**Можно ли полагаться на данные, извлекаемые WMI классами?**

*На примере нескольких WMI-классов Win32\_xxxx, показано, что как минимум некоторые из свойств объектов, возвращаемые указанными классами, совершенно не соответствуют реальным значениям этих свойств.*

Ниже рассматриваются примеры выдач WMI-классами Win32\_CDROMDrive, Win32\_DiskDrive, Win32\_DiskPartition, Win32\_LogicalDisk, Win32\_Volume и демонстрируется, что выдаваемая информация, мягко говоря, не соответствует действительности или выдается в очень неудобном формате или просто опускается. В качестве стенда для выполнения WQL-запросов к WMI-классам используется программа ScriptomaticV2.hta, разработанная группой разработчиков “Scripting Guy” фирмы Microsoft. ScriptomaticV2.hta поддерживает VBS, JS, Perl, Python. В силу отсутствия двух последних, использовались лишь выдачи JS и VBS. Поэтому заранее приношу извинения разработчикам упомянутых WMI-классов, если Perl и / или Python выдают корректную информацию. Кроме того, надеюсь, что использование тех же самых WMI-классов в программах на C#, C++ обеспечивает корректную выдачу результатов.

1. **Неверная или опускаемая дата создания или монтирования объекта**

Все перечисленные выше классы в списке свойств содержат свойство, именуемое “InstallDate”. Этому свойству в Windows Registry соответствует REG\_QWORD. В среде Windows 8.1+ (8.1 и всех сборок Windows 10) CDROMDrive, DiskDrive, это значение хранится в InitialTimeStamp, адресуемом

HKLM\SYSTEM\CurrentControlSet\Enum\SCSI\<PnPDeviceID>\Device Parameters\StorPort

Значение InitialTimeStamp МОЖЕТ принимать значение NULL, но в большинстве случаев оно содержит вполне реальную дату в формате, прекрасно описанном в документации Microsoft:

“*Unique value that indicates the time when an event is generated. This is a 64-bit*

*FILETIME value that represents the number of 100-nanosecond intervals after January 1, 1601.*

*The information is in the Coordinated Universal Time (UTC) format. This property is inherited*

*from \_\_Event. To convert this value to other time formats, use the SWbemDateTime methods*

*SetFileTime and GetFileTime.*”

Видимо разработчики Win32-WMI-классов не захотели привлекать еще и методы *SWbemDateTime* и потому выдача JS-кода в поле InstallDate всегда подставляет null date, выдача VBS-кода — это поле скромно опускает. Но ведь никакие методы не нужны, для того, чтобы 64-битное QWORD число, описанного выше формата, преобразовать в число, представляющее дату в миллисекундах, на основе которого можно создать объект Date. Для этого потребуется элементарная арифметика.

2. **Противоречивые данные, ошибка в поле InterfaceType, Win32\_DiskDrive**

Index: 0

InstallDate: null date *должно быть Sun Jul 19 08:12:08 UTC+0300 2015*

InterfaceType: IDE

.....................

Model: ST9320423AS

Name: \\.\PHYSICALDRIVE0

.....................

PNPDeviceID: SCSI\DISK&VEN\_&PROD\_ST9320423AS\4&3516B3B5&0&000000

Кто-нибудь может дать разумное объяснение значению поля InterfaceType, прямо противоречащее как указанному ниже PNPDeviceID так и тому факту, что начиная с Windows 8.1 в HKLM\SYSTEM\CurrentControlSet\Enum\ интерфейс IDE вообще отсутствует и последний раз он присутствовал лишь в среде Windows 7?

3. **Отрицательные значения в полях SerialNumber, Signature классами Win32\_DiskDrive, Win32\_Volume**

SerialNumber: Hitachi HT100219FCC400NEJTBU5G

Signature: -1498719820

....................

Purpose: null

QuotasEnabled: false

QuotasIncomplete: false

QuotasRebuilding: false

SerialNumber: -1408044882

Это вообще любимая “фишка”, выдача очень важной информации в совершенно неудобоваримом формате. И SerialNumber и Signature все привыкли читать в 16-ном виде.

4. **Неверная индексация разделов для Gpt-форматированных дисков, класс Win32\_DiskPartition**

Caption: Disk #3, Partition #0

................................

Description: GPT: Basic Data

Ну как, объясните мне, Gpt-форматированный раздел типа “Basic Data” приобрел индекс 0, который по праву принадлежит MSR-разделу? Естественно, что MSR-раздел Win32\_DiskPartition в списке разделов вообще холодно игнорирует.

5. **Наличие полей, содержащих NULL для JS или пусто для VBS для всех объектов в WQL-запросе.**

В качестве примера могу привести значения свойств Access и Description для класса Win32\_Volume. У меня на 9 смонтированных устройствах, включающих два внутренних диска, пару виртуальных DVD-ROM, несколько VHD с Gpt-форматированными разделами, USB-диском, SD-картой и флэшкой, расположены 16 томов и все они в указанных полях содержат или NULL для JS-выдачи или пусто для VBS-выдачи. Спрашивается каким же должен быть том, на каком устройстве располагаться для того, чтобы для него хоть одно из указанных свойств оказалось значимым? А если такого быть не может, то зачем вообще включать эти свойства в выдачи?

Решил проверить с помощью PS и “gwmi win32\_volume” для всех томов выдала пустые значения свойств Access, Description.

**Заключение.**

Основной смысл этой короткой, возможно излишне эмоциональной публикации состоит в том, чтобы привлечь внимание всех, кто использует WMI-классы в своем коде, критически оценивать извлекаемые данные, перепроверять их из других источников. Сразу же добавлю, что MSFT-классы, аналогичные описанным выше Win32-классам, выдают всегда корректную информацию, хотя и они грешат выдачей отрицательных чисел в полях Signature.