящее время, практически любая программа, предназначенная для записи дисков, поддерживает возможность создания загрузочных дисков. Однако перед записью данных на диск рекомендуется протестировать дистрибутив, а именно: создать его образ и установить на него операционную систему, используя VmWare Workstation (**http://www.vmware.com**).

На этом процесс формирования дистрибутивного диска завершен. Осталось его протестировать и записать на диск.

### Создание файла-образа диска

Для компиляции ISO-файлов на основе предоставляемых в папке данных существует множество различных программ. Рекомендуется использовать программу CDImage GUI — это маленькая программа (1,4 Мбайт) с графическим интерфейсом, не требующая установки и обладающая необходимым функционалом (рис. 12.11).

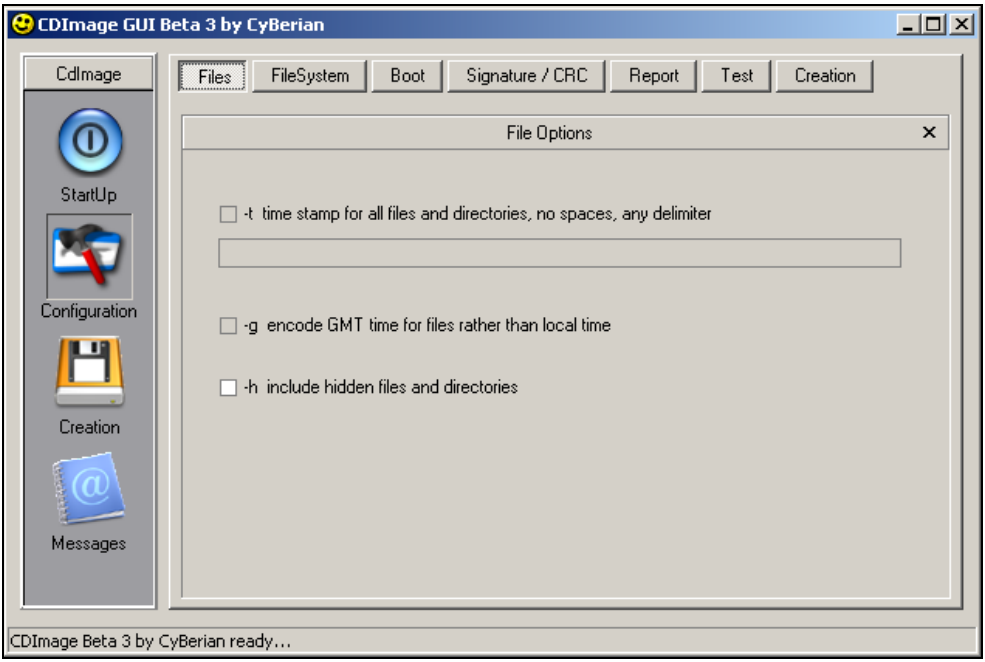



**Рис. 12.11.** Программа создания файла-образа диска

Создавая файл-образ с помощью данной программы, необходимо указать метку тома будущего диска (табл. 12.8), убрать ограничение размера файла в 650 Мбайт, включить скрытые файлы и каталоги в дистрибутив, указать файл, содержащий загрузчик диска. Загрузчик диска можно скачать из Интернета или указать ISO-образ-файл лицензионного диска, содержащего загрузчик.

***Таблица 12.8.*** *Метка тома компакт-диска дистрибутива Windows*

| Тип ОС | Метка тома | Тип ОС | Метка тома |
|---|---|---|---|
| Windows 2000 Professional | W2PFPP_EN | Windows 2000 Advanced Server | W2AFPP_EN |
| Windows 2000 Server | W2SFPP_EN | Windows 2000 Datacenter | W2DFPP_EN |
| Windows XP Professional | WXPCCP_EN | Windows XP Home | WXHCCP_EN |
| Windows XP Professional OEM | WXPOEM_EN | Windows XP Home OEM | WXHOEM_EN |

В корневом каталоге дистрибутива, в зависимости от версии ОС и встроенного SP, должны присутствовать определенные файлы (табл. 12.9 и 12.10).

***Таблица 12.9.*** *Идентификационные файлы версии Windows на компакт-диске дистрибутива ОС*

| Версия Windows | Необходимые файлы |
|---|---|
| XP Home | WIN51 и WIN51IC |
| XP Professional | WIN51 и WIN51IP |
| XP Professional | WIN51 и WIN51IP |
| 2000 Professional | CDROM_NT.5 и CDROM_IP.5 |
| 2000 Server | CDROM_NT.5 и CDROM_IS.5 |
| 2000 Advanced Server | CDROM_NT.5 и CDROM_IA.5 |
| 2000 Datacenter Server | CDROM_NT.5 и CDROM_ID.5 |

***Таблица 12.10.*** *Идентификационные файлы версий Windows, содержащих пакет исправления*

| Версия Service Pack | Необходимые файлы |
|---|---|
| XP Home Service Pack 1 | WIN51IC.SP1 |
| XP Professional Service Pack 1 | WIN51IP.SP1 |
| XP Home Service Pack 2 | WIN51IC.SP2 |
| XP Professional Service Pack 2 | WIN51IP.SP2 |
| 2000 Service Pack 1 | CDROM_SP.TST |
| 2000 Service Pack 2 | CDROMSP2.TST |
| 2000 Service Pack 3 | CDROMSP3.TST |
| 2000 Service Pack 4 | CDROMSP4.TST |

Загрузчик диска ОС, необходимый для автозапуска установки с компакт-диска, представляет собой файл с расширением bin. Его можно скопировать из Интернета или сделать файл с расширением img с помощью любой соответствующей программы, например WinImage.

## Тестирование ISO-файла

Для тестирования ISO-файла рекомендуется использовать VMWare. Эта программа предназначена для эмуляции различных операционных систем на персональном компьютере. После установки VMWare необходимо запустить Virtual Machine Wizard: **File | New | New Virtual Machine**. Результатом работы мастера должен быть файл настроек, с помощью которого осуществляется эмуляция ОС Windows XP Professional. После этого необходимо изменить свойства загрузки Virtual Machine: **Edit | Virtual Machine Settings**. На вкладке **Hardware** нужно сделать активной вкладку **CD-ROM** и в качестве компактного диска указать путь к файлу, содержащему образ будущего диска (рис. 12.12).

Теперь можно приступить к тестированию ISO-файла, запустив эмуляцию Windows XP Pro: **Power | Power On** (<Ctrl>+<B>). После запуска Virtual Machine будет осуществлена попытка установки ОС с одного из доступных носителей, включая CD-ROM. Результатом автоматической установки ОС с файла-образа будут ОС и типовые программы.

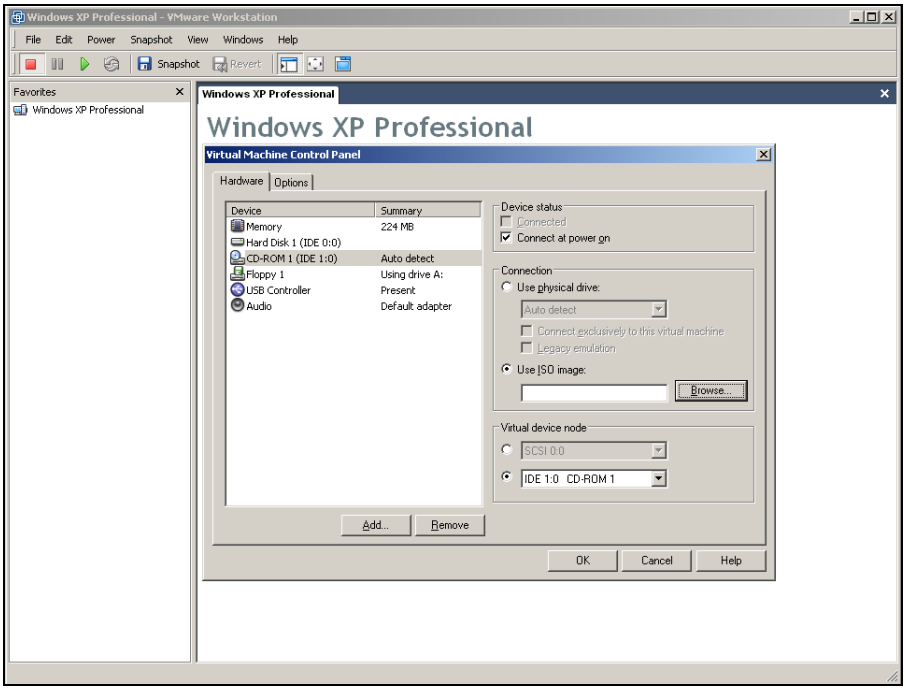



**Рис. 12.12.** Программа VMWare
для тестирования созданного файла-образа дистрибутива Windows

## Запись файла-образа на диск

Сегодня почти любая программа, предназначенная для записи компакт-дисков, поддерживает возможность записи дисков из файлов-образов. Опишем процесс записи ISO-файла на диск на примере программы Ahead Nero. Запустив Nero, необходимо открыть ISO-файл: **File | Open File**, затем запустить процесс записи: **Recorder-Burn Compilation**. В появившемся диалоговом окне нужно проверить, что выбран метод записи диска **Disk-at-once** и отмечена опция **Finalize CD** (рис. 12.13).

Если вы уверены в том, что вы не допустили ошибок в файле ответов и формировании структуры подкаталогов дистрибутива Windows, то вы можете сразу приступить к записи дистрибутива на диск без создания и тестирования файла-образа. При этом следует учесть, что необходимо правильно указать метку тома диска, путь к файлу, содержащему загрузчик, отключить имитацию дискеты и выбрать количество загрузочных секторов — **4** (рис. 12.14).

Созданный диск предназначен для автоматической установки Windows и типового набора программ как для рабочей станции, которая будет впоследствии введена в домен, так и для домашнего компьютера.

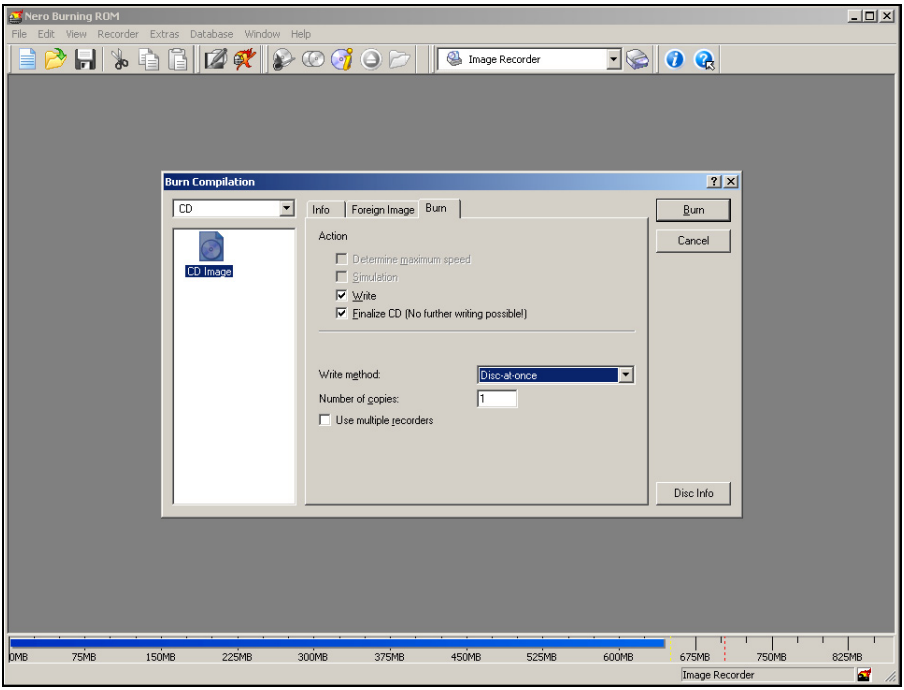



**Рис. 12.13.** Запись файла-образа с помощью Ahead Nero. Шаг 1

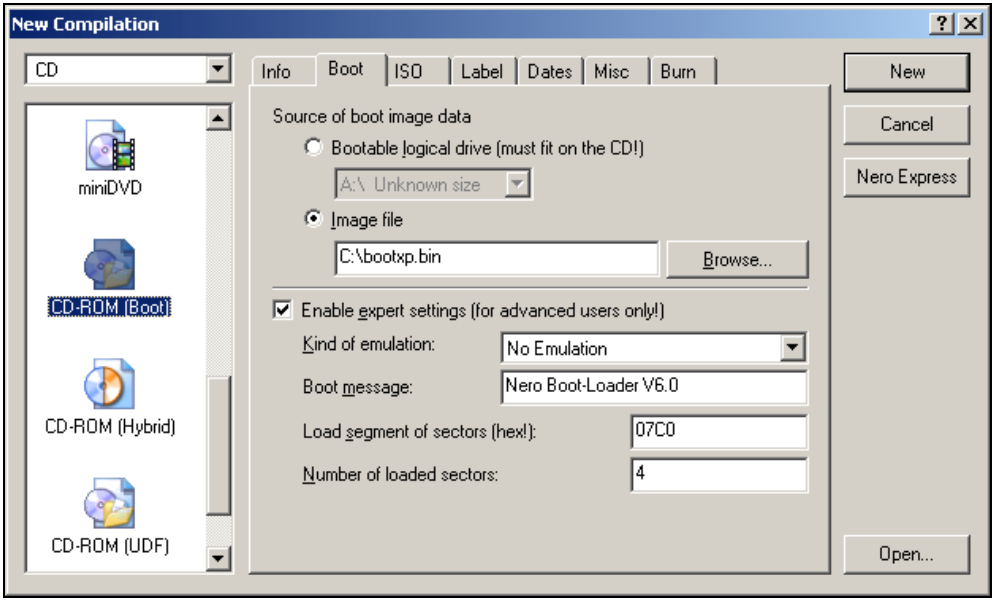



**Рис. 12.14.** Запись файла-образа с помощью Ahead Nero. Шаг 2

# ПРИЛОЖЕНИЯ



# Приложение 1

# Управление сетевой печатью

## Сетевой принтер

Любой сетевой принтер представляет собой печатающее устройство, снабженное сетевым аппаратным обеспечением, а именно: сетевой картой, дополнительной памятью, а в некоторых случаях, жестким диском для буферизации больших документов. Для обеспечения связи эти принтеры используют протоколы, поддерживаемые вашей сетью, как правило, таким транспортным протоколом является TCP/IP, и имеют собственные IP-адреса. Сетевой принтер имеет Web-интерфейс, позволяющий быстро и просто удаленно изменять настройки печатающего устройства.

Соединение локальных принтеров с сетью может быть реализовано с помощью Jet Direct. Jet Direct — это устройство, преобразующее интерфейс LPT/USB в сетевой. С помощью этого устройства к сети может быть подключено несколько принтеров.

Сетевые принтеры чаще всего используются с серверами печати, но могут функционировать и самостоятельно в сети, что экономично, поскольку не требует приобретения сервера.

Тем не менее, если нет сервера, формирующего очередь печати, то каждый пользователь создает свою собственную очередь печати и не может видеть, где его документ находится относительно других документов в глобальной очереди принтера. Из-за отсутствия единой очереди печати невозможно централизованное управление заданиями. Только пользователь, документ которого в настоящее время печатается на принтере, в случае ошибки печати, может ее видеть. Наконец, предварительная печать документов осуществляется на рабочей станции пользователя, что увеличивает нагрузку на нее.

По этим причинам желательно наличие сервера печати при работе с сетевыми принтерами.

# Сервер печати

Сервер печати — это компьютер, который управляет связями между принтерами и рабочими станциями в сети, желающими воспользоваться предоставляемыми услугами.

В сетях, построенных на основе Microsoft Windows, в качестве сервера печати может быть использован Windows 2000 Professional или Windows 2000/2003 Server. В качестве сервера выгоднее использовать Windows 2000/2003 Server, поскольку Windows 2000 Professional поддерживает всего лишь 10 подключений одновременно и не может осуществлять поддержку клиентов Macintosh или NetWare, если таковые имеются в сети.

Принтеры могут соединяться с сервером печати либо по сети, либо напрямую — через параллельный интерфейс, хотя второй вариант менее предпочтителен, поскольку требует значительных затрат процессорного времени на обслуживание порта и физически "привязывает" печатающее устройство к серверу, т. к. кабель, соединяющий LPT-порты сервера и принтера, имеет ограничения по длине.

Принтеры, соединяющиеся с сервером печати через USB или IEE1394 (FireWire), позволяют уменьшить нагрузку на процессор и увеличить скорость печати, но, несмотря на это, самым популярным способом подключения является сетевое соединение.

# Соглашение об именах

Имя должно содержать как можно больше информации о принтере, и при этом быть удобным для использования. На рабочих станциях под управлением операционной системы Windows, пользователь имеет дело с двумя именами — именем принтера и сетевым именем. Имя принтера — это имя, назначаемое принтеру во время установки. Длина имени ограничивается 220 символами. Сетевое имя назначается принтеру для использования в сети. Максимальная длина сетевого имени составляет 80 символов, хотя его не рекомендуется делать длиннее 8 символов для обеспечения совместимости с клиентами MS-DOS и Windows 3.x. Некоторые приложения не могут работать с принтерами, у которых полное составное имя (имя компьютера, объединенное с сетевым именем принтера по шаблону `\\Server_Name\Printer_Share_Name`) длиннее 31 символа.

Чаще всего, имя, назначаемое принтеру, представляет собой реальное имя принтера с порядковым номером, если есть несколько принтеров одинаковой модели, например, `HP LaserJet 2300 (1)`, `HP LaserJet 2300 (2)`. Такой способ именования рекомендуется использовать в небольших организациях.

В крупных корпорациях принцип именования принтеров может быть другим. Например, названия могут быть даны по именам отделов и офисов, где территориально находится принтер, например `Краснодар ОКС` или `Ростов АТС-34`. Сетевое имя, как отмечено ранее, должно быть более коротким, но при этом не должно терять смысловой нагрузки. Оно чаще всего представляет собой общую характеристику принтера, например, `HP2300_1`.

# Установка и настройка сетевого принтера

Для подключения принтера, имеющего USB- или LPT-интерфейс к сети, необходимо использовать Jet Direct, который представляет собой небольшой компьютер, снабженный сетевой платой, оперативной памятью, имеющий программное обеспечение — Web-интерфейс, с помощью которого осуществляется настройка.

В принтере, имеющем встроенный сетевой интерфейс, находится компьютер, в котором присутствует оперативная память, жесткий диск для обеспечения буферизации заданий и более сложное программное обеспечение, доступ к которому также осуществляется через Web-интерфейс или через панель управления на принтере.

Перед тем как начать установку принтера на сервере печати, необходимо выполнить настройку сетевого интерфейса принтера. Настройку удобнее всего производить через Web-интерфейс.

Доступ к Web-интерфейсу принтера осуществляется с помощью любого браузера (Internet Explorer, Opera или Netscape Navigator) по IP-адресу или по сетевому имени принтера. Некоторые принтеры имеют заводскую предустановку IP-адреса, о чем обязательно написано в инструкции. В том случае, если IP-адрес не известен, необходимо распечатать конфигурационный лист принтера. В том случае, если в принтер сетевой интерфейс встроен, то печать осуществляется с помощью выбора соответствующего пункта меню на панели управления принтера. Если сетевой интерфейс не встроен в принтер, т. е. используется Jet Direct, необходимо нажать на кнопку, находящуюся на корпусе Jet Direct, которая инициализирует вывод текущей конфигурации устройства на печать.

Конфигурационная страница принтера, имеющего встроенный сетевой интерфейс или Jet Direct, содержит следующую информацию: модель и серийный номер принтера, параметры настройки сетевого адаптера, его серийный номер, MAC-адрес, количество распечатанных и застрявших в принтере страниц, настройки протоколов TCP/IP, IPX/SPX, AppleTalk и др.

После вывода конфигурационного листа на печать в разделе, посвященном настройкам протокола TCP/IP, необходимо определить IP-адрес сетевого принтера для того, чтобы соединиться с его Web-интерфейсом.

Поскольку сеть функционирует на основе протокола TCP/IP, то все остальные протоколы связи в принтере/Jet Direct можно отключить через Web-интерфейс.

В настройках оставшегося протокола необходимо указать сетевое имя принтера, по которому в будущем будет удобно обращаться к Web-интерфейсу принтера. Сетевое имя принтера (host name) должно соответствовать принятому соглашению об именах, например, `HP2300_1`. Это имя автоматически будет прописано в службе DNS. Если необходимо задать альтернативное сетевое имя принтера, то можно вручную создать соответствующую запись в службе DNS.

Также нужно определить способ получения IP-адреса. Для этого необходимо выполнить настройку свойств протокола TCP/IP так, чтобы IP-адрес выделялся службой DHCP.

В том случае, если служба DHCP не используется, то необходимо на Web-интерфейсе принтера указать статический IP-адрес принтера.

Чтобы служба DHCP всегда выделяла один и тот же адрес, необходимо его зарезервировать для данного сетевого устройства. Резервирование адреса осуществляется на основе MAC-адреса сетевого адаптера. MAC-адрес (Medium Access Control) — это уникальное число длиной 48-бит, использующееся для установки соответствия между TCP/IP и адресом канального уровня.

Значение MAC-адреса можно найти на конфигурационном листе, Web-интерфейсе принтера или на корпусе принтера/Jet Direct.

Следующий этап — установка сетевого принтера. На сервере печати сетевой принтер устанавливается в качестве локального устройства с той разницей, что вместо локального порта (параллельного или USB) создается стандартный порт TCP/IP, который имеет два параметра — IP-адрес и название порта. В качестве IP-адреса задается зарезервированный в службе DHCP адрес, если служба DHCP не используется, то указывается статический IP-адрес, введенный в настройках TCP/IP на Web-интерфейсе принтера.

В процессе установки принтера задаются имя принтера и сетевое имя принтера. Оба имени должны соответствовать принятому соглашению об именах. Желательно, чтобы сетевое имя принтера (share name) совпадало с сетевым именем принтера (host name), заданным в Web-интерфейсе. Таким образом, сетевое имя принтера (host name), по которому можно войти в Web-

интерфейс, будет совпадать с сетевым именем принтера (share name), по которому будет осуществляться подключение принтера к пользователю. Такая унификация имен дает нам отсутствие путаницы в настройках, обеспечивает простоту визуального восприятия.

После установки принтера на сервере печати необходимо убедиться, что в его настройках выбрана опция **List in Active Directory**.

Для реализации автоматического подключения принтера к пользователям, имеющим право печатать и управлять очередью печати, необходимо опубликовать принтер в службе каталогов Active Directory. В AD для каждого сетевого принтера рекомендуется создать OU (Organization Unit), в котором разместится опубликованный в AD принтер и 2 группы безопасности, определяющие уровни доступа к нему пользователей. В соответствии с принятым соглашением об именах рекомендуется использовать следующий шаблон для создания названий групп. Название первой группы, члены которой могут только выводить задания на печать, рекомендуется строить следующим образом: к сетевому имени принтера через дефис прибавляется слово `Print`, например, `HP2300_1 – Print`. Название второй группы строится аналогично, с той разницей, что слово `Print` рекомендуется заменить на словосочетание `Print Managers`. Члены второй группы могут управлять очередью печати и принтером. О том, как задаются права на группы — чуть позже.

Подключение сетевых принтеров осуществляется при регистрации пользователя в сети в автоматическом режиме. Пользователю, входящему хотя бы в одну из двух групп безопасности, осуществляется подключение соответствующего принтера. Если пользователь не входит ни в одну из этих групп, то сценарий отключает принтер у пользователя.

Автоматизирование подключения/отключения принтеров осуществляется с помощью сценария регистрации. Сценарий регистрации будет подробно рассмотрен во второй части статьи.

Для того чтобы обеспечить подключения по группам, необходимо на сервере печати изменить права на доступ к данному принтеру, удалив из списка объектов (пользователей и групп), которые могут осуществлять печать, группу **Everyone** (Все), и добавив туда две соответствующие данному принтеру группы. Группе `Print` необходимо выставить флажок напротив свойства **Print** (печать), группе `Print Managers` — **Manage Documents** (управление документами).

Изложив теоретические аспекты установки и настройки сетевого принтера, рассмотрим этот вопрос на практике.

# Описание установки принтера

Подключим принтер HP LaserJet 1200 с помощью сетевого интерфейса к серверу печати. Обеспечим сетевому принтеру статический IP-адрес `192.168.2.1` службой DHCP. Опубликуем его в Active Directory для обеспечения автоматического управления подключением принтеров с помощью сценария регистрации пользователей в сети и создадим две группы, определяющие различный уровень доступа к принтеру и очереди печати.

Принтер HP LaserJet 1200 имеет два интерфейса — LPT и USB. Для подключения принтера к сети необходимо использовать Jet Direct, который преобразует один из интерфейсов в сетевой. После соединения принтера с Jet Direct и подключения получившегося сетевого принтера к сети, необходимо вывести на печать конфигурационный лист.

Для этого нажмите на корпусе Jet Direct соответствующую кнопку. Анализируя конфигурационный лист, необходимо определить IP-адрес, который получен сетевым адаптером Jet Direct. Воспользовавшись любым браузером, например, Internet Explorer, войдите на Web-страницу принтера, вводя к адресной строке IP-адрес, приведенный в листе конфигурации сетевого принтера, в данном случае Jet Ditect.

С помощью Web-интерфейса необходимо выключить все протоколы, кроме TCP/IP. Протокол TCP/IP следует настроить так, чтобы IP-адрес сетевой адаптер получал от службы DHCP и назначить сетевое имя принтера (host name) в соответствии с поставленной задачей и принятым соглашением об именах — `HP1200_1` (рис. П1.1). На этом настройка сетевого интерфейса принтера с помощью Web-интерфейса завершена.

Следующим этапом установки сетевого принтера в сети является резервирование для него IP-адреса. Для того чтобы зарезервировать IP-адреса в службе DHCP, выделите папку Reservation и, нажав правую кнопку на этой папке, выберите пункт меню **New Reservation**. Появится окно (рис. П1.2), в котором необходимо указать имя, IP-адрес и MAC-адрес принтера.

На рис. П1.2 показано, что служба DHCP будет выдавать устройству с именем `HP1200_1`, обладающим сетевым интерфейсом с MAC-адресом `0001e64a49cb`, один и тот же IP-адрес — `192.168.2.1`.

После настройки службы DHCP и Web-интерфейса можно приступить к установке сетевого принтера на сервере печати. Принтер устанавливается как локальный, с той разницей, что он подключается к стандартному порту TCP/IP. Порт имеет два параметра — имя и IP-адрес.

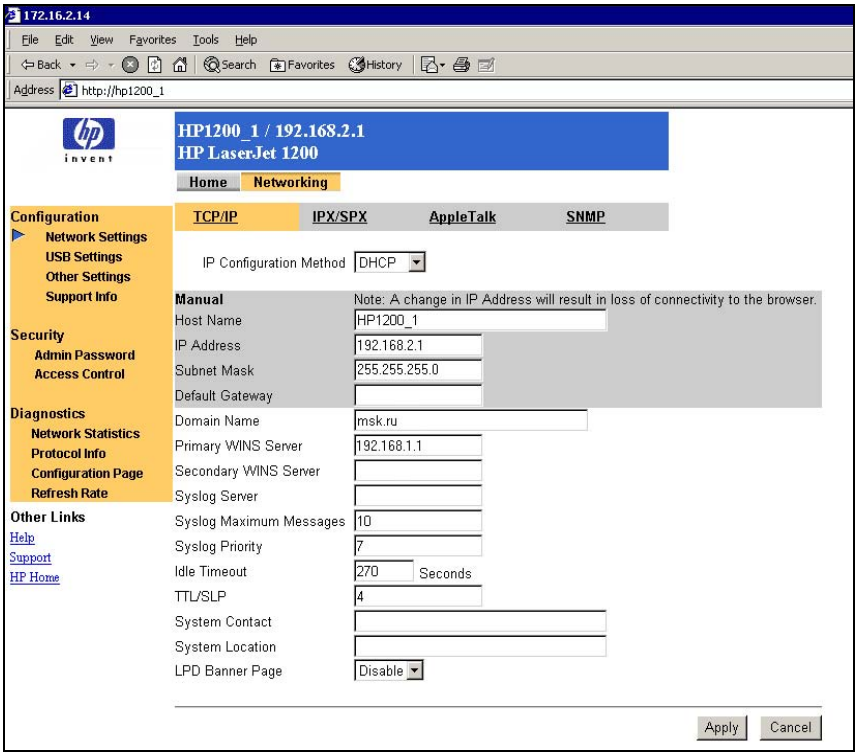



**Рис. П1.1.** Web-интерфейс сетевого принтера

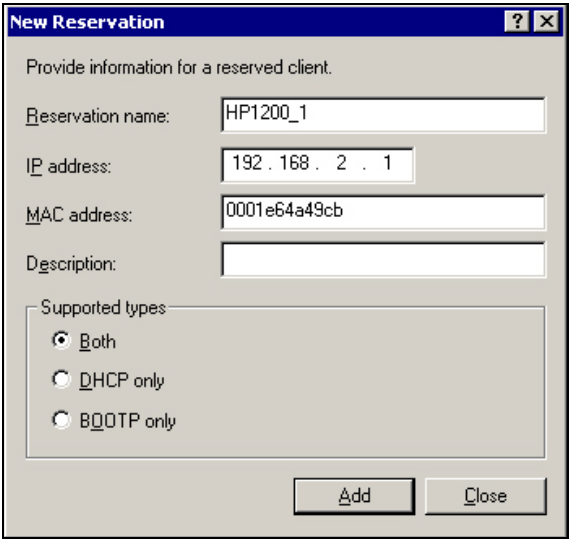



**Рис. П1.2.** Резервирование в DHCP-службе IP-адреса для принтера

Имя порта образуется исходя из IP-адреса: `IP_xxx.xxx.xxx.xxx`. Однако рекомендуется сделать так, чтобы имя порта совпадало с сетевым именем принтера. Этот шаг упростит администрирование принтера в будущем. Для установки принтера на сервере печати необходимо выполнить следующее:

1. Щелкнуть по кнопке **Start**, выбрать команду **Settings**, затем щелкнуть на **Printers**, чтобы открыть папку Printers.
2. Дважды щелкнуть на значке **Add Printers**, чтобы запустить мастер установки принтера.
3. Щелкнуть по кнопке **Next**, чтобы мастер **Add Printer Wizard** начал работу.
4. В появившемся окне предлагается сделать выбор способа подключения принтера — в качестве локального или сетевого. Необходимо выбрать опцию **Local Printer** (Локальный принтер) и удалить флажок из поля **Automatically Detect My Printer** (Автоматическое определение принтера).
5. В следующем окне нужно определить тип порта. Выберите опцию **Create A New Port** (Создать новый порт), а в раскрывающемся списке — **Standard Port TCP/IP** (Стандартный порт TCP/IP). После этого Windows 2000/2003 запустит мастер **Add Standard TCP/IP Printer Port Wizard**.
6. В появившемся окне необходимо ввести IP-адрес и название порта в соответствии с поставленной задачей (рис. П1.3). Помните, желательно, чтобы принтер был подключен к сети, т. к. после создания порта Windows пытается соединиться с принтером и в случае возникновения ошибки связи будет запрошена дополнительная информация (рис. П1.4). После успешного соединения с принтером мастер **Add Standard TCP/IP Printer Port Wizard** заканчивает свою работу.
7. Выберите устанавливаемый принтер из списка. В том случае, если драйвера принтера не являются стандартными, нажмите на кнопку **Have Disk** и укажите путь к драйверу.
8. После установки драйвера в появившемся окне **Printer Sharing** выберите опцию **Share As** для того, чтобы сделать принтер доступным для других пользователей сети, и укажите сетевое имя принтера в соответствии с принятым соглашением об именах, например, `HP1200_1` (рис. П1.5). Щелкните по кнопке **Next**.
9. Введите локальное имя принтера (рис. П1.6) и заполните поля, характеризующие принтер, например его местоположение. Локальное имя принтера также должно удовлетворять принятому соглашению об именах, например, `HP LaserJet 1200 (1)`.
10. По окончании установки принтера вам будет предложено распечатать тестовую страницу.

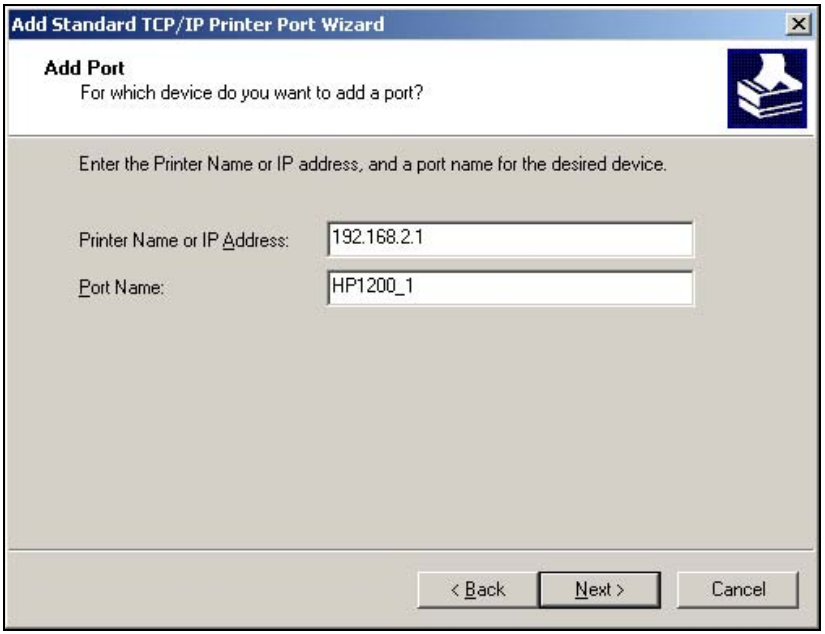



**Рис. П1.3.** Мастер создания порта TCP/IP. Задание параметров

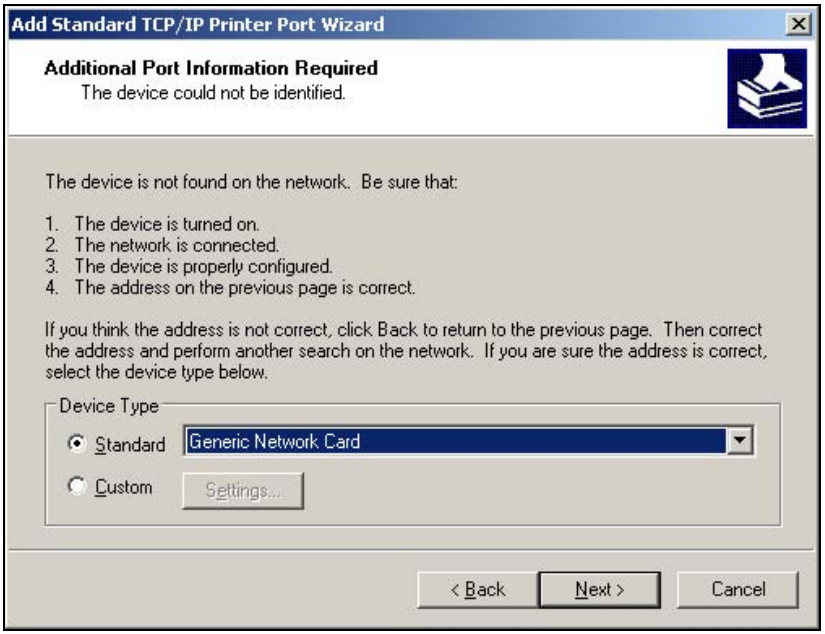



**Рис. П1.4.** Мастер создания порта TCP/IP. Определение типа сетевой карты

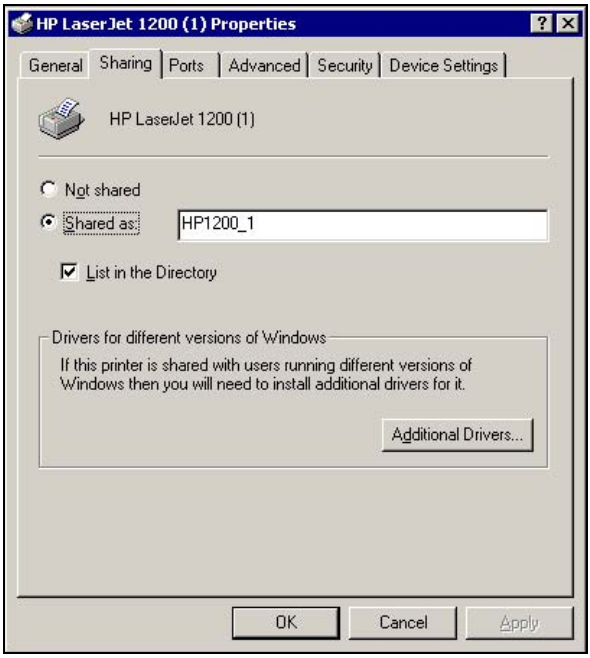



**Рис. П1.5.** Сетевое имя принтера

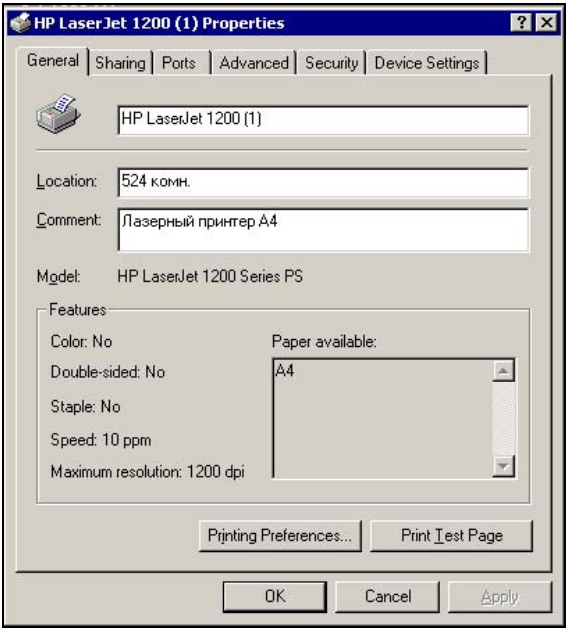



**Рис. П1.6.** Название и комментарии принтера

На этом настройка принтера не закончена: необходимо определить параметры безопасности принтера, для чего в Active Directory необходимо создать две группы безопасности. Члены одной из этих групп смогут только печатать на этом принтере, члены другой — управлять всей очередью печати и состоянием принтера. В соответствии с принятыми правилами наименования групп, первая группа будет называться `HP1200 – Print`, вторая `HP1200 – Print Managers`. Поскольку в средних и крупных организациях используется несколько сетевых принтеров, то в AD рекомендуется создать `OU`, например, `Network Printers`, в котором будут находиться `OU`, названия которых совпадают с названиями принтеров. В каждом из этих `OU` будет содержаться три объекта — сетевой принтер, опубликованный в AD, и две группы безопасности, определяющие уровень доступа к принтеру (рис. П1.7). Наличие такой структуры позволяет реализовать автоматизированное подключение сетевых принтеров тем пользователям, которые имеют права работать с этим принтером.

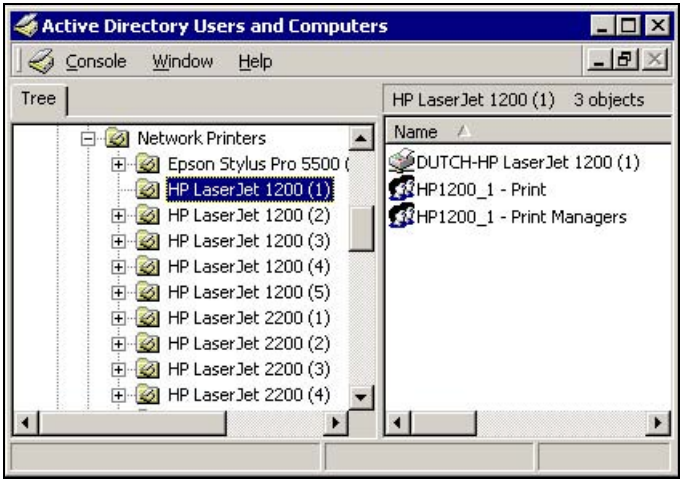



**Рис. П1.7.** Опубликованный в AD принтер

После окончания настройки нужно выполнить настройку безопасности принтера. Для этого необходимо в свойствах принтера на сервере печати открыть вкладку **Security** (Безопасность). На этой вкладке нужно удалить группу **Everyone** (Все), т. к. в противном случае принтер будет подключаться ко всем пользователям сети, и добавить две группы безопасности, соответствующие данному принтеру, в данном случае `HP1200_1 – Print` и `HP1200_1 – Print Managers`. Для группы `HP1200_1 – Print` необходимо установить в разделе **Permissions** (Разрешения) флажок напротив свойства **Print** (рис. П1.8), а для группы `HP1200_1 – Print Managers` — флажки напротив **Print** (Печать)

и **Manage Documents** (Управление документами). Ставить флажок напротив **Manage Printers** не рекомендуется, поскольку управление принтерами подразумевает возможность изменять свойства принтера, удалять его. По мнению автора, такими привилегиями может обладать только системный администратор.

На этом установка и настройка сетевого принтера завершены.

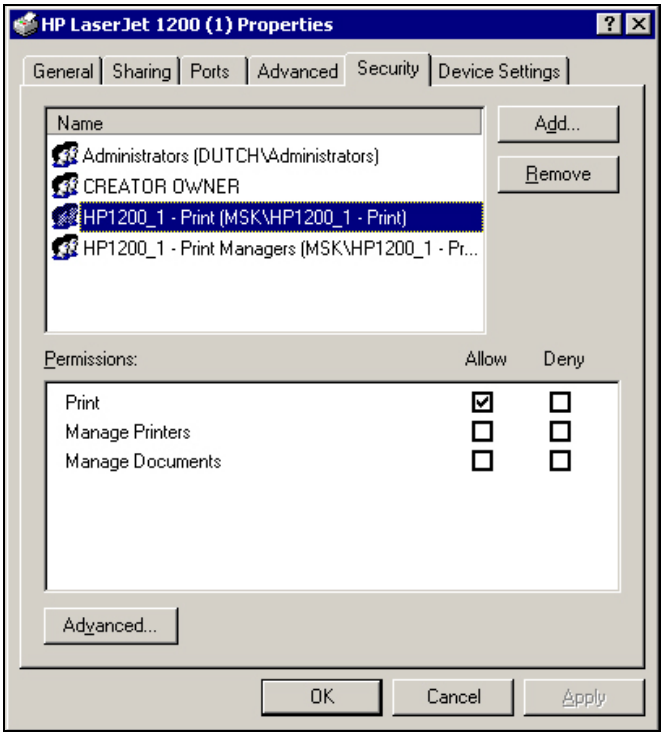
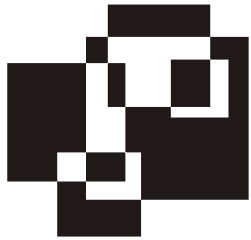



**Рис. П1.8.** Параметры безопасности принтера



# Приложение 2

# Ошибки выполнения сценариев в WSH

Ошибки, которые могут возникнуть при выполнении WSH-сценариев, с описанием возможных причин приведены в табл. П2.1.

***Таблица П2.1.*** *Ошибки выполнения сценария WSH*

| Сообщение об ошибке | Причина |
|---|---|
| A duplicate name for a named or unnamed element was encountered: xxx | Попытка повторного использования имени аргумента |
| Argument list too long | Связано с запуском сценария с помощью технологии Drag-and-Drop: превышена максимальная длина командной строки |
| Cant save settings | Ошибка при сохранении файла с настройками сценария (*.wsh) |
| Environment variable <name> could not be removed | Вызов метода `Environment.Remove` для несуществующей переменной среды |
| Invalid attempt to call Exec without a command | Вызов метода `WshShell.Exec` без указания аргумента (команды для выполнения) |
| Invalid shortcut path name | Попытка создать ярлык с неправильным расширением файла (расширение должно быть ink или url) |
| Printer <name> not found | Неправильно указано имя принтера при вызове метода `SetDefaultPrinter` |
| Protocol handler for <name> could not be found | Попытка установить ярлык на сетевой ресурс, использующий некорректно зарегистрированный обработчик протокола |
| Registry key <name> contains invalid root | Вызов метода `RegRead` или `RegWrite` для некорректного ключа реестра |

***Таблица П2.1*** *(окончание)*

| Сообщение об ошибке | Причина |
|---|---|
| Registry key <name> could not be opened | Вызов метода `RsgRead` для несуществующего ключа реестра |
| Registry key <name> could not be removed | Вызов метода `RegDelete` для несуществующего ключа реестра |
| Remote script object can only be executed once | Попытка повторно запустить объект — удаленный сценарий |
| Shortcut <name> contains invalid syntax | Сохранение ярлыка на сетевой ресурс, имеющий некорректный URL |
| Shortcut <name> could not be saved | Попытка сохранить новый ярлык в файле, который уже существует и имеет атрибут "Только для чтения" |
| Shortcut <name> failed to execute protocol handler | Попытка установить ярлык на сетевой ресурс, использующий несуществующий обработчик протокола |
| Unable to execute remote script | Невозможно создать процесс — удаленный сценарий |
| Unable to find job <job identifier> | В WS-файле нет задания с идентификатором `<jcb identifier>` |
| Unable to wait for process | С помощью метода `Run` дано указание ожидать завершение процесса, которое из сценария определить нельзя |



# Приложение 3

# Объектная модель провайдера WinNT

## objectClass *Domain*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| `AutoUnlockInterval` | Число | Интервал в секундах, после которого учетная запись автоматически разблокируется, если она была заблокирована. Для установки бесконечного времени ожидания устанавливается значение, равное "–1" |
| `LockoutObservationInterval` | Число | Интервал в секундах, в течение которого контроллер домена хранит число неверных попыток регистрации в домене |
| `MaxBadPasswordsAllowed` | Число | Максимальное количество попыток неправильного ввода пароля, после которого блокируется учетная запись. Значение параметра должно быть в промежутке от 0 до 999. Для снятия ограничения на длину пароля значение параметра должно быть "–1" |
| `MaxPasswordAge` | Число | Срок действия пароля. Пароль имеет неограниченный срок действия, если значение параметра равно "–1" |
| `MinPasswordAge` | Число | Временной интервал в секундах, в течение которого пользователь не сможет сменить свой пароль |

*(окончание objectClass Domain)*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| MinPasswordLength | Число | Минимальная длина пароля. Если необходимо разрешить использовать пустые пароли, установите значение, равное "–1" |
| Name | Строка | Краткое имя домена |
| PasswordHistoryLength | Число | Число паролей, которое хранится контроллером домена для каждого пользователя. Данное свойство предотвращает повторное использование паролей. Для деактивации хранения истории паролей установите значение параметра равным "–1" |

## objectClass *User*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| AccountExpirationDate | Дата | Не поддерживается Windows 2k |
| AutoUnlockInterval | Число | Интервал в секундах, после которого учетная запись автоматически разблокируется, если она была заблокирована. Для установки бесконечного времени ожидания устанавливается значение, равное "–1" |
| BadPasswordAttempts | Число | Количество неправильно введенных паролей в течение времени, указанного в параметре LockoutObservationInterval |
| Description | Строка | Описание пользователя. В Windows 2k, как правило, используются другие поля, доступные только через протокол LDAP |
| FullName | Строка | Полное имя пользователя. Имеет формат Second Name, First Name |
| HomeDirDrive | Буква | Буква домашнего каталога. Формат: буква: |
| HomeDirectory | Строка | Путь в формате UNC к домашнему каталогу в виде \\Server\Folder |

*(продолжение objectClass User)*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| `UserFlags` | Число | Флаг пользователя, с помощью которого определяется тип пользователя |
| `LockoutObservationInterval` | Число | Интервал в секундах, в течение которого контроллер домена хранит число неверных попыток регистрации в домене |
| `LoginHours` | Массив | Не поддерживается Windows 2k |
| `LastLogin` | Дата | Дата и время последней регистрации пользователя в сети |
| `LastLogoff` | Дата | Не поддерживается Windows 2k |
| `LoginScript` | Строка | Сценарий загрузки |
| `LoginWorkstations` | Строка | Не поддерживается Windows 2k |
| `MinPasswordAge` | Строка | Временной интервал в секундах, в течение которого пользователь не сможет сменить свой пароль |
| `MinPasswordLength` | Строка | Минимальная длина пароля. Если необходимо разрешить использовать пустые пароли, установите значение, равное "–1" |
| `MaxBadPasswordsAllowed` | Число | Максимальное количество неправильного ввода пароля, после которого блокируется учетная запись. Значение параметра должно быть в промежутке от 0 до 999. Для снятия ограничения на длину пароля значение параметра должно быть "–1" |
| `MaxLogins` | Строка | Не поддерживается Windows 2k |
| `MaxPasswordAge` | | Срок действия пароля. Пароль имеет неограниченный срок действия, если значение параметра равно "–1" |
| `ObjectSid` | Строка | Не поддерживается Windows 2k |
| `PasswordAge` | Число | Время в секундах с момента смены пароля |
| `PasswordExpirationDate` | Дата | Дата и время окончания действия пароля |
| `PasswordExpired` | Число | Принимает значение 0/1.0 |

*(окончание objectClass User)*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| PasswordHistoryLength | | Число паролей, которое хранится контроллером домена для каждого пользователя. Данное свойство предотвращает повторное использование паролей. Для деактивации хранения истории паролей установите значение параметра равным "–1" |
| PrimaryGroupID | Число | ID-номер основной группы пользователя |
| Profile | Строка | Содержит путь, где хранится профиль в UNC-формате |

## objectClass *Group*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| Description | Строка | Описание группы |
| objectSid | Строка | SID группы |

## objectClass *Computer*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| Division | Строка | Название организации |
| Owner | Строка | Владелец компьютера |
| OperatingSystem | Строка | Тип операционнной системы |
| OperatingSystemVersion | Строка | Версия операционной системы |
| Processor | Строка | Тип процессора |
| ProcessorCount[1] | Строка | Количество процессоров |

[1] Логично предположить, что свойство возвращает количество установленных процессоров, однако это не так: оно отображает используемый в системе HAL (аппаратный уровень абстракций).

## objectClass *PrintQueue*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| PrinterPath | Строка | Путь к принтеру в формате UNC |
| PrinterName | Строка | Название принтера |
| Model | Строка | Модель принтера |
| Datatype | Строка | Тип данных |
| PrintProcessor | Строка | Процессор печати |
| PrintDevices | Строка | Порт печати (USB, LPT, IP_xxx.xxx.xxx.xxx) |
| Description | Строка | Описание |
| HostComputer | Строка | Компьютер, к которому подключен принтер |
| Location | Строка | Размещение устройства |
| StartTime | Строка | Время начала печати |
| UntilTime | Строка | Время окончания печати |
| DefaultJobPriority | Число | Приоритет, присваиваемый документу при постановке в очередь печати |
| JobCount | Число | Количество заданий в очереди печати |
| Priority | Число | Приоритет задания |
| BannerPage | Строка | Путь к файлу, используемому для отделения заданий печати |

## objectClass *PrintJob*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| HostPrintQueue | Строка | Имя принтера (ShareName) |
| User | Строка | Имя учетной записи пользователя, пославшего задание на печать |
| TimeSubmitted | Число | Общее время печати страниц |
| TotalPages | Число | Количество страниц в задании |
| Size | Число | Размер задания |

*(окончание objectClass PrintJob)*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| Description | Строка | Описание задания |
| Priority | Число | Приоритет задания |
| StartTime | Число | Время начала печати задания |
| UntilTime | Число | Время окончания печати задания |
| Notify | Строка | Идентификатор пользователя, по которому будет отправлено уведомление об окончании печати |
| TimeElapsed | Число | Время, оставшееся до окончания печати |
| PagesPrinted | Число | Количество уже отпечатанных страниц задания |

## objectClass *FileService*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| HostComputer | Строка | Название рабочей станции |
| DisplayName | Строка | Имя ресурса |
| Version | Строка | Версия |
| Path | Строка | Путь в формате UNC |
| Description | Строка | Описание |
| MaxUserCount | Число | Максимальное количество одновременно подключенных пользователей |

## objectClass *FileShare*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| CurrentUserCount | Строка | Количество пользователей, использующих ресурс в настоящее время |
| Description | Строка | Описание |

*(окончание objectClass FileShare)*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| `HostComputer` | Строка | Название рабочей станции, на которой расположен ресурс, предоставленный в общее пользование |
| `Path` | Строка | Путь в формате UNC |
| `MaxUserCount` | Число | Максимальное количество одновременно подключенных пользователей |

## objectClass *Service*

| Поддерживаемые свойства | Тип данных | Описание |
|---|---|---|
| `HostComputer` | Строка | Компьютер, на котором запущен сервис |
| `ServiceAccountName` | Строка | Учетная запись названия сервиса |
| `Dependencies` | Массив | Связывание служб |
| `StartType` | Строка | Тип запуска службы |
| `ServiceType` | Строка | Тип сервиса |
| `DisplayName` | Строка | Показываемое имя сервиса |
| `Path` | Строка | Путь к сервису |
| `ErrorControl` | Строка | Контроль ошибок |



# Приложение 4

# Таблица ASCII (American Standard Code for Information Interchange)

***Таблица П4.1.*** *ASCII*

| Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | NUL | 32 | 20 | (sp) | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | SOH | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | STX | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | ETX | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | EOT | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | ENQ | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | ACK | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | BEL | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | BS | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | TAB | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | LF | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | VT | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | FF | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | CR | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | SO | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | SI | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | DLE | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | DC1 | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |

***Таблица П4.1*** *(окончание)*

| Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | 12 | DC2 | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | DC3 | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | DC4 | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | NAK | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | SYN | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | ETB | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | CAN | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | EM | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | SUB | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | ESC | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | FS | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | GS | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | RS | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | US | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | DEL |

# Назначение специализированных символов

## Форматирование

- BS или Backspace (Возврат на один символ) — указывает на движение механизма печати или курсора дисплея назад на одну позицию.
- HT или Horizontal Tabulation (Горизонтальное табулирование) — указывает на движение механизма печати или курсора дисплея до следующей предписанной позиции табуляции.
- LF или Line Feed (Перевод строки) — указывает на движение механизма печати или курсора дисплея к началу следующей строки (на одну строку вниз).
- VT или Vertical Tabulation (Вертикальное табулирование) — указывает на движение механизма печати или курсора дисплея к следующей группе строк.

- FF или Form Feed (Перевод страницы) — указывает на движение механизма печати или курсора дисплея к исходной позиции следующей страницы, формы или экрана.
- CR или Carriage Return (Перевод каретки) — указывает на движение механизма печати или курсора дисплея к исходной (крайней левой) позиции текущей строки.

## Передача данных

- SOH или Start of Heading (Начало заголовка) — используется для указания начала заголовка, который может содержать информацию о маршрутизации или адрес.
- STX или Start of Text (Начало текста) — указывает на начало текста и одновременно на конец заголовка.
- ETX или End of Text (Конец текста) — используется при завершении текста, который был начат с символа STX.
- ENQ или Enquiry (Запрос) — запрос идентификационных данных от удаленной станции.
- ACK или Acknowledge (Подтверждение) — приемное устройство передает этот символ отправителю в качестве подтверждения успешного приема данных.
- NAK или Negative Acknowledgement (Неподтверждение) — приемное устройство передает этот символ отправителю в случае отрицания (неудачи) приема данных.
- SYN или Synchronous/Idle (Синхронизация) — используется в синхронизированных системах передачи. В моменты отсутствия передачи данных система непрерывно посылает символы SYN для обеспечения синхронизации.
- ETB или End of Transmission Block (Конец блока передачи) — указывает на конец блока данных для коммуникационных целей. Используется для разбиения на отдельные блоки больших объемов данных.

## Разделительные знаки при передаче информации

- FS или File Separator (Разделитель файлов).
- GS или Group Separator (Разделитель групп).
- RS или Record Separtator (Разделитель записей).
- US или Unit Separator (Разделитель элементов).

## Другие символы

- NUL или Null (No character, нет данных) — используется для передачи в случае отсутствия данных.
- BEL или Bell (Звонок) — используется для управления устройствами сигнализации.
- SO или Shift Out — указывает, что все последующие кодовые комбинации должны интерпретироваться согласно внешнему набору символов до прихода символа SI.
- SI или Shift In — указывает, что последующие кодовые комбинации должны интерпретироваться согласно стандартному набору символов.
- DLE или Data Link Escape (Переключение) — изменение значения идущих следом символов. Используется для дополнительного контроля или для передачи произвольной комбинации бит.
- DC1, DC2, DC3, DC4 или Device Controls (Контроль устройства) — символы для управления вспомогательными устройствами (специальными функциями).
- CAN или Cancel (Отмена) — указывает, что данные, которые предшествовали этому символу в сообщении или блоке, должны игнорироваться (обычно в случае обнаружения ошибки).
- EM или End of Medium (Конец носителя) — указывает на физический конец ленты или другого носителя информации.
- SUB или Substitute (Заместитель) — используется для подмены ошибочного или недопустимого символа.
- ESC или Escape (Расширение) — используется для расширения кода, указывая на то, что следующий символ имеет альтернативное значение.
- (sp) или Space (Пробел) — непечатаемый символ для разделения слов или перемещения механизма печати или курсора дисплея вперед на одну позицию.
- DEL или Delete (Удаление) — используется для удаления (стирания) предыдущего знака в сообщении.



# Приложение 5

# Файл ответов Winnt.sif

```
[Data]
AutoPartition=0
MsDosInitiated="0"
UnattendedInstall="Yes"

[Unattended]
UnattendMode=FullUnattended
OemSkipEula=Yes
OemPreinstall=Yes
TargetPath=\windows
FileSystem = ConvertNTFS

[GuiUnattended]
AdminPassword=*
OEMSkipRegional=1
TimeZone=145
OemSkipWelcome=1
AutoLogon=yes

[UserData]
ProductID=xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx
FullName=Fname
OrgName=OName
ComputerName=xxxxx


[RegionalSettings]
LanguageGroup=5
```

```
SystemLocale=00000419
UserLocale=00000419
InputLocale=0419:00000419


[Networking]
 InstallDefaultComponents=Yes

[Shell]
DefaultStartPanelOff=Yes
DefaultThemesOff=Yes

[GuiRunOnce]
"c:\Install\MUIINST\MUISETUP.EXE /i 0419 /d 0419 /r /s"
"c:\Install\NAV\setup.exe /qn"
"c:\Install\WinCmd\install.exe"
"c:\Install\WinRar\Wrar.exe /s"
"regedit /s c:\Install\WinRar\Wrar304.reg"
"c:\Install\Nero\Nero.exe /silent /noreboot /sn=xxxx-xxxx-xxxx-xxxx-xxxx-
xxxx /write_sn"
"c:\Install\Areader\Areader.msi /qb"
"c:\Install\Office\setup.exe transforms= "C:\Install\Office\answer.mst
/qb- /noreboot"
"c:\Install\hotfix\Qxxxxxx.exe /q /n "
"%windir%\system32\cmd.exe /c rmdir c:\Install /s/q"
```

[1] Справедливо только для Windows XP. Параметры включают классический вид меню **ПУСК** и классическую схему оформления Windows, соответственно.



# Приложение 6

# Файл ответов Install.inf

```
[Installation]
program=Total Commander 6.0
progname=Total Commander
copyright=Copyright © 1993-2003 by Christian Ghisler, All Rights
reserved

[auto]
;Значение auto=1 обозначает автоматическую установку
auto=1
;Язык установки - английский
lang=1
alllang=0
;Расположение INI-файла. Можете изменить на свое.
iniloc="%programfiles%\TotalCmd"
iniall=0
;Параметр mkgroup=1 создаст группу в меню Пуск,
;но тогда в конце установки откроется окно Проводника,
;показывающее ярлыки
mkgroup=0
;Создает ярлык на Рабочем столе
mkdesktop=1

[Versioncheck]
Vernum=2

[Not running]
1=WINDOWSCMD,Windows Commander
2=TTOTAL_CMD,Total Commander
3=TAPPLICATION,Windows Commander,Windows Commander
```

```
4=TAPPLICATION,Total Commander,Total Commander

[Destination]
;Директория, в которую будет установлена программа.
;Измените на свою.
Dir="%programfiles%\TotalCmd"
Ini=wincmd.ini,Configuration,InstallDir

[Languages]
;Лишние языки удалены
Count=1
Default=1
1=English

[LangName]
langdir=language
0=wcmd_eng

[Backup data]
1=default.bar
2=no.bar

[Install]
1=install.cab,c

[Installd]
1=install.cab,c

[Desktop]
1=totalcmd.exe","Total Commander.lnk

[Group]
Groupname=Total Commander
1=totalcmd.exe","Total Commander 32
2=totalcmd.hlp","Total Commander Help
3=tcuninst.exe","Uninstall or Repair Total Commander

[ini]
1=wincmd.ini,configuration,languageini
2=wincmd.ini,configuration,Mainmenu
```

# Предметный указатель


### A

Active Directory (AD) 24, 25, 35, 95, 99, 165, 168, 180, 199, 202, 211, 213, 215, 217, 218, 219, 247, 253
Active Directory Viewer 204
ActiveX-компоненты 23
ADM-файл 120, 126, 127, 134, 135
Adobe Acrobat Reader, установка 306
ADODB 24
ADODB-соединение 220, 247, 250, 255, 262
ADSI 165, 170, 171, 200
ADSI Resource Kit 152
ASP 95, 104, 105
ASP.NET 96, 105
AutoIt 19

### B, C

BAT-файл 286
CDImage GUI 309
CIM 221
CIMOM 223
CIPHER 31
CSVDE-файл 215

### D

DHTML-файл 275, 276
DNS 166
DOS 16

### E, F, G, H

Encrypting File System (EFS) 31
FrameWork 23
Ghost 285
Hotfix 287, 299, 300, 302

### I, J

INI-файл 28, 111
Inno Setup 292
ISO-файл 309, 311, 312
JScript 17

### K

KIX2EXE 274
KIXForms 274
KIXTart 20, 30, 32, 117, 118, 237, 238, 239, 240, 272, 273, 278
KIXWin 275, 276

### L

LDAP 199, 201, 211
LDAP Browser 204, 207
LDIF-файл 214

### M

Microsoft Assembly Registration Tool 23
Microsoft Office, установка 306
Microsoft Script Encoder 32
MOF-файл 221

MOM (Microsoft Operations Manager) 234, 236
msiexec.exe 290
MUI 286, 303

**N**

Nero Burning Rom, установка 305
Norton Antivirus, установка 303
NSIS 293

**R**

Regedit 113
Regedt32 114
REG-файл 115

**S**

SAM 180, 184
Service Pack в Windows 299
SMS (Systems Management Server) 234
SQL 250
SQL-запрос 255
Symantec Ghost 284
SysPrep 285

**T, V**

Taskkill 287
VB.NET 154, 158
VBA (Visual Basic for Application) 18
VBScript 18, 33, 95, 152, 158, 174, 232, 240
Visual Studio .NET 99
VMWare 311

**W**

WBEM 221
WinBatch 17
WinCMD, установка 304
Windows 286
Windows Installer 287
WinNT 173, 176, 189, 201, 211
WinRar, установка 304
Wise Installer 295
WMI (Windows Management Instrument) 221, 222, 223, 225, 232
WQL (WMI Query Language) 224
WSH 63, 65, 240
WSH (Windows Script Host) 19, 61, 63
WSH-объект
FileSystemObject 137
Wscript 67
WScript 64
Wscript.Shell 76
WshArguments 69
WshNetwork 88
WshShell 70, 115
WshShortcut 72
WshURLShortcut 75

**X**

X.500 199
XML 248
XML-файл 249

**А**

Автоматическое преобразование типа переменной 107
Административные шаблоны 120, 134
Аппаратная конфигурация рабочей станции 221
Аппаратное обеспечение 242
Аутентификация 226

**Б**

Безопасность 225
Быстрые исправления (hotfix) 299, 300

**В**

Вывод сообщений на экран 81
Вызов диалогового окна ввода информации в VBScript 55

**Г**

Группа безопасности 253
Групповые политики 120, 125, 178

**Д**

Динамическое связывание 108
Дистрибутив Windows 295, 297, 299
Добавление существующих групп или пользователей 161
Доверительные отношения 101
Домен 203
  имя 201
Доступ к спецпапкам Windows 86
Доступ к файловой системе 138
Драйверы устройств 303

**З**

Загрузочный диск Windows 309
Запуск приложений из сценария 83
Запуск процессов 70
Запуск сценариев WSH 61

**И**

Иерархическая база данных 165
Имперсонация 225
Инсталлятор 286
Инструкции в VBScript 42, 43
Интерпретируемый язык 16

**К**

Класс 49
Классы в VBScript 49
Клонирование жестких дисков 283
Командный интерпретатор 16
Комментарии в VBScript 34
Компилируемый язык 16
Константы в VBScript 37

**М**

Массив 38, 107, 190, 219, 255
Массивы в VBScript 38, 39, 50
Метод 11
  создание 54
Многопоточные компоненты 109

**Н**

Набор 11
Неявное объявление переменных 106
Неявное объявление переменных в VBScript 36

**О**

Обработка ошибок в ASP.NET 110
Общее пользование в домене 189
Объект 9, 182
  "дочерний" 10
  атрибуты 219, 220
  перемещение 218
  поиск 220
  свойства 9, 52
  создание 215
  удаление 217
Объектная модель 11, 28, 63, 173, 204, 263
Объектно-ориентированное программирование 9
Оператор 39
Операторы в VBScript 39, 40
Отображение информации в VBScript 56
Очередь печати 192, 193

**П**

Папка 140, 141, 144
  копирование 148
  перемещение 148
  права 154
  создание 147
  удаление 147
Передача значений параметров в VBScript 49
Передача параметров подпрограммам и методам в ASP.NET 108

Переменные 33
в ASP 106
в ASP.NET 107
в VBScript 33, 37, 106
окружения 76, 79, 80
среды 112
Поддержка OLE-объектов 95
Подключение сетевых дисков 92, 260
Подключение сетевых принтеров 89, 172
Права доступа на файлы и папки 152
Привилегии пользователей 227
Принтер 189, 190, 191, 211, 252, 253, 255, 258, 259
Присвоение переменной в ASP.NET 109
Провайдер 223
Просмотр списка безопасности объекта 159
Пространства имен 165
Профиль оборудования 112
Профиль пользователя 120
Процедуры в VBScript 48

**Р**

Регистрация пользователей в сети 237, 241
Реестр 111
доступ 115
редактирование 117
редакторы 113
типы данных 113
Режим отладки сценариев в KIXTart 239

**С**

Свойства интерпретатора языка VBScript 51
Свойства объекта 174
Связывание служб 197
Система исчисления 12, 13
Скрипт:
создание 24
шифрование 31
Совместно используемые ресурсы 194, 195
Сравнение возможностей языков программирования 20
Субобъект 10

**Т**

Типы данных
в ASP.NET 107, 108
в VBScript 34, 41, 107

**У**

Удаление существующих групп или пользователей 163
Управление службами 196, 197
Управление учетной записью пользователя 98
Учетная запись компьютера 181
Учетная запись пользователя 181, 183, 187, 188, 225, 227, 247
Учетные записи пользователей 203

**Ф**

Файл 140, 141
атрибуты 144, 146
копирование 148
перемещение 148
текстовый, управление 149, 150, 151, 152
удаление 149
Файл ответов для установки Windows 296
Файл-образ диска 309
Функции в VBScript 47

**Х, Ц, Э**

Характеристики файлов 144
Циклы в VBScript 44, 45, 46
Эмуляция нажатия клавиш 85

**Я**

Языки программирования для создания скриптов 15
Ярлык 72, 73, 74, 75