Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA

WMI: Instrumentação e gerenciamento em ambientes distribuídos

por
Francisco K. do Amaral
(francisco.amaral@pucrs.br)
Rodrigo Wesz
(rwesz@yahoo.com)
Marcelo Juchem
(juchem@gmail.com)

Porto Alegre, 11 de Abril de 2007.

1 INTRODUÇÃO

O custo total de manutenção de uma rede distribuída de computadores pode ser muito elevado, dependendo desde o valor e quantidade dos equipamentos, até o treinamento necessário para a manutenção. Por causa disso, várias empresas começaram com iniciativas para diminuir esses custos nas companhias. Entre essas iniciativas está o *Web-Based Enterprise Management* (WBEM), o qual estabelece padrões de gerenciamento de infra-estrutura e provê uma forma de combinar informações de vários sistemas de gerenciamento de hardware e software.

A proposta original do WBEM foi iniciada em 1996 por uma série de companhias lideradas pela Microsoft, Compaq Computer, BMC Software, Cisco Systems e Intel. A visão inicial era definir um ambiente aberto de gerenciamento, onde todos os sistemas de gerenciamento e aplicações poderiam acessar, controlar e compartilhar informações de gerenciamento entre si. De 1996 a 1998 a Microsoft trabalhou para desenvolver uma implementação do WBEM para a plataforma Windows, a qual originou a *Windows Management Instrumentation* (WMI). O WMI é a principal ferramenta para monitoração e gerenciamento de aplicações construídas sobre o sistema operacional Windows da Microsoft (LAVY, 2001).

O WMI está baseado em conceitos como Monitoração e Instrumentação. Monitoração significa estar a par do estado do um sistema, no caso do WMI, do estado de todos os componentes do sistema que são gerenciáveis, sejam eles hardware ou software. Instrumentação nada mais é do que "a arte e a ciência da medida e do controle", ela pode se referir aos métodos de medida e controle e aos instrumentos que facilitam esse trabalho. No caso do WMI, instrumentação se refere aos métodos e medidas disponíveis para o gerenciamento de hardware e software.

O WMI utiliza o *Common Information Model* (CMI), base do WBEM, que define um modelo independente de linguagem utilizado para representar os recursos gerenciados do mundo real através do paradigma de orientação a objetos. Nele os recursos do sistema são representados como objetos gerenciáveis que possuem propriedades que descrevem seus dados e métodos que descrevem seu comportamento (DMTF, 2002).

2 ARQUITETURA WMI

A arquitetura do WMI é formada pelos seus consumidores, pela sua infra-estrutura, pelos provedores e pelos objetos gerenciados (GOLOMSHTOK, 2003). A figura 2.1 ilustra estes componentes.

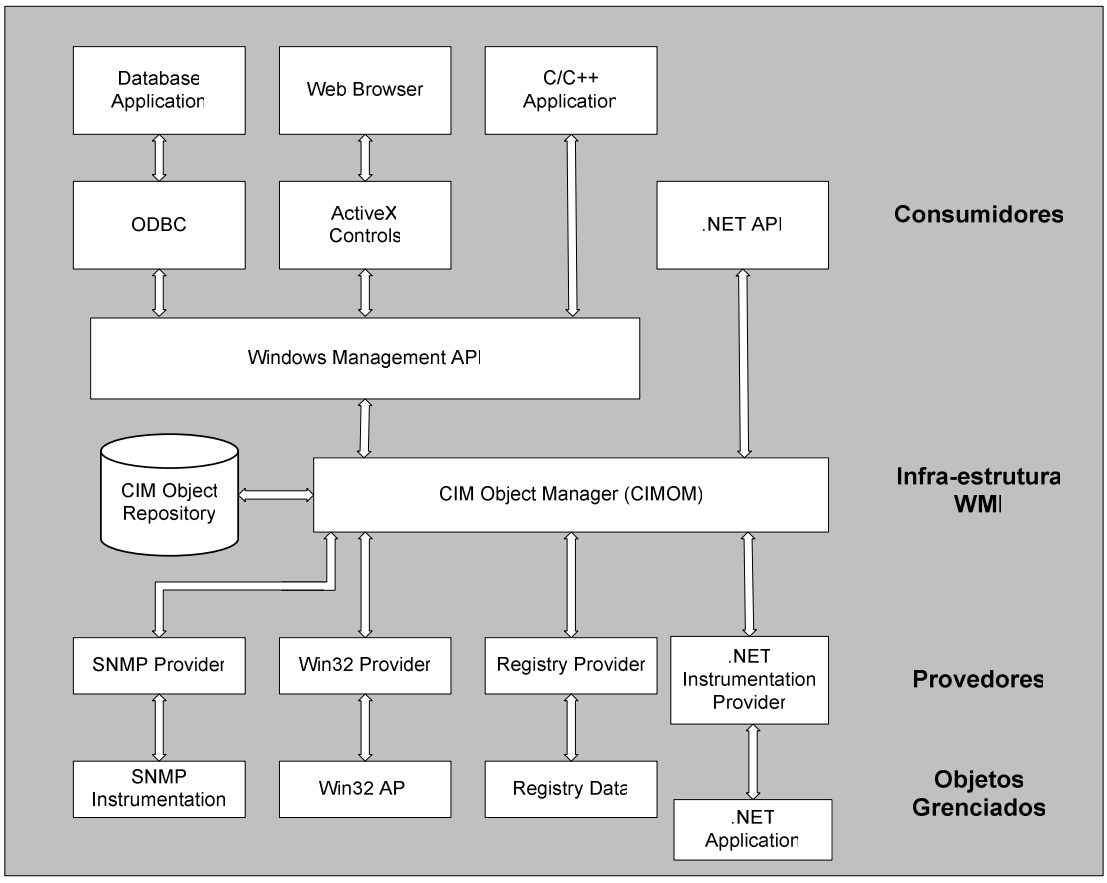

Figura 2.1: Arquitetura do WMI.

Os consumidores são componentes e aplicações que interagem com o WMI para obter informações dos recursos monitorados. Estas informações são coletadas do repositório do WMI ou obtidas pela assinatura de eventos.

A infra-estrutura WMI, oferecida pelo sistema operacional, engloba o CIM *Object Manager* (CIMOM) e o CIM *Object Repository* (CIMOR). O CIMOM manipula a comunicação entre os provedores e as aplicações consumidoras. O CIMOR armazena as informações providas pelos recursos monitorados e as definições dos modelos de dados dos objetos gerenciados.

Os provedores são componentes de software que atuam como intermediários entre o CIMOM e os objetos gerenciados (componentes lógicos ou físicos do sistema). Eles utilizam o serviço oferecido pelo CIMOM para abastecer o repositório com as informações que os consumidores consultarão. O serviço também pode passar estas informações diretamente às aplicações gerenciadoras. Além disso, os provedores manipulam as requisições das aplicações de gerenciamento e geram eventos notificando uma mudança de estado no objeto gerenciado.

A grande vantagem do WMI é sua capacidade de gerenciar informações de sistemas remotos. As aplicações de gerenciamento (ou consumidores) se comunicam com a infra-estrutura WMI usando uma variedade de tecnologias como Visual Basic, C++, ODBC e ActiveX. Todas essas interfaces são baseadas no Component Object Model (COM) ou Distributed COM (para conexões remotas).

3 APLICAÇÃO

O WMI suporta linguagens como C/C++, Visual Basic, e linguagens de script que executam em Windows, tal como VB Script. O WMI pode ser usado na execução de várias tarefas, tais como: monitorar o status de aplicativos, detectar gargalos ou falhas na rede, gerenciar e configurar aplicativos, acessar informações em um ambiente corporativo, automatizar tarefas administrativas, executar operações de gerencia em uma máquina local ou remota (MSDN, 2007). As tarefas de maior destaque são listadas a seguir:

1. Disponibilidade de Aplicativos: Apresenta o status do aplicativo. Se o aplicativo ficou indisponível, por exemplo, o WMI reporta esse estado para o sistema. Além disso, o WMI tem a capacidade de reiniciar a aplicação que apresenta o problema.
2. Métricas de Aplicativos: São métricas baseadas em atributos que permitem monitorar informações sobre os aplicativos.
3. Mensagens e Eventos: As mensagens e eventos podem incluir informações sobre que tipo de alerta elas representam: apenas informação, um aviso ou uma mensagem de erro.
4. Mecanismos de Controle: Controle e configuração dinâmica de aplicativos. Automatização de comandos, *triggers* para eventos, *updates*, etc.

A coleta e gerencia de todas essas informações podem ser feitas através de várias ferramentas, tais como: extensões WMI para o Visual Studio .Net, o navegador de objetos WMI e ferramentas do próprio sistema operacional, como o Windows Administrative Tools e o Microsoft Systems Management Server.

4 UTILIZANDO WMI

Apesar de ser o mais poderoso e sofisticado mecanismo de instrumentação e gerenciamento de aplicações distribuídas disponível no Windows, o WMI, antes do surgimento do .NET, tinha, entre os desenvolvedores, a reputação de ser de difícil utilização. A implementação de provedores, a instrumentação de aplicações e a obtenção de dados do WMI requeriam um conhecimento avançado a respeito da sua infra-estrutura (do esquema de classes do CIM), além de uma profunda experiência com COM ou DCOM. No entanto, atualmente, por ser um componente do sistema operacional, muitas técnicas de monitoração e gerenciamento podem ser implementadas interagindo com o modelo de objetos do WMI através de classes existentes na biblioteca de classes do .NET *Framework*.

O *namespace System.Management.Instrumentation* minimiza o trabalho requerido para tornar as informações de uma aplicação visíveis através da infra-estrutura WMI abstraindo os detalhes desta implementação. Já o *namespace System.Management* permite que aplicações .NET possam acessar informações providas por qualquer fonte instrumentada com WMI (MICROSOFT, 2001).

4.1 Expondo dados via WMI

Expor objetos ou disparar eventos de uma aplicação pode ser intuitivo para desenvolvedores que trabalham com .NET, uma vez que o WMI utiliza um modelo orientado a objetos. Usando atributos do Framework, expõem-se declarativamente eventos e classes. Atributos são uma forma declarativa de se associar informações a um código em .NET. Uma vez associado a alguma entidade do programa o atributo pode ser consultado em tempo de execução. Este método é bastante útil uma vez que exige pouca codificação adicional. Quando uma classe é marcada com os atributos adequados, é estabelecido um mapeamento direto entre as classes da aplicação e o esquema de classes WMI.

Tendo a aplicação instrumentada ela torna-se acessível através do WMI. Seus objetos e eventos podem ser pesquisados, monitorados e configurados. Os eventos marcados para monitoração são automaticamente disparados como eventos WMI, e os objetos escolhidos são registrados na infra-estrutura com suas devidas propriedades e métodos.

A figura 4.1 mostra como o atributo *InstrumentationClass* é adicionado a declaração de uma classe. O atributo iniciado com o parâmetro *InstrumentationType* igual a *Instance*, gera automaticamente (em tempo de compilação) um esquema de classe WMI para a classe implementada em .NET. Instâncias da classe declarada poderão, então, ser expostas via WMI. Estes tipos utilizados na instrumentação encontram-se agrupados no *namespace System.Management.Instrumentation* da biblioteca de classes do .NET 1.1.

```
//Namespace utilizado
using System.Management.Instrumentation;

// Especifica o namespace WMI onde a classe deve ser publicada
[assembly:Instrumented("root/SD/WMIApp")]

//Classe registrada no WMI
[InstrumentationClass(InstrumentationType.Instance)]
public class WMIAppClass{ //... }
```

Figura 4.1: Declaração da classe instrumentada

Também na figura 4.1, o atributo *Instrumented* indica que a biblioteca ou o executável gerado na compilação possui instrumentação para gerenciamento. O parâmetro passado no construtor do atributo informa o *namespace* do esquema WMI em que a classe será criada. O WMI, assim como .NET, trabalha com o conceito de *namespaces* permitindo que as classes sejam agrupadas de forma lógica. O *namespace* raiz é chamado "*root*" e abriga outros *namespaces* filhos com classes WMI para gerenciamento de uma grande quantidade de recursos do sistema.

Qualquer aplicação que declare tipos instrumentados (eventos ou instâncias) precisa que seu esquema seja registrado no repositório WMI. Isto é feito utilizando o mecanismo padrão de instalação no .NET. O *namespace* de instrumentação possui a classe *DefaultManagementProjectInstaller* que auxilia a instalação de aplicações instrumentadas. Para utilizá-la, basta derivar uma nova classe desta classe que realiza a instalação. O atributo *RunInstaller* com parâmetro igual a *true* na declaração da classe, informa ao instalador do .NET (*InstallUtil.exe*) que existe um passo instalação a ser executado.

Por fim, para iniciar o instalador, adiciona-se uma chamada para o método *InstallHelper* da classe *ManagedInstallerClass* no início da aplicação instrumentada. Nos argumentos exigidos para execução deste método é preciso especificar qual componente que contém a classe com os passos de instalação.

```
//Namespaces utilizado
using System.ComponentModel;

// Instalador
[RunInstaller(true)]
public class InstrumentationInstaller :
DefaultManagementProjectInstaller   {     }
```

Figura 4.2: Declaração da classe instaladora.

Por fim, é preciso publicar o objeto na infraestrutura WMI. A figura 4.3 exibe o código necessário para publicar o objeto no repositório. O método estático *Publish* da classe *Instrumentation* é responsável por expor os dados de monitoração via WMI, basta passar como parâmetro a instância da classe instrumentada.

```
WMIAppClass devObj = new WMIAppClass();

// publica objeto no repositório WMI
Instrumentation.Publish(devObj);
```

Figura 4.3: Variável do tipo WMIAppClass é publicada no WMI.

4.2 Obtendo dados do WMI

O *namespace System.Management* do .NET também oferece um conjunto de classes para interagir com o WMI. Com elas é possível construir uma aplicação consumidora e consultar os objetos publicados pelas aplicações instrumentadas, ou até mesmo aqueles objetos do sistema já existentes.

A classe *ManagementClass* permite referenciar qualquer classe WMI. Para instanciar uma *ManagementClass* é preciso passar como parâmetros um objeto do tipo *ManagementScope*, um do tipo *ManagementPath* e outro do tipo *ObjectGetOptions*. *ManagementScope* especifica a máquina de que se deseja obter informações e o *namespace* do WMI onde está definida a classe. Se a conexão WMI for realizada sobre uma máquina remota, deve-se utilizar a classe *ConnectionOptions* na definição do escopo para especificar informações de usuário e senha. O segundo parâmetro do construtor da *ManagementClass* define a classe da qual busca-se instâncias. Já o último parâmetro carrega algumas opções da consulta a ser realizada, como o *timeout*, mas pode receber valor nulo no caso de se optar pelas configurações padrão.

Com a referência para classe WMI, é possível executar a sua operação *GetInstances* e obter como retorno um objeto do tipo *ManagementObjectCollection* que armazena uma coleção de instancias de *ManagementObject*. Um *ManagementObject* representa um objeto publicado no repositório WMI e, através de seus métodos e propriedades, são executadas as funções de gerenciamento e consulta as informações monitoradas deste objeto. A figura 4.4 exibe o trecho de código que trabalha com as classes mencionadas anteriormente para obter ocorrências de *WMIAppClass*.

```
//Opções de conexão
ConnectionOptions options = new ConnectionOptions();
                            options.Username = USER;
                            options.Password = PASSWORD;

//Define o escopo da conexão WMI
ManagementScope Scope = new ManagementScope
(@"\\" + MACHINE_NAME + @"\SD\WMIApp", options);

//Associa a classe registrada no WMI
ManagementPath mngPath = new ManagementPath();
mPath.RelativePath = "WMIAppClass";
WMIClass = new ManagementClass(Scope, mngPath, null);

//Obtém a coleção de objetos do tipo WMIAppClass
ManagementObjectCollection mngColl = WMIClass.GetInstances()
```

Figura 4.4: Código que consulta por instâncias de WMIAppClass no WMI.

Outra forma de realizar consultas ao repositório WMI é usar a *Wmi Query Language*, uma linguagem de consulta baseada na sintaxe padrão do SQL (LAVY, 2001). A grande diferença é que a WQL permite apenas consultas, sem atualizações ou inserções. A figura 4.5. exibe o trecho de código que realiza a consulta pelas instâncias de *SyncPerformance*. Repare que aqui também é definido um *ManagementScope*, e a consulta é representada por *WqlObjectQuery*. Da mesma forma que o método *GetInstances* de *ManagementClass*, o método *Get* de *ManagementObjectSearcher* efetua a consulta e retorna a coleção de objetos publicados.

```
//A consulta a ser executada
WqlObjectQuery Query = new WqlObjectQuery
("SELECT * FROM WMIAppClass");

//Instancia o "buscador" passando
//o escopo (mesmo da figura 5.x) e a consulta
ManagementObjectSearcher Searcher = new
ManagementObjectSearcher(Scope, Query);

//Executa a consulta
ManagementObjectCollection mngColl = Searcher.Get();
```

Figura 4.5: Consulta WQL para obter instâncias de WMIAppClass no WMI.

O suporte da versão 1.1 do *.NET Framework* ao WMI ainda possui limitações. Objetos instrumentados com o auxílio da biblioteca de classes não podem sofrer alterações em suas propriedades nem ter seus métodos executados. No entanto, é possível efetuar essas operações sobre aqueles objetos expostos pelos provedores WMI nativos escritos em C++. Felizmente, versões futuras do *Framework* removerão estas limitações (MICROSOFT, 2001).

5 CONCLUSÃO

O WMI já há alguns anos mostra-se como uma poderosa tecnologia para instrumentação e monitoração de sistemas distribuídos. Com o advento do .NET o que antes poderia ser considerado muito oneroso passou a se tornar mais ágil e fácil. A abstração de passos complexos para implementação de provedores e instrumentação de aplicações torna a utilização de WMI mais prática. Além disso, o fato de ser uma tecnologia totalmente orientada a objetos aproxima mais a implementação do modelo utilizado no WMI.

Desta forma, recomenda-se fortemente a utilização de WMI para instrumentar aplicações utilizando a biblioteca de classes do Framework .NET. Com isso. É possível utilizar ferramentas já disponíveis no mercado ou ainda implementar os seu próprios consumidores para monitoração das aplicações distribuídas.

6 REFERÊNCIAS

DMTF – DISTRIBUTED MANAGEMENT TASK FORCE. **Common Information Model Tutorial**, 2002. Disponível em: <http://www.dmtf.org/education/tutorials>

LAVY, M. M. et al. **Windows Management Instrumentation (WMI)**. 1ª ed. New York: Sams, 2001.

GOLOMSHTOK, A. **.NET System Management Services**. 1ª ed. Berkeley: Apress, 2003.

LAM, H. et al. **.NET Framework Essentials**. 3ª ed. [S.l.]: MSPress, 2002.

JEFFREY, R. **Applied Microsoft .NET Framework Programming**. [S.l.]: O'Reilly, 2003.

MICROSOFT CORPORATION. **Microsoft Framework SDK Documentation**. [S.l.:s.n]. 2001.

FANCEY, Jon. **Instrumentation: Powerful Instrumentation Options in .NET Let You Build Manageable Apps with Confidence**. MSDN Magazine v. 1 9, n. 4, Abr. 2004.

c Library**. Windows Management Instrumentation (WMI)**. Disponível em: <http://msdn2.microsoft.com/en-us/library/aa394582.aspx> Acesso em: abril 2007.