

# **ТЕМА 4. МЕТОДЫ АДМИНИСТРИРОВАНИЯ БОЛЬШИХ КОМПЬЮТЕРНЫХ СИСТЕМ**

В данной теме рассматриваются следующие вопросы:

- особенности администрирования больших компьютерных систем;
- администрирование компьютерных систем на базе операционных систем Windows;
- инфраструктура групповых политик;
- настройка параметров безопасности с помощью групповых политик.
- управление рабочими столами;
- административные шаблоны;
- централизованное хранилище групповых политик;
- определение результирующей политики;
- использование PowerShell для администрирования Windows.

Лекции – 4 часа, лабораторные занятия – 4 часа, самостоятельная работа – 12 часов.

Минимальный набор знаний:

Структура объекта групповой политики

Особенность административных шаблонов

Порядок применения объектов групповой политики в доменной среде

Базовые элементы программирования в Powershell (переменные, массивы и хэш-таблицы, условный оператор, циклы)

Виды команд в Powershell

Конвейер в Powershell, передача параметров

## 4.1. Администрирование корпоративных информационных систем

### 4.1.1. Особенности администрирования больших компьютерных систем

**Администрирование** — процедуры управления, регламентирующие некоторые процессы или их часть. Как правило, оно фиксирует и руководит процессами и ситуациями, нуждающимися в ограничениях или целевом управлении.

Построение компьютерных сетей вызвало необходимость управления (администрирования) ими и созданными на их основе компьютерными вычислительными и информационными системами. В результате появилось системное администрирование.

Основной целью системного администрирования является приведение сети в соответствие с целями и задачами, для которых она предназначена. Достигается эта цель путём управления сетью, позволяющего минимизировать затраты времени и ресурсов, направляемых на управление системой, и в тоже время максимизировать доступность, производительность и продуктивность системы.

Увеличение размеров сети приводит как к увеличению количества устройств, обеспечивающих её функционирование, так и к увеличению их сложности и числа пользователей сети. В различных сетях эта проблема может осложняться гетерогенностью программного и аппаратного обеспечения. При этом такие информационные системы и среды являются многопользовательскими.

***Многопользовательская среда*** – компьютерная сетевая программа, создающая среду для реализации различных типов поведения различных пользователей.

При организации сети компании распределённой средствами Интернета по большой площади, нужно заботиться о надёжной маршрутизации, своевременном обмене информацией и о защите этой информации от несанкционированного доступа. Кроме того, следует организовать информационное обслуживание, единое для всех частей такой распределённой структуры. При построении корпоративных информационных сетей, как правило, используют две базовые архитектуры: Клиент-сервер и Интернет/Инtranет.

В этом курсе мы не будем рассматривать особенности использования сетевого оборудования при построении корпоративных информационных систем, а ограничимся лишь компьютерной частью этих систем.

Одной из самых распространённых архитектур построения корпоративных информационных систем является архитектура «клиент-сервер». В реализованной по данной архитектуре информационной сети клиенту предоставлен широкий спектр приложений и инструментов разработки, ориентированных на максимальное использование вычислительных возможностей клиентских рабочих мест. При этом ресурсы сервера в основном используются для хранения и обмена документами, а также с целью выхода во внешнюю среду. Для программных систем, имеющих разделение на клиентскую и серверную части, применение данной архитектуры позволяет лучше защитить серверную часть приложений, предоставляя возможность приложениям непосредственно

адресоваться к другим серверным приложениям, или маршрутизировать запросы к ним. При этом частые обращения клиента к серверу снижают производительность работы сети. Кроме того, приходится решать вопросы безопасной работы в сети, так как приложения и данные распределены между различными клиентами. Распределённый характер построения системы обуславливает сложность её настройки и сопровождения. Чем сложнее структура сети, построенной по архитектуре «клиент-сервер», тем выше вероятность отказа любого из её компонентов.

В основе архитектуры Интернет/Интранет корпоративных информационных систем лежит принцип «открытой архитектуры», определяющий независимость реализации корпоративной системы от конкретного производителя.

**Открытая архитектура (Open architecture)** – архитектура компьютера или периферийного устройства, имеющая опубликованные спецификации, позволяющие другим производителям разрабатывать дополнительные устройства к системам с такой архитектурой.

При этом важное место в любой многопользовательской среде отводится сетевым операционным системам.

Для работы в локальной сети на серверы обычно устанавливают серверную ОС, например, Windows Server 2012 или Windows Server 2016, а на рабочие узлы сети — клиентскую ОС Windows 7, Windows 8.1 или Windows 10.

#### **4.1.2. Методы и инструменты администрирования операционных систем Windows**

Как правило, большинство системных инструментов требует наличия у запускающего их пользователя административных привилегий. Чтобы иметь возможность выполнения операций по управлению системой, администратор должен зарегистрироваться в системе под учётной записью, обладающей соответствующими правами. Однако правила безопасности требуют от администратора постоянно не пользоваться в системе подобными учётными записями. Первый способ решения проблемы — использование команды RunAs (запуск от имени). Второй способ — использование User Access Control, при котором система автоматически запрашивает у пользователя разрешение при попытке выполнить задачу, требующую административных привилегий.

Использование инструментов с графическим интерфейсом требует большого объема ручного труда администратора и практически нереализуемо при большом количестве обслуживаемых компьютерных систем.

Так как большинство приложений Windows и сама операционная система хранит большинство своих настроек в реестре Windows, то очевидно, что намного проще было бы вносить эти параметры непосредственно в реестр каждого управляемого компьютера. Именно для реализации этого процесса были изобретены групповые политики.

Другой способ массового администрирования — это создание текстовых командных файлов, еще называемых скриптами, которые можно принудительно запустить на выполнение на множестве компьютеров. При администрировании Windows удобнее всего использовать встроенное средство автоматизации

PowerShell, состоящее из оболочки с интерфейсом командной строки, сопутствующего языка сценариев и исполняющей системы.

В последнее время становится популярным декларативная конфигурация системы. В системах Microsoft этот подход реализован в PowerShell Desired State Configuration (можно перевести «настройка требуемого состояния»). Используя PowerShell DSC, вы описываете как хотите, чтобы ваша система выглядела в конечном итоге, и далее происходит ее автоматическая настройка в соответствии с заданными требованиями.

В этой теме мы рассмотрим лишь групповые политики и основы работы с PowerShell.

## 4.2. Применение групповых политик

Будем изучать групповые политики последовательно, от простого к сложному, от мелкого к крупному. В качестве вступления рассмотрим реестр Windows, затем локальные объекты групповой политики, доменные объекты групповой политики и некоторые примеры их применения.

### 4.2.1. Реестр Windows

**Реестр Windows (Registry)** — это иерархическая централизованная база данных, используемая в современных операционных системах семейства Windows для хранения сведений, необходимых для настройки операционной системы для работы с пользователями, программными продуктами и устройствами.

В реестре хранятся данные, которые необходимы для правильного функционирования Windows. К ним относятся профили всех пользователей, сведения об установленном программном обеспечении и типах документов, которые могут быть созданы каждой программой, информация о свойствах папок и значках приложений, а также установленном оборудовании и используемых портах.

Системный реестр заменяет собой большинство текстовых INI-файлов, которые использовались в Windows 3.x, а также файлы конфигурации MS-DOS, такие как Autoexec.bat и Config.sys. Версии реестра для разных версий операционных систем семейства Windows имеют определенные различия.

**Куст реестра (hive)** — это группа разделов, подразделов и параметров реестра с набором вспомогательных файлов, содержащих резервные копии этих данных. Вспомогательные файлы для всех кустов за исключением HKEY\_CURRENT\_USER хранятся в папке %SystemRoot%\System32\Config. Вспомогательные файлы для куста HKEY\_CURRENT\_USER хранятся в папке %SystemRoot%\Profiles\Имя\_пользователя.

**HKEY\_CURRENT\_USER.** Данный раздел является корневым для данных конфигурации пользователя, вошедшего в систему в настоящий момент. Здесь хранятся папки пользователя, цвета экрана и параметры панели управления. Эти сведения сопоставлены с профилем пользователя. Вместо полного имени раздела иногда используется аббревиатура HKCU.

**HKEY\_USERS.** Данный раздел содержит все активные загруженные профили пользователей компьютера. Раздел HKEY\_CURRENT\_USER является подразделом раздела HKEY\_USERS. Вместо полного имени раздела иногда используется аббревиатура HKU.

**HKEY\_LOCAL\_MACHINE.** Раздел содержит параметры конфигурации, относящиеся к данному компьютеру (для всех пользователей). Вместо полного имени раздела иногда используется аббревиатура HKLM.

**HKEY\_CLASSES\_ROOT.** Является подразделом HKEY\_LOCAL\_MACHINE\Software. Хранящиеся здесь сведения обеспечивают выполнение необходимой программы при открытии файла с использованием проводника. Вместо полного имени раздела иногда используется аббревиатура HKCR.

**HKEY\_CURRENT\_CONFIG.** Данный раздел содержит сведения о профиле оборудования, используемом локальным компьютером при запуске системы.

Внутри разделов находится древовидная структура ключей реестра. Каждый из них во многом напоминает папку или файл в файловой системе. В каждом ключе могут содержаться как данные, так и другие ключи. Если ключ содержит данные, то они представлены последовательностью значений. Каждое значение имеет ассоциированное с ним имя, тип данных и собственно данные. Вдобавок ключ имеет безымянное значение по умолчанию.

Записи в системном реестре могут иметь формат одного из четырех типов данных:

**REG\_SZ** – приблизительно соответствует экземпляру строки в .NET, но это сходство не точное, поскольку типы данных реестра не являются типами данных .NET;

**REG\_MULTI\_SZ** – приблизительно соответствует массиву строк в .NET;

**REG\_DWORD** – приблизительно соответствует типу uint;

**REG\_BINARY** – массив байтов.

Приложение, предусматривающее сохранение каких-то данных в реестре, будет делать это за счет создания ряда ключей, причем обычно внутри ключа HKLM\Software\<Название компании>. В этих ключах не обязательно должны содержаться какие-либо данные. Порой сам факт существования ключа позволяет приложению получать те данные, которые ему необходимы.

Для работы с реестром в состав операционной системы Windows входит утилита regedit.exe. Также доступ к ключам реестра можно получить программно с помощью классов Registry и RegistryKey платформы .NET.

На данном этапе нам достаточно увидеть, что в реестре есть один раздел, в котором хранятся параметры компьютера, которые действуют независимо от того, какой пользователь зарегистрировался в системе. Эти параметры хранятся в разделе HKEY\_LOCAL\_MACHINE. Также есть раздел HKEY\_CURRENT\_USER, в котором хранятся параметры, специфичные для каждого пользователя, например, размер окна Microsoft Word. Физически этот раздел хранится в профиле пользователя.

Соответственно, при изучении групповых политик мы увидим, что они также состоят из двух частей: компьютерной и пользовательской, что можно увидеть на рис. 4.1.

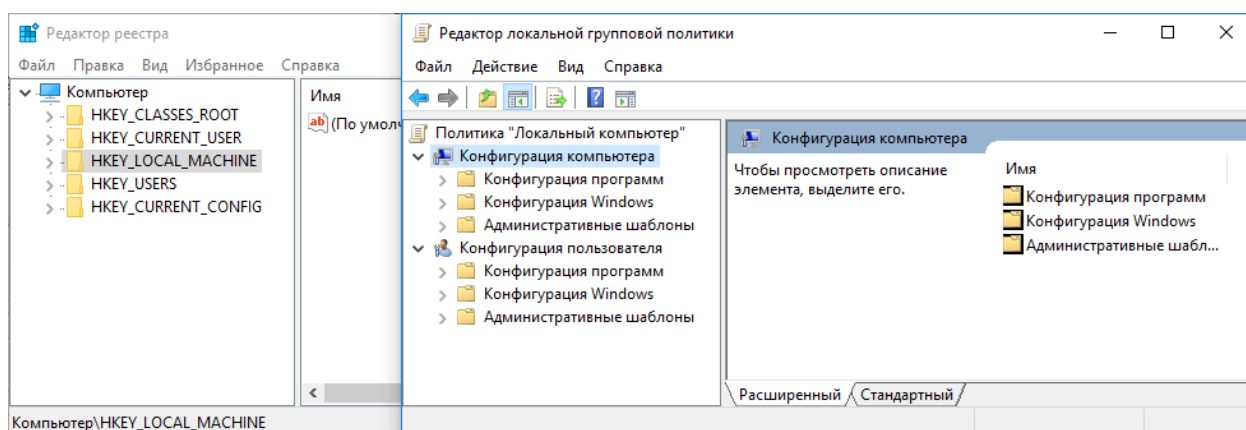


Рис.4.1. Сопоставление разделов реестра и разделов групповой политики

## 4.2.2. Локальные объекты групповой политики

Управление и конфигурирование операционных систем Windows может быть упрощено и стандартизовано за счет использования групповых политик. Групповые политики предназначены для централизации настройки и управления операционными системами семейства Windows, а также пользователями, входящими в системы. Управление групповыми политиками разделено на два узла политик — Computer Configuration (Конфигурация компьютера) и User Configuration (Конфигурация пользователя). Параметры, содержащиеся в узле Computer Configuration, можно применять для настройки системного реестра и прав доступа к файловой системе, для определения пользовательских политик паролей, ограничения использования съемных носителей, изменения конфигурации сети и параметров брандмауэра, управления системными службами, определения и управления профилями электропитания и многого другого. Узел User Configuration содержит параметры политик, которые можно применять для управления параметрами среды рабочего стола: автоматическое включение стандартной заставки и определение продолжительности блокировки, выполнение входных сценариев, добавление ярлыков на рабочий стол, перенаправление папок пользователей в общий сетевой ресурс и настройка синхронизации папок, ограничение функций среды рабочего стола и т. д.

Управление системами Windows может осуществляться индивидуально с помощью локальных групповых политик, а если системы являются членами доменов Active Directory, то ими можно управлять посредством групповых политик доменов. Локальные групповые политики и групповые политики доменов имеют похожие функции, однако групповые политики доменов обеспечивают дополнительную функциональность, т.к. многие параметры, включенные в шаблоны политик, применяются только к членам доменов Active Directory. На компьютерах, не включенных в домен, можно использовать только локальные групповые политики.

Каждая система Windows будет содержать локальную политику компьютера по умолчанию. Локальная политика компьютера представлена в виде объекта локальной групповой политики (Local Group Policy Object — LGPO). Данная политика содержит отдельные узлы Computer Configuration (Конфигурация компьютера) и User Configuration (Конфигурация пользователя). Исходя из названия, локальная политика компьютера применяет заданные параметры только к отдельной локальной компьютерной системе, а также к пользователям, которые входят в нее. В новой системе локальная политика компьютера не содержит ничего, кроме параметров по умолчанию, определенных в узле политики Computer Configuration\Windows Settings\Security Settings (Конфигурация компьютера\Параметры Windows\Параметры безопасности). Узел политики Security Settings тоже является локальной политикой безопасности.

Операционные системы Windows теперь поддерживают множество локальных групповых политик для пользовательских учетных записей. Кроме политики по умолчанию, можно еще создать пользовательские политики для членов группы Администраторы и для пользователей, не являющихся членами группы Администраторы, а также для каждого локального пользователя компьютера.

По умолчанию пользователи, входящие в систему с современной версией Windows (начиная с Windows Vista), подчиняются локальной политике компьютера, затем политике Administrators (Администраторы) или Non-Administrators (Рядовые пользователи), после чего любой локальной политике, определенной для пользователей. Вот пример применения множества политик: можно создать локальную политику компьютера, которая запрещает всем пользователям чтение и запись на съемные носители, и локальную политику пользователя группы Administrators, которая позволяет производить чтение и запись информации на съемные запоминающие устройства. Поскольку локальная политика пользователя группы Administrators применяется после локальной политики компьютера, только администраторы могут записывать информацию на съемное запоминающее устройство.

Чтобы открыть локальную политику по умолчанию в редакторе, можно открыть консоль gredit.msc. Для редактирования пользовательских локальных политик, нужно создать новую консоль MMC и добавить в нее оснастку редактора групповых политик (Group Policy Management Editor) с необходимыми параметрами.

### **4.2.3. Групповые политики домена**

Групповые политики домена очень похожи на локальные групповые политики, однако они содержат много дополнительных параметров. Созданные в домене групповые политики могут быть назначены на разных уровнях: ко всему домену, организационному подразделению или сайту.

*Примечание. Контейнеры Computers и User не являются организационными подразделениями и к ним нельзя применять групповые политики.*

Объекты групповых политик (Group Policy Object — GPO) представляют собой определенный набор доступных значений, которые могут быть применены к объектам компьютеров и/или пользователей Active Directory. Параметры, доступные в каком-либо GPO, создаются с помощью сочетания файлов административных шаблонов, включенных в данный GPO или упомянутых в нем. Если управление компьютером или пользователем нужно изменить, в этот объект GPO можно импортировать дополнительные административные шаблоны, расширяющие его функциональность.

Объекты групповых политик хранятся и в файловой системе, и в базе данных Active Directory. Каждый домен в лесу Active Directory хранит полную копию всех GPO этого домена.

Внутри Active Directory связи и сведения о версиях GPO хранятся в разделе базы данных, хранящем контекст именования доменов, в контейнере System\Policies. Параметры GPO хранятся в файловой системе всех контроллеров домена, в папке SYSVOL. Эта папка совместно используется всеми контроллерами домена. У каждого GPO домена имеется соответствующая ему папка, которая находится в подпапке SYSVOL\<имя\_домена>\Policies.

В качестве имени папки GPO берется глобально уникальный идентификатор (globally unique identifier — GUID), присвоенный этому GPO при его создании. Этот GUID выводится при просмотре свойств GPO домена в консоли управления групповыми политиками. В папке GPO находится обычный набор подпапок и



файлов: папка User, папка Machine (иногда папки ADM, Preferences, Scripts и т.д.) и файл gpt.ini. Каждая подпапка в иерархии папок GPO содержит файлы и папки, связанные с конкретным разделом политики или предпочтения.

### **Порядок применения групповых политик**

Если система Windows содержит несколько локальных политик или является членом домена Active Directory, то при включении компьютера или входе пользователя в систему будет обрабатываться несколько политик. Каждая политика, которая применяется к определенному компьютеру или пользователю, обрабатывается последовательно, поэтому очень важно понимать порядок обработки политик. В тех случаях, когда несколько политик имеют одинаковые параметры, но разные значения, значение параметра будет совпадать со значением в последней обработанной политике.

#### **Обработка политик для компьютеров**

Параметры политики применяются к компьютерам во время загрузки компьютера, его выключения, а также в периоды фоновой обработки. Обработка политик применительно к объектам компьютера производится в следующем порядке.

1. Локальная политика компьютера.
2. Политики доменов, связанные с сайтом Active Directory.
3. Политики доменов, связанные с доменом Active Directory.
4. Политики доменов, связанные с иерархией организационной единицы, в которой находится учетная запись компьютера.

#### **Обработка политик для пользователей**

Параметры политик применяются к пользователям во время входа пользователя в систему и в периоды фоновой обработки. Обработка политик для доменов и локальных пользователей производится в описанном ниже порядке.

1. Локальная политика компьютера.
2. Локальная политика Non-Administrators (Рядовые пользователи) или локальная политика Administrators (Администраторы), если такие политики существуют.
3. Локальная политика, определенная для данного пользователя; она применяется только в том случае, если пользователь входит с локальной учетной записью, и для этого пользователя существует политика.
4. Политики доменов, связанные с сайтом Active Directory.
5. Политики доменов, связанные с доменом Active Directory.
6. Политики доменов, связанные с иерархией организационной единицы, в которой находится учетная запись пользователя.

#### **Порядок обработки групповых политик**

Если с одним сайтом Active Directory, доменом или организационной единицей связывается множество политик, то каждая политика будет применяться последовательно.

Порядок применения политик или их обработка основывается на порядке присоединения политики. Политика с номером 1, связанным с именем политики,

является последней политикой, примененной в контейнере, поэтому она обрабатывается первой.

### **Циклическая обработка (loopback processing)**

Когда пользователь обрабатывает групповые политики, то политики, применяемые к пользователю, основываются на местонахождении объекта пользователя в иерархии Active Directory. Это касается и применения политик домена к компьютерам. Однако бывают ситуации, когда администраторам или организациям необходимо убедиться в том, что все пользователи подчиняются одной политике во время входа в систему на определенном компьютере или сервере. Например, если на компьютере, который используется для обучения, или на хосте сеансов удаленных рабочих столов (Remote Desktop Session Host) пользовательская среда рабочего стола должна быть одинаковой для всех пользователей, это можно сделать посредством циклической обработки.

Имеются два различных режима циклической обработки: замена и слияние. Режим слияния применяет пользовательские политики, которые обычно применяются к пользовательским учетным записям, а также пользовательские политики к контейнеру, содержащему учетную запись компьютера, в которую входит пользователь. Режим замены обрабатывает только пользовательские политики, применяемые к компьютеру, в который входит пользователь.

#### **4.2.4. Некоторые параметры групповой политики**

По умолчанию групповые политики в Windows Server 2012 содержат примерно 1860 параметров в узле Computer Configuration (Конфигурация компьютера) и еще 1560 параметров в узле User Configuration (Конфигурация пользователя). Еще больше установок в узлах Windows Settings (Параметры Windows) и узле Preferences (Предпочтения), которые существенно увеличивают объем параметров. Это, естественно, приводит к тому, что указание каждого параметра оказывается очень неудобным и длительным процессом.

Здесь мы упомянем только некоторые типы доступных параметров, которые считаются наиболее часто используемыми и полезными для управления средами Windows.

Многие из параметров политик, содержащихся в узлах политик Computer Configuration и User Configuration, применяются только к определенным службам ролей Windows Server 2012, таким как Encrypting File System (Шифрующая файловая система, EFS), Remote Desktop Services (Служба удаленных рабочих столов, RDS), Network Access Protection (Защита сетевого доступа, NAP) и Distributed File System (Распределенная файловая система, DFS). Для этих определенных служб, как и для параметров групповой политики, очень важно, чтобы администратор понимал, к чему может привести настройка любого из этих параметров. Прежде чем создавать, изменять или связывать групповые политики предприятия, нужно проверить политику в изолированной среде и подготовить и проверить в действии план отката.

#### **Узел политики Computer Configuration**

Узел Computer Configuration (Конфигурация компьютера) групповой политики содержит параметры, предназначенные для настройки и управления системой

Windows (рис. 4.2). Многие из параметров этого узла присутствуют и в узле User Configuration (Конфигурация пользователя), однако они дают разные результаты. В одних случаях параметры политики компьютера будут использоваться всегда, даже при настройке пользовательских параметров.

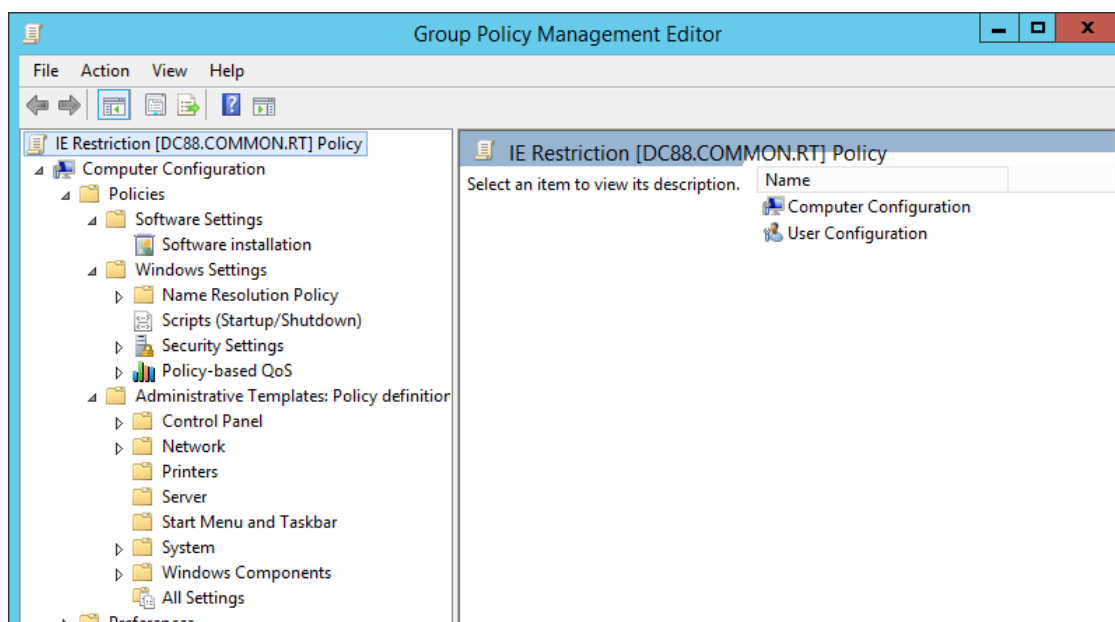


Рис. 4.2. Узел Computer Configuration (Конфигурация компьютера)

В других случаях будет использоваться последняя применённая политика. Например, в локальной групповой политике, внутри каждого узла в разделе Administrative Templates\System\Scripts (Административные шаблоны\Система\Сценарии), имеется параметр Run Logon Scripts Synchronously (Синхронное выполнение сценариев входа в систему), и, если этот параметр будет сконфигурирован в разделе Computer Configuration, он будет реализован независимо от того, как этот параметр будет сконфигурирован в узле политики User Configuration.

В корне узла Computer Configuration присутствуют три узла политики: Software Settings (Конфигурация программ), Windows Settings (Параметры Windows) и Administrative Templates (Административные шаблоны). В групповых политиках домена эти три узла находятся в узле Computer Configuration\Policies (Конфигурация компьютера\Политики).

### **Узел Computer Configuration\Software Settings**

Узел Software Settings (Конфигурация программ) используется для добавления пакетов программ в компьютеры, которые обрабатывают определенную политику. В этот узел можно добавить заранее упакованные или специальные пакеты программ Windows Installer MSI и применять их для автоматической установки программ на компьютере во время следующего цикла перезагрузки. Такой пакет называется назначенным пакетом программ.

### **Узел Computer Configuration\Windows Settings**

Узел Windows Settings (Параметры Windows) позволяет администраторам управлять общей безопасностью и конфигурацией системы Windows. Параметры, содержащиеся в этом узле, можно использовать для того, чтобы определить, как локальные пользователи и пользователи домена могут взаимодействовать и

управлять системой, и как система будет поддерживать связь по сети. Этот узел содержит пять дочерних узлов.

- **Name Resolution Policy (Политика преобразования имен).** Этот узел позволяет администраторам групповой политики создавать правила для построения содержимого таблицы политики преобразования имен (Name Resolution Policy Table) для поддержки реализаций DNSSEC и централизованно конфигурировать параметры Windows Server 2012 DirectAccess DNS.

- **Scripts (Startup/Shutdown) (Сценарии (запуск/завершение)).** Этот узел позволяет администраторам добавлять сценарии запуска или завершения в объекты компьютера.

- **Deployed Printers (Развернутые принтеры).** Этот узел позволяет администраторам автоматически устанавливать и удалять принтеры в системах Windows. При использовании редактора Group Policy Object Editor (Редактор объектов групповых политик) в системах Windows Server 2008 и Windows Server 2012 этот узел может не отображаться, если не будет установлена консоль Print Management (Управление печатью).

- **Security Settings (Параметры безопасности).** Этот узел является копией локальной политики безопасности, хотя он не синхронизируется и не извлекает информацию из локальной политики безопасности. Параметры, содержащиеся в этом узле, могут использоваться для определения политик паролей, политик аудита, ограничений на использование программного обеспечения, полномочий на использование файлов и на работу с системным реестром, и многого другого.

- **Policy-base QoS (Качество обслуживания на основе политики).** Этот узел можно сконфигурировать для управления, ограничения и установки приоритетов в исходящем сетевом трафике между исходной системой Windows и хостом назначения. При этом учитывается приложение, IP-адрес источника или пункта назначения, протоколы и порты источника и пункта назначения.

### **Узел Security Settings**

Узел Security Settings (Параметры безопасности) позволяет администратору безопасности задавать уровни безопасности для объектов домена или локальной групповой политики. Это можно сделать вручную или путем импортирования существующего шаблона безопасности.

Узел Security Settings объекта групповой политики можно применять для настройки разных параметров, связанных с безопасностью, включая разрешения на использование файловой системы NTFS, и многих других параметров, содержащихся в перечисленных ниже узлах.

- **Account Policies (Политики учетных записей).** Эти параметры безопасности компьютера управляют политикой паролей, политикой блокировки учетной записи, а также политикой Kerberos в доменах Active Directory от Windows 2000 Server до Windows Server 2012.

- **Local Policies (Локальные политики).** Эти параметры безопасности управляют политиками аудита, назначением полномочий пользователям, а также параметрами безопасности, включая параметры User Account Control (Управление учетной записью пользователя) для систем, к которым применяется политика.

- **Event Log (Журнал событий).** Управляет параметрами безопасности и размером журналов событий для журналов приложений, безопасности и системного журнала событий.

- **Restricted Groups (Ограниченные группы).** Эти параметры позволяют администратору управлять членством в локальной группе или в домене. Эти параметры можно использовать для добавления членов в существующую группу, не удаляя существующих членов, или для реализации и замены членства в группе на основе конфигурации политики.

- **System Services (Системные службы).** Эти параметры можно использовать для управления режимом запуска службы и для определения полномочий на управление конфигурацией службы или ее состоянием. Конфигурирование этих параметров не приводит к запуску или останову какой-либо службы, но после применения групповой политики это состояние изменится.

- **Registry (Реестр).** Этот параметр используется для конфигурирования полномочий на работу с определенными ключами системного реестра и, при желании, всеми подключами и значениями. Этот параметр полезен при поддержке унаследованных приложений, требующих доступа к специфическому ключу реестра, доступ к которому обычно невозможен или запрещен для стандартных пользовательских учетных записей.

- **File System (Файловая система).** Этот параметр используется для конфигурирования прав NTFS в отношении определенных папок на дисках с файловой системой NTFS. Также с его помощью можно разрешить аудит и конфигурирование владением папки и распространением этих параметров на вложенные папки и файлы.

- **Wired Network (IEEE 802.3) Policies (Политики проводной сети (IEEE 802.3)).** Этот узел политики можно использовать для конфигурирования дополнительного уровня безопасности в адаптерах проводных сетей, чтобы разрешить или затребовать аутентификацию и шифрование сертификата смарт-карты или компьютера.

- **Windows Firewall with Advanced Security (Брандмауэр Windows с расширенными параметрами безопасности).** Этот узел политики позволяет администраторам производить настройку брандмауэра (межсетевого экрана) в Windows Vista, Windows Server 2008 и более поздних версиях ОС. Выбранные параметры могут задавать определенные исходящие и входящие правила, а также определять конфигурацию брандмауэра на основании профиля брандмауэра. Конфигурация может заменить локальные правила брандмауэра или групповую политику и может объединять локальные правила.

- **Network List Manager Policies (Политики диспетчера списка сетей).** Брандмауэр Windows в Windows Vista, Windows Server 2008 и более поздних версиях ОС использует профили брандмауэра на основе сети. Этот узел параметров можно применять для определения полномочий, которыми будут обладать конечные пользователи, в зависимости от того, как идентифицируется и классифицируется новая сеть — публичная или частная, чтобы разрешить применение подходящего профиля брандмауэра.

- **Wireless Network (IEEE 802.11) Policies (Политики беспроводной сети (IEEE 802.11))**. Эти политики позволяют настраивать параметры широкого круга устройств, обеспечивающих доступ к сети по беспроводным технологиям, включая предварительное определение предпочтительной беспроводной сети, в том числе идентификатор набора служб (service set identifier — SSID) и тип безопасности для сети. Этот узел включает политики, совместимые с операционными системами Windows Vista и Windows XP.

- **Public Key Policies (Политики открытого ключа)**. Эти параметры используются для того, чтобы компьютер автоматически выдавал запрос центру сертификации на получение сертификата и устанавливал выданный сертификат. Также политики открытого ключа создаются и используются при распределении списка доверенных сертификатов. Политики открытого ключа могут устанавливать обычные доверенные корневые центры сертификации. Этот узел политики используется также параметрами Encrypting File System (Шифрующая файловая система).

- **Software Restriction Policies (Политики ограниченного использования программ)**. Эти политики позволяют администратору управлять приложениями, которым будет разрешено работать в системе Windows, основываясь на свойствах файла, включая его имя. Кроме того, эти политики можно создавать на основе сертификатов или определенной зоны сети, из которой будет осуществляться доступ или запуск приложения. Например, можно создать правило, согласно которому будут блокироваться попытки установить приложение из зоны Интернета, как это определено в браузере Microsoft Internet Explorer.

- **Network Access Restriction Policies (Политики ограничения доступа к сети)**. Этот параметр можно использовать для развертывания конфигурации клиента Network Access Protection (Защита сетевого доступа, NAP).

- **Application Control Policies (Политики управления приложениями)**. Этот узел позволяет администраторам групповой политики создавать правила, которые определяют то, какие группы доступа или конкретные пользователи могут запускать исполняемые программы, сценарии или установочные файлы Windows. Он также служит для тонкой настройки путей, имен файлов и публикаторов файлов с цифровой подписью, разрешенных или запрещенных на компьютерах, к которым будут применены установки политики.

- **IP Security Policies on Active Directory (Политики безопасности IP-адреса в Active Directory)**. Политики IP Security (IPsec) могут быть применены к объекту групповой политики объекта Active Directory, чтобы определить, когда и где будет разрешена/необходима связь по протоколу IPsec.

- **Advanced Audit Policy Configuration (Расширенная конфигурация политики аудита)**. Этот узел может использоваться для определения более детализированных и гранулированных настроек аудита для использования в системах Windows Server 2012 и Windows 8.

## **Узел Computer Configuration\Administrative Templates**

Узел Computer Configuration\Administrative Templates (Конфигурация компьютера/Административные шаблоны) содержит связанные с системным

реестром параметры политик, которые применяются к системе Windows. Эти параметры применяются главным образом для управления, конфигурирования и защиты установки и способа использования системы Windows. Эти параметры отличаются от параметров безопасности, в которых определенным пользователям или группам передаются полномочия, поскольку параметры конфигурации, доступные в административных шаблонах, применяются к системе и ко всем пользователям, получающим доступ к системе. Однако многие параметры не применяются к пользователям, являющимся членами локальной группы администраторов системы.

### **Узел политики User Configuration**

Узел User Configuration (Конфигурация пользователя) содержит параметры для конфигурирования и управления пользовательской средой рабочего стола в системе Windows. В отличие от параметров конфигурации компьютера, определяющих системные параметры и действия, которые пользователи могут выполнять в определенной системе, параметры конфигурации пользователя могут настраивать аспекты рабочего стола, включая настройку опций меню Start (Пуск), сокрытие или отключение значков панели управления (Control Panel), перенаправление папок в общие сетевые ресурсы, запрет записи данных на съемные носители и многое другое. В корне узла User Configuration имеются три узла политики: Software Settings (Конфигурация программ), Windows Settings (Параметры Windows) и Administrative Templates (Административные шаблоны), однако параметры, содержащиеся в этих узлах, отличаются от параметров, содержащихся в узле Computer Configuration (Конфигурация компьютера), а в групповой политике домена эти узлы находятся в узле User Configuration\Policies (Конфигурация пользователя\Политики).

#### **Узел User Configuration\Software Settings**

Узел Software Settings (Конфигурация программ) в разделе User Configuration (Конфигурация пользователя) политики позволяет администратору публиковать или назначать программные приложения отдельным пользователям, к которым применяется политика. Если пользователю назначается упакованное программное приложение, то его можно настроить так, чтобы оно автоматически устанавливалось во время входа пользователя в систему, или просто сделать так, чтобы пользователь имел к нему доступ для установки через значок Programs (Программы) панели управления, как и во время его публикации. Когда упакованное приложение публикуется для пользователя, этот пользователь может установить его, найдя его в панели управления. Для Windows 10 опубликованные программы можно найти по пути **Панель управления\Программы\Программы и компоненты\Установка новой программы из сети**.

#### **Узел User Configuration\Windows Settings**

Узел Windows Settings (Параметры Windows) в разделе User Configuration (Конфигурация пользователя) политики позволяет администраторам конфигурировать сценарии входа пользователей в систему, настраивать перенаправление папок в пользовательском профиле, определять политики ограниченного использования программ, автоматически устанавливать и при необходимости удалять принтеры.

## Узел User Configuration\Administrative templates

Параметры в узле Administrative Templates (Административные шаблоны) являются наиболее часто настраиваемыми параметрами политики при развертывании групповых политик домена. Параметры, содержащиеся в этом узле, можно применять для того, чтобы помочь администраторам автоматически конфигурировать пользовательскую среду рабочего стола. Конечно, теперь с помощью предпочтений групповых политик многие из новых доступных настроек также будут весьма востребованы, как только администраторы групповых политик начнут исследовать и искать наилучшие способы использования предпочтений.

Объекты групповых политик делятся на параметры политик и параметры предпочтений, как показано на рис. 4.3.

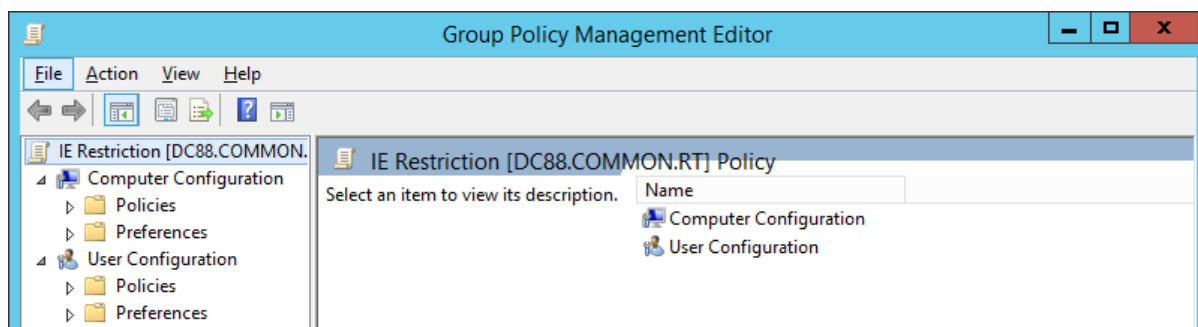


Рис. 4.3. Настройки (Preferences)

Предпочтения предлагают многие функциональные средства, которых не хватало в предыдущих версиях инфраструктуры Group Policy, и множество функций, которые обычно обрабатывались посредством сложных сценариев входа в систему и запуска, с помощью задач импортирования файла реестра, причем администраторами, конфигурирующими профиль пользователя по умолчанию на серверах и рабочих станциях. Многие параметры предпочтений, такие как Registry (Системный реестр) и Drive Maps (Отображения дисков), можно было предварительно применять с помощью сценариев, которые требовали входа в рабочую станцию или запуск во внутренней сети. Теперь же, благодаря параметрам предпочтений в групповых политиках доменов, эти параметры можно применять в период обновления групповой политики, что может благотворно сказаться на успешном применении этих типов параметров.

### 4.2.5. Административные шаблоны

В большинстве случаев административные шаблоны GPO представляют собой набор текстовых или XML-файлов, содержащих четко определенные параметры, которым можно присвоить ряд различных значений.

Административные шаблоны предоставляют администраторам легкий доступ ко многим конфигурационным параметрам, обычно используемым для управления компьютерами серверов и рабочих станций, а также конечными пользователями. При создании нового GPO в эту политику импортируется базовый набор административных шаблонов или ссылка на них. Можно импортировать и дополнительные административные шаблоны, чтобы добавить в какую-либо политику нужные функции. Когда в существующей сети устанавливаются новые



операционные системы (например, Windows 8 и Windows Server 2012), администраторы групповых политик увидят другие значения в редакторах групповых политик при редактировании политики в более новой ОС. Это может привести к путанице и проблемам, поэтому всем администраторам следует применять новые административные шаблоны. Быстрый способ для эффективного использования таких шаблонов в организации — задействование центрального хранилища групповых политик и обновление административных шаблонов в этом хранилище при появлении каждой новой операционной системы.

В новой инфраструктуре групповых политик, которая появилась в Windows Vista и Windows Server 2008, созданные GPO хранят только файлы и папки, нужные для хранения установленных параметров, сценариев, registry.pol и других файлов, имеющих отношение к GPO. При открытии GPO для редактирования или обработки на компьютере Windows Vista, Windows Server 2008 или более поздней версии используется ссылка на локальную копию административных шаблонов, но они не копируются в папку нового GPO в папке SYSVOL. Вместо этого в файлах, которые хранятся на локальных рабочих станциях или в центральном хранилище домена, имеются ссылки на административные шаблоны. Сами административные шаблоны хранятся в папке Windows\PolicyDefinitions.

Рассмотрим внутреннее устройство административных шаблонов на примере параметра Internet Explorer **Отключить вкладку «Общие»**. В редакторе групповой политики он выглядит следующим образом (рис. 4.4).

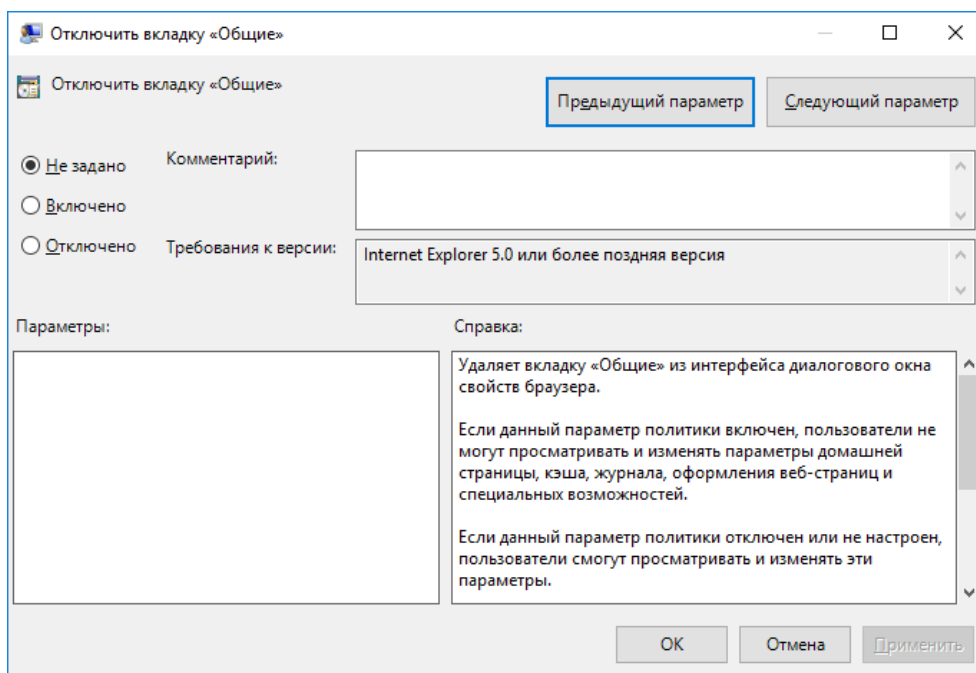


Рис. 4.4. Параметр **Отключить вкладку «Общие»**

В файле inetres.admx мы можем найти следующее описание этого параметра:

```
<policy name="ControlPanel_RestrictGeneralTab" class="Both"
  displayName="$(string.ControlPanel_RestrictGeneralTab)"
  explainText="$(string.IE_ExplainControlPanel_RestrictGeneralTab)"
  key="Software\Policies\Microsoft\Internet Explorer\Control Panel"
  valueName="GeneralTab">
  <parentCategory ref="InternetCPL" />
  <supportedOn ref="SUPPORTED_IE5" />
</policy>
```

Здесь можно увидеть, что внутреннее имя параметра, используемое в системе — **ControlPanel\_RestrictGeneralTab**, на целевом компьютере он будет храниться в ключе реестра **Software\Policies\Microsoft\Internet Explorer\Control Panel** в виде значения **GeneralTab**. Около знака доллара мы видим ссылки на файл .adml, в котором мы можем найти следующие фрагменты:

```
<string id="ControlPanel_RestrictGeneralTab">Отключить вкладку «Общие»</string>
```

```
<string id="IE_ExplainsControlPanel_RestrictGeneralTab">Удаляет вкладку «Общие» из интерфейса диалогового окна свойств браузера.
```

Если данный параметр политики включен, пользователи не могут просматривать и изменять параметры домашней страницы, кэша, журнала, оформления веб-страниц и специальных возможностей.

Сравнив вышеприведенные фрагменты с рис. 4.4, легко увидеть, что редактор отображает именно информацию, взятую из административного шаблона. Наконец, на рис.4.5 мы можем увидеть в редакторе реестра, как применение этого параметра групповой политики приводит к появлению в реестре Windows соответствующего значения.

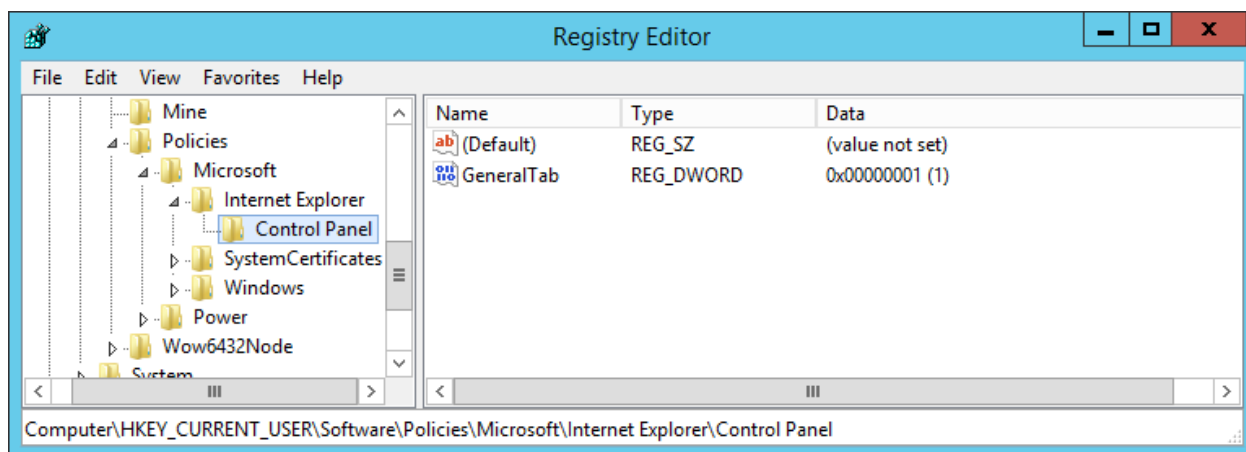


Рис. 4.5. Параметр GeneralTab в реестре Windows

#### 4.2.6. Централизованное хранилище шаблонов групповых политик

Если в сети имеется несколько контроллеров домена, работающих под управлением разных версий операционной системы, то консоль редактора групповых политик будет использовать локальные административные шаблоны. Чтобы создать одинаковую рабочую среду для администраторов, можно использовать центральное хранилище на томе SYSVOL.

Центральное хранилище GPO — это файловое хранилище, в котором хранятся все административные шаблоны следующего поколения. Это центральное хранилище предназначено для хранения всех новых административных шаблонов ADMX и ADML, и в каждой рабочей станции хранятся ссылки на файлы в контроллере домена, который применяется для обработки ее групповых политик. Теперь при открытии или обработке GPO система вначале проверяет наличие центрального хранилища, а затем использует шаблоны, хранимые в этом хранилище.

Создание центрального хранилища объектов GPO предоставляет администраторам простой, но эффективный способ управления административными шаблонами с сервера.

Для создания центрального хранилища GPO выполните следующие шаги.

1. Войдите в нужную административную систему, работающую под Windows 8 или Windows Server 2012.
2. Откройте папку C:\Windows\ и скопируйте в буфер обмена папку PolicyDefinitions.
3. В домене откройте папку \\имя\_домена\sysvol\ имя\_домена\policies.
4. Вставьте из буфера обмена папку PolicyDefinitions в папку, указанную в предыдущем шаге.
5. Закройте все открытые в окнах папки.

#### 4.2.7. Определение результирующей политики

В ситуациях, когда групповая политика не дает ожидаемых результатов, можно запустить средство GPO Results (Результаты объекта групповой политики), чтобы прочитать и отобразить хронологию обработки групповой политики. Средство Group Policy Results выполняется в отношении определенного компьютера, хотя его можно применять и для сбора результатов обработки политики пользователя. В следующем примере мы используем существующую систему с именем DC88 и существующего пользователя с именем Administrator (рис. 4.6).

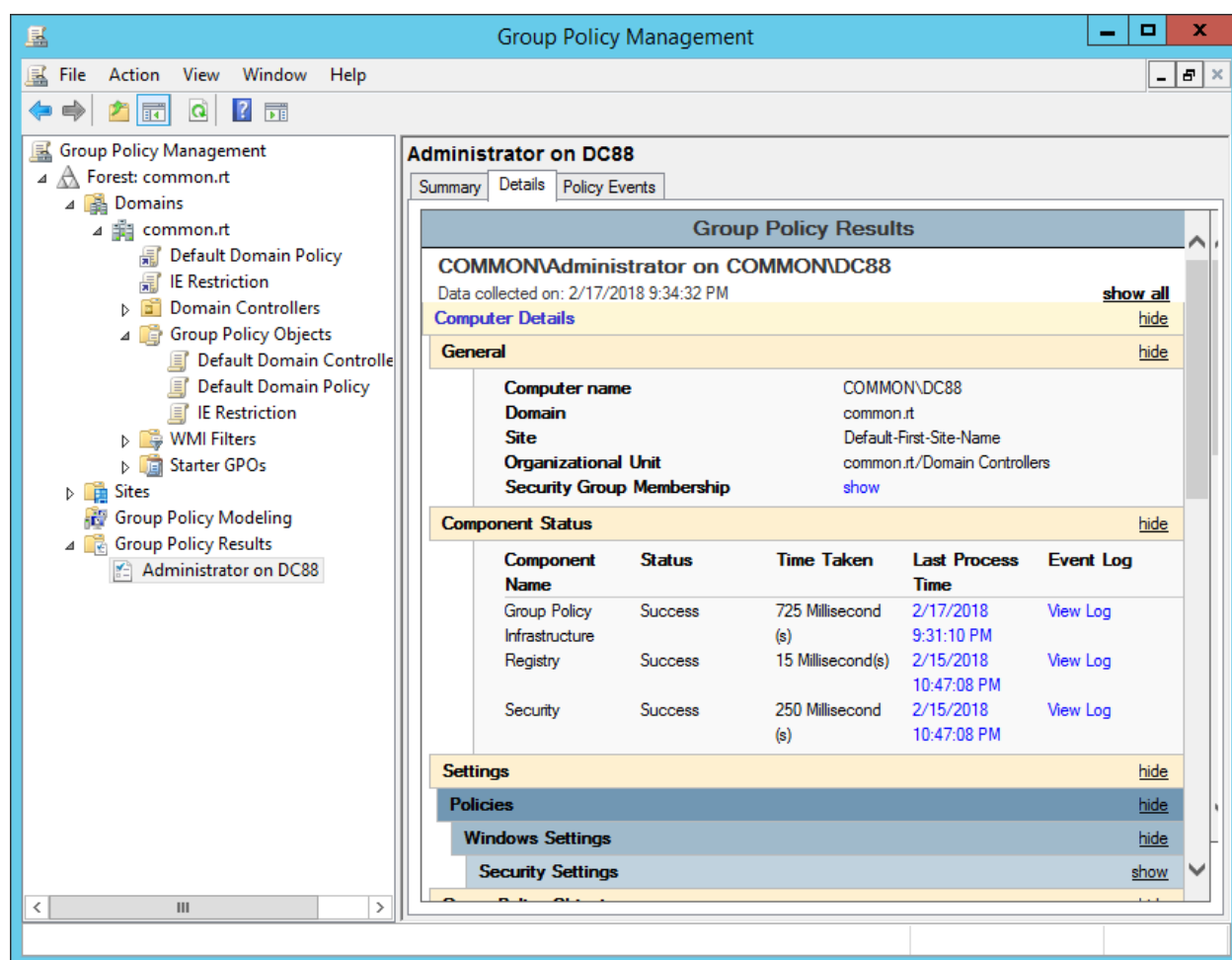


Рис. 4.6. Результат работы мастера Group Policy Results.

Также можно использовать утилиту командной строки `gpresult /v`.

Третий способ — открыть консоль Resultant Set of Policy, запустив файл `rsop.msc`.

#### 4.2.8. Управление рабочими столами

Групповая политика позволяет администраторам определять, как будут сконфигурированы личные настройки конечного пользователя и его рабочий стол. Также, благодаря групповым политикам, определяемым для пользователей, конечным пользователям может быть предоставлен или запрещен доступ к некоторым приложениям и функциональным средствам Windows, и может быть даже введено ограничение на чтение или запись информации на съемное запоминающее устройство. Обычные конфигурации пользовательской групповой политики могут содержать:

- конфигурацию меню Start (Пуск);
- ограничения параметров экрана и панели управления;
- параметры браузера Internet Explorer;
- ограничения на использование программного обеспечения;
- ограничения на использование консоли управления Microsoft (Microsoft Management Console);
- параметры экранной заставки;
- отображения сетевых дисков;
- установку принтеров;
- создание ярлыков на рабочем столе;
- конфигурации, характерные для приложений, включая настройку Microsoft Office, если в политику загружены и используются административные шаблоны;
- параметры конфигурации сети;
- планы электропитания;
- параметры перенаправления папок и автономных файлов.

По большому счету, управление средой пользователя и рабочим столом через групповые политики может применяться для конфигурирования графического пользовательского интерфейса и накладывать ограничения безопасности для повышения надежности используемых компьютерных систем. В большинстве случаев ярлыки программ могут быть помещены на рабочий стол, а значки скрыты от просмотра в панели управления или в меню Start (Пуск), но в более ограничивающих случаях они могут быть скрыты и запрещены для выполнения. Безопасность рабочего стола тоже является большой проблемой для компаний, особенно в настоящее время. Чтобы рабочие столы конечных пользователей были как можно более защищенными, можно реализовать экранную заставку, защитив ее паролем.

Профили службы удаленных рабочих столов удобны для систем Remote Desktop Services, однако реализация перемещаемых профилей на каждом компьютере в компании — не самое удачное решение, поскольку каждый раз, когда пользователь будет входить в систему, весь профиль будет копироваться на локальный компьютер, а когда пользователь будет выходить из системы, профиль будет копироваться обратно на сервер. Чем больше размер профиля, тем дольше он будет копироваться между общими папками на сервере и компьютером.

В системах удаленных рабочих столов администраторы могут без труда удаленно выходить и копировать профиль обратно в общую папку на сервере. А для рабочих станций конечных пользователей, когда перемещаемые профили становятся большими, многие пользователи не ждут, пока закончится копирование профиля, и вручную выключают систему, отключают ее от сети или переводят в спящий режим и забирают с собой. Это, естественно, может привести к повреждению профиля и, что еще хуже, к потере данных. Чтобы улучшить производительность профиля удаленного рабочего стола и стандартного перемещаемого профиля, администраторы могут использовать групповую политику для перенаправления пользовательских папок в общие папки на сервере с использованием механизма перенаправления папок.

Перенаправление папок может применяться для перенаправления некоторых специальных папок, включенных в профиль конечного пользователя, в общие папки на сервере.

В общие папки на сервере можно перенаправлять специальные папки, такие как Documents (Документы), которая является папкой по умолчанию, где пользователь хранит свои данные.

Групповые политики в Windows Vista, Windows Server 2008 и более поздних клиентских и серверных операционных системах предлагают несколько новых параметров, которые можно применять для управления использованием переносных устройств и съемных запоминающих устройств. Некоторые из этих параметров применяются к CD/DVD-дискам, однако большая часть предназначена для управления полномочиями на чтение и запись съемных дисков, таких как внешние USB-приводы и карты памяти. Эти параметры можно сконфигурировать в групповой политике компьютера, а также в узле User Configuration (Конфигурация пользователя), чтобы отказать в доступе на запись данных на съемный носитель.

## **4.3. Использование PowerShell для администрирования Windows**

Оболочки (shell) — это средство, необходимое для работы в операционных системах. Они дают возможность выполнять произвольные команды и просматривать содержимое файловой системы. Все пользователи компьютеров имели дело с оболочкой - либо вводя команды в командной строке, либо щелкая на значке для запуска текстового процесса. Каждый пользователь так или иначе пользовался оболочкой. Это неизбежно, в какой бы форме он ни работал с компьютерной системой.

Ранее пользователи и администраторы в основном взаимодействовали с большинством операционных систем Windows с помощью проводника Windows или командной строки cmd (и то, и другое — оболочки). Но с выпуском PowerShell — новой оболочки и языка написания сценариев — привычный стандарт взаимодействия с Windows и управления системой начал быстро изменяться. Это изменение стало еще более заметным после выпуска Microsoft Exchange Server 2007, где PowerShell применялся как основное средство управления (зачастую с функциями, которых нет в графическом интерфейсе) и добавления PowerShell в Windows Server 2008. С каждым очередным выпуском операционных систем семейства Windows увеличивается количество функций, которые можно выполнять в PowerShell.

### **4.3.1. Начальные сведения о PowerShell**

Windows PowerShell, представленная в 2006 году, является платформой, построенной на платформе Microsoft .NET. Вы можете использовать Windows PowerShell для автоматизации административных задач. Поскольку он построен на .NET, Windows PowerShell объектно-ориентирован. Это обеспечивает большую мощность и гибкость при работе с данными по сравнению с работой со всеми данными в виде текста. У вас также есть доступ к тем же функциям, что и .NET Framework, и к другим библиотекам программ, которые используют разработчики программного обеспечения.

Команды предоставляют основные функции Windows PowerShell. Существует множество разновидностей команд, в том числе командлеты (cmdlet), функции, фильтры, сценарии, приложения, конфигурации и рабочие процессы. Команды — это блоки, которые вы объединяете, используя язык сценариев Windows PowerShell. Используя команды, вы можете создавать собственные решения для сложных административных проблем. Кроме того, вы можете просто запускать команды непосредственно в консоли Windows PowerShell для выполнения одной задачи. Консоль — это интерфейс командной строки (CLI) для Windows PowerShell и является основным способом взаимодействия с Windows PowerShell. Windows PowerShell заменяет предыдущий CLI cmd.exe и ограниченную функциональность языка сценариев командного файла.

Приложения Microsoft и облачные сервисы предоставляют специализированные командлеты, которые можно использовать для управления этими службами. Фактически, вы можете управлять некоторыми функциями только с помощью Windows PowerShell. Во многих случаях, даже когда приложение предоставляет

графический интерфейс пользователя (GUI) для управления программным обеспечением, он подключается к Windows PowerShell и запускает командлеты за кулисами.

Приложения и службы с административными функциями Windows PowerShell являются последовательными в том, как они работают. Это означает, что вы можете быстро применить полученные уроки. Кроме того, когда вы используете сценарии автоматизации для администрирования программного приложения, вы можете повторно использовать их и среди других приложений.

Начиная с 2016 года, Windows PowerShell стала проектом с открытым исходным кодом, в настоящее время называемым PowerShell Core, и поддержкой Microsoft для запуска Windows PowerShell в Linux. Это расширяет полезность Windows PowerShell за пределами среды администрирования Windows. Вы можете использовать Windows PowerShell для управления:

- Linux-серверами и macOS из Windows Server;
- Windows Server из Linux и macOS.

Администратор может с помощью PowerShell выполнять целый ряд задач, в том числе описанные ниже.

- **Управление файловой системой** — создание, удаление, изменение файлов и папок и задание прав доступа к ним.

- **Управление службами** — просмотр, остановка, запуск, перезапуск и даже изменение служб.

- **Управление процессами** — просмотр (мониторинг), остановка и запуск процессов.

- **Управление системным реестром** — создание, изменение, удаление и чтение значений, хранящихся в системном реестре.

- **Инструментальные средства управления Windows (WMI)** — управление не только Windows, но и другими платформами, например, IIS и терминальной службой.

- **Использование существующих объектов COM (Component Object Model)** — выполнение широкого спектра задач автоматизации.

- **Управление рядом ролей и компонентов Windows** — добавление и удаление ролей и компонентов.

- **Выполнение административных задач** — выполнение множества задач, от сброса паролей и добавления записей DNS до настройки виртуальных машин. Многие приложения, например, Microsoft Exchange Server, Microsoft SharePoint Server, Microsoft System Center и другие, предоставляют модули для управления этими приложениями из среды PowerShell.

Windows PowerShell имеет пять версий: Windows PowerShell 5.0, Windows PowerShell 4.0, Windows PowerShell 3.0, Windows PowerShell 2.0 и Windows PowerShell 1.0. Windows PowerShell 5.0 обратно совместим со старыми версиями Windows PowerShell, кроме версии 1.0. Помните, что многие серверные продукты зависят от конкретной версии Windows PowerShell. Например, Microsoft Exchange Server 2010 требует Windows PowerShell 2.0 и несовместим с более новыми версиями Windows PowerShell. Перед установкой новой версии Windows

PowerShell на компьютер проверьте совместимость этой версии с программным обеспечением, установленным на этом компьютере.

Кратко охарактеризуем несколько последних версий PowerShell.

### Windows PowerShell 4.0

Windows PowerShell 4.0 является установкой по умолчанию в Windows 8.1 и Windows Server 2012 R2. Он также доступен в виде бесплатной загрузки для Windows 7 с пакетом обновления 1 (SP1), для Windows Server 2008 R2 с пакетом обновления 1 (SP1), Windows 8 и Windows Server 2012. Windows PowerShell 4.0 несовместим с более ранними версиями Windows. Windows Management Framework 4.0 — это бесплатный пакет загрузки, в котором доступен Windows PowerShell 4.0. Для Windows PowerShell 4.0 требуется версия 4.5 .NET Framework.

### Windows PowerShell 5.0

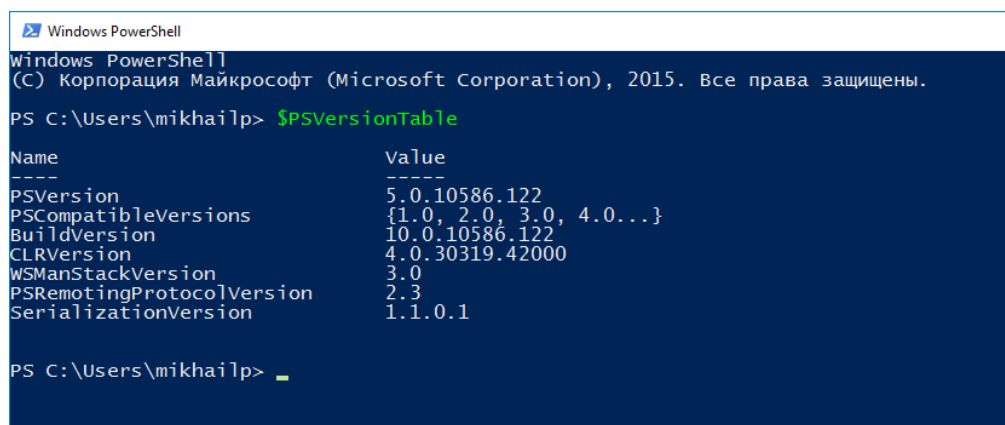
Windows PowerShell 5.0 является установкой по умолчанию в Windows 10 Version 1511. Вы можете установить Windows PowerShell 5.0 на Windows Server 2008 R2, Windows Server 2012, Windows 10, Windows 8.1 Enterprise, Windows 8.1 Pro или Windows 7 SP1. Для этого вы можете загрузить и установить Windows Management Framework 5.0 из Центра загрузки Windows.

### Windows PowerShell 5.1

Windows PowerShell 5.1 выпущен как часть Windows Management Framework (WMF) 5.1 и включен в Windows Server 2016 и Windows 10 Anniversary Edition. Вы можете установить Windows PowerShell 5.1 на все операционные системы, на которые можно установить Windows PowerShell 5.0. Для этого вы можете загрузить и установить Windows Management Framework 5.1 из Центра загрузки Windows.

Когда вы просматриваете сеанс Windows PowerShell, может быть сложно определить, какую версию вы используете. Все версии Windows PowerShell устанавливаются в папку %systemdir%\WindowsPowerShell\v1.0. Это относится к языку Windows PowerShell 1.0. Это означает, что имя папки не поможет узнать номер версии самого Windows PowerShell.

Чтобы правильно определить версию, введите `$PSVersionTable` в Windows PowerShell, а затем нажмите Enter. Windows PowerShell отобразит номера версий для различных компонентов (рис. 4.7). Сюда входит основной номер версии Windows PowerShell. Помните, что этот метод не будет работать в Windows PowerShell 1.0. Вместо этого он вернет пустой результат.



```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation), 2015. Все права защищены.

PS C:\Users\mikhailp> $PSVersionTable

Name                           Value
----                           -
PSVersion                      5.0.10586.122
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.10586.122
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1

PS C:\Users\mikhailp> _
```

Рис. 4.7.Содержимое переменной `$PSVersionTable` в Windows 10 Version 1511



Чтобы взаимодействовать с Windows PowerShell, вы можете использовать любое приложение, которое использует движок Windows PowerShell. Некоторые из этих приложений могут иметь графический интерфейс, такие как консоль управления Exchange в Exchange Server 2007 и более новые версии. Два таких приложения входят в состав Windows: консоль Windows PowerShell и Windows PowerShell ISE.

### Консоль Windows PowerShell

Консоль использует встроенное консольное приложение Windows. Это в походе на обычную командную строку из старой оболочки cmd.exe. Однако Windows Server 2016 и Windows 10 предоставляют обновленную консоль с большей функциональностью, включая окраску синтаксиса. Эта консоль обеспечивает простую среду, не поддерживает двухбайтовые наборы символов и имеет ограниченные возможности редактирования. В настоящее время он предоставляет самую широкую функциональность Windows PowerShell. Консоль в Windows PowerShell 5.1 включает значительные улучшения.

### Windows PowerShell ISE

ISE — это приложение Windows Presentation Foundation (WPF), которое предоставляет расширенные возможности редактирования, подсказки и завершение кода IntelliSense и поддержку двухбайтовых наборов символов.

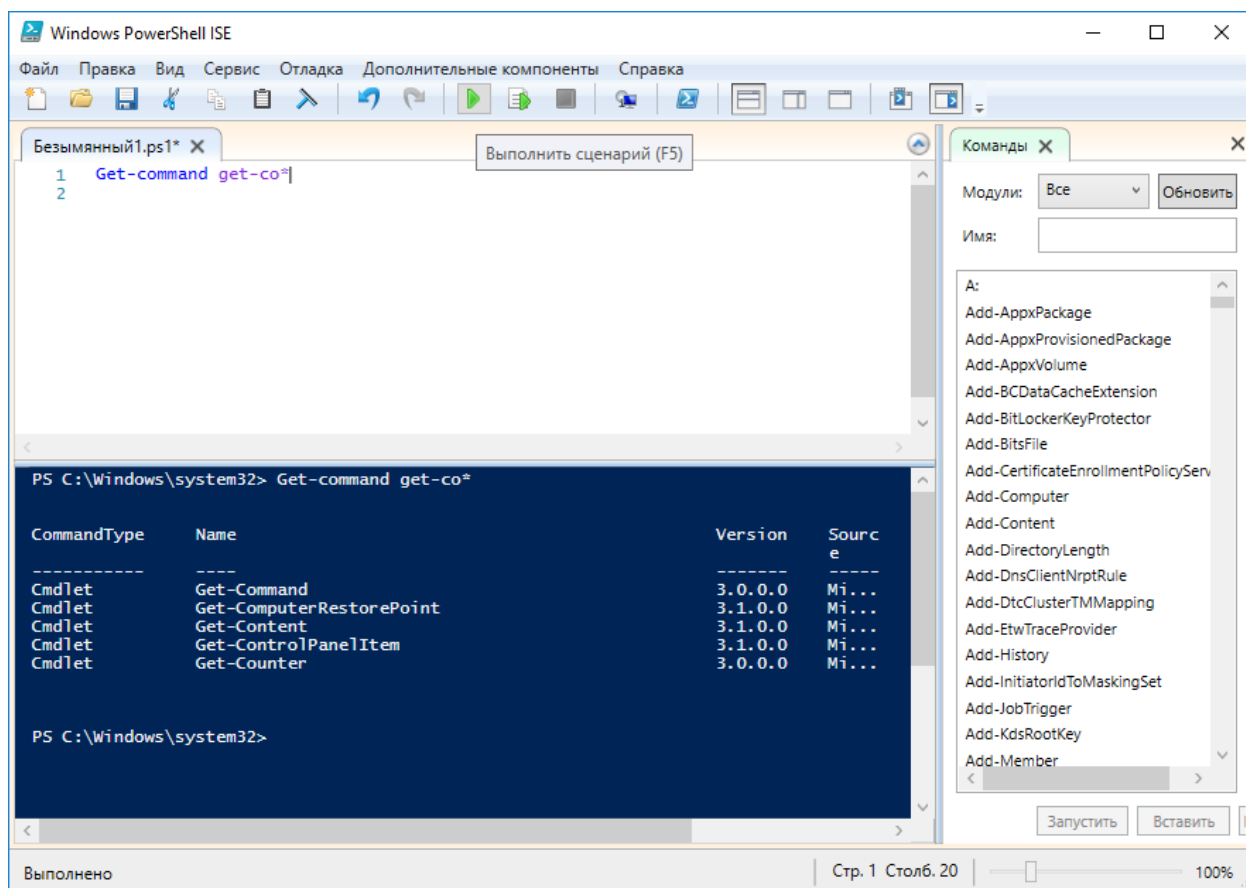


Рис. 4.8. Windows PowerShell ISE

Чтобы создать свое приложение, использующее платформу PowerShell, нужно включить в проект ссылку на библиотеку System.Management.Automation.dll и добавить директивы using для подключения соответствующих пространств имен (рис. 4.9).

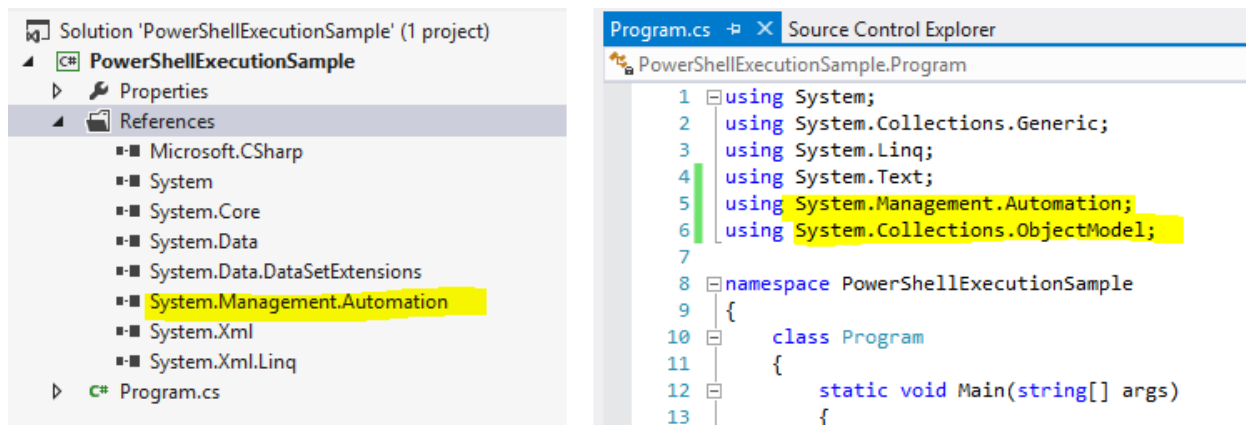


Рис. 4.9. Добавление в проект ссылки на библиотеку System.Management.Automation.dll

Более подробную информацию можно найти по ссылке:

#### Executing PowerShell scripts from C#

<https://blogs.msdn.microsoft.com/kebab/2014/04/28/executing-powershell-scripts-from-c/>

Синтаксис применения PowerShell из командного интерфейса (Command-Line Interface — CLI) похож на синтаксис других командных оболочек. Одним из основных компонентов команд PowerShell является, конечно же, имя команды, которая должна быть выполнена. Кроме того, действие команды можно уточнить с помощью параметров и аргументов этих параметров. То есть команды PowerShell могут быть записаны в одном из следующих форматов:

- [имя\_команды]
- [имя\_команды] -[параметр]
- [имя\_команды] -[параметр] -[параметр] [аргумент1]
- [имя\_команды] -[параметр] -[параметр] [аргумент1], [аргумент2]

В PowerShell параметр — это переменная, принимаемая командой, сценарием или функцией. Аргумент — значение, присвоенное параметру. Поскольку эти термины часто путают, запомните данные определения, они понадобятся при обсуждении использования PowerShell.

При работе с командным интерфейсом PowerShell, как и со всеми другими командными оболочками, нужно знать принципы навигации в нем. Ниже приведен перечень операций редактирования и соответствующих им клавиш при работе в консоли PowerShell.

Клавиши	Операция редактирования
Клавиши со стрелками влево и вправо	Перемещение курсора влево и вправо в текущей командной строке
Клавиши со стрелками вверх и вниз	Перемещение по списку последних введенных команд
<PgUp>	Вывод первой команды в истории команд
<PgDn>	Вывод последней команды в истории команд
<Home>	Перемещение курсора в начало командной строки
<End>	Перемещение курсора в конец командной строки
<Insert>	Переключение между режимами ввода текста: вставка или замена

<Delete>	Удаление символа в позиции курсора
<Backspace>	Удаление символа непосредственно перед позицией курсора
<F3>	Вывод предыдущей команды
<F4>	Удаление указанного количества символов от текущего положения курсора
<F5>	Перемещение назад по истории команд
<F7>	Вывод списка последних введенных команд во всплывающем окне. Выберите с помощью стрелок вверх и вниз одну из введенных ранее команд, а затем нажмите клавишу <Enter>, чтобы выполнить эту команду
<F8>	Продвижение вперед по истории тех команд, которые соответствуют тексту, введенному в командной строке
<F9>	Выводит номер команды и выполняет указанную команды из истории команд (номера команд соответствуют списку команд, выводимому клавишей <F7>)
<Tab>	Автозаполнение текста команды. Нажатие <Shift+Tab> позволяет просматривать в обратном порядке список потенциальных совпадений

## Переменные

Переменная — это место для хранения данных. В большинстве оболочек переменные могут хранить только текстовые данные. В самых продвинутых оболочках и языках программирования данные, хранящиеся в переменных, могут быть практически любого типа, от строк и до последовательностей объектов. В PowerShell также переменные могут хранить информацию практически любого вида.

Для определения переменной в PowerShell ей нужно назначить имя, начинающееся с префикса \$, который помогает отличить переменные от псевдонимов, командлетов, имен файлов и других элементов, требуемых программисту. Имена переменных могут содержать любое сочетание алфавитно-цифровых символов (a-z и 0-9) и символа подчеркивания (\_). Для переменных PowerShell нет специальных соглашений по именованию, но лучше, если имя отражает тип данных, которые хранит переменная. Регистр символов значения не имеет.

## Политики выполнения

Политика выполнения определяет ограничения на выполнение сценариев в PowerShell или загрузкой конфигурационных файлов. В PowerShell имеются четыре основных политики выполнения: **Restricted**, **AllSigned**, **RemoteSigned** и **Unrestricted**. **Restricted** вообще запрещает выполнение любых сценариев, **AllSigned** разрешает выполнение лишь тех сценариев, которые имеют цифровую подпись от доверенного издателя, **RemoteSigned** разрешает выполнение локально созданных сценариев, сценарии с удаленных систем должны иметь цифровую подпись. **Unrestricted** разрешает выполнять любые сценарии, но выводит предупреждение для удаленных сценариев без цифровой подписи.

При использовании политики **Bypass** (Обход) выполнения ничего не блокируется, и нет никаких предупреждений. Эта политика обычно применяется,

если PowerShell используется другим приложением с собственной моделью безопасности или сценарий PowerShell встроен в другое приложение.

После первоначальной установки PowerShell политика выполнения по умолчанию устанавливается в Restricted. Для изменения политики выполнения используется командлет Set-ExecutionPolicy:

```
PS C:\>Set-ExecutionPolicy AllSigned
```

### 4.3.2. Типы команд

Когда команда выполняется в PowerShell, интерпретатор команд определяет по имени команды, какую задачу следует выполнить. В это время определяется и тип команды, и способ ее выполнения. Имеется четыре типа команд PowerShell: командлеты, функции оболочки, команды сценариев и встроенные команды.

#### Командлеты

Первым типом команд являются командлеты (cmdlet), которые похожи на встроенные команды в других командных оболочках. Различие состоит в том, что командлеты реализованы с помощью классов .NET, компилированных в динамически подключаемую библиотеку (DLL) и загруженных в PowerShell во время выполнения. Это различие означает, что не существует фиксированного класса встроенных командлетов: каждый может, воспользовавшись набором средств разработки ПО (Software Developers Kit — SDK) PowerShell, написать собственный командлет, расширив таким образом функциональность PowerShell.

Имя командлета всегда составляется из глагола и существительного, разделенных дефисом ("-"). Глагол описывает действие, выполняемое командлетом, а существительное — обрабатываемый объект. Ниже приведен пример выполнения командлета Get-Process для получения списка процессов, имена которых начинаются с буквы "P".

```
PS C:\Users\mikhailp> Get-Process p*
```

Handles	NPM(K)	PM(K)	WS (K)	VM (M)	CPU(s)	Id	SI	ProcessName
-----	-----	-----	-----	-----	-----	--	--	-----
1211	93	562544	370260	1118	43,16	2752	2	Photoshop
1104	71	112992	147568	699	41,11	7388	2	POWERPNT
743	29	84056	95312	...01	1,31	9848	2	powershell
202	27	27120	22340	523		4616	0	PresentationFontCache
87	7	936	4540	27		2176	0	PsiService_2

При выполнении командлетов в PowerShell следует учитывать несколько моментов. Во-первых, синтаксис в PowerShell легок в использовании и прощает многие ошибки. А во-вторых, PowerShell пытается заполнять информацию за пользователя.

- Имена командлетов всегда состоят из глагола и существительного в единственном числе.
- Параметры и аргументы являются позиционными: `Get-Process winword`;
- Во многих аргументах допустимы обобщенные символы: `Get-Process w*`.
- Разрешены также частичные имена параметров: `Get-Process -P w*`.

Глаголы в PowerShell стандартизированы, и рекомендуется строго следовать этим стандартам. Например, для изменения объекта используется глагол Set, а

использование Change и Modify запрещено. Для получения списка доступных глаголов используется командлет Get-Verb:

```
PS C:\Users\mikhailp>Get-Verb | Sort-Object Verb
```

Verb	Group
-----	-----
Add	Common
. . .	
Block	Security
Checkpoint	Data
. . .	
Merge	Data
Mount	Data
. . .	
Send	Communications
Set	Common
Show	Common
. . .	
Write	Communications

Ниже перечислены чаще всего употребляемые глаголы:

- **Get.** Извлекает ресурс, например, файл или пользователя.
- **Set.** Изменяет данные, связанные с ресурсом, такие как файл или пользовательское свойство пользователя.
- **New.** Создает ресурс, такой как файл или пользователь.
- **Add.** Добавляет ресурс в контейнер из нескольких ресурсов.
- **Remove.** Удаляет ресурс из контейнера из нескольких ресурсов.

Они представляют собой лишь некоторые из глаголов, которые используют командлеты. Кроме того, некоторые глаголы выполняют аналогичные функции. Например, глагол Add может создать ресурс, как и глагол New. Некоторые глаголы могут казаться похожими, но имеют очень разные функции. Например, глагол Read получает информацию, содержащуюся в ресурсе, такую как содержимое текстового файла, тогда как Get извлекает сведения о файле.

Вторая часть имени командлета (Noun, существительное) указывает, на какие ресурсы или объекты влияет командлет. Все командлеты, которые работают с одним ресурсом, должны использовать одно и то же существительное. Например, существительное Service используется для командлетов, работающих с службами Windows, и существительное Process используется для управления процессами на компьютере.

Существительные могут также иметь префиксы, которые помогают группировать родственные существительные в семьи. Например, существительные имен Active Directory начинаются с букв AD (таких как ADUser, ADGroup и ADComputer). Командлеты Microsoft SharePoint Server начинаются с префикса SP и командлеты Microsoft Azure Active Directory начинаются с префикса AzureAD.

### Получение справочной информации

Если неизвестно точное имя командлета, то можно использовать командлет Get-Command с указанием предположительной части названия. Например, для

поиска командлетов для работы с заданиями, можно использовать следующую команду:

```
PS C:\Users\mikhailp> get-command *Job*
```

CommandType	Name	Version	Source
-----	----	-----	-----
Function	Get-PrintJob	1.1	PrintManagement
Function	Get-StorageJob	2.0.0.0	Storage
Function	Remove-PrintJob	1.1	PrintManagement
Function	Restart-PrintJob	1.1	PrintManagement
Function	Resume-PrintJob	1.1	PrintManagement
Function	Stop-StorageJob	2.0.0.0	Storage
Function	Suspend-PrintJob	1.1	PrintManagement
Cmdlet	Add-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	Debug-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Disable-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	Disable-ScheduledJob	1.1.0.0	PSScheduledJob
Cmdlet	Enable-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	Enable-ScheduledJob	1.1.0.0	PSScheduledJob
Cmdlet	Get-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Get-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	Get-ScheduledJob	1.1.0.0	PSScheduledJob
Cmdlet	Get-ScheduledJobOption	1.1.0.0	PSScheduledJob
Cmdlet	New-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	New-ScheduledJobOption	1.1.0.0	PSScheduledJob
Cmdlet	Receive-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Register-ScheduledJob	1.1.0.0	PSScheduledJob
Cmdlet	Remove-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Remove-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	Resume-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Set-JobTrigger	1.1.0.0	PSScheduledJob
Cmdlet	Set-ScheduledJob	1.1.0.0	PSScheduledJob
Cmdlet	Set-ScheduledJobOption	1.1.0.0	PSScheduledJob
Cmdlet	Start-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Stop-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Suspend-Job	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Unregister-ScheduledJob	1.1.0.0	PSScheduledJob
Cmdlet	Wait-Job	3.0.0.0	Microsoft.PowerShell.Core

Далее можно получить справочную информацию о найденной команде с помощью командлета Get-Help:

```
PS C:\Users\mikhailp> Get-Help Start-Job
```

ИМЯ

Start-Job

ОПИСАНИЕ

Starts a Windows PowerShell background job.

СИНТАКСИС

```
Start-Job [-ScriptBlock] <ScriptBlock> [[-InitializationScript] <ScriptBlock>]
[-ArgumentList <Object[]>] [-Authentication <AuthenticationMechanism> {Default |
Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest |
Kerberos}] [-Credential <PSCredential>] [-InputObject <PSObject>] [-Name
String] [-PSVersion <Version>] [-RunAs32] [<CommonParameters>]
```

. . .

Есть встроенная справка по самому PowerShell. Это набор статей, начинающихся с фразы "about\_". Например, так можно получить список всех доступных статей справки:

```
PS C:\Users\mikhailp> Help about_*
```

Name	Category	Module	Synopsis
----	-----	-----	-----
about_ActivityCommonParameters	HelpFile		Describes the parameters that...
about_Aliases	HelpFile		Describes how to use alternat...
about_Arithmetic_Operators	HelpFile		Describes the operators that ...
about_Arrays	HelpFile		Describes arrays, which are d...
about_Assignment_Operators	HelpFile		Describes how to use operator...

about\_Automatic\_Variables      HelpFile      Describes variables that stor...  
...

Операционная система Windows поставляется с минимальным набором справочной информации. Перед началом работы со справкой нужно выполнить командлет Update-Help, который загрузит из Интернет самую свежую информацию (рис. 4.10).

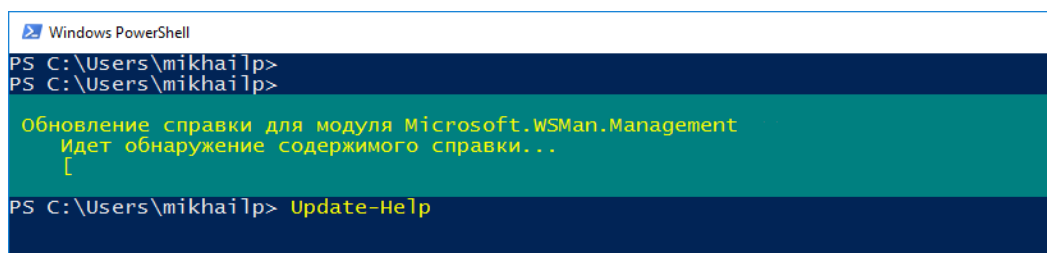


Рис. 4.10. Обновление справочной системы PowerShell

## Функции

Следующим типом команд являются функции. Эти команды дают возможность назначить имя последовательности команд. Функции похожи на подпрограммы и процедуры в других языках программирования. Их главное отличие от сценариев состоит в том, что для каждого командного сценария запускается новый экземпляр оболочки, а функции выполняются в том же экземпляре текущей оболочки.

Функции, определенные в командной строке, доступны только на протяжении текущего сеанса PowerShell. Кроме того, их область действия локальна, и они не применяются к новым сеансам PowerShell.

Функции, определенные в командной строке, — удобный способ динамического создания последовательности команд в среде PowerShell, но они хранятся только в памяти и стираются при закрытии и перезапуске PowerShell. Поэтому, хотя динамическое создание сложных функций и возможно, лучше все-таки писать их в виде команд сценариев. Вот пример командной функции приведен на рис. 4.10.

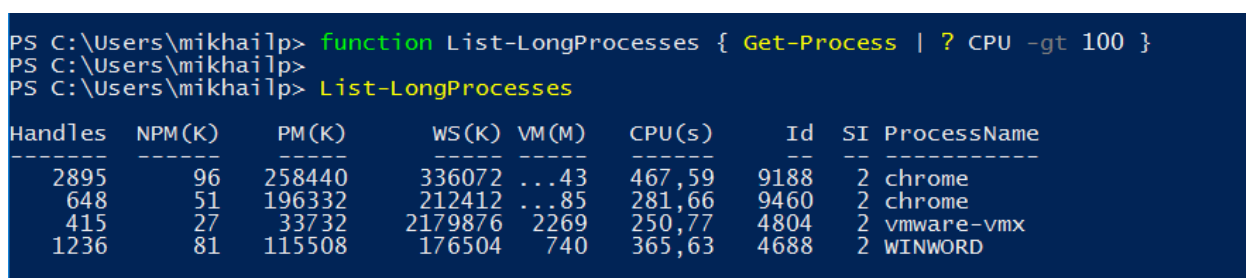


Рис. 4.10. Пример функции, выводящей список текущих процессов, занимавших процессор более 100 секунд

## Сценарии

Третий тип команд — сценарии — это команды PowerShell, хранящиеся в .ps1-файле. Основное их отличие от функций в том, что сценарии хранятся на диске и могут быть вызваны в любой момент, а функции не сохраняются между сеансами PowerShell.

Сценарии можно запускать в сеансе PowerShell либо из командной строки cmd. Чтобы запустить сценарий в сеансе PowerShell, введите имя этого сценария без расширения. За ним можно указать нужные параметры. Оболочка выполняет

первый .ps1-файл, имя которого совпадает с введенным именем, из любого каталога, указанного в переменной PowerShell по имени \$ENV:PATH.

Чтобы запустить сценарий PowerShell из командной строки cmd, вначале нужно с помощью команды CD открыть каталог, в котором находится этот сценарий. Затем запустите исполняемый файл PowerShell с параметром -command и указанием нужного сценария, например:

```
C:\Scripts>PowerShell -command .\myscript.ps1
```

Если вы не хотите входить в каталог сценария с помощью команды CD, его можно запустить и с помощью абсолютного пути:

```
C:\>PowerShell -command C:\Scripts\myscript.ps1
```

Важным моментом относительно сценариев в PowerShell являются стандартные ограничения безопасности. По умолчанию сценарии запускать запрещено - это один из методов защиты от вредоносных сценариев. Такую политику можно изменить с помощью командлета Set-ExecutionPolicy.

### Встроенные команды

Последний тип команд — встроенные команды — включает в себя внешние программы, которые может выполнять операционная система. Поскольку для их выполнения требуется создание нового процесса, они не так эффективны, как остальные типы команд PowerShell. Встроенные команды имеют свои собственные параметры, которые обычно отличаются от параметров PowerShell.

### 4.3.3. Передача параметров по конвейеру

Раньше данные передавались от одной команды к другой с помощью конвейера, что давало возможность выполнить сразу последовательность команд, чтобы выбрать нужную информацию из системы. Но, как уже было сказано, у большинства оболочек имеется серьезный недостаток: информация, выдаваемая командами, является текстовой — это просто сцепление выходного потока stdout первой команды с входным потоком stdin второй команды. Такой текст приходится преобразовывать в формат, который понимает следующая команда, а уже затем передавать этой команде.

PowerShell передает по конвейеру полноценные объекты .NET, со свойствами и методами. В этом можно убедиться, отправив вывод командлета Get-Process на вход командлета Get-Member, который отображает тип объекта и его свойства и методы. Часть этого вывода приведена ниже:

```
PS C:\Users\mikhailp> Get-Process WINWORD | Get-Member
```

```
TypeName: System.Diagnostics.Process
```

Name	MemberType	Definition
----	-----	-----
Handles	AliasProperty	Handles = Handlecount
Name	AliasProperty	Name = ProcessName
. . .		
Disposed	Event	System.EventHandler Disposed(System.Ob...
. . .		
InitializeLifetimeService	Method	System.Object InitializeLifetimeService()
Kill	Method	void Kill()
Refresh	Method	void Refresh()
Start	Method	bool Start()
. . .		
BasePriority	Property	int BasePriority {get;}
Container	Property	System.ComponentModel.IContainer Conta...



EnableRaisingEvents	Property	bool EnableRaisingEvents {get;set;}
ExitCode	Property	int ExitCode {get;}
ExitTime	Property	datetime ExitTime {get;}
Handle	Property	System.IntPtr Handle {get;}
HandleCount	Property	int HandleCount {get;}
. . .		
ProductVersion	ScriptProperty	System.Object ProductVersion {get=\$thi...

Здесь можно увидеть тип объекта — `System.Diagnostics.Process` — и различные события, свойства и методы, например, метод `Kill()`, позволяющий аварийно завершить процесс.

Одно из применений конвейера — **форматирование результатов**. Windows PowerShell предоставляет несколько способов управления форматированием вывода конвейера. Форматирование вывода по умолчанию зависит от объектов, которые существуют в выводе, и файлов конфигурации, которые определяют выход. После того, как Windows PowerShell примет решение о соответствующем формате, он передает результат на один из командлетов форматирования без вашего указания.

Также можно явно указать один из следующих командлетов форматирования:

- `Format-List`
- `Format-Table`
- `Format-Wide`
- `Format-Custom`

Команда `Sort-Object` принимает одно или несколько имен свойств для **сортировки**. По умолчанию команда выполняет сортировку в порядке возрастания. Если вы хотите изменить порядок сортировки на противоположный, добавьте параметр `-Descending`. Если вы указываете более одного свойства, команда сначала сортирует по первому свойству, затем по второму свойству и так далее. В одной команде невозможно сортировать по одному свойству в порядке возрастания, а другое — в порядке убывания.

Следующие команды — примеры сортировки:

```
Get-Service | Sort-Object -Property Name -Descending
Get-Service | Sort Name -Desc
Get-Service | Sort Status,Name
```

Вы можете использовать **Select-Object** для указания свойств для отображения. Вы используете параметр `-Property`, за которым следует список свойств, которые вы хотите отобразить. Если представить себе коллекцию объектов как таблицу, то вы просто выбираете столбцы для отображения. После отбора свойств, которые вы указали, `Select-Object` удаляет все другие свойства (или столбцы). Если вы хотите отсортировать по свойству, но не отобразить его, сначала используйте `Sort-Object`, а затем используйте `Select-Object`, чтобы указать свойства, которые вы хотите отобразить.

Следующая команда отображает таблицу, содержащую имя, идентификатор процесса, размер виртуальной памяти и использование ЦП для всех процессов, запущенных на локальном компьютере:

```
Get-Process | Select-Object -Property Name,ID,VM,CPU | Format-Table
```

## Фильтрация результатов

Команда **Where-Object** (и ее псевдонимы, **Where** и **?**) имеет две синтаксические формы: базовую и расширенную. Эти две формы определяются обширным списком наборов параметров — по одному для каждого оператора сравнения. Это позволяет легко прочитать синтаксис, особенно в базовой форме. Например, чтобы отобразить список только запущенных служб, введите следующую команду в консоли и нажмите «Ввод»:

```
Get-Service | Where Status -eq Running
```

В расширенном синтаксисе Where-Object используется скрипт фильтра. Скрипт фильтра представляет собой блок сценария, который содержит сравнение и передается с помощью параметра **-FilterScript**. Внутри этого блока скриптов вы можете использовать встроенную переменную **\$PSItem** (или **\$\_**, которая также действительна в версиях Windows PowerShell до 3.0), чтобы сослаться на любой объект, который был передан в команду. Скрипт фильтра запускается один раз для каждого объекта, который передается в команду. Когда скрипт фильтра возвращает True, этот объект передается по конвейеру в качестве вывода. Когда скрипт фильтра возвращает False, этот объект удаляется из конвейера.

Следующие две команды функционально идентичны. Первый использует базовый синтаксис, а второй использует расширенный синтаксис, чтобы сделать то же самое:

```
Get-Service | Where Status -eq Running
```

```
Get-Service | Where-Object -FilterScript { $PSItem.Status -eq 'Running' }
```

Параметр **-FilterScript** является позиционным, и большинство пользователей опускают его. Большинство пользователей также используют псевдоним Where или ?, который еще короче. Опытные пользователи Windows PowerShell также используют переменную **\_** вместо **\$PSItem**, потому что в Windows PowerShell 1.0 и Windows PowerShell 2.0 допускается только **\_**. Следующие команды выполняют ту же задачу, что и предыдущие две команды:

```
Get-Service | Where { $PSItem.Status -eq 'Running' }
```

```
Get-Service | ? { $_.Status -eq 'Running' }
```

Расширенный синтаксис позволяет комбинировать несколько критериев с помощью логических операторов **-and** и **-or**. Вот пример:

```
Get-EventLog -LogName Security -Newest 100 |  
    Where { $PSItem.EventID -eq 4672 -and $PSItem.EntryType -eq  
        'SuccessAudit' }
```

## Перечисление результатов

Команда **ForEach-Object** выполняет перечисление. Он имеет два общих псевдонима: **ForEach** и **%**. Подобно Where-Object, ForEach-Object имеет базовый синтаксис и расширенный синтаксис.

В базовом синтаксисе вы можете запустить один метод или получить доступ к одному свойству объектов, которые были переданы в команду. Вот пример:

```
Get-ChildItem -Path C:\Encrypted\ -File | ForEach-Object -MemberName  
    Encrypt
```

В этом синтаксисе вы не добавляете круглые скобки после имени члена, если этот элемент является методом. Поскольку базовый синтаксис должен быть коротким, его часто используют без имени параметра `-MemberName` и с псевдонимом вместо полного имени команды. Например, обе следующие команды выполняют одно и то же действие:

```
Get-ChildItem -Path C:\Encrypted\ -File | ForEach Encrypt
```

```
Get-ChildItem -Path C:\Encrypted\ -File | % Encrypt
```

Расширенный синтаксис для перечисления обеспечивает большую гибкость и функциональность, чем базовый синтаксис. Вместо того, чтобы позволить вам получить доступ к одному члену объекта, вы можете запустить весь скрипт. Этот скрипт может включать только одну команду или может включать в себя множество команд в последовательности.

Например, чтобы зашифровать набор файлов с помощью расширенного синтаксиса, введите следующую команду в консоли и нажмите клавишу «Ввод»:

```
Get-ChildItem -Path C:\ToEncrypt\ -File | ForEach-Object -Process {  
    $PSItem.Encrypt() }  
}
```

Команда Windows PowerShell может использовать только один из своих параметров при каждом запуске, чтобы указать для него входной объект. При передаче данных из одной команды в другую вам не нужно указывать этот параметр. Это упрощает чтение всего командного выражения. Однако может быть неясно, как команда работает при запуске команды.

Когда вы соединяете две команды в конвейере, привязка параметра к конвейеру берет вывод первой команды и решает, что с ней делать. Процесс выбирает один из параметров второй команды для получения этого вывода. У Windows PowerShell есть два метода, которые он использует для принятия такого решения. Первый метод — тот, который Windows PowerShell всегда пытается использовать первым — называется **ByValue**. Второй метод называется **ByPropertyName**, и он используется только тогда, когда **ByValue** терпит неудачу.

Если вы прочитаете полную справку для команды, вы можете увидеть возможности ввода конвейера для каждого параметра. Например, в файле справки для `Sort-Object` вы найдете следующую информацию:

```
-InputObject <PSObject>  
    Specifies the objects to be sorted.  
    To sort objects, pipe them to Sort-Object.  
    Required?                false  
    Position?                Named  
    Default value            None  
    Accept pipeline input? true (ByValue)  
    Accept wildcard characters? false
```

Атрибут `Accept pipeline input?` Имеет значение `true`, следовательно, параметр `-InputObject` принимает вход из конвейера. Кроме того, Справка показывает список методов, поддерживаемых параметром. В этом случае он поддерживает только технику **ByValue**.

При передаче данных с помощью **ByValue** параметр может принимать полные объекты из конвейера, если эти объекты имеют тип, который принимает

параметр. Одна команда может иметь более одного параметра, принимающего входной поток ByValue, но каждый параметр должен принимать объект другого типа.

Windows PowerShell распознает два обобщенных объекта: Object и PObject. Параметры, которые принимают такие объекты, могут принимать любые объекты. Когда вы выполняете привязку параметров конвейера ByValue, Windows PowerShell сначала ищет наиболее подходящий тип объекта. Если конвейер содержит строку, а параметр может принимать String, этот параметр получит объекты.

Если нет определенного типа данных, Windows PowerShell попытается сопоставить общие типы данных. Это поведение объясняет, почему работают команды Sort-Object и Select-Object. Каждая из этих команд имеет параметр с именем -InputObject, который принимает объекты типа PObject из конвейера ByValue. Вот почему вы можете передавать любые типы объектов этим командам. Их параметр -InputObject получит любой объект из конвейера, потому что он принимает объекты любого типа.

Если Windows PowerShell не может связывать вход конвейера с помощью метода ByValue, он пытается использовать технику ByPropertyName. При использовании метода ByPropertyName Windows PowerShell пытается сопоставить свойство объекта, переданного параметру команды, которой был передан объект. Это совпадение происходит простым образом. Если входной объект имеет свойство Name, он будет сопоставляться с параметром Name, потому что они написаны одинаково. Однако он будет передавать свойство только в том случае, если параметр запрограммирован на принятие значения по имени свойства. Это означает, что вы можете передавать выходные данные из одной команды в другую, когда они логически не объединяются.

#### 4.3.4. Функции

Когда вы создаете много скриптов, у вас будут фрагменты кода, которые вы хотите использовать повторно. У вас также будут фрагменты кода, которые вы хотите повторно использовать в одном скрипте. Вместо того, чтобы один и тот же код появлялся несколько раз в скрипте, вы можете создать функцию, которая вызывается несколько раз. Если вам нужно использовать один и тот же код для нескольких сценариев, вы можете поместить функцию в модуль, который может использоваться несколькими сценариями.

Итак, функция представляет собой блок многоразового кода. Вы можете использовать функции для выполнения повторяющихся действий внутри скрипта, а не для того, чтобы один и тот же код в скрипте несколько раз. То есть, используется та же концепция функции, что и в большинстве языков программирования.

Когда вы вызываете функцию, вы можете передавать ей данные. Вы используете блок **Param ()** для функции так же, как и для скрипта. После объявления функции вставьте блок **Param ()** и определения для любых переменных, которые, как ожидается, будут переданы функции. Следующий пример — это функция, которая использует блок **Param ()** для принятия имени компьютера:

```
Function Get-SecurityEvent {
    Param (
        [string]$ComputerName
    ) #end Param
    Get-EventLog -LogName Security -ComputerName -$ComputerName
        -Newest 10
}
```

Чтобы вызвать функцию внутри скрипта, используйте следующий синтаксис:

```
Get-SecurityEvent -ComputerName LON-DC1
```

В приведенном выше примере значение параметра -Computer передается переменной \$Computer в функции. Затем Get-EventLog запрашивает последние 10 событий из журнала безопасности этого компьютера и отображает их на экране. Если вы хотите, чтобы эти события были помещены в переменную и доступны для последующего использования в остальной части скрипта, используйте этот синтаксис:

```
$securityEvents = Get-SecurityEvent -ComputerName LON-DC1
```

Интуитивно, вы бы предположили, что переменная с именем \$computer, которую вы установили в функции, может быть доступна в скрипте, когда функция будет завершена. Однако это не так. Переменные имеют определенную область действия и ограничены в том, как они взаимодействуют между областями.

Ниже описаны три области и способы их использования.

**Глобальная область действия** — это уровень подсказки Windows PowerShell. Переменные, установленные в приглашении Windows PowerShell, могут быть прочитаны во всех сценариях, запущенных в этом сеансе Windows PowerShell.

Переменные, созданные в подсказке Windows PowerShell, не существуют в других сессиях Windows PowerShell или в экземплярах Windows PowerShell ISE.

**Область сценария** предназначена для одного сценария. Переменные, установленные внутри скрипта, могут быть прочитаны всеми функциями внутри этого скрипта.

Если вы установите значение переменной в области сценария, которая уже существует в глобальной области, в области сценария создается новая переменная. В двух отдельных областях есть две переменные с тем же именем. В этот момент, когда вы просматриваете значение переменной в скрипте, возвращается значение переменной в области сценария.

**Область действия функции** предназначена для одной функции. Переменные, установленные внутри функции, не используются совместно с другими функциями или скриптом.

Если вы установите значение переменной в области функций, которая уже существует в глобальной или сценарии, в области функций создается новая переменная. В двух отдельных областях могут быть две переменные с одним и тем же именем.

Помимо чтения переменной в области более высокого уровня, вы также можете изменить эту переменную, указав конкретную область действия переменной при

ее изменении. Чтобы изменить переменную области сценария из функции, используйте следующий синтаксис:

```
$script:var = "Изменено из функции"
```

Если данные в функции находятся в переменной, вы можете использовать **Return()**, чтобы передать ее в скрипт.

Ниже приведен пример использования функции Return () в конце функции для передачи значения переменной обратно в область сценария:

```
Return($users)
```

*Примечание. Использование функции Return () в функции добавляет указанные данные в конвейер возвращаемых данных, но не заменяет существующие данные в конвейере. В рамках разработки скриптов вам необходимо точно проверить, какие данные возвращаются функцией.*

### 4.3.5. Модули PowerShell

Вы можете создавать модули для хранения функций. После того, как вы добавите свои функции в модули, они будут доступны, как и командлеты. Кроме того, как и модули, входящие в состав Windows, модули, которые вы создаете, загружаются автоматически, когда требуется функция.

Во многих случаях у вас уже есть свои функции в файле сценария Windows PowerShell. Чтобы преобразовать этот файл сценария в модуль, переименуйте его с расширением файла .psm1. Никаких структурных изменений в файле не требуется.

Windows PowerShell использует переменную окружения \$PSModulePath для определения путей, из которых загружаются модули. В Windows PowerShell 5.0 используются следующие пути:

- C:\Users\UserID\Document\WindowsPowerShell\Modules
- C:\Program Files\WindowsPowerShell\Modules
- C:\Windows\System32\WindowsPowerShell\1.0\Modules

*Примечание. Если вы храните модули в C:\Users\UserID\Document\WindowsPowerShell\Modules, они доступны только одному пользователю.*

Модули не размещаются непосредственно в каталоге модулей. Вместо этого вы должны создать подпапку с таким же именем, что и файл модуля, и поместить файл в эту папку. Например, если у вас есть модуль с именем AdatumFunctions.psm1, вы поместите его в папку C:\Program Files\WindowsPowerShell\Modules\AdatumFunctions.

Dot sourcing — это метод импорта другого скрипта в текущую область. Если у вас есть файл сценария, содержащий функции, вы можете использовать символ точки для загрузки функций в память в командной строке Windows PowerShell. Обычно, когда вы запускаете файл сценария с функциями, функции удаляются из памяти, когда скрипт завершается. Когда вы используете символ точки, функции остаются в памяти, и вы можете использовать их в командной строке Windows PowerShell. Вы также можете использовать символ точки в сценарии для импорта контента из другого сценария.

Источник с точкой может загружаться из локального файла или по сети с использованием пути UNC (Universal Naming Convention). Ниже приведен синтаксис для использования источника с точкой:

```
. C:\scripts\functions.ps1
```

В свое время, источник с точкой был единственным методом, позволяющим поддерживать централизованный репозиторий функций, которые можно было бы повторно использовать в нескольких сценариях. Тем не менее, модули являются более стандартизированным и предпочтительным методом для поддержки функций в сценариях.