Руководство
сисадмина

Коробко И. В.

PowerShell
как средство автоматического администрирования

ДМК
издательство

УДК 004.438PowerShell
ББК 32.973.26-018.1
К68

Коробко И. В.
К68 PowerShell как средство автоматического администрирования. – М.: ДМК Пресс, 2012. – 224 с.

ISBN 978-5-94074-755-0

В этой книге описываются различные средства автоматизации процессов в сети с помощью PowerShell. Уделено внимание особенностям интеграции и взаимодействия PowerShell с другими языками программирования, что позволяет значительно расширить возможности по автоматизации рутинных процессов на примере подключения различных сетевых ресурсов (дисков, принтеров, приложений) с помощью сценария регистрации пользователей в сети.

УДК004.438PowerShell
ББК 32.973.26-018.1

ISBN 978-5-94074-755-0

Содержание

Глава 1
УСТАНОВКА И НАСТРОЙКА WINDOWS POWERSHELL

В этой главе кратко описывается процедура установки и удаления Windows PowerShell. Рассказывается об особенностях настройки оболочки для выполнения настройки.

Установка PowerShell

Для каждой версии операционной системы, каждой платформы существует индивидуальная сборка оболочки PowerShell. Дополнительно почти для каждой ее версии присутствует русификация либо в виде локализованной версии, либо в виде отдельного многоязыкового пакета (Multilingual User Interface – MUI). Подробно обо всех возможных вариантах – в табл. 1.1.

Таблица 1.1. Дистрибутивы PowerShell

Операционная система	Платформа	Статья KB	Язык	Файл
Winodws XP	x86	926139	Англ.	WindowsXP-KB926139-x86-ENU.exe
			Рус.	WindowsXP-KB926139-x86-RUS.exe
		926141	MUI	WindowsXP-KB926141-v2-x86-ENU.exe
	x64	926140	Англ.	WindowsXP-KB926140-x64-ENU.exe
			Рус.	WindowsXP-KB926140-x64-RUS.exe
		926141	MUI	WindowsServer2003.WindowsXP-KB926141-x64-ENU.exe
WindowsVista	x86	928439	–	Windows6.0-KB928439-x86.msu
	x64			Windows6.0-KB928439-x64.msu
Windows 2003	x86	926139	Англ.	WindowsServer2003-KB926139-x86-ENU.exe
			Рус.	WindowsServer2003-KB926139-x86-RUS.exe
		926141	MUI	WindowsServer2003-KB926141-x86-ENU.exe

Таблица 1.1. Дистрибутивы PowerShell *(окончание)*

Операционная система	Платформа	Статья KB	Язык	Файл
	x64	926139	Англ.	WindowsServer2003-KB926139-x64-ENU.exe
			Рус.	WindowsServer2003-KB926139-x64-RUS.exe
		926141	MUI	WindowsServer2003.WindowsXP-KB926141-x64-ENU.exe
Windows 7	x86, x64	Интегрирован в операционную систему		
Windows 8	x86, x64	Интегрирован в операционную систему		

В зависимости от используемой версии операционной системы необходимо установить соответствующее обновление (см. табл. 1.1). Для операционной системы Windows 7 или Windows 8 – выбрать соответствующий компонент из стандартного комплекта операционной системы, если он не установлен.

Windows PowerShell, любой из выбранных версий, устанавливается в папку `%SystemRoot%\System32\WindowsPowerShell\v1.0`. Для запуска оболочки используется файл `powershell.exe`.

Среда разработки PowerShell

Начиная с Windows PowerShell CTP 2.0, в комплект дистрибутива включено приложение для разработки Windows PowerShell Integrated Script Environment (ISE), которое запускается с помощью файла `powershell_ise.exe`.

В новой, третьей версии Windows PowerShell оболочка претерпела значительные изменения. Цель, поставленная разработчиками, – сделать интерфейс максимально удобным для разработчиков, как, например, в Visual Studio, – успешно достигнута. Рассмотрим основные нововведения:

- панель **Commands** (рис. 1.1). Предназначена для оперативного получения справочной информации по командлетам и быстрому их созданию;
- поддержка технологии **IntelliSense**, позволяющая выводить подсказку в момент написания сценария в режиме реального времени. Именно эта система дает возможность в Visual Studio быстро и просто создавать сценарии;
- поддержка шаблонов сценариев **(snippet)**.

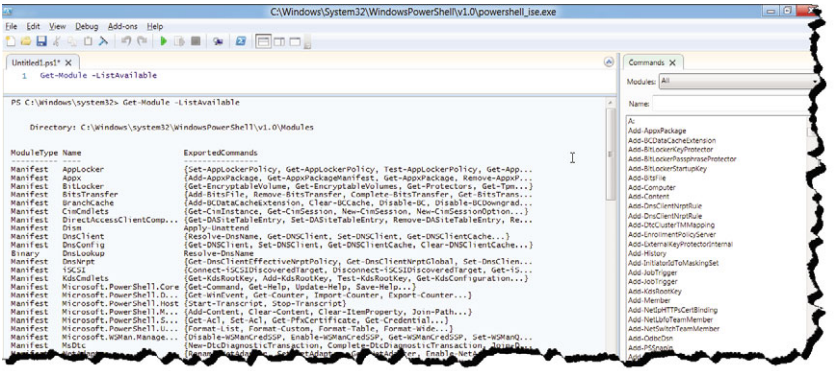

Рис. 1.1. Внешний вид Windows PowerShell ISE 3.0

Панель Commands

К трем панелям, имевшимся в предыдущей версии PowerShell, добавилась вертикальная панель **Commands**, расположившаяся с правой стороны относительно трех ранее существовавших в предыдущей версии горизонтальных панелей. В ней отображается список доступных командлетов, сгруппированных по модулям.

Непосредственно из самой панели с командлетами можно выполнять следующие действия:

- ***Run* (*запустить*)**. Для большинства командлетов реализована возможность определения параметров запуска прямо в этой панели. Обязательные параметры отмечены звездочкой;
- ***Insert* (*поместить*)**. Помещает выбранный командлет в окно консоли. Причины, по которым оно не помещается в панель, где ранее был установлен курсор, или непосредственно в сценарий, не понятны;
- ***Copy* (*скопировать*)**. Помещает командлет в буфер обмена.

Здесь же сосредоточены элементы управления отображения самой панели.

Быстрое создание сценариев

Основное назначение панели – упростить разработку новых сценариев. Рассмотрим простую задачу: необходимо вывести список всех файлов с расширением `EXE`, находящихся в папке `c:\windows`.

Очевидно, что для ее решения используется команда `dir` или соответствующий этому псевдониму командлет `Get-ChildItem`.

Если известно, к какому модулю относится командлет, то необходимо выбрать нужный модуль **(1)** (рис. 1.2) из списка, затем установить курсор непосредственно на командлет **(3)** или набрать его название в поле **Name (2)**. В процессе набора список будет автоматически сужаться. Если вы не помните название модуля, то с помощью значения фильтра **Modules** выведите все командлеты, выбрав значение **All** (1).

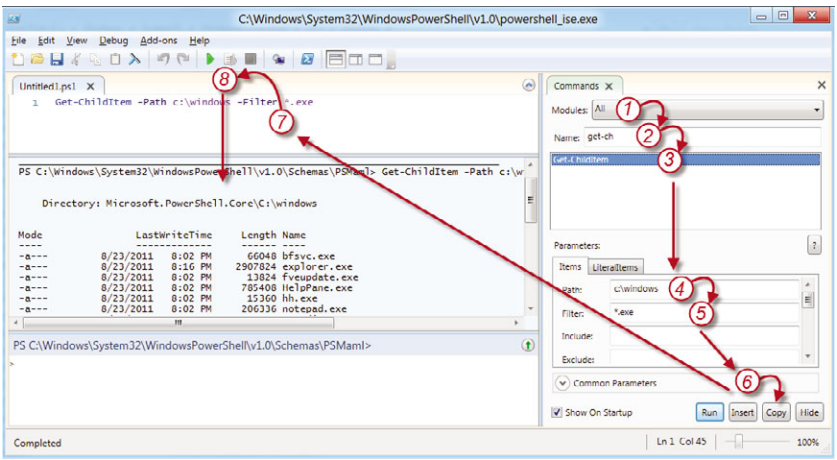

Рис. 1.2. Быстрое создание командлетов

После того как курсор установлен на нужном командлете, ниже в панели **Commands** отображается группа характеристик **Parameters**, определяющих его параметры запуска. В соответствии с условиями задачи определим значения для **Path (4)** и **Filter (5)**. Затем скопируем сформированный текст запуска командлета в буфер обмена, нажав на кнопку **Copy (6)**. Вставим скопированные данные в сценарий. Для этого установим курсор в окно сценария и нажмем комбинацию кнопок **Shift+Ins (7)**. Запустим сценарий, воспользовавшись соответствующей кнопкой **(8)** или нажав на кнопку **F5**. Результат будет отображен в соответствующем окне.

Замечание

Все стандартные аргументы запуска командлетов, такие как `Debug`, `Error Action` и др., сгруппированы в разделе **Common Parameters**.

Усовершенствованная справка по командлетам

Кроме традиционного вызова справки с помощью командлета `get-help`, в новой панели *Commands* появилась возможность оперативного доступа к справочной информации, разработан интерфейс для настройки выводимой информации.

Получить справку по интересующему вас командлету очень просто. Выбрав из списка нужный командлет, необходимо нажать на кнопку с вопросительным знаком (*?*) (рис. 1.3). Автоматически сгенерируется диалоговое окно, содержащее информацию, выводимую командлетом, – `get-help`. Кроме того, в нем реализованы механизм поиска данных в выведенной справке и возможность настройки вывода данных. При нажатии на кнопку *Settings* появится диалоговое окно, в котором можно определить список отображаемых разделов справки и настройки системы поиска. По умолчанию отображаются все секции справки.

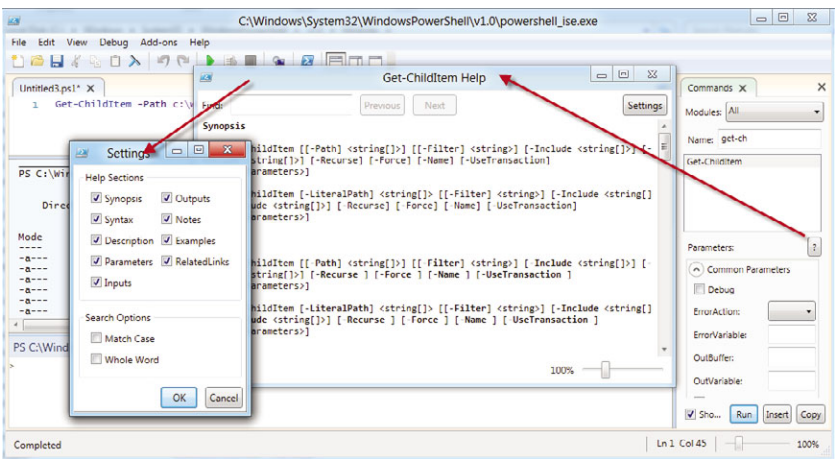

Рис. 1.3. Оперативное получение справочной информации по командлетам

Технология IntelliSense

Технология **IntelliSense** хорошо известна программистам по Microsoft Visual Studio. Сегодня она наконец-то появилась в Windows PowerShell. Если в предыдущих версиях для быстрого ввода имени

существующего командлета или его атрибута запуска было необходимо многократно нажимать кнопку **Tab**, перебирая предлагаемые варианты, то теперь автоматически выводится интеллектуальная подсказка (рис. 1.4). Если это по каким-то причинам не происходит, то достаточно нажать сочетание клавиш **Ctrl+Space**. На этом разработчики не остановились: рядом со списком доступных командлетов отображается еще окно, содержащее существующие варианты синтаксиса.

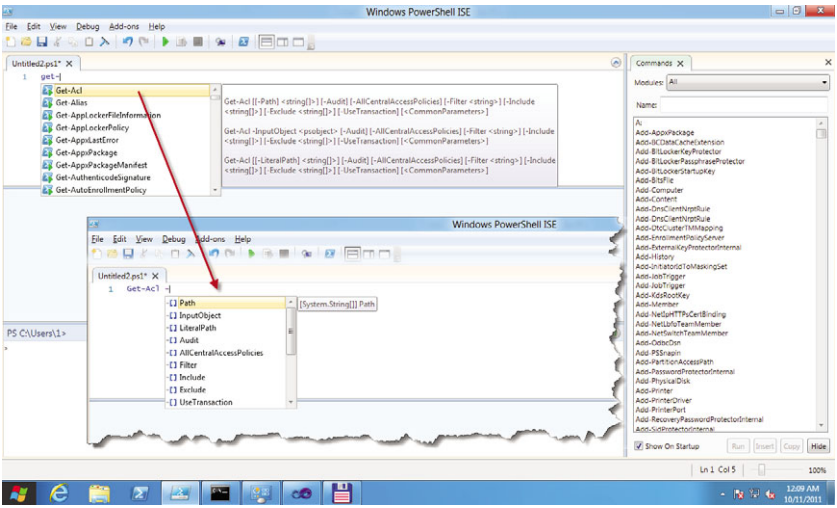

Рис. 1.4. Иллюстрация работы всплывающих подсказок

Аналогичная ситуация – с подсказкой названий атрибутов командлета. Как только после названия команды вводится дефис, сразу отображается список используемых в нем параметров (см. рис. 1.4).

Поддержка шаблонов сценариев

В новой версии реализована возможность создания своих шаблонов сценария, которые позволяют формировать конечный сценарий из уже готовых фрагментов. По умолчанию в PowerShell CTP 3 всего 12 шаблонов для условных операций, например `If`, `For`, `Switch`, и для пользовательских функций, имеющих различное количество аргументов.

Для вызова списка шаблонов достаточно в меню **Edit** выбрать **Start Snippets** (рис. 1.5) или воспользоваться клавиатурным сокращением **Ctrl+J**.

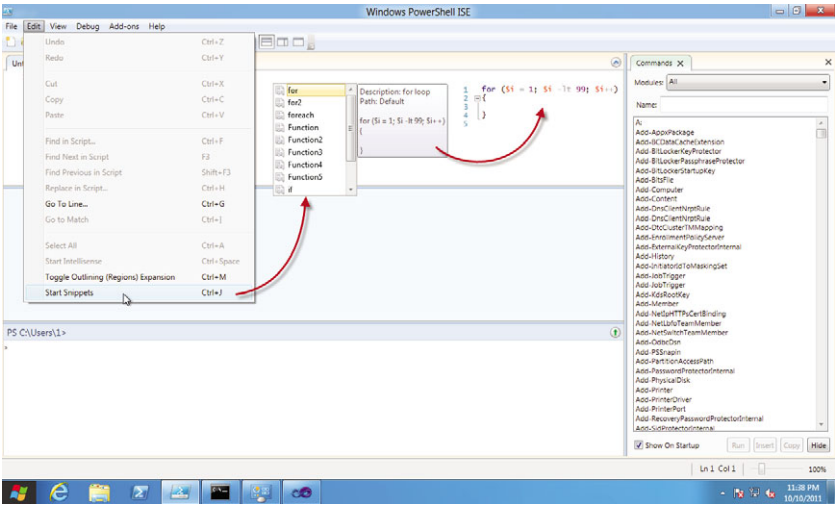

Рис. 1.5. Использование шаблонов сценариев

Первый запуск PowerShell

Для запуска PowerShell необходимо в меню **Пуск ⇨ Программы ⇨ Windows PowerShell 1.0** кликнуть по ярлыку **Windows PowerShell**. В результате выполнения команды появится диалоговое окно, очень похожее на оболочку CMD.

Запуск сценариев PowerShell

По умолчанию выполнение скриптов Windows PowerShell отключено. Установлен режим *Restricted*, позволяющий работать с консолью только в интерактивном варианте. Существуют четыре уровня безопасности запуска сценариев:

- ***AllSigned*** – все файлы, в том числе и на локальной машине, содержащие сценарии на языке PowerShell, должны иметь цифровую подпись. При запуске файла оболочка задаст вопрос о том, доверяете ли вы данному издателю и можно ли доверять в дальнейшем файлам с этой подписью;
- ***RemoteSigned*** – все сценарии, находящиеся в сети; должны быть подписаны. Файлы, запускаемые с локальной машины; могут не иметь цифровой подписи;
- ***Restricted*** (по умолчанию) – выполнение сценариев запрещено;

- ***Unrestricted*** – все сценарии, запускаемые с локального диска или из сети, могут не иметь цифровой подписи. Запуск файлов из сети сопровождается соответствующим предупреждением. Для его подавления необходимо в свойствах файла выбрать *Unblock* (разблокировать).

Уровень безопасности задается с помощью параметра реестра `ExecutionPolicy`. Значение параметра соответствует названию уровня безопасности (листинг 1.1).

Листинг 1.1. Настройка параметра безопасности PowerShell по умолчанию

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\Shell\Ids\
                                                  Microsoft.PowerShell]
"ExecutionPolicy"="Restricted"
```

Управление уровнем безопасности исполнения сценариев средствами PowerShell осуществляется с помощью двух командлетов – `Get-ExecutionPolicy` и `Set-ExecutionPolicy`. Командлет `Get-ExecutionPolicy` предназначен для просмотра установленного уровня безопасности (по умолчанию `Restricted`), а командлет `Set-ExecutionPolicy` – для изменения текущего уровня безопасности. В отличие от предыдущего командлета, при использовании данного командлета в качестве аргумента необходимо указать новый уровень безопасности. Недостаток данного метода заключается в том, что пользователь, выполняющий данные команлеты, должен обладать административными привилегиями.

Из всех существующих уровней безопасности поставленным условиям удовлетворяет один из двух:

- ***RemoteSigned***. Файл сценария должен иметь электронную подпись;
- ***Unrestricted***. Файлы могут быть не подписаны.

Использование первого способа позволяет повысить уровень безопасности выполнения различных сценариев в сети, по сравнению со вторым способом.

Документация по PowerShell

Пакет документации можно скачать с официального сайта Microsoft со страницы *Windows PowerShell 1.0 Documentation Pack* (http://

www.microsoft.com/downloads/details.aspx?familyid=B4720B00-9A66-430F-BD56-EC48BFCA154F&displaylang=en). В комплект входят четыре документа, краткое описание которых приведено в табл. 1.2.

Таблица 1.2. Содержание пакета Windows PowerShell 1.0 Documentation Pack

Название документа (англ.)	Название документа (в локализации для России)	Имя файла	Краткое описание
Windows PowerShell Getting Started Guide	Знакомство с Windows PowerShell	GettingStarted.rtf	Рассказывается о целях и задачах PowerShell, основах построения, о навигации и работе оболочки
Windows PowerShell Primer	Введение в Windows PowerShell	userguide.rtf	Рассказано о приемах программирования, о функциях и командах, доступных в PowerShell
Windows PowerShell Language Quick Reference	Краткий обзор языка Windows PowerShell	quadfold.rtf	Описаны основы синтаксиса языка (типы данных, комментарии, поддерживаемые циклы и др.)
Windows PowerShell V1.0 Release Notes	Примечания к выпуску Windows PowerShell v 1.0 (для .NET Framework 2.0 RTM)	releasenotes.rtf	Описаны различия версий PowerShell, PowerShell RC1, PowerShell RC2

Еще один важный документ, который обязательно пригодится программистам, знающим VBScript и желающим перейти на PowerShell, – *Translating VBScript to Windows PowerShell*. Этот документ входит в пакет *Windows PowerShell Week Script Samples* (https://www.microsoft.com/downloads/details.aspx?FamilyId=264CE487-1D36-4466-BD8B-23A7F1FA967E&displaylang=en). После установки пакета из файла `PowerShellWeekSamples.exe` в указанной для установки папки будут присутствовать четыре папки и текстовый файл с детальным описанием их содержания. Стоит отметить, что первые три папки содержат примеры на PowerShell, которые будут очень полезны. В последней папке – `VBStoPSGuide` – находится файл `vbscript_to_powershell.doc`, содержащий описание перехода от VBScript к PowerShell.

Глава 2
УПРАВЛЕНИЕ КАТАЛОГОМ ACTIVE DIRECTORY

В этой главе рассказано об основных объектах каталога Active Directory. Описаны основные действия, выполняемые с объектами, и характерные для них примеры. Вы узнаете, как с помощью скрипта определить имя текущего домена и как осуществляется взаимодействие сценария регистрации пользователей в сети с объектами каталога Active Directory.

Объекты Active Directory

Active Directory представляет собой иерархическую структуру, в которой находятся объекты различного типа. Обращение и управление к объектам осуществляются по определенным, заранее оговоренным правилам.

Поддерживаемые форматы имен

Обращение к объектам каталога Active Directory осуществляется в соответствии с одним из существующих стандартов, указанных в табл. 2.1.

Таблица 2.1. Форматы имен, используемые в Active Directory

Формат	RFC	Описание	Пример
UPN	RFC 822	Формат основного имени пользователя (User Principial Name, UPN). 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	Plvanov@Island.ru

Таблица 2.1. Форматы имен, используемые в Active Directory *(окончание)*

Формат	RFC	Описание	Пример
RDN	RFC 1779, RFC 2247	RDN-имена (Relative Distinguished Name) имеют более сложную структуру, по сравнению с URL-именами. Имена URL LDAP состоят из следующих частей: ❑ CN – расшифровывается как Common Name (общее имя); ❑ OU – означает организационную единицу (Organization Unit); ❑ DN – контроллер домена (Distinguished Name – отличительное имя). Часть имени «DC=» обеспечивает подключение к виртуальному объекту RootDSE, с помощью которого осуществляется подключение к домену	LDAP://ISLAND.RU/CN=Plvanov,OU=WorkSpace LDAP://CN=Plvanov,OU=WorkSpace,DC=Island,DC=ru
UNC	RFC 1123	UNC (Universal Naming Context) имеет вид `Server.Doamin/OU1/OU2/./Login`. В Active Directory данный формат является путем в иерархической структуре к объекту	Hoda.Island.ru/WorkSpace/Plvanov

Виды объектов

В Active Directory поддерживаются несколько типов объектов, каждый из которых имеет индивидуальный набор свойств. Для доступа к каталогу используется провайдер LDAP. В табл. 2.2 перечислены характерные объекты Active Directory. Для совместимости доменов, построенных на основе Windows NT и Windows 2k, Active Directory обеспечивает доступ к некоторым объектам с помощью провайдера WinNT.

Таблица 2.2. Типы объектов Active Directory

Тип объекта	Создается с Active Directory	Поддержка WnNT	Описание
RootDSE	Да	Нет	Виртуальный объект, с помощью которого обеспечивается доступ ко всем объектам каталога Active Directory
Computer	Да	Да	Учетная запись рабочей станции, зарегистрированной в домене
Contact	Нет	Нет	Контакт, использующийся в почте. Созданные контакты отображаются в почтовом клиенте, например в Microsoft Oulook
Group	Да	Да	Группе безопасности в Active Directory присваивается тип Group
InetOrgPerson	Нет	Нет	Учетная запись пользователя, используемая Active Directory (LDAP)
OU	Да	Нет	Объекту соответствует папка (OU, Organizational Unit) в Active Directory
SharedFolder	Нет	Да	Сетевая папка, опубликованная в Active Directory
User	Да	Да	Учетная запись пользователя, используется для совместимости с Windows NT
Printer	Нет	Да	Сетевой принтер, опубликованный в Active Directory

Идентификация объектов

Идентификация объектов в Active Directory осуществляется с помощью массива ObjectClass, который представляет собой массив строк. Однозначная идентификация объектов осуществляется по последнему элементу массива. В табл. 2.3 приведена информация, позволяющая однозначно идентифицировать объект: полный список значений массива ObjectClass; значение, идентифицирующее объект и префикс в каноническом имени объекта, используемом в Active Directory.

Таблица 2.3. Идентификация типа объекта для его создания программным способом

Объект	Тип объекта	Значение *objectClass*	Идентификатор объекта для сценария
Учетная запись компьютера	Computer	Top Person OrganizationalPerson User Computer	Computer
Контакт, используется в почтовых приложениях	Contact	Top Person OrganizationalPerson Contact	Contact
Группа безопасности	Group	Top Group	Group
Учетная запись пользователя, не совместимая с доменами Windows 2k	InetOrg-Person	Top Person OrganizationalPerson User InetOrgPerson	InetOrgPerson
Папка дерева каталогов Active Directory	OU	top organizationalUnit	organizationalUnit
Опубликованный в Active Directory сетевой принтер	Printer	Top Leaf ConnectionPoint PrintQueue	PrintQueue
Опубликованная в Active Directory сетевая папка	Shared Folder	Top Leaf ConnectionPoint Volume	Volume
Учетная запись пользователя, совместимая с доменами Windows NT	User	Top Person OrganizationalPerson User	user
Контейнер (папка), создаваемая в каталоге Active Directory по умолчанию	Container	Top Container	container

Данные, приведенные в таблице, будут активно использоваться для осуществления манипуляций с объектами различных типов.

Определение имени домена

Доступ к домену, так же как и его имя, определяется виртуальным объектом RootDSE, который является вершиной логического пространства имен и к тому же вершиной дерева. В этом виртуальном объекте публикуется информация о LDAP-сервере, в том числе сведения о поддерживаемой им версии LDAP, механизмах протокола SASL (Simple Authentication and Security Layer), элементах управления.

Объект RootDSE был определен в RFC 2251 как часть спецификации LDAP версии 3. Поэтому его имя отсутствует в пространстве имен Active Directory. Каждый контроллер домена самостоятельно поддерживает этот искусственный объект.

Для подключения к RootDSE используется бессерверное подключение. В таких случаях для определения имени текущего домена применяется локатор контроллеров домена. Доступ к RootDSE осуществляется анонимно.

На практике виртуальный каталог RootDSE используется для определения имени текущего домена, а также косвенным образом для определения сервера к тому или иному сайту.

Как известно, существуют три вида имени домена:

- ***RDN* (*Relative Distinguished Name*)** – относительное составное имя домена, например DC=ISLAND,DC=RU;
- ***FQDN* (*Fully Qualified Domain Name*)** – полное доменное имя или DNS-имя, в котором составные части разделены точкой, например ISLAND.RU;
- ***NetBIOS-имя*** – первая часть составного имени домена, например ISLAND.

На практике для управления объектами Active Directory с помощью провайдера LDAP используется имя домена, записанное в формате RDN.

Домен имеет древовидную структуру, в которой доступны несколько пространств имен. Каждое из них имеет свою точку входа:

- ***defaultNamingContext***. Описываемое этим параметром пространство имен используется для управления учетными записями пользователей, групп, контейнеров и других объектов в оснастке Active Directory Users and Computers;

- ***shemaNamingContext***. Данным параметром описывается местоположение схемы домена;
- ***configurationNamingContext***. Содержит RDN-путь к разделу, включающему путь к конфигурации леса текущего домена;
- ***rootDomainNamingContext***. Значением параметра является RDN-путь к корню домена (домен, который был создан первым в лесу).

Имя текущего домена является значением параметра `defaultNamingContext`. Поскольку PowerShell представляет собой нечто среднее между Visual Studio и VBScript, то рассмотрим управление объектом `RootDSE` с помощью четырех языков программирования (C#, VB.NET, VBScript и PowerShell), чтобы наглядно показать все преимущества PowerShell.

Удаленное подключение к каталогу Active Directory (провайдер LDAP) обеспечивается с помощью *Active Directory Services Interface* (ADSI). В VBScript для этого используется функция `GetObject()`, в качестве аргумента которой фигурирует путь к объекту. В PowerShell для решения идентичной задачи вместо функции `GetObject()` в квадратных скобках указывается ключевое слово `ADSI`, а в кавычках, следующих далее, – путь к объекту.

Определение RDN-имени текущего домена с помощью виртуального объекта `RootDSE` состоит из двух этапов. Первый этап – получение доступа к `RootDSE`, второй – чтение строкового значения параметра `defaultNamingContext`. На языке VBScript для этого используются функция `GetObject()`, аргументом которой является путь к виртуальному объекту: `LDAP://RootDSE`, – и чтение значения с помощью функции `GET`, аргументом которой, в свою очередь, является имя параметра (листинг 2.1).

Листинг 2.1. Определение RDN-имени домена (VBScript)

```
Set obj = GetObject("LDAP://rootDSE")
domain = obj.Get("defaultNamingContext")
MsgBox domain
```

Эту же операцию в DOT.NET можно осуществить двумя способами:

- с помощью COM-объекта. Так же как и в VBScript, осуществляется вызов функции `GetObject()` и с помощью свойства `GET` – чтение значения параметра `defaultNamingContext`;

- с помощью класса `EntryDirectory`, входящего в состав библиотеки `Directory Services .Net Framework` (листинг 2.2 на C# и листинг 2.3 на VB.NET).

Первый способ рассматриваться не будет, поскольку его скорость работы, по сравнению со вторым, оставляет желать лучшего. Для реализации второго способа необходимо к созданному в Visual Studio проекту подключить библиотеку `System.DirectoryServices` (рис. 2.1).

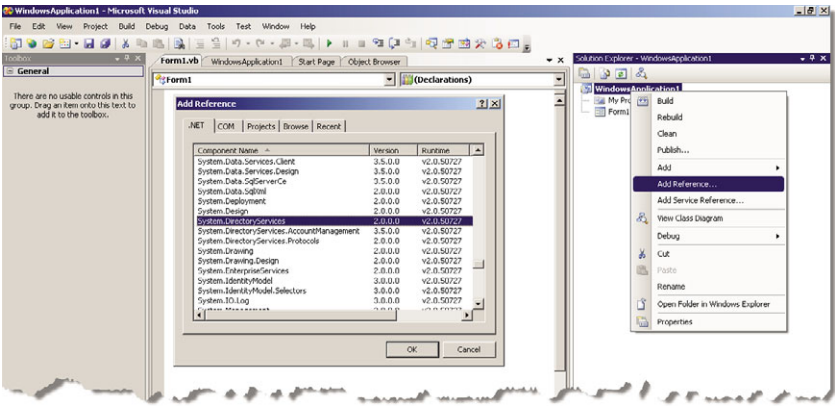

Рис. 2.1. Подключение к проекту библиотеки System.DirectoryServices

Листинг 2.3. Определение RDN-имени домена (C#)

```
Using System.DirectoryServices;
...
DirectoryEntry obj = New DirectoryEntry("LDAP://RootDSE");
String domain = obj.Properties("defaultNamingContext").Value;
MessageBox.Show (domain);
```

Листинг 2.3. Определение RDN-имени домена (VB.NET)

```
Imports System.DirectoryServices
...
Dim obj As New DirectoryEntry("LDAP://RootDSE")
Dim domain As String=obj.Properties("defaultNamingContext").Value
MsgBox (domain)
```

В отличие от VBScript, PowerShell не поддерживает COM-объекты в явном виде, зато он ориентирован на .NET Framework. Для по-

лучения имени домена все так же используется библиотека `System.DirectoryServices`, однако ее нет необходимости подключать, поскольку все имеющиеся библиотеки сразу доступны после запуска оболочки PowerShell. Листинг сценария очень похож на листинг DOT.NET (листинг 2.4). Обратите внимание: все имена переменных в PowerShell начинаются с символа доллар (`$`). Как видно, листинг по своему объему сопоставим с листингом на VBScript, а по скорости работы – с DOT.NET.

Листинг 2.4. Определение RDN-имени домена (PowerShell)

```
$obj=[ADSI]"LDAP://RootDSE"
$domain=$obj.defaultNamingContext
Write-Host $domain
```

Действия над объектами

Со всеми объектами каталога Active Directory доступны следующие действия: создание, копирование, переименование, перемещение, удаление, изменение свойств, поиск и чтение свойств объектов. Сценарий, запускаемый от имени пользователя, входящего в сеть, как правило, не имеет каких-либо аминистративных привилегий, поэтому для него актуальны только поиск объектов и чтение его свойств. Используя трехзвенную схему, сценарий может выполнять код от имени администратора. В этом случае все остальные действия с объектами каталога Active Directory могут быть реализованы. Рассмотрим весь перечень возможных операций.

Создание объекта

Создание объекта – стандартная процедура для всех объектов Active Directory. Для создания любого объекта необходимо определить несколько параметров:

- ***Тип объекта.*** Определяется значением последнего элемента массива **objectClass** (табл. 2.1);
- ***Местоположение объекта.*** Относительный составной путь к контейнеру, в котором создается объект;
- ***Имя объекта.*** В зависимости от типа объекта перед именем указывается префикс. Во время создания объекта присвоенное имя дублируется атрибутом `name`.

В листинге 2.5 приведен шаблон создания объекта в общем виде. Создание объекта осуществляется с помощью метода `Create()`, кото-

рый имеет два параметра. В качестве первого параметра указывается тип создаваемого объекта. Тип объекта определяется по последнему элементу массива `ObjectClass` (табл. 2.1). Второй параметр – составное имя объекта, например `cn=testGroup`. Большинство объектов имеют один и тот же префикс – `cn`.

Управление свойствами объекта осуществляется с помощью метода `Put()`, который имеет два параметра. Значением первого параметра является имя атрибута, второго – его значение. Далее осуществляется физическое создание объекта в каталоге Active Directory. Заметим, что параметры можно изменить только у существующего объекта.

Листинг 2.5. Создание произвольного объекта

```
# Определение имени домена
$domain = ([ADSI]"LDAP://RootDSE").defaultNamingContext
# Назначение пути к объекту
$path="LDAP://OU=…"
# Получение доступа к объекту-родителю
$obj=[ADSI]("$path, $domain")
# Создание объекта в памяти
$NewObj=$obj.Create(ТИП_ОБЪЕКТА, ИМЯ_ОБЪЕКТА)
# Назначение обязательных свойств объекта
$NewObj.Put(ИМЯ_АТРИБУТ, ЗНАЧЕНИЕ_АТРИБУТА)
…
# Запись данных в каталог Active Directory
$NewObj.SetInfo()
```

Копирование объекта

Копирование объектов позволяет быстро создать объект на основе существующего, уменьшив затраты на администрирование и количество ошибок, связанных с влиянием человеческого фактора.

Копирование объекта по шаблону является производным действием от создания объектов. После завершения процедуры создания объекта осуществляется копирование указанных свойств с существующего объекта. Атрибуты, значения которых необходимо перенести с существующего объекта, указываются в массиве.

При создании объекта по шаблону с помощью мастера подразумевается, что новый объект будет создан в том же контейнере, что и шаблонный. Используя сценарий, путь может меняться как угодно. В листинге 2.6 приведен пример создания нового объекта на основе