The Windows Management Instrumentation (WMI)  
... and the technologies it is build on

Jelmer Vernooij
<jelmer@samba.org>
Samba Team
http://www.samba.org/~jelmer/
Agenda

• WMI required technologies:
  – WBEM
  – COM
  – Distributed COM
• WMI itself and how to use it
WBEM
Web-Based Enterprise Management (WBEM)

- Created by DMTF (Distributed Management Task Force)
- Open Source implementations: OpenPegasus, OpenWBEM
WBEM - Components

- CIM (Common Information Model) - Standard set of classes / objects
  - Core schema
  - Common schema
  - Win32 Extended schema
- CIM Query language
- CIM URI standard
- CIM-XML (default “transport”)
CIM schemas – MOF notation

Superclass

class HardwareDevice {
    }

Derived class

class Computer : HardwareDevice {
    string Model;
    uint32 ProcessorClockFrequency;
}

Instance

instance of Computer {
    ManufacturerName = “ASUS”;
    Model = “M3700N”;
    ProcessorClockFrequency = 1500;
}

- Compile and register with mofcomp
- IDL-like
CIM Standard Schemas

- Core Model
- Common Model
- Extension Model
WQL

- Subset of SQL92

```
SELECT * FROM Win32_LogicalDisk WHERE FileSystem = "FAT"

ASSOCIATORS OF {Win32_Service = 'DHCP'}

CALL[\server\root\cimv2:Win32_Process.Handle="2236"].Terminate(Reason=0)

UPDATE Win32_Environment SET VariableValue = 'bla' WHERE __PATH="\.\root\cimv2:Win32_Environment.Name="Test",UserName="server\user"
```
WBEM URI's

scheme://[user[:pass]@]host/namespace/model

e.g.
  https://foo:bar@bla/interop/cim_namespace.name=unknown

for CIM-XML over HTTPS

not used in WMI
SNIA and WBEM

• Worked on CIM as part of the SMI workgroup
COM
Introduction to COM (1)

• Key part of Windows
• Around since ~1993, actively used since ~1997
• Used as the basis for various other technologies:
  – DCOM
  – OLE2/ActiveX
• Several enhancements in Windows 2000: COM+
Introduction to COM (2)

- Object-oriented language-independent framework
- Implementation and interface clearly separated
- Implementation only specified at activation time
IUnknown

- All interface in OO style based upon the IUnknown interface
- IUnknown contains GetInterface(), AddRef() and Release()
• ODL (extended IDL)
  – `coclass` data type
  – inheritance for `interfaces`
  – Identification by UUID's
• Activated using `GetObject()`
• "Activation" information all stored in the registry (`HKEY_CLASSES_ROOT`)

```cpp
IBird *pBird = CoCreateInstance(CLSID_Penguin, IID_IBird, ...)
pBird->EatFish()
```
DCE/RPC

• Traditional OpenGroup DCE/RPC
• NDR encoding version 1 defined
• Should be well known to most CIFS vendors
Introduction to DCOM

• Distributed version of COM
• Documented in an internet draft
• Microsofts' answer to CORBA
• “hidden” from the application programmer
• Once “hyped” as the way of providing services over the internet
DCOM – RPC extensions

- ORPC (NDR revision 2)
  - uses extra field in dcerpc bind with object GUID
  - new primitive data type: MInterfacePointer (i.e. pointing)
  - additional `this` and `that` arguments
  - uses alternate binding contexts a lot
Frame 74 (198 bytes on wire, 198 bytes captured)


Internet Protocol, Src: 192.168.1.8 (192.168.1.8), Dst: 192.168.1.3 (192.168.1.3)

Transmission Control Protocol, Src Port: 1035 (1035), Dst Port: 2467 (2467), Seq: 276, Ack: 415, Len: 144

DCE RPC Request, Fragment: Single, FragLen: 144, Call: 1 Ctx: 0, [Resp: #78]

Version: 5
Version (minor): 0
Packet type: Request (0)

Packet Flags: 0x83

Data Representation: 10000000

Frag Length: 144
Auth Length: 16
Call ID: 1
Alloc hint: 68
Context ID: 0
Opnum: 5
Object UUID: 00007c00-051c-0000-ba02-42e0ad315410
Auth type: NTLMSSP (10)
Auth level: Connect (2)
Auth pad len: 12
Auth Rsrvd: 0
Auth Context ID: 4360412

[Response in frame: 78]

NTLMSSP Verifier

Version Number: 1
Verifier Body: 00000000000000000000000000000000

IRemUnknown, RemRelease

Operation: RemRelease (5)

DCOM, ORPCThis, V5.4, Causality ID: 000000003-e7ab-0000-cd1d-f9ff951ff9ff
InterfaceRefs: 1

RemInterfaceRef[1]: IPID=00008402-051c-0000-b666-3e015606be8f, PublicRefs=5, PrivateRefs=0

Auth Padding (12 bytes)
DCOM – RPC interfaces

- Activation
  - IRemoteActivation
  - ISystemActivator (since Win2k)
- Management
  - IOXIDResolver
  - IROT
DCOM – Garbage collection

• Destroy objects if
  – Clients don't “ping” an object for 3 minutes
• Mechanism for pinging groups of objects
  – ComplexPing() and SimplePing() in IOXIDResolver
WMI
Mixing it all together...

- Windows Management Instrumentation
  - basically WBEM with DCOM as transport
  - Core Model, Common Model, Microsoft-specific model (CIMv2)
WMI

- Runs on Win9x/NT4 and above
  - Included by default since Win2K
- Available thru DCOM using the IWbemServices class
- Special Activation mechanism for WMI: the winmgmmts: namespace
- Can manage pretty much everything:
  - Hardware devices
  - Several applications such as Office
  - .NET Framework
  - AD Related
WMI – User tools

- WBEMtester
- MMC
- VBScript / API
- Available to all COM-enabled languages (VB/C++/Python/...)
- Part of .NET (System.Management)
- WMIC
WMI - WBEMtester

Windows WBEM test client
WMI – Microsoft Management Console

- WMI used for snap-ins
Set WMIService = _
    GetObject("winmgmts:{impersonationLevel=impersonate}!" + _
    "\.oot\cimv2")

Set users = WMIService.ExecQuery("SELECT * FROM Win32_UserAccount WHERE Name="" + _
    WScript.Arguments(0) + "")

For Each User In users
    Wscript.Echo(user.Domain)
    Wscript.Echo(user.SID)
    Wscript.Echo(user.FullName)
Next
Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.root\cimv2")

Set objRefresher = CreateObject("WbemScripting.SWbemRefresher")

Set colDiskDrives = objRefresher.AddEnum_
  (objWMIService, "Win32_PerfFormattedData_PerfDisk_LogicalDisk").objectSet

objRefresher.Refresh

For i = 1 to 500
  For Each objDiskDrive in colDiskDrives
    Wscript.Echo "Drive name: " & objDiskDrive.Name
    Wscript.Echo "Disk bytes per second: " & objDiskDrive.DiskBytesPerSec
    Wscript.Sleep 2000
    objRefresher.Refresh
  Next
Next
Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\root\cimv2")

Set colComputers = objWMIService.ExecQuery("Select * from Win32_ComputerSystem")

For Each objComputer in colComputers
    errReturn = ObjComputer.Rename("NewName")
    WScript.Echo "Computer name is now " & objComputer.Name
Next
using System;
using System.Management;

class Class1
{
    static void Main(string[] args)
    {
        ManagementClass mc = new ManagementClass("Win32_Share");

        ManagementObjectCollection mcc = mc.GetInstance();

        foreach (ManagementObject mo in mcc) {
            Console.WriteLine("'{0}' path is '{1}'", mo["__REL_PATH"], mo["Path"]);
        }
    }
}
Future

- Replace by a full .NET equivalent (no DCE/RPC)
- Move towards standardised transport CIM-XML (?)
Implementation Considerations

- DCOM or CIM-XML as transport protocol?
  - CIM-XML advantages
    • Simpler to implement
    • Less security
    • Standardised
  - DCOM
    • Natively supported on Windows
    • Superceded by .NET (?)
Further resources

• WMI  
• DCOM white-paper http://samba.org/~jelmer/dcom.pdf  
• WBEM standard http://www.dmtf.org/standards/wbem/  
• DCOM standard  
• DCE/RPC  
  http://www.opengroup.org/onlinepubs/9629399/toc.html  
• SNIA CIM/WBEM http://www.snia.org/tech_activities/SMI/cim/