Windows PowerShell — это новая командная оболочка Windows, разработанная в первую очередь для системных администраторов. Она включает интерактивную командную строку и среду исполнения сценариев, которые можно использовать вместе или по отдельности. В Windows PowerShell реализована новая концепция командлетов — простых, узко специализированных средств командной строки, встроенных в оболочку.

Windows PowerShell позволяет системным администраторам автоматизировать большинство рутинных задач. С ее помощью можно менять настройки, останавливать и запускать сервисы, а также производить обслуживание большинства установленных приложений. Воспринимать синее окошко как еще один интерпретатор команд было бы неправильно. Такой подход не отражает сути предложенных корпорацией Microsoft инноваций.

Возможности Windows PowerShell:

- Менять настройки операционной системы;
- Управлять службами и процессами;
- Настраивать роли и компоненты сервера;
- Устанавливать программное обеспечение;
- Управлять установленным ПО через специальные интерфейсы;
- Встраивать исполняемые компоненты в сторонние программы;
- Создавать сценарии для автоматизации задач администрирования;
- Работать с файловой системой, реестром Windows, хранилищем сертификатов и т.д.

Отличие Cmd от Power Shell

PowerShell — это мощный интерфейс командной строки и среда сценариев, которая позволяет выполнять сложные сценарии для простого и эффективного выполнения задач администрирования системы Windows, а CMD — интерфейс командной строки, который предоставляет пользователю набор команд для взаимодействия с компьютером для выполнения задач.

Разница между cmd и PowerShell: не только в цветах.

Более могущественная, и столь же усложнённая. Но по сравнению с cmd она может гораздо больше, и это на фоне того факта, что практически всё, что может cmd, прокатит и в PowerShell. Однако для работы с shell придётся использовать уже отдельный вид команд, которые, дабы на русском не звучало косноязычно, переводчики назвали «командлетами». В отличие от cmd, в shell команды исполняются

по нескольким каналам. Это значит, что выход одной команды (в смысле, командлета) может быть одновременно входом в другую. И всё потому, что командлеты PowerShell — это вполне себе определённые объекты, представители конкретной структуры данных. Даже те командлеты, которые встречают в ответе shell на запрос пользователя. Выражаясь языком программистов, PSH — объектно-ориентированный, а cmd обрабатывает только символы или последовательность символов. Проще говоря, PowerShell позволяет работать с некоторыми программами изнутри, в режиме реального времени, интерактивно. Cmd, в сущности, может только запускать утилиты, которые в Windows уже существуют (почти все они в папке C:\Windows).

Более того, PowerShell — это вполне себе законченная среда для написания и исполнения скрипта. Так что можно создавать очень сложные и объёмные скрипты для управления системой, чем те, на какие была способна консоль cmd.

Основная разница между PowerShell и cmd в том, что последняя — это обновлённая версия «отжившей» в своё время программной оболочки DOS, а в первую, как видно, Windows вдыхает новую жизнь. Очевидно демонстрируя, что от DOS команд разработчики отказываться вообще не собираются. Сравнение с DOS уже неверно, та очень ограничена в своих возможностях; cmd существовала в Windows как «наследие DOS для избранных» или ремонтный, прямой вариант самых необходимых команд. А ввод в работу PowerShell — это как своеобразное предложение, если не покопаться во внутренностях системы, то поучаствовать в изучении её возможностей и способ заняться её модификацией вполне официально: интеграция в среду .Net тому подтверждение.

Знакомство с PowerShell.

Итак, на компьютерном языке Shell — это пользовательский интерфейс, который позволяет получить доступ к службам самой ОС-и. Этот shell может иметь и вариант консоли (как мы его знаем), а может иметь и более дружелюбный интерфейс — оформленный в окна с кнопочками, по которым можно тыцкать мышкой (PowerShell ISE). Windows PowerShell работает именно с ОС от Microsoft последних версий, основан на среде .NET framework. Первая версия Windows PSH появилась почти сразу после выпуска Windows XP, в ноябре 2006 г. Его основное призвание — автоматизация ежедневных задач и процессов, запускаемых администратором системы.

Можно выделить несколько первых команд, с помощью которых можно познакомиться с оболочкой в общих чертах (а не просто смотреть на пустую консоль с синюшним фоном).

Get-help

Справка как она есть. Русифицирована. Информацию о командах получаем отсюда.

Get-Command

Основные командлеты PSH. Сама по себе командлет вам не даст ничего, однако для начала можно было бы ознакомиться с её именем и исполнительным модулем. Например, если нужно увидеть командлеты, работающие с IP функциями или просто сетевые командлеты, мы можем попросить у get-command:

Get-Command –Name *IP*

Командлетов сотни, а может и тысячи. Понимая это, Microsoft ввела в работу целый сервис, который позволяет в режиме онлайн компоновать атрибуты команд в командлеты и проверять их работоспособность. Знакомьтесь: Windows PowerShell Command Builder.

Работа с журналами событий Под событием в Windows понимается любой значительный инцидент в операционной системе или приложении, о котором пользователи (чаще всего администраторы операционной системы) должны быть извещены. Как пра-вило, подобное извещение производится путем записи соответствующей информации в специальные журналы событий (Event Logs). События и журналы событий являются важными инструментами администрирования, так как с их помощью можно следить за состоянием операционной системы и исполняемых в ней приложений, анализировать возникающие проблемы. Кроме того, грамотно построенная политика безопасности информационных систем на платформе Windows обязательно должна обеспечивать протоколирование в журналах событий всех наиболее критичных с точки зрения информационной безопасности событий (например, попыток несанкционированного доступа). Ясно, что администратору системы необходимо иметь простые, гибкие и удобные средства для анализа информации, хранящейся в журналах событий. Также очень важно правильно настроить максимальный размер и режим хранения журналов событий. Журналы могут перезаписываться (новые события будут заменять старые) при достижении определенного размера, однако это может привести к потере важной информации. Лучше отвести для каждого журнала достаточное количество дискового пространства и периодически архивировать и очищать эти журналы, сохраняя тем самым информацию и освобождая дисковое пространство.

На компьютере, работающем под управлением Windows, запись событий производится **в следующие журналы.**

- **Журнал безопасности (Security Log)** содержит события, которые генерируются в том случае, когда на компьютере настроен аудит. Записи о событиях в журнале безопасности делятся на два типа:

• успехи (Success events) указывают на то, что действие, для которого был настроен аудит, завершилось успешно (например, пользователь 276 Часть II. Используем PowerShell успешно зарегистрировался в сети или успешно получил доступ к файлу на общедоступном ресурсе);

• отказы (Failure events) протоколируют, что попытка выполнить действие, для которого был настроен аудит, завершилась безуспешно (например, пользователь попытался зарегистрироваться, однако ввел неправильный пароль, попытался обратиться к сетевому ресурсу, не имея соответствующих разрешений на доступ, и т. п.).

- **Журнал системы (System Log**) хранит события, которые были сгенерированы в результате действий операционной системы (например, запуск служб, сбои в работе драйверов, изменения роли сервера с рядового члена до контроллера домена и т. д.). Записи о событиях системы делятся на три типа:
  • уведомления (Information events) просто описывают произведенные действия (например, успешный запуск самой службы Event Log, установление удаленного

подключения и т. п.). Некоторые уведомления также содержат информацию о сбоях при выполнении действий, которые реально не влияют на сетевые операции;

• предупреждения (Warning events) содержат информацию о событиях, которые могут стать источником различных проблем (например, сбой динамической регистрации DNS-имен из-за неправильной настройки клиента DNS, сбой службы Windows Time Service при поиске контроллера домена, нехватка пространства на диске и т. п.). Реагировать на подобные предупреждения следует как можно более оперативно;

• ошибки (Error events) сохраняют информацию о критических событиях, которые могут привести к потерям данных или другим серьезным проблемам (сбой при инициализации рабочей станции, отказ в динамическом обновлении со стороны сервера DNS, сбой драйвера устройства и т. п.).

- **Журнал приложений (Application Log)** предназначен для записи событий, сгенерированных запущенными на компьютере приложениями. Для генерации подобных событий разработчики должны вставлять определенный код в свои приложения. Как правило, события приложений оказываются полезными лишь в том случае, когда вы отсылаете информацию об этих событиях разработчикам для решения возникающих проблем. Тем не менее, в журнале приложений также сохраняются некоторые системные события Windows, например, события, возникающие в результате сбоя приложений (они записываются программой Dr. Watson), события, связанные с групповой политикой, нарушения ограничений криптографического экспорта для протокола IPSec, действия IIS, связанные с работой Active Server Pages (ASP), и т. д. События в журнале приложений также делятся на уведомления, предупреждения и ошибки.

  В зависимости от того, какие дополнительные компоненты Windows установлены на компьютере, в системе кроме трех основных журналов могут вестись дополнительные журналы событий:
  - журнал службы каталога (Directory service log), в который записывается информация о действиях Active Directory. Располагается журнал на контроллерах домена Windows;
  - журнал сервера DNS (DNS server log), где сохраняется информация о действиях сервера DNS; журнал службы репликации файлов (File Replication Service log), в котором сохраняются данные о действиях службы File Replication Service (FRS) на тех машинах, где настроена служба DFS (Distributed File System, распределенная файловая система).

  Во всех трех перечисленных журналах записи о событиях делятся на уведомления, предупреждения и ошибки. Кроме этого, собственный журнал событий ведет и сама оболочка Windows PowerShell. В этом журнале регистрируются запуск и остановка интерпретатора PowerShell, а также некоторые ошибки, приводящие к завершению сеанса работы оболочки.

Оболочка и среда разработки

Существует Windows PowerShell в двух ипостасях: помимо эмулятора консоли с командной оболочкой есть интегрированная среда сценариев (Integrated Scripting Environment — ISE). Чтобы получить доступ к интерфейсу командной строки достаточно выбрать соответствующий ярлык в меню Windows или запустить powershell.exe из меню «Выполнить». На экране появится синее окошко, заметно отличающееся по возможностям от допотопного cmd.exe. Там есть автодополнение и другие фишки, привычные пользователям командных оболочек для Unix-систем.

Windows PowerShell ISE является полноценной средой разработки с поддерживающим вкладки и подсветку синтаксиса редактором кода, конструктором команд, встроенным отладчиком и другими программистскими радостями. Если в редакторе среды разработки после имени команды написать знак дефис, вы получите в выпадающем списке все доступные параметры с указанием типа. Запустить PowerShell ISE можно либо через ярлык из системного меню, либо с помощью исполняемого файла powershell_ise.exe.

Командлеты

В Windows PowerShell появились т.н. командлеты (cmdlets). Это специализированные классы .NET, в которые заложена разнообразная функциональность. Именуются они по принципу «Действие-Объект» (или «Глагол-Существительное, если вам так больше нравится), а разделенная дефисом связка напоминает сказуемое и подлежащее в предложениях естественных языков. Например, Get-Help буквально означает «Получить-Помощь» или в контексте PowerShell: «Показать-Справку».

Помимо Get в командлетах для обозначения действий используются и другие глаголы (и не только глаголы, строго говоря). В списке ниже мы приведем несколько примеров:

`Add` — добавить;
`Clear` — очистить;
`Enable` — включить;
`Disable` — выключить;
`New` — создать;
`Remove` — удалить;
`Set` — задать;
`Start` — запустить;
`Stop` — остановить;
`Export` — экспортировать;
`Import` — импортировать.

Есть системные, пользовательские и опциональные командлеты: в результате выполнения все они возвращают объект или массив объектов. К регистру они не чувствительны, т.е. с точки зрения интерпретатора команд нет разницы между Get-

Help и get-help. Для разделения используется символ ‘;’, но ставить его обязательно только если в одной строке выполняется несколько командлетов.

По умолчанию команда отображает краткую справку, но в командлеты при необходимости передаются параметры (аргументы). С их помощью можно, например, получить детальную (параметр -Detailed) или полную (параметр -Full) справку, а также вывести на экран примеры (параметр -Examples):

```
Get-Help Get-Command -Examples
```

Справка в Windows PowerShell обновляется командлетом Update-Help. Если строка команд получается слишком длинной, аргументы командлета можно перенести на следующую, написав служебный символ ‘`’ и нажав Enter — просто закончить писать команду на одной строке и продолжить на другой не получится.

Ниже приведем несколько примеров распространенных командлетов:

`Get-Process` — показать запущенные в системе процессы;
`Get-Service` — показать службы и их статус;
`Get-Content` — вывести содержимое файла.

Сценарии, функции, модули и язык PowerShell

Скрипты Windows PowerShell хранятся в виде обычных текстовых файлов с расширением .ps1. Запустить их двойным кликом нельзя: нужно правой кнопкой мыши вызвать контекстное меню и выбрать пункт «Запустить в PowerShell». Из консоли придется либо указать полный путь к скрипту, либо перейти в соответствующий каталог и написать имя файла. Запуск сценариев также ограничен системной политикой, а для проверки текущих настроек можно использовать командлет Get-ExecutionPolicy, который выдаст одно из следующих значений:

`Restricted` — запуск сценариев запрещен (по умолчанию);
`AllSigned` — разрешен только запуск подписанных доверенным разработчиком сценариев;
`RemoteSigned` — разрешен запуск подписанных и собственных сценариев;
`Unrestricted` — разрешен запуск любых сценариев.

Пишутся скрипты на объектно-ориентированном языке программирования, команды которого именуются по тому же принципу, что и рассмотренные ранее командлеты: «Действие-Объект» («Глагол-Существительное»). Основное его предназначение — автоматизация задач администрирования, но это полноценный интерпретируемый язык, в котором есть все необходимые конструкции: условный переход, циклы, переменные, массивы, объекты, обработка ошибок и т.д. Для написания сценариев годится любой текстовый редактор, но удобнее всего запустить Windows PowerShell ISE.

Конвейеры

В последнем примере мы применили знакомую пользователям оболочек для Unix-систем конструкцию. В Windows PowerShell вертикальная черта также позволяет передать выход одной команды на вход другой, но в реализации конвейера есть и существенная разница: речь здесь идет уже не о наборе символов или каком-то тексте. Встроенные командлеты или пользовательские функции возвращают объекты или массивы объектов, а также могут получать их на входе. Как в Bourne shell и его многочисленных последователях, в PowerShell с помощью конвейера упрощается выполнение сложных задач.

Простейший пример конвейера выглядит так:

```
Get-Service | Sort-Object -property Status
```

Сначала выполняется командлет Get-Service, а потом все полученные им службы передаются на сортировку по свойству Status командлету Sort-Object. В какой именно аргумент передается результат работы предыдущего участка конвейера, зависит от его типа — обычно это InputObject. Подробнее этот вопрос будет рассматриваться в посвященной языку программирования PowerShell статье.

При желании цепочку можно продолжить и передать результат работы Sort-Object еще одному командлету (выполняться они будут слева направо). Кстати, пользователям Windows доступна и привычная всем юниксоидам конструкция для постраничного вывода:

```
Get-Service | Sort-Object -property Status | more
```

Запуск задач в фоновом режиме

Довольно часто бывает нужно запустить некую команду в фоне, чтобы не дожидаться результата ее выполнения в сессии оболочки. В Windows PowerShell есть несколько командлетов на такой случай:

`Start-Job` — запуск фоновой задачи;
`Stop-Job` — остановка фоновой задачи;
`Get-Job` — просмотр списка фоновых задач;
`Receive-Job` — просмотр результата выполнения фоновой задачи;
`Remove-Job` — удаление фоновой задачи;
`Wait-Job` — перевод фоновой задачи обратно в консоль.

Удаленное выполнение команд

Windows PowerShell позволяет выполнять команды и сценарии не только на локальном, но и на удаленном компьютере и даже на целой группе машин. Для этого существует несколько способов:

- У многих командлетов есть параметр `-ComputerName`, но таким способом не получится, например, создать конвейер;
- Командлет `Enter-PSSession` позволяет создать на удаленной машине интерактивный сеанс;
- С помощью командлета `Invoke-Command` можно выполнять команды или сценарии на одном или нескольких удаленных компьютерах.