# ВВЕДЕНИЕ

Любая технология, следящая за состоянием компьютера, так или иначе использует счетчики производительности. Напрямую общаться с провайдерами счетчиков производительности дело утомительное и совсем не нужное, когда давно разработана не одна технология по управлению доступом к значениям счетчиков производительности. Каждая технология старается максимально облегчить программисту работу по написанию кода для доступа к значениям счетчиков производительности. Иногда такое упрощение ведет к длительной работы по их выборке, либо за простотой скрывается огромный механизм, который непосильно изучить в деталях. Поэтому важно изучить все технологии до определенной глубины и выбрать наиболее подходящую конкретному случаю — мониторингу работы серверов с ежесекундной записью значений всех счетчиков производительности со всех выбранных серверов.

# О счетчиках производительности

Счетчики производительности — это расширяемый механизм сбора различной информации, заложенный в операционной системе линейки Windows [1].

Начиная с ОС Windows NT 4.0 и до сих пор, разработали четыре технологии по работе со счетчиками производительности [5]:

1. первоначальный оригинальный интерфейс, производит запрос к реестру непосредственно;
2. начиная с версии Windows NT 4.0, компания Microsoft сопровождает интерфейс Performance Data Helper (PDH);
3. начиная с Windows 2000, существует механизм Windows Management Instrumentation (WMI), который используется стандартным монитором производительности Windows — System Monitor;
4. начиная с четвертой версии платформы .NET существуют классы пространства имен System.Diagnostics.PerformanceData, относящиеся к API .NET Framework. Классы работают поверх первоначального оригинального интерфейса.

Вышеописанный список технологий расположен в порядке их появления. Сравнение технологий проводится по критериям: количество найденных провайдеров и время получения всех найденных счетчиков производительности.

# Технологии получения значений счетчиков производительности

## Технология WMI

Distributed Management Task Force (DMTF) является некоммерческой ассоциацией участников отрасли, занимающаяся продвижением управления и взаимодействия предприятий и систем. Web-Based Enterprise Management (WBEM) и Common Information Model (CIM) являются одними из разработанных DMTF стандартов.

Идея WBEM заключается в создании универсального интерфейса мониторинга и управления системами и компонентами распределенной информационной среды предприятия, используя объектно-ориентированную идеологию и широко распространенные веб-технологии: XML и HTTP. WMI является адаптацией Microsoft стандарта WBEM.

В основе WBEM лежит стандарт CIM, отвечающий за реализацию объектно-ориентированной парадигмы WBEM [6].

Поскольку WMI построена по объектно-ориентированному принципу, то все данные о компьютере, операционной системе и ее свойствах, представлены в виде объектов. Каждый тип объекта описан классом, в состав которого входят свойства и методы.

## Первоначальный оригинальный интерфейс

Получить значения счетчика производительности можно только программным путем через Win32 API-функции реестра типа RegQueryValueEx, открыв специальный раздел с именем HKEY\_PERFORMANCE\_DATA.

Интерфейс, осуществляемый через функцию RegQueryValueEx — наиболее базовый, то есть всего одна функция без дополнительных уровней абстракций, в отличие от других интерфейсов.

### Схема получения данных счетчиков производительности через функцию RegQueryValueEx

На рис. 3. представлена схема получения данных счетчиков производительности через функцию RegQueryValueEx.

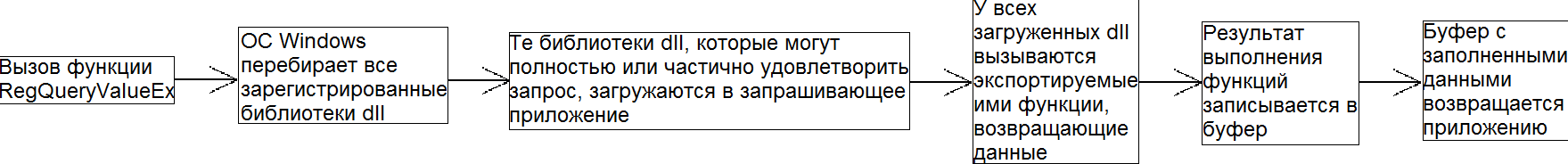


Рис. 3. Схема получения данных счетчиков производительности через функцию RegQueryValueEx

Функция RegQueryValueEx вызывается с соответствующими параметрами.

Далее управление передается ОС Windows, которая перебирает все зарегистрированные библиотеки, пути к которым хранятся в реестре: HKLM\SYSTEM\CurrentControlSet\Services\<имя приложения>\Performance значение Library [3].

Dll-библиотеки, содержащие счетчики производительности запроса, загружаются в запрашиваемое приложение.

Далее у каждой библиотеки вызываются необходимые функции для экспорта из них данных о счетчиках производительности.

После вызова всех необходимых функций всех библиотек, заполненный данными буфер возвращается приложению.

## NET пространство имен System.Diagnostics

Среди классов пространства имен System.Diagnostics. PerformanceData есть класс PerformanceCouterСategory, который является шаблоном объекта производительности. Функции данного класса используют функции реестра Win32 API.

Среди прочих методов, класс PerformanceCouterСategory содержит метод ReadCategory, который считывает все счетчики и данные экземпляра объекта производительности. После выполнения данного метода возвращается коллекция InstanceDataCollectionCollection.

Наглядный пример структуры коллекции InstanceDataCollectionCollection представлен в [10].

2.4. Performance Data Helper

Данная технология позволяет абстрагироваться от работы с реестром и, посредством встроенных функций PDH работать с запросами на получение указанных в запросе счетчиков производительности. Позволяет получать значения счетчиков в отформатированном виде.

2.5. Отбора технологий для сравнения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии | Технологии | | | |
| Реестр | PDH | WMI | API .NET |
| Удаленный доступ к счетчикам | есть | есть | есть | есть |
| Автоматическое форматирование значений счетчиков | нет | есть | есть | Есть |
| Язык разработки | C | C | PowerShell | C# |
| Минимальные системные требования | Любая ОС Windows | Windows NT 4.0 | Windows 2000 | Windows 7, .NET Framework 4.0 |
| Количество уровней абстракций |  |  |  |  |

# Различия в количестве провайдеров счетчиков производительности, у различных технологий

Точное число провайдеров счетчиков производительности сложно определить, так как технологии работают по различным алгоритмам с различными проверками на корректность провайдеров.

Разрабатывая сложную программу, разработчики, как правило, заботятся о мониторинге занимаемых программой ресурсов. Для того, чтобы счетчики производительности обнаруживались стандартной утилитой мониторинга производительности необходимо зарегистрировать провайдера этих счетчиков производительности. В случае, если регистрация прошла некорректно с точки зрения конкретной технологии, то она будет отсутствовать в списке провайдеров только для этой технологии, в то время, как другая технология обнаруживает этого провайдера.

Таким образом, в идеале, количество провайдеров для всех технологий должно совпадать, но на практике картина совсем иная.

В приложении 1 представлен текст программы выборки всех провайдеров счетчиков производительности, используя три различные технологии.

Результат работы программы представлен в таб. 1.

*Таблица 1*

Количество провайдеров, найденных технологиями, у серверов DIRECTUM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя сервера | Классы PerformanceData System.Diagnostics | Реестр | WMI | PDH |
| Сервер WorkFlow | 139 | 167 | 141 |  |
| Сервер обработки событий | 145 | 174 | 147 |  |
| Файловое хранилище | 138 | 167 | 140 |  |
| Сервер сеансов | 138 | 167 | 140 |  |
| Сервер DCTS | 152 | 181 | 154 |  |
| Сервер распознавания документов | 118 | 136 | 130 |  |
| Сервер назначенных заданий | 139 | 167 | 141 |  |
| В среднем: | 138,3 | 165,6 | 141,9 |  |

Как видно из таб. 1, наибольшее число провайдеров были идентифицированы стандартными функциями доступа к реестру. На втором месте — технология WMI. Наименьшее число идентифицированных провайдеров у метода GetCategories класса PerformanceCounterCategory

Наименьшее число идентифицированных провайдеров у технологий, лежащих поверх стандартных функций доступа в реестр. Такая тенденция происходит из-за внутренних особенностей идентификации провайдеров, которые могут забраковать некоторых провайдеров, не отвечающих их внутренним стандартам.

# Различия в правах доступа на чтение счетчиков производительности у различных технологий

В данной главе речь пойдет о реализации удаленного доступа к компьютерам у различных технологий.

## Технология WMI

Для интеграции, приложение должно зарегистрировать свои классы в существующей модели CIM и обеспечить стандартные вызовы встроенных процедур для создания объектов этих классов и наполнения их свойствами и методами. Набор этих процедур оформляется как WMI Provider — специальная библиотека, являющаяся мостом между любым приложением (управляемой сущностью) и ядром службы WMI (WMI Core или CIM Object Manager).

Вся технология WMI нацелена на использование COM-серверов.

Безопасность в WMI можно разбить на три уровня:

1. Уровень протокола DCOM. WMI работает поверх протокола прикладного уровня DCOM. DCOM, в свою очередь, работает поверх RPC. Данный протокол использует 135 порт.
2. Уровень пространств имен. Все счетчики производительности WMI группирует в пространства имен по какому-то определенному признаку. WMI поддерживает безопасность пространств имен, сравнивая токен доступа пользователя, пытающегося подключиться к пространству имен с дескриптором безопасности данного пространства имен.
3. Уровень ОС. ОС Windows использует множество технологий защиты компьютера от нежелательной внешней информации. Брандмауэр является одной из таких технологий, который может разрешить/запретить получать данные с конкретного компьютера и от конкретного пользователя.

Таким образом, для удаленного доступа WMI (создания удаленного процесса) необходимо следующее [11]:

1. быть локальным администратором на удаленной машине;
2. пользователь, под которым выполняется удаленное подключение должен быть в домене и состоять в группе администраторов;
3. оба компьютера должны находится в одном домене или оба домены должны быть в отношении «trust»;
4. иметь пароль на локальной машине;
5. локальная и удаленная машины должны быть в одном домене;
6. у пользователя должны быть права на выполнение и удаленное разрешение пространства имен root\cimv2;
7. группе администраторов должно быть выдано разрешение на удаленное подключение DCOM в утилите DCOMCnfg.exe;
8. минимальный уровень олицетворение пользователя — Impersonate;
9. в брандмауэре должен стоять чек-бокс у Windows Management Instrumentation (WMI) в таблице исключений.

Большой минус WMI еще заключается в том, что технология не всегда возвращает все инстанции некоторых счетчиков производительности из-за фильтрации UAC, когда приложение, использующее WMI, работает под доменным пользователем со всеми вышеперечисленными условиями, но без повышенных привилегий (режим обычного пользователя).

## Функции работы с реестром

Удаленный доступ к реестру регулируется одноименным сервисом. Сервис запускается как часть процесса svchost.exe.

Ключи реестра используют стандартные дескрипторы безопасности Windows.

При попытке получения значения ключа реестра, система сравнивает права доступа запрашивающего пользователя с дескриптором безопасности ключа реестра. Если пользователь не имеет нужных прав для доступа к ключу реестра, то операция чтения значения заканчивается ошибкой.

При успешном доступе к ключам реестра, далее происходит попытка доступа к провайдеру, необходим доступ к данному объекту, который регулируется, как и все объекты Windows, дескриптором безопасности.

## Функция ReadCategory класса PerformanceCounterCategory

Ситуация с регулированием доступа к значениям счетчиков производительности, используя функцию ReadCategory класса PerformanceCounterCategory, аналогична работе стандартных методов доступа к реестру, так как обе технологии используют реестр для поиска провайдеров.

5.4. Performance Data Helper

# Сравнение длительности выборки счетчиков производительности у различных технологий

## Функция ReadCategory класса PerformanceCounterCategory

Длительность выборки значений всех счетчиков производительности без вывода их на консоль представлено в таб. 2. Также в таб. 2 представлено среднее значение длительности выборки по всем серверам, не было использовано среднеквадратичное отклонение, как более точное описание характеристики выборки, так как серверов представлено не много и можно вручную убедиться в точности такой характеристики выборки как среднее значение.

Листинг программы по расчету длительности выборки всех счетчиков производительности с использованием функции ReadCategory класса PerformanceCounterCategory представлен в приложении 2.

*Таблица 2*

Длительность выборки значений всех счетчиков производительности без вывода их на консоль по серверам DIRECTUM с использованием функции ReadCategory класса PerformanceCounterCategory

|  |  |
| --- | --- |
| Имя сервера | Длительность выборки, с |
| Сервер WorkFlow | 8,1 |
| Сервер обработки событий | 7,8 |
| Файловое хранилище | 6,9 |
| Сервер сеансов | 7 |
| Сервер DCTS | 8,7 |
| Сервер распознавания документов | 5 |
| Сервер назначенных заданий | 7,6 |
| В среднем: | 7,16 |

## Технология WMI

Длительность выборки значений всех счетчиков производительности без вывода их на консоль представлено в таб. 3.

Листинг программы по расчету длительности выборки всех счетчиков производительности с использованием технологии WMI представлен в приложении 3.

*Таблица 3*

Длительность выборки значений всех счетчиков производительности без вывода их на консоль по серверам DIRECTUM с использованием технологии WMI

|  |  |
| --- | --- |
| Имя сервера | Длительность выборки, с |
| Сервер WorkFlow | 36 |
| Сервер обработки событий | 36 |
| Файловое хранилище | 36 |
| Сервер сеансов | 33 |
| Сервер DCTS | 36 |
| Сервер распознавания документов | 92 |
| Сервер назначенных заданий | 31 |
| В среднем: | 44 |

Различие в длительности выборки значений счетчиков производительности связано с особенностями архитектуры обеих технологий. Технология WMI использует COM-сервера из-за чего происходит постоянный маршаллинг между COM-сервером и процессом, так как некоторые COM-сервера выполняют свои функции в отдельном от вызвавшего его процессе. Некоторые же провайдеры WMI представляют собой Hi-Perf COM-сервера, которые загружаются в память вызвавшего его процесса, таким образом исключается ресурсозатратный маршаллинг. Также, большая доля времени ушло у технологии WMI на фильтр классов Win32\_PerfFormattedData\_\* для выборки значений только счетчиков производительности.

Получение же значения ключа реестра не требует больших затрат ресурсов. Максимум выполняется стандартная сверка дескрипторов безопасности.

# Выводы по сравнительному анализу

В рамках данного анализа выполнено следующее:

1. Проведено сравнение количества провайдеров разных серверов, используя различные технологии: функции System.Diagnostics, WMI и функции доступа к реестру.
2. Проведено детальное описание работы блоков безопасности у различных технологий.
3. Проведено сравнение длительности выборки всех найденных у всех провайдеров счетчиков производительности без вывода их экран, используя две технологии: методы System.Diagnostics и технологию WMI.

По количеству провайдеров, в среднем, лидирует стандартные методы доступа к реестру. Дать однозначный ответ, что данная технология может собрать значения с большего количества счетчиков, чем остальные технологии нельзя. Возможно, что среди идентифицированных провайдеров каждой технологией есть, непересекающееся с подмножествами провайдеров остальных технологий, подмножество провайдеров, где, например, один провайдер извлекает сравнительно большое количество значений счетчиков производительности и их инстанций. Таким образом, количество идентифицированных провайдеров может и не сыграть роль решающего фактора в выборе подходящей технологии для мониторинга работы серверов DIRECTUM.

В простоте безопасности технологий выигрывают стандартные функции доступа к реестру и функция ReadCategory класса PerformanceCounterCategory. Все три технологии используют дескрипторы безопасности для доступа к объектам. Требование WMI намного обширнее, чем требования остальных технологий. Также, при создании нового процесса на удаленной машине, технология WMI использует стандартные средства защиты Windows: токены доступа.

В сравнении длительности получения значений со всех найденных счетчиков производительности на первом месте — функция ReadCategory класса PerformanceCounterCategory. Технология WMI проиграла по двум причинам:

1. долгая фильтрация WMI-классов для нахождения только классов, содержащих данные об счетчиках производительности;
2. маршаллинг некоторых COM-серверов с вызывающими их процессами, что значительно увеличивает длительность выборки некоторых счетчиков производительности.

# ЗАКЛЮЧЕНИЕ

Были проведены сравнения технологий по трем аспектам: количество идентифицируемых провайдеров, сложность в контроле безопасности объектов технологий и длительность выборки всех счетчиков производительности всех идентифицированных провайдеров.

Можно сделать выводы по каждой из технологий:

1. технологи WMI имеет огромную инфраструктуру — переплетения нитей взаимосвязи COM-серверов, а также высокие требования к безопасности и доступу;
2. стандартные функции доступа к реестру — самый быстрый и чистый (без посредников) метод получения значений счетчиков производительности. Но данные функции заставляют использовать язык Си для работы с ними, что усложняет работу над системой мониторинга серверов DIRECTUM;
3. функция ReadCategory класса PerformanceCounterCategory наиболее оптимальный вариант, так как работают поверх стандартных функций доступа к реестру, а также используют реестр для поиска провайдеров, непосредственно, без сложной инфраструктуры технологии.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТЦРЫ

1. Холодилов, С. Счетчики производительности Часть 1. Чтение / С. Холодилов //The RSDN Group.— 2004.— №4
2. Microsoft Docs RegQueryValueExA function [Электронный ресурс].— Режим доступа: https://docs.microsoft.com/ru-ru/windows/desktop/api/winreg/nf-winreg-regqueryvalueexa
3. Холодилов, С. Счетчики производительности Часть 2. Запись / С. Холодилов //The RSDN Group.— 2004.— №4
4. Microsoft Docs About-performance-counters [Электронный ресурс].— Режим доступа: https://docs.microsoft.com/en-us/windows/desktop/perfctrs/about-performance-counters
5. Odysseas Pentakalos. Windows 2000 Performance Guide / Mark Friedman, Odysseas Pentakalos.— Sebastopol: O’Reilly & Associates, Inc., 2002.— P. 705
6. Distributed Management Task Force counters [Электронный ресурс].— Режим доступа: https://dmtf.org
7. Microsoft Docs https://docs.microsoft.com/en-us/windows/desktop/wmisdk/wmi-architecture
8. Microsoft Docs Access to WMI securable objects https://docs.microsoft.com/en-us/windows/desktop/wmisdk/access-to-wmi-securable-objects
9. Microsoft Docs InstanceDataCollectionCollectionClass https://docs.microsoft.com/ru-ru/dotnet/api/system.diagnostics.instancedatacollectioncollection?view=netframework-4.8
10. Microsoft Docs Security Descriptors counters [Электронный ресурс].— Режим доступа: https://docs.microsoft.com/en-us/windows/desktop/secauthz/security-descriptors