Gestion des imprimantes réseau avec WMI et C#

Table des matières

Gestion des imprimantes réseau avec WMI et C# 1

I Introduction 2

II Prérequis 2

1. Mise en place d'un serveur d'impression 2
2. Mise en place de WMI 3

III Utilisation de WMI avec C# 3

1. Ouvrir une connexion sur le serveur distant 3
2. Les requêtes WMI 4

IV Codes utiles 5

1. Lister les imprimantes et leur configurations 5
2. Obtenir la qualité d'impression d'une imprimante 6
3. Obtenir la résolution horizontale 7

V Conclusion 8

I Introduction

A travers ce tutoriel je vais vous montrer comment réaliser un inventaire de vos imprimantes réseau et ainsi obtenir des informations intéressantes sur chacune d'entre elle. Nous allons utiliser le système de gestion de ressources informatiques fourni par Microsoft, je veux bien sur parler de WMI (Windows Management Instrumentation). Dans un article précédent je vous ai introduit quelques notions concernant le protocole SNMP en utilisant la bibliothèque de fonctions oleprn.dll. En combinant ces deux technologies vous serai capable de gérer dans de bonnes conditions toutes vos imprimantes réseau… Nous allons voir qu'il est assez facile à mettre en œuvre mais pour cela il faut respecter quelques requis.

II Prérequis

1. Mise en place d'un serveur d'impression

Pour réaliser l'inventaire de vos imprimantes réseau, il est nécessaire dans un premier temps de créer un serveur d'impression. C'est-à-dire un serveur sur lequel toutes les imprimantes en état de fonctionnement sur votre parc doivent être installées. Ceci permet d'établir la liste des imprimantes plus facilement…

2. Mise en place de WMI

Dans un deuxième temps il faut s'assurer que le serveur accepte les requêtes WMI . Vous devez de même vous assurer que vous avez les droits d'accès sur cette machine. Mieux vaut être administrateur de la machine, pour ne pas avoir de problème…

Il faut de même s'assurer que le service RPC « Appel de procédure distante » soit démarré (RpcSs), ainsi que le service « Lanceur de processus serveur DCOM » (DcomLaunch), et aussi le service « Infrastructure de gestion Windows » (Winmgmt).

III Utilisation de WMI avec C#

1. Ouvrir une connexion sur le serveur distant

Pour se connecter à notre serveur d'impression distant il est nécessaire de faire appel à une instance de l'objet ConnectionOptions qui se trouve dans le namespace System.Management. Cet objet va nous permettre de mettre en place une connexion de type Kerberos. Le code suivant réalise cette opération :

```
ConnectionOptions opt = new ConnectionOptions();
opt.Username = sUserName;
opt.Password = sPwd;
opt.Authority = "ntlmdomain:”+ sAuthority;
```

Avec sUserName et sPwd les identifiants de connexion de l'utilisateur qui doit faire partie du groupe administrateur de la machine.

sAuthority peut :

soit référencer le nom du serveur distant auquel cas nous nous connectons en tant qu'administrateur local ;

soit référencer un domaine auquel cas n'importe quel administrateur de ce domaine est en mesure de pouvoir se connecter.

Une fois ces identifiants notifiés il est nécessaire de se connecter à la machine. Nous réalisons cette tâche en utilisant un scope sur celle-ci :

```
ManagementScope scope = new
ManagementScope(@"\\adresseip\root\cimv2", opt);
```

Où adresseip est l'adresse ip de la machine distante, bien entendu.

Une fois défini ce scope avec les bonnes informations, il suffit de se connecter :

```
scope.Connect();
```

Nous avons vu les principales informations nécessaires pour obtenir un droit d'accès à une machine distante.

2. Les requêtes WMI

Alors à partir de ce moment la connexion doit être établie. Il ne reste plus qu'à interroger cette machine pour obtenir des informations voulues. Dans notre cas ce sera des informations concernant les imprimantes. La syntaxe des requêtes ressemble à celle de SQL, donc cela ne devrait pas être trop compliqué. Voici un exemple de ce que l'on peut définir comme requêtes :

```
SelectQuery q = new SelectQuery();
q.QueryString = "Select * from win32_printer";
```

Comme vous pouvez le constater il suffit de créer une instance d'objet SelectQuery. Comme vous pouvez le constater on effectue ces opérations de recherche sur des classes. Concernant les imprimantes nous pourrons utiliser 3 classes :

- Win32_printer qui permet d'obtenir des informations globales sur le périphériques (nom, emplacement, port, statut, etc…) ;
- Win32_printerconfiguration qui comme son nom l'indique vous permet d'obtenir des informations telles que la taille du papier, la largeur du papier, le nombre de copies par défaut, etc …;
- Win32_printjob qui donne des informations utiles sur un travail d'impression (comme le statut, le nombre total de pages, etc…).

Je vous envoie sur cette page, vous obtiendrez des informations plus détaillées sur ces trois classes.

Pour pouvoir envoyer des requête il est nécessaire dans un premier temps de créer une instance d'objet ManagementObjectSearcher que l'on crée en lui ajoutant en paramètres la requêtes que nous venons de définir.

```
ManagementObjectSearcher mos = new ManagementObjectSearcher(scope, q)
```

Il nous faut ensuite récupérer les collections de données retournées par la requêtes. Ceci est réalisé en créant une instance d'objets ManagementObjectCollection.

```
ManagementObjectCollection moc = mos.Get();
```

Il suffit ensuite d'énumérer tous les éléments de la collection en utilisant une boucle foreach en utilisant un objet ManagementObject comme suit :

```
foreach (ManagementObject mo in moc)
{

    //opérations à réaliser

}
```

Voilà l'essentiel de ce qu'il faut connaître pour utiliser WMI. Nous allons voir maintenant quelques exemples de code.

IV Codes utiles

1. Lister les imprimantes et leur configurations

Voici un exemple de code qui est intéressant pour récupérer la liste de toutes les imprimantes se trouvant dans votre réseau d'entreprise. Dans cet exemple nous allons récupérer le nom de l'imprimante, le port de l'imprimante, ainsi que son statut.

```
      public Printers getListeImprimantes()
        {
            ConnectionOptions opt = new ConnectionOptions();
            opt.Username = sUserName;
            opt.Password = sPwd;
            opt.Authority = "ntlmdomain:Domain";

            ManagementScope scope = new
ManagementScope(@"\\adresseip\root\cimv2", opt); //remplacer adresseip par la
valeur correspondant à celle de votre serveur distant
            scope.Connect();
            SelectQuery q = new SelectQuery();
            q.QueryString = "Select * from win32_printer";


            prnlist = new Printers();
            ManagementObjectSearcher mos = new
            ManagementObjectSearcher(scope, q);
            ManagementObjectCollection moc = mos.Get();

            foreach (ManagementObject mo in moc)
            {
                Printer prn = new Printer();

                string tempstatus = mo["PrinterStatus"].ToString();
                comboBox1.Items.Add(mo["Name"].ToString());

                prn.Nom = mo["Name"].ToString();
                String temp;
                temp = mo["PortName"].ToString();
                prn.IpAddress = temp.Substring(3);



                try
                {

                    prn.Emplacement = mo["Location"].ToString();

                }

                catch (Exception)
                {

                    //Ne rien faire
                }
```

```
                prnlist.Add(prn);


            }

            return prnlist;
        }
```

Dans cet exemple, j'ai utilisé les deux classes que j'avais créées (Printer et Printers qui est une liste d'objets Printer) pour le tutoriel concernant la gestion des imprimantes en réseau avec la bibliothèque de fonctions oleprn.dll, que vous pouvez consulter ici.

2. Obtenir la qualité d'impression d'une imprimante

Dans cet exemple je vais vous montrer comment utiliser la classe Win32_PrinterConfiguration pour obtenir la qualité d'impression d'une imprimante.

```
      public string getPrintQuality(string PrinterName)
        {
            int tempQuality=0;
            string tempQualityString;

            ConnectionOptions opt = new ConnectionOptions();
            opt.Username = sUserName;
            opt.Password = sPwd;
            opt.Authority = "ntlmdomain:Domain"; //remplacer Domain par le
nom de domaine ou le nom du serveur distant

            ManagementScope scope = new
ManagementScope(@"\\adresseip\root\cimv2", opt); //remplacer adresseip par
l'adresse ip de votre serveur

            scope.Connect();
            SelectQuery q = new SelectQuery();
            q.QueryString = "Select Name, PrintQuality from
            win32_PrinterConfiguration";


            ManagementObjectSearcher mos = new
ManagementObjectSearcher(scope, q);
            ManagementObjectCollection moc = mos.Get();

            foreach (ManagementObject mo in moc)
            {
                try
                {
                    string Temp = mo["Name"].ToString();
                    if (nom == Temp)
                    {

                        tempQuality = Convert.ToInt32(mo["PrintQuality"]);
```

```
                        break;
                    }



                }
                catch
                {
                    tempQualityString = "informations non disponibles...";
                }
            }

            if (tempQuality < 0)
            {
                switch (tempQuality)
                {
                    case -1: tempQualityString = "Brouillon";
                        break;
                    case -2: tempQualityString = "Faible";
                        break;
                    case -3: tempQualityString = "Moyen";
                        break;
                    case -4: tempQualityString = "Haute";
                        break;
                    default: tempQualityString = "Information
indisponible...";

                        break;
                }
            }
            else
            {
                tempQualityString = tempQuality.ToString();
            }


        return tempQualityString;

        }
```

En fait si l'on regarde de plus près la définition liée à la propriété PrintQuality, on s'aperçoit que celle-ci renvoie un entier. Si cet entier est négatif on peut donc définir une qualité mais si celui-ci s'avère positif l'information récupérée vous donne le nombre de points par pouce.

3. Obtenir la résolution horizontale

Dans ce dernier exemple nous allons demander la résolution horizontale configurée pour l'impression d'un document. Nous requêtons toujours la classe Win32_PrinterConfiguration.

```
public string getHResolution(string ipAdress, string PrinterName)
        {

            string tempRes = "";

            ConnectionOptions opt = new ConnectionOptions();
```

```
            opt.Username = sUserName;
            opt.Password = sPwd;
            opt.Authority = "ntlmdomain:Domain"; //remplacer Domain par le
nom de domaine ou le nom du serveur distant



            ManagementScope scope = new
ManagementScope(@"\\adresseip\root\cimv2", opt); //remplacer Domain par le
nom de domaine ou le nom du serveur distant

            scope.Connect();
            SelectQuery q = new SelectQuery();
            q.QueryString = "Select Name, HorizontalResolution from
win32_PrinterConfiguration";


            ManagementObjectSearcher mos = new
ManagementObjectSearcher(scope, q);
            ManagementObjectCollection moc = mos.Get();

            foreach (ManagementObject mo in moc)
            {
                try
                {
                    string Temp = mo["Name"].ToString();
                    if (nom == Temp)
                    {

                        tempRes = mo["HorizontalResolution"].ToString();
                        break;
                    }

                }
                catch
                {
                    tempRes = "informations non disponibles...";
                }
            }



        return tempRes;
    }
```

V Conclusion

Nous avons vu au cours de ce tutoriel un moyen simple de récupérer des informations concernant vos imprimantes réseau en utilisant la technologie WMI. Combinée avec l'utilisation de la bibliothèque de fonctions oleprn.dll, elle vous permet de disposer d'un moyen efficace pour pouvoir administrer dans de bonnes conditions votre parc d'imprimantes.