Windows Remote Management

Kirill Nikolaev

MCSE, MCITP

TOC

Windows Remote Management: Overview

Overview of remote management technologies in Windows-based infrastructure.

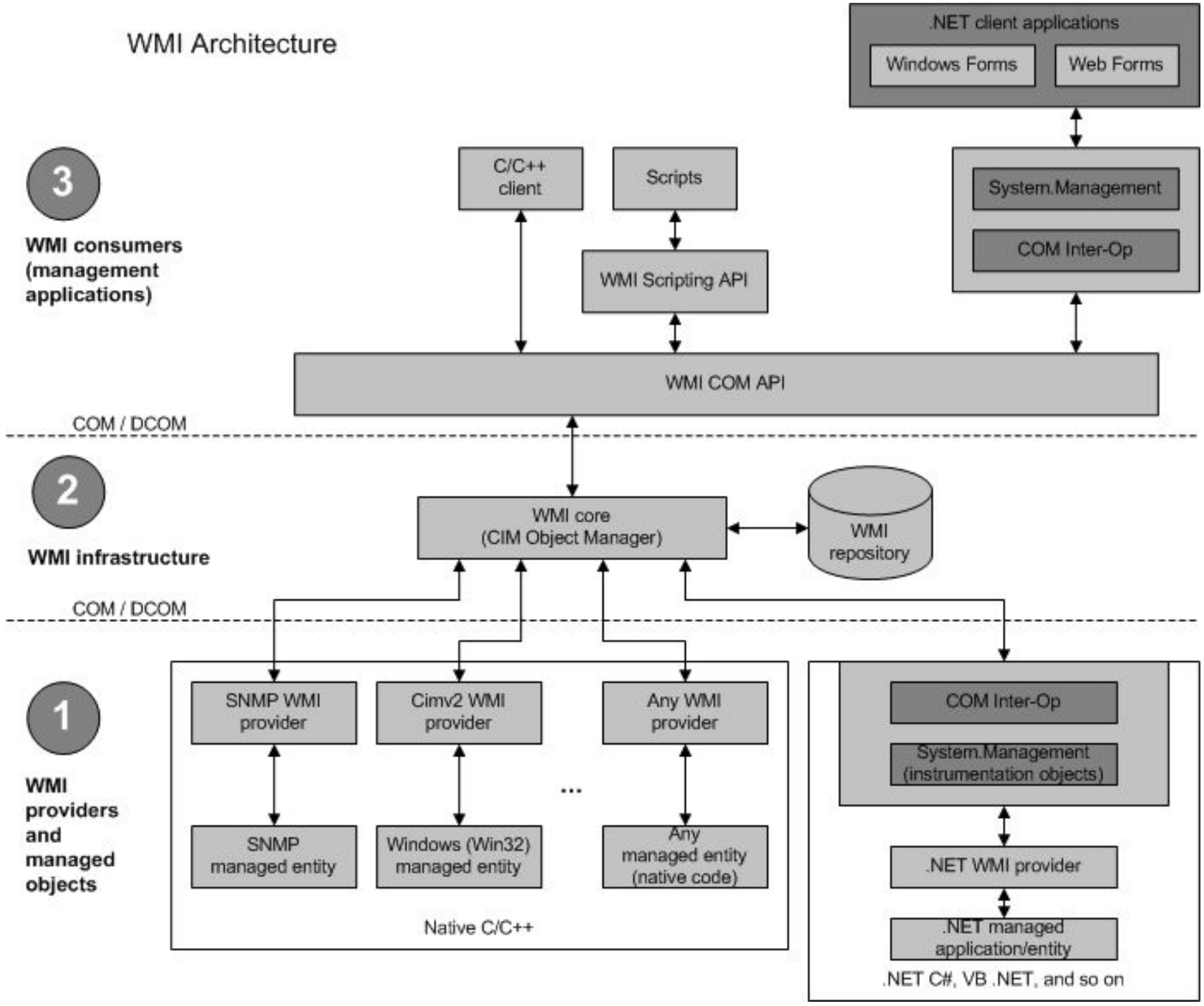
Windows Management Instrumentation

- Одна из первых технологий Windows для управления локальным и удалёнными компьютерами (NT5.0+).
- WMI частная реализация Web-Based Enterprise Management (WBEM)
- WBEM – стандартная технология доступа к информации в корпоративных средах, частная реализация Common Information Model (CIM).
- CIM – описывает управляемые элементы ИТ-инфраструктуры и их связи. (http://dmtf.org/standards/cim)
- CIM, WBEM, WMI, в случае Windows, – одно и то же.

Что можно сделать при помощи WMI:

- Управлять локальными дисками, службами, системным журналом и т.п.
- Управлять сетевыми настройками: IP-адрес, использование DHCP, DNS-серверы.
- Получение информации для мониторинга состояния системы: место на диске
- Получение информации о конфигурации системы: имя компьютера, объём RAM, установленная ОС и обновления.
- Получение конфигурации установленных приложений: SCCM, Exchange, SQL Server.
- etc.

WMI Architecture



Управляемые ресурсы

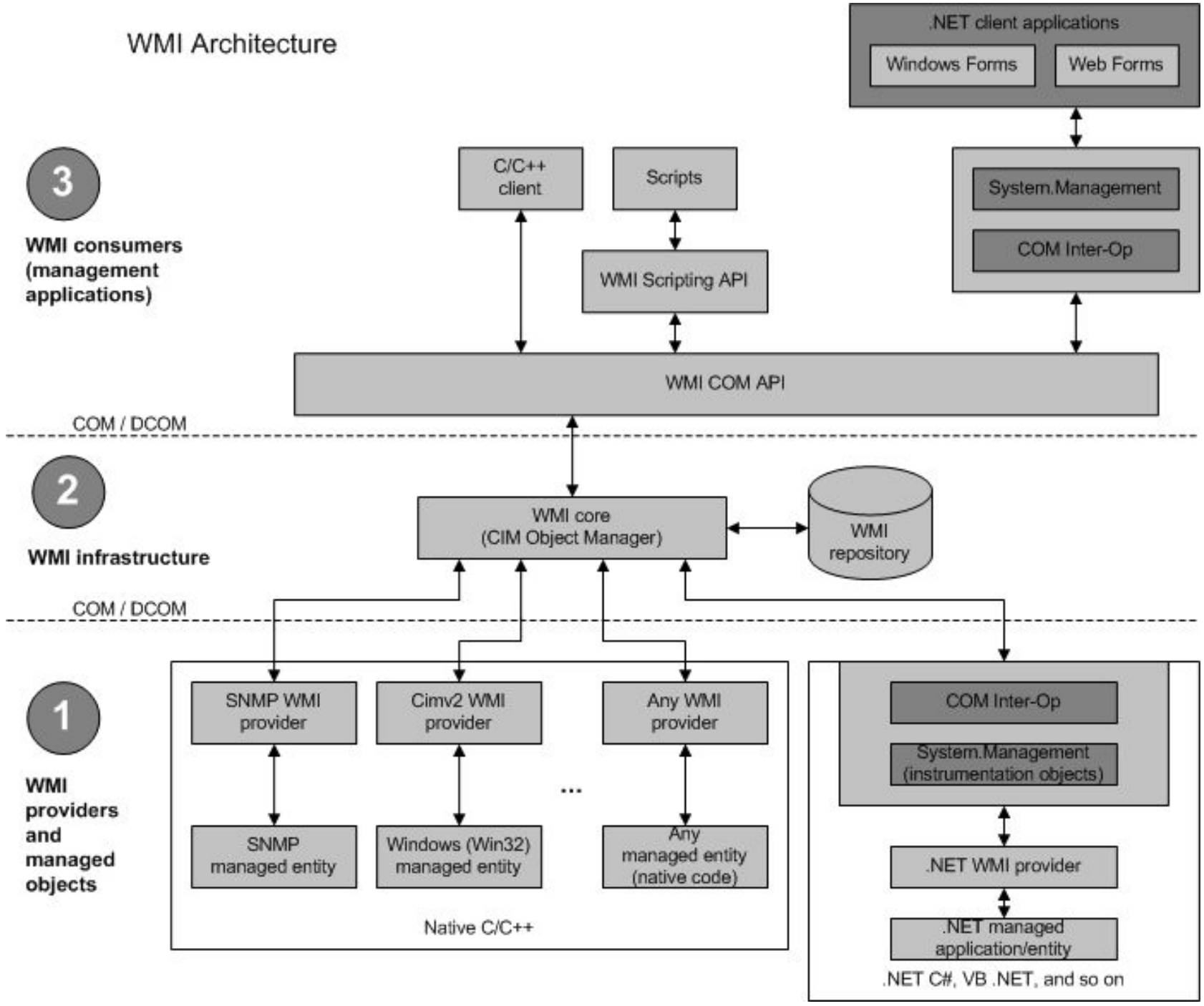
Любой компонент системы или установленного приложения:

- Локальные диски
- Журналы ОС
- Службы ОС
- SQL Server
- SCCM
- Exchange
- Сама система (глобальные свойства)
- Принтеры
- Общие папки
- Оборудование

Классы WMI

- Каждый управляемый ресурс принадлежит к какому-либо классу.
- Класс – описание свойств ресурса и доступных методов (команд).
- Примеры:
  - Win32_LogicalMemoryConfiguration
  - Win32_Service
  - Win32_NTLogEvent
  - Exchange_Mailbox
  - CCM_SoftwareDistributionClientConfig

WMI Architecture



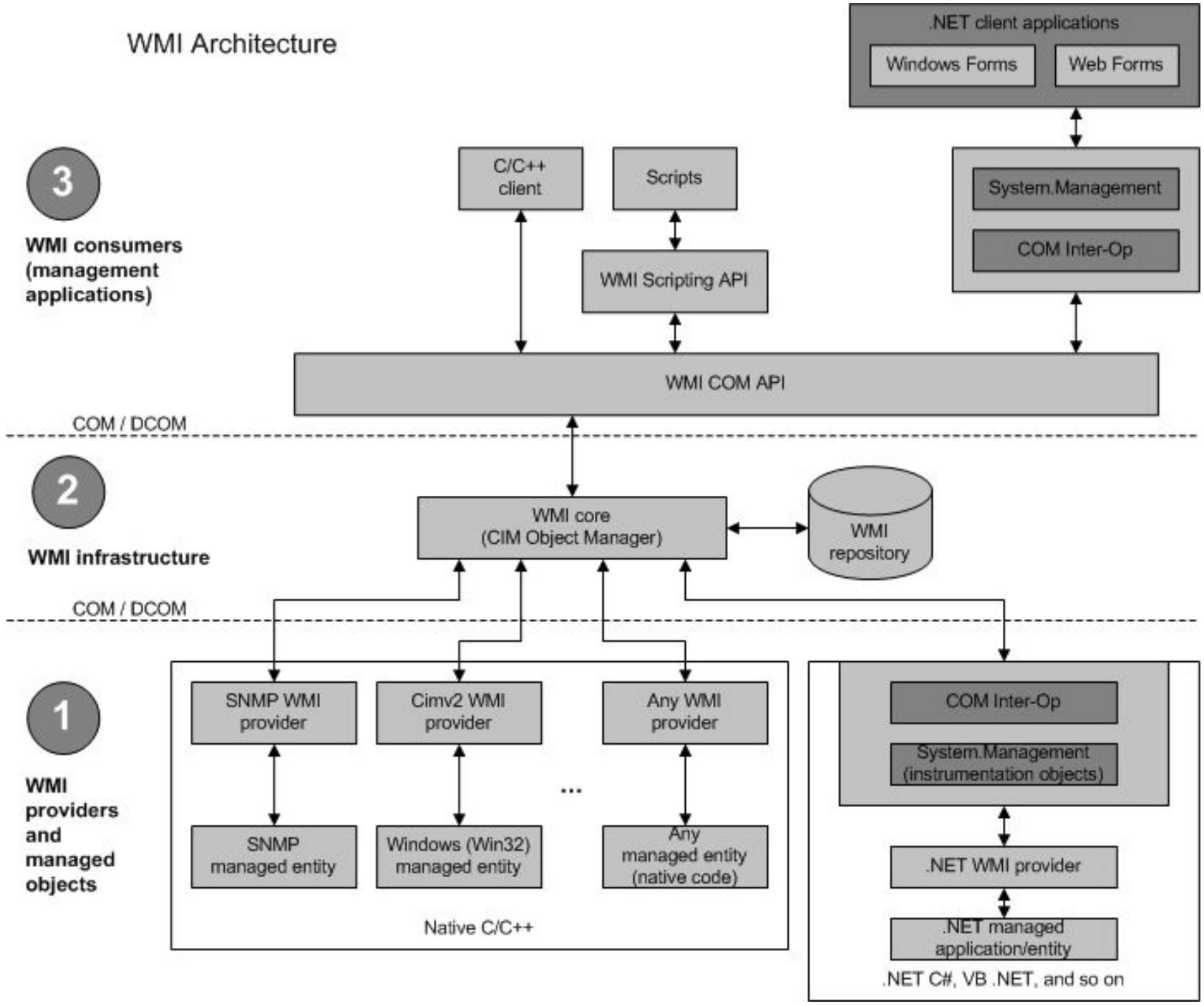
Провайдеры WMI

- Каждый ресурс имеет свой собственный API
- В WMI используется стандартная модель доступа.
- Провайдер транслирует запросы между службой WMI и управляемыми ресурсами.
- Поэтому, провайдеры напоминают драйверы.
- Провайдер может представлять:
  - один класс (Registry – StdRegProv)
  - несколько классов (Win32 - Win32_Process, Win32_LogicalDisk etc.)

Примеры провайдеров

Provider	DLL	Namespace	Description
Active Directory	dsprov.dll	root\directory\ldap	Maps Active Directory objects to WMI
Event Log	ntevt.dll	root\cimv2	Manages Windows event logs (for example, reads, backs up, clears, copies, deletes, monitors, renames, compresses, and uncompresses event log files and changes event log settings)
Performance Counter	wbemperf.dll	root\cimv2	Provides access to raw performance data
Registry	stdprov.dll	root\default	Reads, writes, enumerates, monitors, creates, and deletes registry keys and values
SNMP	snmpincl.dll	root\snmp	Provides access to SNMP MIB data and traps from SNMP-managed devices
WDM	wmiprov.dll	root\wmi	Provides access to information about WDM device drivers
Win32	cimwin32.dll	root\cimv2	Provides information about the computer, disks, peripheral devices, files, folders, file systems, networking components, operating system, printers, processes, security, services, shares, SAM users and groups, and more
Windows Installer	msiprov.dll	root\cimv2	Provides access to information about installed software
Exchange Server		root\MicrosoftExchangeV2	
SCCM		root\CCM	

WMI Architecture

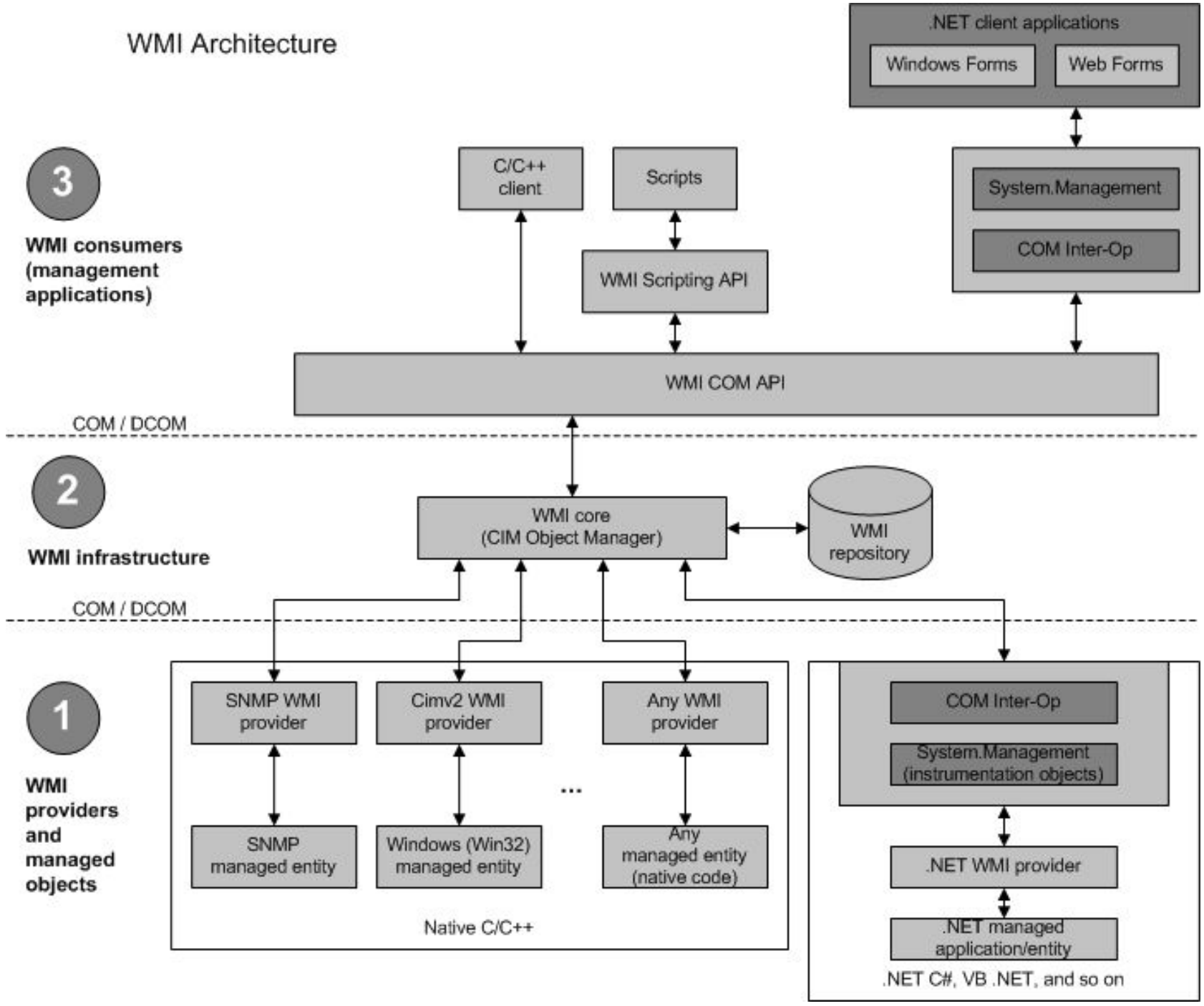


Инфраструктура WMI

1. Служба WMI (winmgmt)
   - Обеспечивает взаимодействие между провайдерами, репозиторием и приложениями.

2. WMI-репозиторий
   - Организован в виде namespaces (root\default, root\cimv2)
   - Namespaces используются для разграничения доступа (ala папки в ф. с.)
     - http://wutils.com/wmi/namespaces.html
   - Хранит только статические данные (описания классов)
   - Физически - %SYSTEMROOT%\System32\wbem\

WMI Architecture



Как получить доступ к WMI?

Для программистов: COM API (WMI Component Object Model (COM) API), Microsoft.Management.Infrastructure (C#)

Для администраторов:

1. GUI
   - WMI Explorer
   - wbemtest.exe
   - WMI Administrative Tools
   - Scriptomatic 2.0
   - Coretech WMI and PowerShell Browser

http://goo.gl/sySC5o

Как получить доступ к WMI?

2. CLI
   - wmic
   - PowerShell

3. Scripting:
   - VBScript - Scripting API for WMI (http://goo.gl/EWt23b)
   - PowerShell - Get-WmiObject, Get-CimInstance

Demo

WMI Explorer

Вопросы?

Архитектура WMI, общие концепции.

Example: VBS – Total Visible Memory

```
strComputer = "."

Set objWMIService = GetObject("winmgmts:\\" & strComputer & _
"\root\cimv2")
Set colItems = objWMIService.InstancesOf("Win32_OperatingSystem")

For Each objItem In colItems
        Wscript.Echo "Total Physical Memory (KB): " & _
        objItem.TotalVisibleMemorySize
Next
```

Example: VBS – Installed Updates

```
strComputer = "."

Set objWMIService = GetObject("winmgmts:\\" & strComputer)
Set colItems = objWMIService.ExecQuery(_
        "Select * from Win32_QuickFixEngineering")

For Each objItem in colItems
        Wscript.Echo "HotFixID: " & objItem.HotFixID
        Wscript.Echo "Caption: " & objItem.Caption
        Wscript.Echo "Description: " & objItem.Description
        Wscript.Echo "InstalledOn: " & objItem.InstalledOn
Next
```

Example: VBS – Disable User

```
strComputer = "."

Set objWMIService = GetObject("winmgmts:\\" & strComputer & _
"\root\cimv2")
Set colItems = objWMIService.ExecQuery(_
   "SELECT * FROM Win32_UserAccount WHERE Name = 'User'")

For Each objItem In colItems
        objItem.Disabled = TRUE
        objItem.Put_()
Next
```

Example: VBS – Restart Service

```
strComputer = "."

Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
Set colServices = objWMIService.ExecQuery _
   ("SELECT * FROM Win32_Service where Name = 'Themes'")
For Each objService In colServices
    Return = objService.StopService()
    If Return <> 0 Then
        Wscript.Echo "Failed " & VBNewLine & "Error code = " & Return
    Else
       WScript.Echo "Succeeded"
       objService.StartService()
    End If
Next
```

wmic qfe

wmic syntax

- wmic qfe | find "2998527" << external filtering using “find” command

- qfe where HotfixID= "KB2998527" << built-in filtering,
  strict compliance only, works in CLI only

- wmic memorychip get Capacity << clear output
  while using property’s name

- wmic **path** win32_QuickFixEngineering get Hotfixid << full path
  w/o usage of aliases

Example: wmic – Rich output

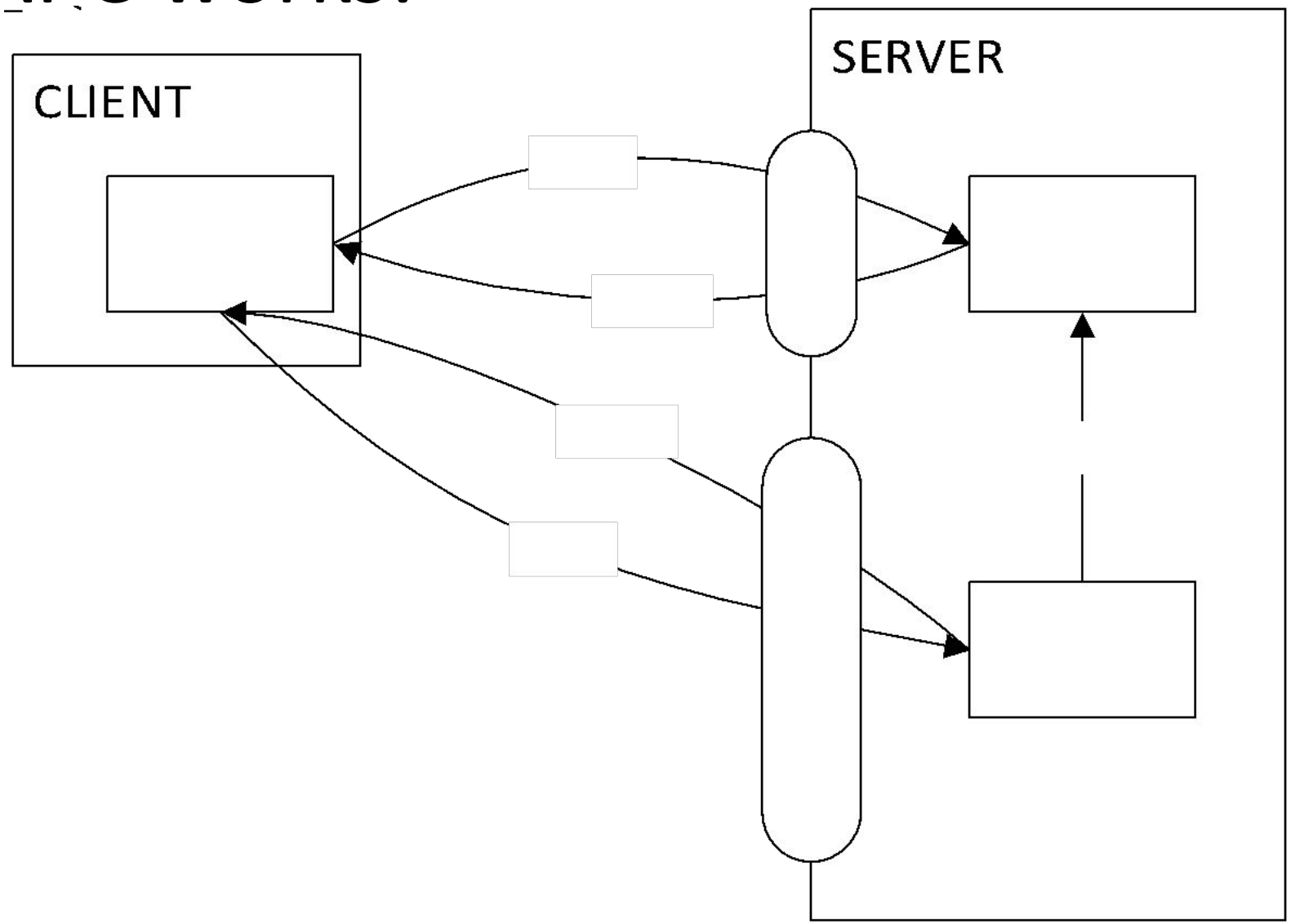
wmic /output:C:\temp\CPU1.htm cpu get Name, MaxClockSpeed, NumberOfCores, SocketDesignation /format:hform

Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz.	
Property Name	Value
MaxClockSpeed	3401 .
Name	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz.
NumberOfCores	4 .
SocketDesignation	LGA1155.
.	

DCOM (distributed component object model)

- Используется WMI для удалённого подключения.
- Состоит из 2-х частей:
  - COM - стандартная модель Microsoft для взаимодействия приложений друг с другом
  - RPC (Remote Procedure Calls) – технология взаимодействия клиент-серверных приложений. Может вызывать удалённо функции, передавать объекты и т.п.

How RPC works:



Example: VBS – Remote DHCP Enable

```
strComputer = “CLT1.exchange12rocks.net“

Set objWMIService = GetObject(_
    "winmgmts:\\" & strComputer & "\root\cimv2")
Set colNetAdapters = objWMIService.ExecQuery _
    ("Select * from Win32_NetworkAdapterConfiguration " _
        & "where IPEnabled=TRUE")

For Each objNetAdapter In colNetAdapters
    errEnable = objNetAdapter.EnableDHCP()
Next
```

Demo

wbemtest

WinRM

- Частная реализация WS-Management
- Единственный порт – 5985/6 (HTTP/S)
- Активация:
  - 2003-2008 R2: winrm qc
  - 2012+: включён по умолчанию

Как использовать WinRM?

- Удалённая командная строка:
  - winrs -r:<ServerName> cmd.exe
- Запуск службы:
  - winrm invoke StartService wmicimv2/Win32_Service?Name=Themes
- Перезагрузка:
  - winrm invoke reboot wmicimv2/Win32_OperatingSystem -r:<ServerName>
- Информация о системе:
  - [xml]$osInfo = winrm get wmicimv2/Win32_OperatingSystem /format:pretty
  - $osInfo.Win32_OperatingSystem

Вопросы?

Работа с WMI при помощи VBScript, wmic, WinRM.

Перерыв

15 минут.

PowerShell: Quick Overview

What is PowerShell?

- Script language
- Command-line interface with auto-completion
- Available as built-in from Windows Vista
- Object-oriented – result of each command is an **object** but no text string
- “Verb-Noun” system of commands (**cmdlets**)
  - Get-Process
  - Stop-Service
  - Set-Mailbox
- Easily extensible

PS primitives

- Pipeline – transfers objects between commands:
  - Get-Process mmc.exe | Stop-Process
- Variable – text string starting with “$” sign:
  - $Counter = 10
  - $Files = Get-ChildItem -*Path* C:\temp\
- “this” variable ($_) – contains current object:
  - 1, 2, 3 | ForEach-Object {echo ($_+5)}
- Properties – each object is described by one or many properties:
  - $Files.Count
- Methods – most objects have methods to execute:
  - $Files.GetType()

PS Aliases

Short aliases exist for some of the built-in cmdlets:

- where -> Where-Object
- cd -> Set-Location
- man -> help

Main cmdlets

- Get-Help (help)
- Get-Command
- Get-Member
- Select-Object (select)
- Get-Content (gc)
- ForEach-Object (foreach, %)
- Write-Output (echo)
- Where-Object (where)

Comparison operators

-eq - Equal to. Includes an identical value.

-ne - Not equal to. Includes a different value.

-gt - Greater-than.

-ge - Greater-than or equal to.

-lt - Less-than.

-le - Less-than or equal to.

-Like - Match using the wildcard character (*).

-NotLike - Does not match using the wildcard character (*).

-Match - Matches a string using regular expressions.

-NotMatch - Does not match a string. Uses regular expressions.

-Contains - Tells whether a collection of reference values includes a single test value.

-NotContains

-In

-NotIn

-Replace - Replace operator. Changes the specified elements of a value.

Complex Example

```
$Files | where {$_.LastWriteTime -gt '01.01.2010'} | select Name, Length
```

PowerShell: Remoting

Cmdlets: CIM vs. WMI

- Get-WmiObject:
  - PowerShell 2.0
  - DCOM/RPC
- Get-CimInstance:
  - PowerShell 3.0
  - WS-Man/HTTP(S)
  - Improved compatibility (non-Windows systems, down-level OS)

Example: PS – OLD

```
$Service = Get-WmiObject -Query "SELECT * FROM
Win32_Service WHERE Name = 'Themes'"

$Return  = $Service.ChangeStartMode("Manual")

if ($Return.ReturnValue -eq 0) { "Success" }
else { "$($Return.ReturnValue) was reported" }
```

Example: PS – NEW

```
$Return = Invoke-CimMethod -Query "SELECT * FROM
Win32_Service WHERE Name = 'Themes'"
-MethodName 'ChangeStartMode'
-Arguments @{StartMode = 'Manual'}

if ($Return.ReturnValue -eq 0) { "Success" }
else { "$($Return.ReturnValue) was reported" }
```

Remote-enabled PowerShell-cmdlets

Get-WmiObject

Remove-WmiObject

Invoke-WmiMethod

Register-WmiEvent

Set-WmiInstance

Get-EventLog

Show-EventLog

New-EventLog

Remove-EventLog

Clear-EventLog

Limit-EventLog

Get-WinEvent

Stop-Computer

Restart-Computer

Get-Service

Set-Service

Get-Process

Get-Counter

Get-HotFix

PowerShell Remoting

1. Произвольные команды PS на удалённых компьютерах.
   Invoke-Command

2. Полноценная удалённая сессия PowerShell
   -PSSession

PS Remoting – minimum requirements

1. Windows XP SP3
2. .NET Framework 2.0 SP1
3. Windows Management Framework
    1. Windows PowerShell 2.0
    2. Windows Remote Management (WinRM) 2.0

PS Remoting - activation

- Enable-PSRemoting
  - Enabled by default on Windows 2012 and later.

- Remote activation:
  - http://gallery.technet.microsoft.com/scriptcenter/Enable-PSRemoting-Remotely-6cedfcb0

- Network ports: 5985 (HTTP), 5986 (HTTPS) (same as WinRM)

PS Remoting – выполнение команд

Invoke-Command -ComputerName SRV1, SRV2 -ScriptBlock {Get-Process}

- ComputerName принимает любой список PowerShell на вход
  - (Get-Content C:\Scripts\Servers.txt)
- ScriptBlock принимает как один командлет (с параметрами или без), так и несколько сразу.
  - {Get-Process mmc | Stop-Process},
  - {$myScript}

- -FilePath {C:\Scripts\TestScript.ps1}

PS Remoting – RunAs

Invoke-Command … -Credential:

1. (Get-Credential)
2. $cred, где $cred = Get-Credential

PS Remoting - Sessions

- Command completion works even if cmdlets aren't installed at your box.
- Get-Help, Get-Command works against remote cmdlet set.
- Less typing, commands are shorter – same as you'd run them locally.

PS Remoting – Session cmdlets

- Enter-PSSession
- Exit-PSSession

- Permanent sessions for Invoke-Command cmdlet:
  1. $S = New-PSSession $ComputerName
  2. Invoke-Command -Session $S -ScriptBlock {Start-Job -ScriptBlock {$Script}}

PS Remoting – Background Jobs

1. Run command as a job:

   Invoke-Command SRV1 -ScriptBlock {(Get-ChildItem C:\ -Recurse).Count} -AsJob

```
Id     Name            PSJobTypeName   State         HasMoreData     Location
--     ----            -------------   -----         -----------     --------
2      Job2            RemoteJob       Running       True            SI-SRV1
```

2. Grab the result:

   Get-Job *-Id 2* | **Receive-Job**

Useful for long operations, especially with multiple computers.

Вопросы

PowerShell

Windows Remote Management: Tools

Tools, which are useful to any network administrator in Windows-based infrastructure.

Administrative shares

- “Hidden” networks share
  - Its name ends with “$” sign. Windows Explorer and “net view” command don’t show such network shares.
- One for each logical volume:
  - C$, D$, E$ etc.
- admin$ - %SYSTEMROOT%
- print$ - contains printer objects
- ipc$ - not a part of a file system. Used for inter-process communication

By default, accessible by administrators only.

MMC

- Microsoft Management Console – GUI which hosts many administrative tools to manage your machines locally and remotely.
- Installed at each Windows PC starting from NT4.0
- Many snap-ins ship separately
  - Remote Server Administration Tools
  - Exchange Management Console
  - DPM Administration Console
  - Kaspersky Security Center

MMC snap-ins

- Standard Microsoft snap-ins located in “Control Panel\All Control Panel Items\Administrative Tools”
- Most useful for you – “Computer Management”
- You can create your own set of snap-ins and save as a single file

Remote registry

- Depends on “Remote Registry” service

- Use common regedit.exe tool
  - File -> Connect network registry

Built-in command-line tools

- tasklist/taskkill
  - /s
- shutdown
  - /m
- netsh
  - -r
- w32tm
  - /computer

Sysinternals PsTools

- PsExec - execute processes remotely
- PsFile - shows files opened remotely
- PsGetSid - display the SID of a computer or a user
- PsInfo - list information about a system
- PsPing - measure network performance
- PsKill - kill processes by name or process ID
- PsList - list detailed information about processes
- PsLoggedOn - see who's logged on locally and via resource sharing (full source is included)
- PsLogList - dump event log records
- PsPasswd - changes account passwords
- PsService - view and control services
- PsShutdown - shuts down and optionally reboots a computer
- PsSuspend - suspends processes

Вопросы?

Любые по рассмотренным темам.

Мои контакты

- Все-все контакты и соцсети:
  - http://about.me/exchange12rocks
- Мой технический блог:
  - http://exchange12rocks.org