基于 WMI .NET 动态修改 IP 地址的实现方法①

An Implementation Method of Dynamic Modification of IP Address Based on WMI .NET

刘邦明 左伍衡 蒋炯辉 (浙江工业大学 之江学院 浙江 杭州 310024)

摘 要: 某些计算机安全管理目标要求动态实时修改 IP 地址,WMI 是目前少有的几种技术方案之一。基于.NET Framework 托管平台的 WMI .NET 技术封装了原始 WMI 细节,并提供统一的 WMI 对象属性读取、方法调用途径。通过 WMI .NET 类访问 WMI 网络对象,执行该对象相关公开方法,可以完成动态修改 IP 地址;结合注册表项的修改,能进一步提高方法执行效率。本方法的有效性和易用性在应用实例中得到验证。

关键词: WMI .NET Framework WQL 动态修改 IP 实现方法

1 引言

动态实时修改 Windows 系统 IP 地址组,是实现某些特定计算机安全管理目标的必要技术之一。但微软并未公开相关的 Windows API 函数,从而徒增实现难度。尽管有方法称借助未文档化的函数 DhcpNotifyConfigChange()可以实现目的,不过却由于蕴含着不可测的风险而不能用于正式系统。本文将介绍基于 WMI .NET 技术实现动态修改 IP 地址组的方法及其实际应用。

2 WMI .NET技术

WMI(Windows Management Instrumentation,Windows 管理规范)是一项核心的 Windows 管理技术,是 Windows2000 及以后版本的一个内置组件。WMI 使用基于“公共信息模型 (CIM)”行业标准的类表示目标计算机的操作系统、进程、网络、设备和其他计算机组件,可以通过脚本语言或 API 编程实现对本地管理客户端系统中几乎一切软硬件信息的访问和管理[1]。正因如此,目前市面上很多专业的网络管理工具都是基于 WMI 开发的。

.NET Framework 中的 WMI(简称“WMI .NET”)是基于原始 WMI 技术的一种托管模式,属于.NET Framework 高级应用范畴。它不但可以实现与原始 WMI 相同的 WMI 数据访问和操作,而且能享有.NET Framework 具有的诸多优势——例如,利用公共语言运行库(CLR)功能、类定义和实例发布规范化、开发和调试简单、对象组织标准化较强等[2]。当然,WMI .NET 的托管模式也带来局限性,例如不支持刷新器、不能在托管代码中创建“事件使用者提供程序”、暂不支持群集环境等原始 WMI 所没有的限制。

WMI.NET 相关类集中组织在 System.Management 和 System.Management.Instrumentation 这两个.NET Framework 命名空间,前者包括用于执行 WMI 查询和操作的类(例如收集 WMI 类信息、查询数据、执行方法等),后者包括实现向应用程序中添加 WMI 提供程序的类(创建数据提供程序、创建事件提供程序、注册提供程序)。

3 实现原理

3.1 WMI .NET 体系结构分层

分层结构是 WMI 体系的显著特征,各层分别组织不同的对象、实现不同的功能。WMI 体系主要分三层:客户访问层(Client Access Layer)、对象管理层(Objects Management Layer)、提供程序层(Provider Layer),如图 1。其中,对象管理器是 WMI 提供程序层与客户访问层之间的中间装置,它提供一

① 收稿时间:2008-12-28

些关键服务，例如标准事件发布和订阅、事件筛选、查询引擎以及其他管理配置服务等[3]。一般的管理工具软件主要作用在对象管理层，而较少涉及提供程序层。

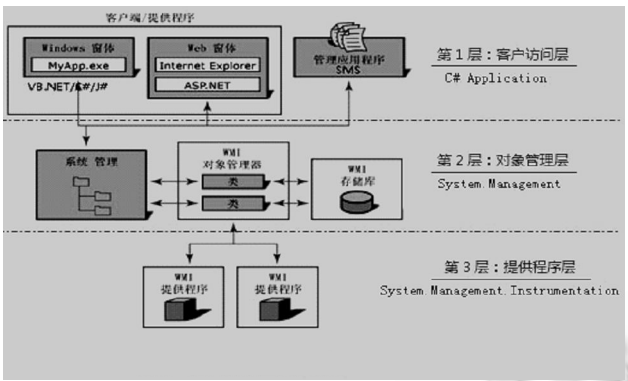

图 1　WMI.NET 三层结构图

在.NET Framework 托管平台中，对应该体系结构分别是：1) .NET 应用程序。采用 VB.NET、C#、J#、C++.NET、ASP.NET 等编程语言开发的 Windows、Web 应用程序或脚本程序。2) System.Management 命名空间。承担了对 WMI 对象枚举或检索、调用 WMI 类方法、订阅事件等诸多功能。其中有 4 个类较常用：ManagementClass，表示单个 WMI 类；ManagementObject，表示单个类实例；ManagementObjectSearch，根据指定的条件查询或枚举检索类或实例的集合；ManagementQuery，用作所有查询类的基础。3) System.Management.Instrumentation 命名空间的类可以实现用户自行创建符合 WMI 规范的提供程序。

3.2 WQL 语句

WMI 类，实际上就是对一些给定软件或硬件组件的属性和功能的“抽象描述”。由于存在许多种类，因此 WMI 将它们组织到命名空间层次结构中(与.NET Framework 命名空间并无关联[3])，核心 Windows OS 类的命名空间称为 root\cimv2，类按“Win32_XXX”格式命名。检索和管理必须基于类实例进行，所谓“实例”是指当前系统中“实际存在的类”。

WQL(WMI Query Language)语言用来检索某 WMI 类的实例集合，它是标准 SQL 的一个子集加上 WMI 的扩展，只能执行 SELECT 操作，而其他如 UPDATE、DELETE 等 DML 语句无效[4]。例如，“SELECT*FROM Win32_NetworkAdapter”语句将检索到当前系统中所有网络适配器对象(包含物理网卡和虚拟网卡)；而条件检索“SELECT * FROM Win32_LogicalDisk WHERE FileSystem IS "NTFS" ”只返回当前启用的 NTFS 格式逻辑磁盘。

3.3 WMI 公开方法的统一调用格式

WMI 类种类繁多，每种类各具有不同的属性和方法，WMI .NET 并没有直接给出针对每个类 WMI 对象的访问方法，而是提供统一的访问途径：首先，通过 ManagementClass 类获取某 WMI 类的实例集合；然后，枚举集合中的每个实例(以 ManagementObject 类实例表示)，读取该实例具有的属性值(可能为空值)。

如果需要调用某 WMI 实例的公开方法，则应先通过 ManagementObject.GetMethodParameters()获取该公开方法的传入参数组，并给该参数组赋值；接着执行 ManagementObject.InvokeMethod()实现对 WMI 公开方法的同步或异步调用；最后，检索.InvokeMethod()返回的参数组(如果存在返回值的话)，判断方法调用是否成功或/和错误代码(代码具体含义请参考 MSDN 文档)。

总之，枚举某 WMI 类的实例集合，然后读取实例的相关公开属性值、执行相关公开方法，从而实现查询和管理单个或多个 WMI 对象的目的，这即是本文阐述方法的实现原理。

4　具体步骤及方法改进

根据上述原理，我们探讨动态修改 IP 地址的具体实现步骤，以及如何提高执行效率。以 C#伪代码为例。

第 1 步，在应用程序项目中添加并引入 System.Management 命名空间。由于本方法不需要创建 WMI 提供程序，故不引入 System.Management. Instrumentation 命名空间。

```
Imports System.Management;
```

第 2 步，利用 WQL 语句检索系统中网络适配卡类及实例集合(Win32_NetworkAdapter，提供约 40

个属性和 4 个公开方法)，根据实例的属性值找出满足某约定条件所需的单个或多个对象，返回标识该目标网卡的唯一设备序号。例如，在 Windows XP 系统中可以根据 AdapterType= “Ethernet 802.3” 和 NetConnectionId!=null && NetConnectId=N 来确定第 N 块以太网类型网卡对象。

```
ManagementClass mc =
    New ManagementClass("Win32_Network Adapter");ManagementObjectCollection mos= mc.GetInstances();
    Foreach(ManagementObject mo in mos)
    If(mo["xx"]= =XX && …) //根据属性值找到目标网卡
        return(uint)mo["Index"]; //返回该网卡设备序号
//没有找到，返回 0(非有效的设备序号)
return 0;
```

第 3 步，根据目标网卡设备序号，检索其对应的可配置对象(可读写的 Win32_NetworkAdapterConfiguration 类)。

```
ManagementObject moSet=new Management Object("Win32_NetworkAdapterConfiguration.Index="
        + Index.ToString());
```

第 4 步，调用 moSet 对象公开方法，完成 IP 地址设置。Win32_NetworkAdapterConfiguration 类公开了近 60 个属性值、约 40 个方法用于操作 IP 地址相关动作。其中，与本方法相关的几个方法及其简要说明见表 1。这些方法，均按 WMI .NET 约定的统一调用格式——唯一区别的就是方法名和传入参数(组)，以下伪代码演示调用 “fnA” 公开方法：

```
//1.获取目标WMI对象公开方法的传入参数(组)
ManagementObject moInParameters =
    moInstanceA.GetMethodParameters("fnA");
//假设方法名为 fnA。
//2．依次给各传入参数赋值
moInParameters["XX1"] = 参数值 1;
    ……
```

表 1　与 IP 修改相关的方法

方法名	传入参数	功能
EnableStatic	string IPAddress[], string SubnetMask[]	同时修改IP地址、子网掩码(必须成对)
SetGateWays	string DefaultIPGateway[], uint16 GatewayCostMetric[]	修改默认网关
SetDHSSerwerSeachOrder	string DNSServerSearch Order[]	修改DNS服务器
EnableDHCP	无	启用DHCP获取IP地址
EnableDNS	……	启用静态DNS地址

```
    返回值:以上函数均返回 unit32 值:0 成功,非 0 错误代码(参见 MSDN 资料)
moInParameters["XXn"] = 参数值 n;
//3．执行方法(同步或异步)，获取返回值(组)
ManagementObject moOutParameters=// 同步执行
    moInstanceA.InvokeMethod("fnA",moInParameters);
//4．简单返回执行结果(0-执行成功；非 0-失败或其他含义)
Return((uint)moOutParameters["ReturnValue"] = = 0));
```

如果成功调用某方法，则可以完成对应的 IP 修改。例如，传入 IP、子网掩码新值(必须成对)给 EnableStatic()方法，将实现对 IP 地址(含子网掩码)的修改。其他修改操作类似。

上述步骤在将地址分配方式由 DHCP 改变为静态时，执行速度比较缓慢(从几秒到十几秒不等)。笔者经过分析和实验，提出了克服该弊端的一种改进方法：先修改 Windows 注册表中该网络适配器 TCP/IP 参数的 “EnableDHCP” 项为 0x0(即不采用 DHCP 分配地址)，然后再执行前述第 1～4 步骤。实验结果显示操作延滞几乎觉察不到，执行效率得到明显提高。

```
//获取该网络适配器的设备标识号:{xxxx-...-xxxx}
string SettingId = moSet["SettingID"];
//修改 Windows 注册表项
RegistryKey
r=Registry.LocalMachine.OpenSubKey
```

```
(@"SYSTEM\CurrentControlSet\Services\Tcpip\
Parameters\Interfaces\" + SettingId，true);
r.SetValue("EnableDHCP", 0x0);//将启用静态分配
r.Close();
```

5 应用实例

现以笔者参与开发的某高校公共计算机房管理软件为例，演示 WMI .NET 动态修改 IP 地址的应用及效果。

项目简介：该管理软件目前同时管理 6 个机房近 500 台计算机，需要根据学生上机的性质类型(教学集体、课外上机、收费上网)而不是属于哪个机房，自动、动态禁止或允许某台计算机访问 Internet——甚至教学集体上机也要动态控制学生端访问 Internent。

实现思路：能访问到机房局域网通往 Internet 的网关 A，是学生端访问 Internet 的必备条件，而通过控制学生端 Windows 系统“默认网关”IP 地址则可以达到目的：当允许上网时，将学生端“默认网关”IP 地址修改成可通往外部的网关 A(不可上网)；反之，则修改成内部网关 B 的 IP 地址。

实现步骤：1)管理服务器端根据学生端的当前上机性质，决定是否准予上网；2)如果允许上网，则发送指令 A(并附带网关 A 的 IP)给目标学生端；3)目标学生端收到指令 A 后，执行本文所述方法，将学生端当前默认网关修改为网关 A 的 IP；4)学生端通过网关 A，顺利访问外部 Internent 资源。如果不再允许上网，则通过修改学生端默认网关为内部网关 B 的 IP 地址，即可切断通往外部的桥梁。本过程可根据管理要求和学生端状态变化实时、动态执行。

效果评价：实际结果表明，利用本方法实现控制访问 Internet 简单易行，执行高效，效果良好。同时，本方法还可用于机房自动批量修改计算机初始 IP 地址的场合。

6 结语

WMI .NET 是.NET Framework 编程的高级部分，利用本技术不但能实现原始 WMI 非常丰富的功能，而且.NET 编程具有的诸多优势能提高开发可能性、可靠性、标准性。本文论述的借助 WMI .NET 技术实现动态修改 IP 地址就是一个较好的例证。

参考文献

1 宋昕,盛晨,王新华.基于 WMI 的计算机管理技术的研究与实现.浙江科技学院报, 2007,19(1):42－43.

2 李明,李廷全,张波. WMI 在网络管理中的应用研究.网络安全技术与应用, 2008,(11):39.

3 Robinson S. 高级.NET 程序设计.冉晓旻,王军 译.北京:清华大学出版社, 2003:340－342.

4 宋昕.WMI 在计算机管理中的应用研究.电脑知识与应用, 2008,(11):370－371.