HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

# WMI – Windows Management Instrumentation

**CCERT-PUBDOC-2005-08-131**

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument, koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

**CARNet CERT**, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

**LS&S**, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

# Sadržaj

# 1. Uvod

Upravljanje informacijskim sustavima sve je kompleksniji i vremenski zahtjevniji proces, osobito kada se u obzir uzmu velika korporativna okruženja koja sadrže desetke tisuća upravljivih entiteta – računala, korisničkih računa, mrežnih komponenti, periferije i sl. Upravo zbog brzog razvoja računalne industrije te potrebe za centraliziranim upravljanjem nastala je DMTF organizacija koja se brine o standardima i inicijativama vezanim uz upravljanje u velikim distribuiranim sustavima. Zahvaljujući toj inicijativi Microsoft je razvio i nadogradio postojeći model CIM standarda te razvio WMI (eng. *Windows Management Instrumentation*) sustav. Centraliziran sustav omogućio je jednostavniju kontrolu informacija i osigurao provođenje sigurnosne politike sustava. WMI nije isključivo vezan za sigurnosne aspekte, ali zahvaljujući detaljnom modelu cijelog operacijskog sustava i fizičkog modela računala upravljanje svim dijelovima, a tako i sigurnosnim aspektima na zavidno je visokoj razini.

Podrška za brojne skriptne jezike te integracija s WSH-om (eng. *Windows Scripting Host*) i drugim inicijativama zadovoljava zahtjeve svih informacijskih sustava od onih najmanjih do onih najvećih, a detaljan objektni model i API omogućuje razvoj aplikacija s podrškom za WMI. Integracija s .NET okruženjem i drugim programskim jezicima preko COM objekata omogućila je razvoj programske podrške koja na relativno jednostavan način krajnjim korisnicima pruža mogućnost nadzora svih aspekata sustava. Kako je WMI duboko integriran u Windows operacijski sustav zajedno sa serverskim tehnologijama, moguć je pouzdan nadzor i striktno provođenje sigurnosne politike informacijskog sustava.

Dokument daje općeniti pregled i arhitekturu WMI-a, proces i razloge njegovog nastajanja te mogućnosti njegove primjene u praksi. Poseban osvrt dan je na primjenu WMI-a u području informacijske sigurnosti, pri čemu je naglasak dan na tehnikama upravljanja te mogućnostima praćenja i nadzora sustava.

S obzirom na sve veći broj sofisticiranih malicioznih programa (virusi, crvi, trojanski konji, spyware, rootkit i sl.) te sve naprednije tehnike koje isti koriste za prikrivanje od korisnika i operacijskog sustava te zaobilaženje antivirusnih i vatrozidnih zaštita, evidentno je da su potrebni efikasniji mehanizmi za njihovu detekciju i uklanjanje. U četvrtom poglavlju su opisane su mogućnosti nadzora i praćenja *registry* datoteke koju maliciozni programi vrlo često modificiraju, a također su analizirane i trenutno vrlo aktualne metode detekcije *rootkit* programa.

Drugi segment koji je dio ozbiljnog problema kako u sigurnosnom tako i u ekonomskom smislu su P2P programi. Osim što predstavljaju sigurnosni rizik, ovi programi generiraju i neželjeni mrežni promet koji ozbiljno može ugroziti funkcionalnost korporativnog mrežnog okruženja. U slučaju nemogućnosti nabave *statefull inspection* vatrozida, implementacija *process firewalla* temeljnog na WMI-u može se pokazati kao prihvatljivo i ekonomično rješenje. Primjer implementacije ovakvog tipa zaštite također je opisan u četvrtom poglavlju.

Pored navedenih skriptnih tehnika biti će riječi o dodatnim alatima koji mogu olakšati rad s WMI-em i doprinijeti jednostavnijoj i praktičnijoj uporabi ove iznimno moćne tehnologije. Zbog opsežnosti Microsoftove implementacije dokument je zamišljen je kao uvod i upoznavanje s mogućnostima WMI-a u implementaciji sigurnosne politike jednog ozbiljnog informacijskog sustava.

# 2. Teoretske osnove

Najsloženiji dio WMI sustava zasigurno je veliki broj akronima koji se uz njega vezuje (DMTF, CIM, WBEM itd.). Iako detaljno poznavanje navedenih pojmova nije neophodno za uspješno korištenje WMI-a, njihovo razumijevanje može znatno pomoći u shvaćanju cijelog koncepta.

## 2.1. DMTF

Kako bi shvatili što je WMI pogledajmo prvo što je DMTF i njegovu povezanost s WBEM-om, CIM-om i WMI-em. DMTF je osnovan 1992. godine tada kao *Desktop Management Task Force*. DMTF je neprofitna organizacija, a njezino djelovanje bilo je usmjereno ka standardima za upravljanje osobnim računalima. Razvojem novih tehnologija i računalne industrije postalo je sve kompliciranije upravljati brojnim entitetima na korporativnoj razini. Ovakva evolucija odrazila se i na DMTF koja je zadržala svoj prvobitni akronim, ali se njegovo značenje promijenilo u *Distributed Management Task Force*. Današnja misija DMTF je koordiniranje razvoja standarda za upravljanje distribuiranim računalnim sustavima te *enterprise* mrežnim i Internet okruženjima. DMTF čini nekoliko vodećih kompanija među kojima su: 3Com, Hewlett-Packard, Cisco, Microsoft, SUN, Intel, IBM/Tivoli Systems, Novell i mnogi drugi (detaljan popis moguće je pronaći na adresi http://www.dtmf.org).

DMTF je pokrenula nekoliko osnovnih inicijativa:

1. *Desktop Management Interface (DMI):* DMI je prvi industrijski standard za upravljanje i praćenje podataka o različitim komponentama u osobnim računalima, prijenosnicima i poslužiteljima. Razvojem ovog standarda proizvođačima navedene opreme omogućeno je da na uniforman i standardizirani način omoguće upravljanje svojim proizvodima. DMI je jedino rješenje koje je optimizirano za rad s današnjim računalima.
2. *Web-based Enterprise Management (WBEM):* WBEM je tehnologija koja omogućuje jednostavnu razmjenu podataka o upravljivim računalnim sustavima. DTMF je razvio temeljni skup standarda koji sadrže podatkovni model, kodnu specifikaciju te transportni model. WBEM se na najjednostavniji način može opisati kao skup standardiziranih apstrakcijskih slojeva koji skrivaju kompleksnost pristupu informacijama za upravljanje. WMI sustav opisan u nastavku dokumenta implementira upravo WBEM.
3. *Common Information Model (CIM)* je općeniti model za opis podataka o upravljivim komponentama sustava (servisi, aplikacije, računala i sl.), neovisan o platformi i tehnologiji koja se koristi. CIM sadrži specifikaciju i podatkovnu shemu. Specifikacija definira detalje integracije s drugim modelima za upravljanje (npr. SNMP, MIB, CMIP...), dok podatkovna shema opisuje potrebne podatkovne modele. Kako **WMI intenzivno koristi CIM** o njemu će više riječi biti u nastavku dokumenta.
4. *Directory Enabled Networks (DEN):* DEN specifikacija je dizajnirana tako da omogući ostvarivanje inteligentnijih računalnih mreža logičkim povezivanjem mrežnih servisa s korisnicima i poslovnim kriterijima. DEN specificira općeniti podatkovni model koji može koristiti brojne tehnologije od CIM-a do X.500.

   Ovakva osebujnost stvara temelje za kreiranje predložaka za razmjenu informacija i omogućuje dobavljačima hardverske i softverske opreme međusobnu razmjenu definicija uređaja, aplikacija i servisa. DEN također dopušta interoperabilnost s rješenjima temeljenim na WBEM-u.

Ovo su samo neke od važnijih inicijativa koje ujedno pružaju uvid u temelje na kojima počiva WMI. Jedan od osnovnih razloga nastajanja ovih inicijativa je stvaranje unificiranog okruženja za upravljanje računalnim sustavima. Računalne mreže sadrže brojne komponente (*embeded* uređaje, računala, periferiju, itd.), kojima je potrebno upravljati na različite načine. Iako se računala različitih proizvođača bitno razlikuju, općenito gledajući, sva ona imaju slične karakteristike. Primjerice, svako računalo ima mrežnu karticu, tvrdi disk, radnu memoriju, procesor itd. Na sličan način moguće je raščlaniti i različite operacijske sustave.

Svim navedenim objektima, odnosno entitetima, različito se i upravlja. Tijekom godina mnogo je truda uloženo u tehnologije koje omogućuju dohvaćanje informacija za upravljanje, kao što su npr. *Simple Network Management Protocol* (SNMP), *Desktop Management Information* (DMI) i *Common Management Information Protocol* (CMIP). Zbog velike količine informacija kojima je potrebno

upravljati unutar modernih računalnih sustava, potrebno je bilo osmisliti rješenje koje će se moći nositi sa svim ovim tehnologijama, a WMI je jedna od implementacija koja se najviše približila tom cilju.

## 2.2. WBEM

Još od 1996. godine cilj je bio osmisliti standard koji će definirati kako je može upravljati s pojedinim entitetima i omogućiti voditeljima projekata jednostavan no ipak moćan način za ispunjavanje njihovih potreba za dobavljanjem informacija, bez da pri tome moraju detaljno poznavati specifikacije pojedinih uređaja. Jedna od DMTF inicijativa je upravo i WBEM (eng. *Web-Based Enterprise Management*).

WBEM predstavlja skup upravljačkih i Internet standarda koji nisu samo novi oblik dohvaćanja podataka (npr. HTTP protokolom), već i pružaju način reprezentacije entiteta čije su nam informacije potrebne, te njihovih karakteristika (CIM). WBEM se ukratko može definirati kao skup standarda koji definiraju apstraktne slojeve koji skrivaju kompleksnost pristupa informacijama pojedinog entiteta.

## 2.3. CIM

CIM (eng. *Common Information Model*) model ključan je u opisu svih elemenata vezanih uz računala i na hardverskoj i softverskoj azini. DMTF je shvatio da različiti proizvođači hardvera ili/i softvera za identične stvari koriste različite nazive te je s ciljem uklanjanja tih razlika odlučio stvoriti CIM. Sve komponente kojima je potrebno upravljati u tipičnom informacijskom okruženju (računala, tvrdi diskovi, procesori, memorija, pisači itd.) predstavljeni su objektima. Svaki takav objekt enkapsulira skup informacija koje je moguće predstaviti u podatkovnom modelu. Kako bi se realizirao jedan takav podatkovni model WBEM inicijativa uključuje CIM standard. CIM je konceptualni informacijski model koji opisuje informacije pojedinog entiteta te nije ovisan o načinu implementacije softverske ili hardverske komponente.

Na primjer, tvrdi disk jednog proizvođača može se opisati karakterističnim modelom koji je zajednički tvrdom disku bilo kojega drugog proizvođača. Čak i kada diskovi međusobno nisu isti, svim diskovima je zajedničko da imaju određeni kapacitet, brzinu pristupa, latenciju, broj cilindara i sl. Ako pojedine komponente imaju određene specifičnosti, i dalje postoji skup parametara koji su zajednički svim komponentama iste vrste. Pomnim razmatranjem i raščlanjivanjem upravljivih entiteta moguće je stvoriti cjelovit model koji opisuje svaku njihovu karakteristiku; upravo takav podatkovni model implementiran je u sklopu CIM-a.

U CIM-u su definirane brojne osnovne klase (eng. *base classes*) koje opisuju entitete kao što su tvrdi diskovi, procesori, matične ploče i sl. Detaljnom hijerarhijom pozabavit ćemo se u poglavlju 2.3.1. Primjerice, osnovna klasa `CIM_DiskDrive` uključuje `MaxBlockSize` i mnoga druga svojstva, no ne uključuje svojstva poput datotečnog sustava budući da su ona specifična za pojedini operacijski sustav. Ova svojstva nisu uključena u CIM model, no budući da je CIM model proširiv, Microsoft je prilikom kreiranja WMI-a stvorio skup Win32 klasa koje su specifične upravo za Windows operacijski sustav. Sve klase iz Win32 skupa su, ili temeljene na, ili naslijeđene iz skupa CIM klasa. Vraćajući se na primjer s tvrdim diskovima, Microsoftova implementacija CIM-a, odnosno WMI, uključuje razred `Win32_DiskDrives`, on nasljeđuje sva svojstva `CIM_DiskDrive` razreda no dodaje i neka nova svojstva, kao što je primjerice `PNPDeviceID` i koja su specifična za sam operacijski sustav.

Veći dio WMI-a čine upravo ovi razredi koji definiraju informacije o hardveru i softveru jednog računala. Jedno računalo primjerice može imati jedan tvrdi disk te sukladno tome imat će i jednu instancu klase `Win32_DiskDrives`. Računalo koje je korištenu u svrhu izrade ovog dokumenta sadrži tri instance `Win32_DiskDrives` razreda, što znači da računalo posjeduje tri tvrda diska. Slijedi primjer.

```
var SWBemlocator;

try {
        strComputer = ".";
        SWBemlocator = new ActiveXObject("WbemScripting.SWbemLocator");
        objWMIService=SWBemlocator.ConnectServer(strComputer,"/root/CIMV2");
        colItems = objWMIService.ExecQuery("Select * from Win32_DiskDrive");
        propEnum = new Enumerator(colItems);
```

```
        for (;!propEnum.atEnd();propEnum.moveNext()) {
                objItem = propEnum.item();

                        WScript.Echo("BytesPerSector: " + objItem.BytesPerSector);
                        WScript.Echo("Model: " + objItem.Model);
                        WScript.Echo("Name: " + objItem.Name);
                        WScript.Echo("Partitions: " + objItem.Partitions);
                        WScript.Echo("Size: " + objItem.Size);
        }
}
catch(e)  {
        WScript.Echo("catch caught " + e.number + " " + e.description);
}
```

U navedenom primjeru korišten je JScript (iako je moguće koristiti i VBScript jezik) te je za njegovo potpuno razumijevanje poželjno poznavanje osnova JScripta i WSH-a. Također, navedeni primjer je samo ilustrativne prirode, a opis ključnih komponenata potrebnih za interakciju s WMI-jem biti će pojašnjen u poglavljima koja slijede.

Izvršavanje skripte odvija se pozivom `cscript.exe` programa:

```
Cscript DiskInfo.js
```

Rezultat izvršavanja skripte je sljedeći:

```
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

BytesPerSector: 512
Model: IBM-DTLA-305020
Name: \\.\PHYSICALDRIVE0
Partitions: 2
Size: 20571425280
BytesPerSector: 512
Model: Maxtor 6Y080L0
Name: \\.\PHYSICALDRIVE1
Partitions: 1
Size: 81956689920
BytesPerSector: 512
Model: AVIXE PDU02_512 2.0 USB Device
Name: \\.\PHYSICALDRIVE2
Partitions: 1
Size: 501742080
Exit code: 0 , 0000h
```

WMI je od samog početka zamišljen tako da radi s višestrukim instancama klasa, iako to u određenim situacijama i nema smisla (npr. računala s dvije matične ploče vrlo su rijetka), no ideja je WMI implementaciju načiniti što fleksibilnijom, a uporabom ovakve logike dizajnerima to i uspijeva.

Nedostatak višestrukih instanci je relativna kompleksnost upita, odnosno dobivenih informacija. Dobar primjer za ovo je otkrivanje IP adresa računala, kako bi doznali IP adresu potrebno je točno znati za koji adapter postaviti upit jer Windows okruženja sadrže brojne adaptere (ne samo fizička Ethernet sučelja), a ovdje se čak ni ne misli na virtualni *loopback* adapter ili virtualne WAN adaptere. Na ovo treba obratiti pozornost prilikom kreiranja upita, odnosno pisanja skripti.

Vratimo se sada na CIM, naime CIM se dijeli na dva dijela, CIM specifikaciju i CIM shemu.

### 2.3.1. CIM specifikacija

CIM specifikacija definira podatkovni model u CIM standardu. Ona opisuje način na koji je podatkovni model kodiran (npr. kako je opisan objekt koji predstavlja tvrdi disk), uporabom MOF (eng. *Managed Object Format*) ili xmlCIM (eng. *eXtensible Markup Language CIM*) formata. MOF datoteka je ASCII kodirana i koristi posebno definiranu sintaksu za opis CIM definicija, a xmlCIM kodiranje definira XML elemente koji se koriste za definiciju objekata. Prednost XML strukture je u otvorenosti standarda, što omogućuje uporabu CIM-a između međusobno različitih sustava. Također, CIM specifikacija omogućuje i tehnike pristupa drugim upravljačkim modelima kao što su SNMP (*Simple Network Management Protocol*), DMI (*Desktop Management Interface*) ili CMIP (*Common Management Information Protocol*).

Također, CIM definira još jednu iznimno važnu stvar, a to je tzv. *metashema*. *Metashema* je formalna definicija podatkovnog modela. Kako CIM predstavlja brojne međusobno ponekad potpuno različite

komponente kao što su diskovi, softver i procesi, podatkovnom modelu je potrebna strogo definirana struktura kako bi se uspješno predstavili ovakvi različiti objekti. Za implementaciju podatkovnog modela stoga je nužno koristiti neke osnovne gradivne elemente, koji nam pomažu u definiranju prirode entiteta. Npr. u prethodnom primjeru diska, disk je disk i on je različit od matične ploče ili procesa, te je stoga on definiran razredom (eng. *class*) (vidi prije `CIM_DiskDrive`). Disk ima također i već navedena svojstva (eng. *properties*) koje predstavljaju drugi stupanj u CIM hijerarhiji. U objašnjenju trećeg stupnja – postupaka, poslužit ćemo se primjerom grafičke kartice (razred) koja posjeduje neku rezoluciju (svojstvo) i koju je moguće nekakvom akcijom promijeniti, upravo ta akcija predstavlja postupak (eng. *method*) i definira treću razinu CIM hijerarhije.

### 2.3.2. CIM shema

CIM shema je ključna komponenta podatkovnog modela koja predstavlja skup objekata različite povezanosti i može se shvatiti kao nacrt koji definira gradivne elemente pri upravljanju s računalima i pripadajućim softverom. CIM objektni model čini opis objekata koji se mogu naći u stvarnom svijetu, CIM shema koristi objektnu terminologiju u definiranju elemenata sadržanih u objektnom modelu. Slijedi sažeti opis ključnih elemenata (proširenje *metasheme*):

- **Razred** (eng. *class*) – opisuje ključne karakteristike objekta, zajedničke nekom skupu koji pripada istom razredu.
- **Podrazred** (eng. *subclass, superclass, parent class* ) – je naslijeđen iz osnovnog razreda i proširuje skup svojstava. Tako primjerice možemo imati razred medij koji je prilično općenit dok bi podrazredi u ovom slučaju bili CD/DDV-ROM, floppy, DAT trake itd.
- **Instanca** (eng. *instance*) – objekt je instanca nekog razreda, razred je samo opis objekta i jedinstven je, dok je instanca ustvari pravi objekt sa svojstvima opisanim u razredu. Razred je samo jedan, dok instanci u teoriji možemo imati beskonačno.
- **Svojstva** (eng. *properties*) – su atributi stvarnog objekta. Svojstva diska su primjerice veličina, broj cilindara, vrijeme pristupa i sl. Svojstva su definirana već u razredima, kada se objekt instancira u njegov memorijski prostor se spremaju i vrijednost koja čine njegova svojstva.
- **Veze** (eng. *relationships*) – se koriste kako bi naznačili nužnu vezu između objekata u CIM shemi. Dobar primjer veze je particija koja ne može postojati bez tvrdog diska. Naravno, ovo je samo jedan primjer veze, u praksi ih postoji mnogo više.

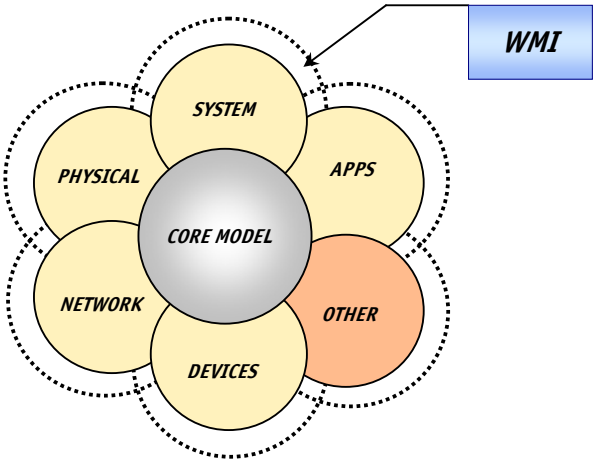
Dani prikaz jasno ukazuje na usku povezanost s objektno orijentiranom paradigmom te se shodno tome i pri modeliranju objekata koristi UML.

### 2.3.3. Struktura CIM sheme

CIM shemu grade dva usko povezana modela, temeljni model (eng. *core model*) i općeniti model (eng. *common model*). Kako CIM model hijerarhijski definira najprije najopćenitije, a zatim sve složenije entitete, ova podjela je logična. Temeljni model se nadograđuje dodatnim modelima i tako čini cjeloviti CIM model. Svaka nova inačica CIM-a dodaje nove modele, trenutno aktivna verzija je 2.9, no bez obzira na proširenja, struktura klasa je uvijek podijeljena na tri razine:

1. **Klase i pripadajući objekti koji odgovaraju svim područjima upravljanja.** Ove klase pružaju osnovni rječnik za analizu i upravljanje sustavima. One su dio temeljnog modela .
2. **Klase koje odgovaraju specifičnim područjima**, ali su neovisne o implementaciji ili tehnologiji. Ove klase su dio općenitog modela. Područja upravljanja koja spadaju u općeniti model su: System, Application, Physical, Device, Network, Metric, Supports, i User. Primjeri ovih klasa su: CIM_LogicalDevice i CIM_MediaAccessDevice
3. **Tehnološki i implementacijski specifične klase koje su dodatak općenitom modelu.** Ovi razredi općenito odgovaraju specifičnim platformama kao što su Linux ili Windows platforme. Primjer ovih klasa su `Win32_CDROMDrive` i `Win32_FloppyDrive` koje predstavljaju CD ROM i *floppy* pogon u Windows okruženju.

Slika 1 prikazuje logičku prezentaciju CIM strukture, u centru je prikazan temeljni model s općenitim modelima oko njega, koji predstavljaju dodatke (eng. *extension schema*) i na kraju su tu implementacijski specifični dodaci koji u ovakvom sklopu čine WMI.



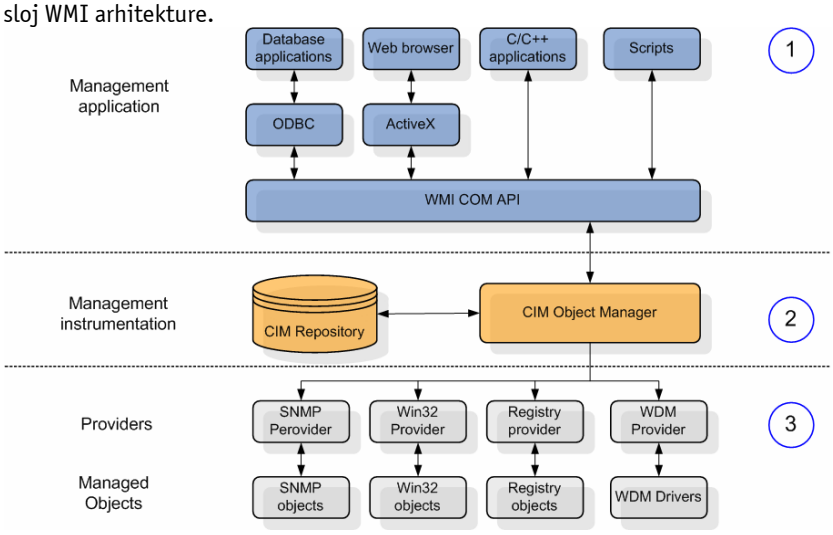

**Slika 1:** Struktura Common Information Modela i Windows Management Instrumentation-a

# 3. WMI – Windows Management Instrumentation

U dosadašnjem dijelu dokumenta detaljno je opisana osnovna tehnologija na kojoj se temelji WMI. U nastavku će biti više riječi o građi WMI-a, odnosno načinu na koji je proširena CIM specifikacija te kako se WMI u praksi može upotrijebiti za dohvaćanje i upravljanje mnogobrojnim aspektima tipičnog korporativnog Windows okruženja. WMI je dizajniran kao ekstenzija CIM općenitog modela i sadrži set servisa koji između ostalog uključuju podršku za postavljanje upita jezikom vrlo sličnim SQL-u (WQL) te dostavu obavijesti o događajima koji su izuzetno važni u .NET okuženju. Dodatno, WMI podržava COM (eng. *Component Object Model*) programsko sučelje na visokoj razini apstrakcije, što omogućuje jednostavan pristup WMI servisima te upravljanje pripadajućim podacima.

## 3.1. WMI arhitektura

WMI je zamišljen kao troslojna arhitektura prikazana na slici (Slika 2). Na vrhu arhitekture nalaze se aplikacije za upravljanje (eng. *management applications)*, koje su usko povezane s WMI COM API-jem i koji sadrži kolekciju COM objekata koji aktivno koriste .NET/win32 (C#, C++, Visual Basic) i skriptni programski jezici (JScript, VBScript). Drugi sloj sadrži *Common Information Model Object Manager* komponentu, čija je uloga interakcija s aplikacijama (eng. *consumer application*), CIM skladištem (eng. *CIM repository*) i WMI dobavljačima (eng. *WMI providers*), koji zajedno s upravljivim objektima čine treći sloj WMI arhitekture.



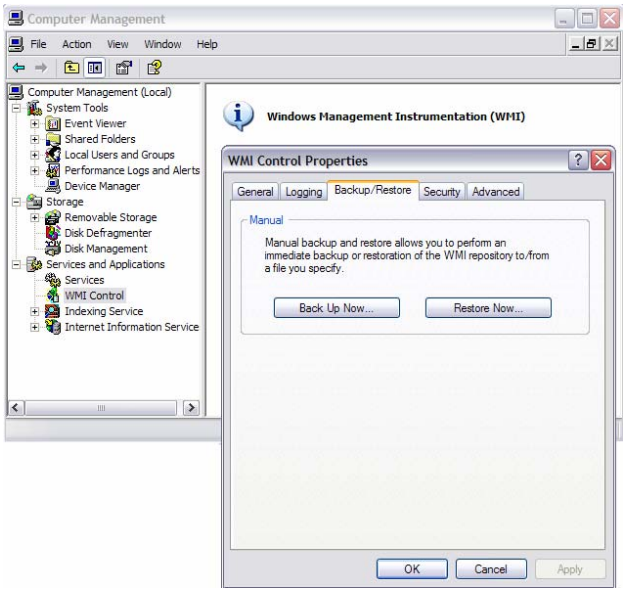

**Slika 2:** WMI arhitektura

### 3.1.1. Dobavljači i potrošači

Još jedan par termina koji se često susreće u WMI svijetu jesu dobavljači i potrošači (eng. *providers and consumers*). Potrošač je aplikacija koja koristi WMI kako bi, ili dohvatila, ili promijenila određenu informaciju iz CIM repozitorija. Primjer potrošača je bilo koja WMI skripta. Dobavljači su pak one aplikacije koje WMI informacije čine dostupnima. Windows sustav dolazi s određenim brojem dobavljača koji omogućuju dostupnost informacija o hardveru, softveru i performansama kroz WMI. Aplikacije drugih proizvođača mogu biti WMI dobavljači, što znači da je tim aplikacijama moguće upravljati kroz WMI. Činjenica da su ovi dobavljači duboko zakopani u Windows operacijskom sustavu prikriva pravu moć i fleksibilnost WMI-a. Preciznije rečeno, WMI nije integralni dio Windowsa, već je skup dodatnih servisa koji se pokreću ovisno o potrebi (`wmiprvse.exe`), a moguće ih je i potpuno onemogućiti putem **Computer Management** konzole.
Jedna od prednosti WMI-a je što donosi mogućnost obavljanja raznih zadataka uporabom skripti uz tehnologiju koja omogućuje nadgledanje cijelog sustava i procesa koji se odvijaju u njemu. Ovo znači kako sada možemo jednostavnije i brže izvršavati administrativne poslove njihovom automatizacijom. Još je jedna bitna činjenica koja WMI čini tako moćnim rješenjem, a to je njegova otvorenost i proširivost, te bilo tko može napisati *provider* aplikaciju koja ju otvara za uporabu u skriptama.

### 3.1.2. WMI skladište

WMI skladište (moguće ga je naći i pod imenom CIM skladište) je baza podataka u koju WMI pohranjuje informacije sukladno CIM podatkovnom modelu. U tipičnom Windows sustavu WMI skladište pohranjeno je u `\%SystemRoot%\System32\WBEM\Repository` direktoriju. Niti jedna aplikacija direktno ne pristupa skladištu, već se pristup podacima uvijek odvija putem CIM Object Managera. WMI skladište se pohranjuje u `.rec` datoteku svakih pola sata, što je ujedno i podrazumijevana vrijednost. Ostale podrazumijevane vrijednosti moguće je mijenjati putem **Computer Management** konzole. (Slika 3). CIM skladište se inicijalizira iz različitih MOF datoteka (locirane u `\%SystemRoot%\System32\WBEM` direktoriju) tijekom procesa instalacije. Ukoliko iz nekog razloga dođe do oštećenja CIM baze ona se reinicijalizira koristeći MOF datoteke čije su informacije pohranjene u *registry* ključu *HKLM\SOFTWARE\Microsoft\WBEM\CIMOM\Autorecover MOFs*. Ukoliko tijekom vremena bilo koja druga aplikacija proširi mogućnosti CIM repozitorija, potrebno je provjeriti jesu li i njezine datoteke sadržane u Windows *registry* datoteci. Ukoliko to nije slučaj u reinicijaliziranu bazu ove datoteke neće biti uključene. CIM skladište može se i ručno rekompajlirati uporabom *MOF compilera (MOFCOMP.EXE)* uključenog u svaku WMI instalaciju.




**Slika 3:** Do opcija CIM repozitorija dolazi se desnim klikom miša na WMI control i izborom Properties

## 3.2. WMI u praksi

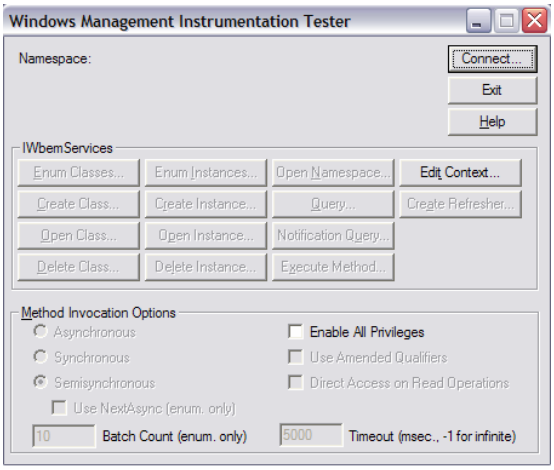
Možda najbolji način da se u potpunosti shvati WMI je primjena u praksi. Mnogi su načini kako se može koristiti WMI, a također su brojni i alati koji pri tom mogu biti od pomoći. U poglavlju koje slijedi opisani su najzanimljiviji alati i načini njihova korištenja, a kroz opise alata bit će dan i kratki uvod u WQL i osnove WSH-a potrebne za uspješno svladavanje primjera koji slijede.

### 3.2.1. Instalacija

WMI je sastavni dio instalacije Windows operacijskog sustava od verzije 2000 nadalje (uključivo tu i manje popularnu inačicu ME). Za uporabu sa starijim inačicama Windows operacijskog sustava, WMI je potrebno posebno instalirati. WMI je potrebno instalirati na svako računalo u mreži s kojim želimo upravljati, bez obzira izvršavaju li se skripte na njemu ili ne (naravno podrazumijeva se da svako računalo na kojemu se izvršavaju skripte također ima instaliran WMI). Instalacija WMI-a za starije operacijske sustave može se dohvatiti s adrese http://www.microsoft.com/downloads/details.aspx?FamilyID=afe41f46-e213-4cbf-9c5b-fbf236e0e875&DisplayLang=en.

### 3.2.2. Wbemtest i WQL

Prva aplikacija koja je opisana je `wbemtest`, koja dolazi sa svakom instalacijom Windows XP i Windows Server 2003 operacijskog sustava, a cilj joj je testiranje funkcionalnost WMI-a i istraživanje njegovih mogućnosti. `wbemtest` aplikaciju moguće je pokrenuti putem **Start -> Run** izbornika. Slika 4 prikazuje sučelje *Windows Management Instrumentation Tester* programa nakon pokretanja.
Prvi korak koji je potrebno učiniti kako bi se započela interakcija s WMI-em je povezivanje s WMI dobavljačem. Općenito, ovo znači povezivanje na WMI servis pokrenut na lokalnom ili nekom drugom računalu u mreži. Kao primjer je pokazano povezivanje na udaljeno računalo, jer upravo ova mogućnost WMI čini vrlo moćnom tehnologijom. Povezivanje se inicira pritiskom na gumb **Connect** što je prikazano na slici (Slika 5). U testne svrhe pokrenuta je testna domena u VMware okruženju s Windows Server 2003 i Windows XP računalima. Ime domene je `wmi.test`, a povezivanje je ostvareno s Windows Server 2003 računala (poslužitelj) na klijentsko računalo `wmi-client` s Windows XP Pro operacijskim sustavom. Ista konfiguracija koristit će se prilikom analize ostalih alata i skriptnih tehnika.

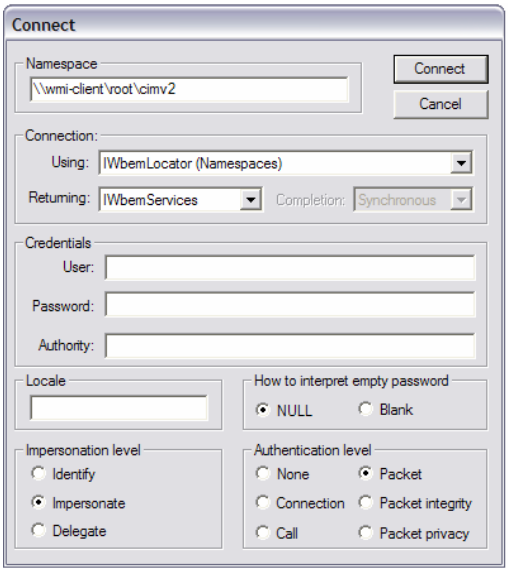



**Slika 4:** Sučelje programa wbemtest.exe tik nakon pokretanja

Za povezivanje na računalo dovoljno je upisati `\\IME_RACUNALA\root\cimv2`, gdje je:

- `IME_RACUNALA` - ime računala s kojim se želimo povezati (u našem slučaju wmi-client),
- `\root` - korijenski imenski prostor WMI-a,
- `\cimv2` - CIMv2 imenski prostor koji sadrži sve Win32 razrede s kojima će korisnici najviše raditi.

Ukoliko se spajanje i postavljanje upita odvija uz administratorske ovlasti na poslužitelju, nije potrebno unijeti korisničko ime i zaporku. U drugim slučajevima potrebno se prijaviti kao korisnik koji ima lokalne administratorske ovlasti. Pritiskom na gumb **Connect** započinje proces spajanja (Slika 5.)



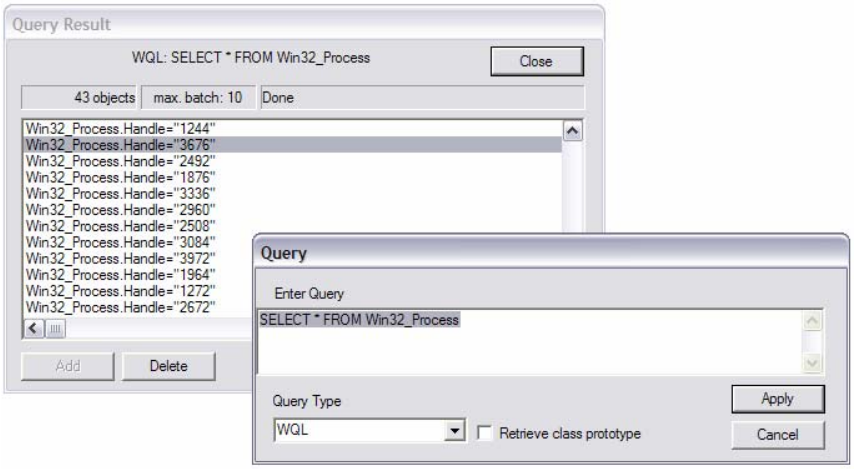

**Slika 5:** Spajanje na WMI provider udaljenog klijenta

Nakon uspješnog spajanja na udaljeno računalo svi gumbi u `IWbemServices` kolekciji glavnog sučelja postaju omogućeni. Prije postavljanja bilo kakvih upita preporučljivo je izvršiti "enumeraciju" razreda pritiskom na gumb **Enum Classes** kako bi ostvarili pristup svim raspoloživim klasama. Tijekom procesa enumeracije pojavit će se upit o *superclass* imenu. Ovo polje je potrebno ostaviti prazno te označiti opciju **Recursive,** kako bi se dobio popis svih objekata. U dijaloškom okviru koje se zatim prikazuju svi razredi u CIM i Win32 imenskom prostoru kojima imate pristup.

Sljedeći zadatak, koji ujedno demonstrira velike mogućnosti WMI-a, je dohvaćanje informacija kreiranjem jednostavnih upita. WMI podržava poseban jezik WQL (eng. *WMI Query Language*), vrlo sličan SQL-u. U svrhu primjera pokazan je postupak dohvaćanja informacije o procesima koji se izvršavaju na udaljenom računalu. Upiti se postavljaju pritiskom na gumb **Query**. U novo otvorenom prozoru potrebno je upisati slijedeći izraz:

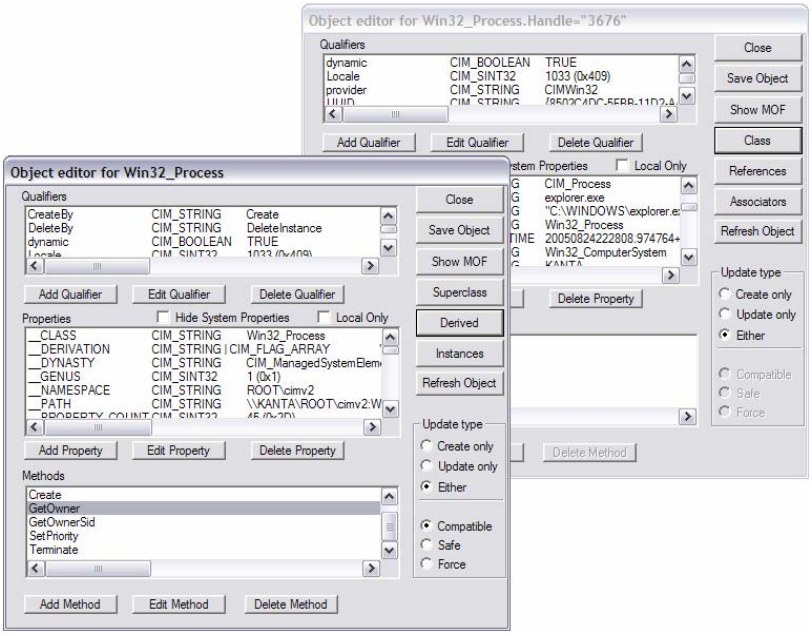
```
SELECT * FROM Win32_Process
```

te pritisnuti na gumb **Apply**. Ovaj upit dohvatit će sva svojstva instance razreda `Win32_Process` i prikazati ih u **Query Result** prozoru (Slika 6.).




**Slika 6:** Postavljanje upita i ispis dohvaćenih informacija

Budući da je u primjeru korišten znak zvjezdice, u postavljenom upitu dobivena su svojstva do kojih je moguće doći dvostrukim klikom na pojedinu instancu `Win32_process` razreda. Neka to primjerice bude proces s 3676 PID brojem. Otvorit će se novi prozor u kojem je moguće vidjeti sva svojstva izabranog procesa. S desne strane sučelja pružaju se dodatne mogućnosti, među kojima je najzanimljivija ona pod imenom **Class,** koja će popuniti prozor s metodama. Na ovaj način moguće je jednostavno vidjeti koje metode je moguće primijeniti u skriptama (primjerice moguće je stvoriti novi proces ili prekinuti izvršavanje izabranog); nakon izbora **Class** novi prozor nam omogućuje uvid u klasu roditelja. Prijašnji gumb **Class** sada je postao **Superclass**, a njegovim izborom moguće je vidjeti informacije o razredu iz kojega je izveden razred `Win32_Process`. Gumb **Derived** prikazuje klase koje su izvedene iz klase `Win32_Process` (u ovom slučaju ni jedna), a **Instances** ponovno prikazuje sve instance `Win32_Process` razreda (Slika 7).




***Slika 7:*** *Prikaz svojstava i postupaka za izabranu instancu Win32_Process razreda*

`Wbemtest` je odličan alat za testiranje WQL upita prije uključivanja u produkcijske skripte, jer omogućuje izravan uvid u informacije koje pojedini upit dohvaća. WQL upiti su relativno jednostavni i nužno ih je poznavati za bilo koji aspekt uporabe WMI-a. Iz tog razloga će WQL jezik biti detaljnije opisan u nastavku dokumenta.

WQL sintaksa ima jednostavan oblik koji se sastoji od slijedećih elemenata:

1. **SELECT** – svaka WQL naredba mora započeti ovom ključnom riječju, nakon čega je moguće navesti točno određena svojstva odvojena zagradom, ili se može navesti znak zvjezdice (*) što znači dohvat svih svojstava,
2. **FROM** –razred u kojemu su definirana svojstva koja dohvaćamo,
3. **WHERE** – ključna riječ koju je moguće opcionalno uključiti za preciznije definiranje upita.

Jednostavan primjer opisane sintakse je sljedeći izraz:

```
SELECT * FROM Win32_Process WHERE Name="emule.exe"
```

Navedeni izraz omogućit će dohvat svih svojstava instance razreda Win32_Process, čije je ime emule.exe. Za razliku od SQL jezika, koji ima na stotine ključnih riječi, WQL ih ima samo 20 (Tablica 1) te je izuzetno jednostavan za uporabu. Iako postoje i drugi načini kako dohvatiti informacije iz WMI skladišta, najjednostavnije je upravo korištenjem WQL upita. Navedeni primjer može se proširiti još i logičkim operatorima, operatorima usporedbe i ključnom riječju LIKE. Oni se dodaju nakon ključne riječi WHERE.

- Dva su booleova operatora koja se koriste u WQL jeziku, AND i OR. Npr.

```
SELECT * FROM Win32_LogicalDisk WHERE (DriveType = 2) OR
```

```
(DriveType = 3 AND FreeSpace < 1000000)
```

- Operatori usporedbe. Osnovni operatori usporedbe su **=, >, <, >=, <=, <>** ili !=, operator jednakosti, veće od, manje od, veće jednako, manje jednako, i nejednakosti redom. Kada se koristi usporedba s NULL vrijednošću tada se ne koriste operatori jednakosti ili nejednakosti već ključne riječi IS i IS NOT. Npr.

```
SELECT * FROM Win32_LogicalDisk WHERE FileSystem IS NOT NULL
```

- LIKE ključna riječ određuje poklapa li se dohvaćeni niz znakova s određenim uzorkom. Npr. sljedeći upit vraća sve instance Win32 klasa.

```
SELECT * FROM Meta_Class WHERE __Class LIKE "Win32"
```

Naravno, ovo su samo primjeri koji imaju široku primjenu i koji zadovoljavaju potrebe ovog dokumenta, za sve ostale preporuča se pogledati msdn referenca na sljedećoj adresi: http://msdn.microsoft.com/library/en-us/wmisdk/wmi/wql_sql_for_wmi.asp

| WQL - ključne riječi | | | | |
|---|---|---|---|---|
| __CLASS | ASSOCIATORS OF | HAVING | REFERENCES OF | GROUP |
| AND | BY | FALSE | TRUE | SELECT |
| OR | WITHIN | LIKE | ISA | IS |
| NOT | NULL | WHERE | FROM | KEYSONLY |

**Tablica 1:** ključne riječi WQ jezika

### 3.2.3. WMI Administrative Tools

WMI Administrative Tools paket skup je dodatnih alata za testiranje WMI komponenti i mogućnosti. Predviđen je za rad s operacijskim sustavima Windows generacije 2000 i više a može se dohvatiti s adrese http://www.microsoft.com/downloads/release.asp?ReleaseID=40804. Alati koji se nalaze u paketu su slijedeći:

- *WMI CIM Studio* – omogućuje pregled i uređivanje razreda, svojstava i instanci u WMI skladištu, izvršavanje metoda te generiranje i prevođenje MOF datoteka.
- *WMI Object Browser* – omogućuje pregled objekata, uređivanje vrijednosti svojstava i izvršavanje metoda pojedinog objekta
- *WMI Event Registration Tool* – dopušta podešavanje potrošača događaja (eng. *event consumer*), stvaranje ili pregled instanci *filter, binding* i *timer* razreda
- *WMI Event Viewer* – prikazuje događaje svih instanci ili registriranih potrošača

Svi navedeni alati temelje se na ActiveX komponentama i prikazuju se u Internet Explorer Web pregledniku. U nastavku su opisani WMI CIM Studio i WMI *object browser*.

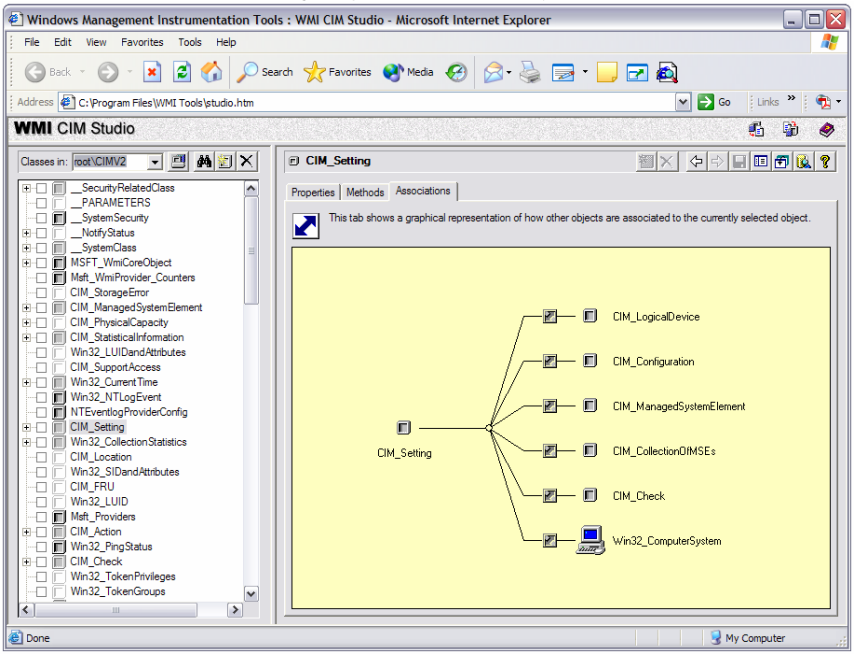
**WMI CIM Studio** je svojevrsna zamjena alata `wbemtest`, koja podržava pregled klasa u svakom imenskom prostoru te detaljan uvid u svojstva, metode i veze svakog razreda. Asocijacija (eng. *associations*) predstavlja odnos između dva ili više objekata, te se kaže da je razred asocijativan ukoliko sadrži dvije ili više reference na druge razrede. Navedeni primjer prikazan je na slici (Slika 8). `CIM_Setting` je asocijativna klasa i sadrži reference na `CIM_LogicalDevice`, `CIM_Configuration` itd. Prilikom prvog pokretanja potrebno je navesti imenski prostor kojim se želi upravljati, a na raspolaganju je standardni skup (Tablica 2).

| Imenski prostori | |
|---|---|
| root/SECURITY | root/Microsoft/HomeNet |
| root/SecurityCenter | root/DEFAULT |
| root/WMI | root/DEFAULT/ms_409 |
| root/WMI/ms_409 | root/directory |
| root/CIMV2 | root/directory/LDAP |
| root/CIMV2/ms_409 | root/directory/LDAP/ms_409 |
| root/CIMV2/Applications | root/subscription |
| root/CIMV2/Applications/MicrosoftACT | root/subscription/ms_409 |
| root/CIMV2/Applications/MicrosoftIE | root/MSAPPS11 |
| root/Microsoft | root/Cli |

**Tablica 2:** standardni skup imenskih prostora

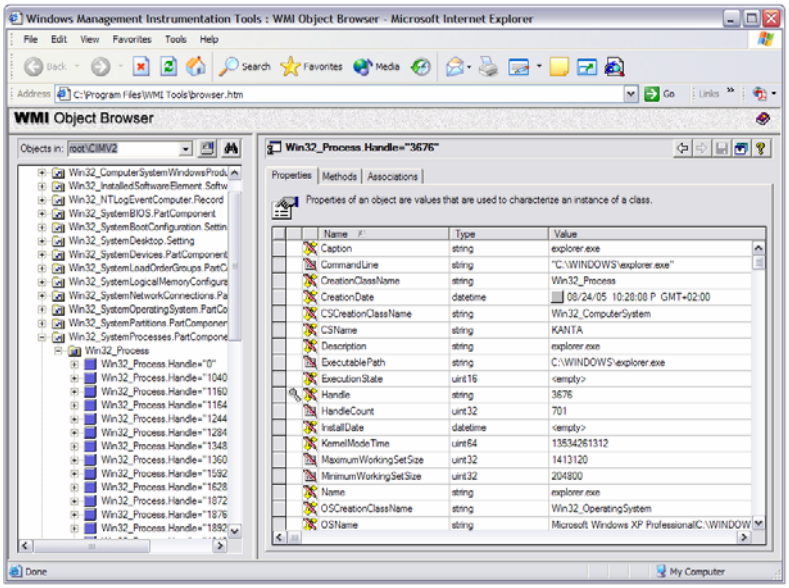
Ovaj skup različit je od računala do računala, te se do liste podkorijenskih imenskih prostora dolazi pritiskom na gumb **Browse For Namespace** u desnom gornjem dijelu sučelja i upisom traženih

informacija u otvoreni dijalog. Za **Machine Name** se upisuje ime računala do čijeg se imenskog prostora želi doći, a za **Starting Namespace** potrebno je opisati *root.* Pritiskom na gumb **Connect,** u okviru niže, dobivamo popis dostupnih imenskih prostora.
Sučelje WMI CIM studija je pregledno, desna strana sadrži popis svih razreda pojedinog imenskog prostora, a odabirom pojedinog razreda ili podrazreda, na lijevoj strani moguće je vidjeti pripadajuća svojstva. WMI CIM studio predviđen je i za testiranje WQL upita. Izborom drugog gumba s lijeve strane, u novootvorenom izborniku moguće je izvršavati WQL upite.



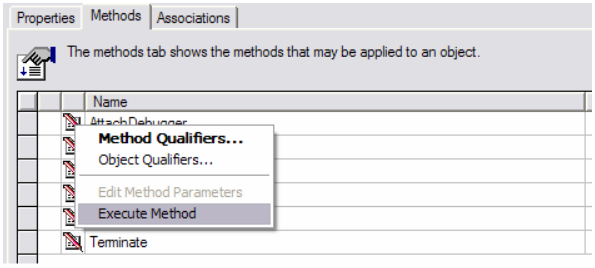

**Slika 8:** Sučelje WMI CIM studia

**WMI Object Browser** – nakon pokretanja WMI Object Browsera te spajanja na `root/CIMV2` imenski prostor (lokalno ili udaljeno) moguće je vidjeti popis svih razreda čije instance su aktivne na računalu/računalima s kojima je ostvarena veza (Slika 9). Ovo je bitna razlika između WMI CIM studia, budući da ovdje imamo pristup "korisnim" klasama, a ne cijelom skupu. WMI Object Browser se povezuje na *Win32_ComputerSystem* razred koji predstavlja model fizičkog računala i sadrži sve informacije sukladno tome. Sučelje je jednako onome WMI CIM studia. Nakon izbora određene instance razreda (desni dio sučelja), primjerice procesa s 3676 PID brojem iz Win32_SystemProcesses skupa, na lijevoj strani moguće je vidjeti sva svojstva i još važnije, pripadajuće metode koje je moguće izvršavati. Za navedeni primjer ti postupci su *AttachDebugger, Create, GetOwner, GetOwnerSID, SetPriority, i Terminate,* a izvršiti ih je moguće odabirom kartice **Method** odabirom željene metode te iz kontekstnog izbornika odabirom opcije **Execute Method** (Slika 10).




**Slika 9:** Sučelje WMI Object Browsera pri radu sa svojstvima odabranog procesa

Ovisno o odabranom razredu različit je i popis metoda koje je moguće izvršavati, a upravo ovakav način prikaza informacija bitno pomaže pri izradi vlastitih skripti. Najvažniji korak u procesu pisanja skripti je određivanje informacija koje se žele dohvatiti i što je moguće učiniti s pripadajućim objektima. Object Browser upravo ovaj korak, proces odluke koji su objekti potrebni i što se s njima želi radit, znatno olakšava svojim jednostavnim i preglednim prikazom razreda, svojstava i metoda.




**Slika 10:** Izvršavanje odabranog postupka

# 4. WMI kao temelj security managementa

## 4.1. WMIC

WMIC je izuzetno moćan alat koji se pokreće putem naredbenog retka, dizajniran kako bi olakšao dohvaćanje WMI informacija. Način na koji to postiže je uporaba tzv. *aliasa* – jednostavnih komandi koje iza sebe skrivaju konkretne WMI (WQL) upite. Zbog jednostavnosti uporabe i robusnosti može se koristiti na brojne načine i u brojne svrhe, a neki aspekti uporabe ovoga alata biti će obrađeni u ovom poglavlju.

WMI konzola se pokreće upisivanjem `wmic` naredbe u konzolu. Prilikom prvog pokretanja sustav će instalirati potrebnu podršku i ući u interaktivni način rada prikazujući u naredbenom retku slijedeće:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>wmic
wmic:root\cli>
```

Interaktivni način rada omogućuje neprekidan rad s WMI-jem odnosno upis slijeda naredbi bez potrebe za novom inicijalizacijom WMI providera (eng. *console mode*). Za izlaz iz interaktivnog načina rada potrebno je upisati naredbu `exit` ili `quit`. Ukoliko se `wmic` klijentu pri samom pokretanju doda željena naredba, ona će se izvršiti, ali program neće ući u interaktivni rad već će prekinuti s radom. Program `wmic` dostupan je samo na Windows XP/Server 2003 sustavima, i nije ga moguće uključiti ili instalirati na starijim inačicama Windows sustava.
WMIC klijentom moguće je upravljati pomoću nekoliko naredbi te konfiguracijskih prekidača ( navedene naredbe i popis svih aliasa prikazan je u tablici (Tablica 3). Do ovih informacija moguće je doći zadavanjem naredbe **`/?`**. U interaktivnom načinu rada dovoljno je upisati ovu naredbu za popis svih podržanih naredbi i aliasa, a za detalje o pojedinoj naredbi ili aliasu dovoljno je upisati odgovarajuće ime i znak `/?` (npr. `/node /?`). Popis navedenih naredbi i aliasa dan je u sljedećim tablicama (Tablica 3, Tablica 4).

| Prekidač | Opis |
|---|---|
| `/NAMESPACE` | Path for the namespace the alias operates against |
| `/ROLE` | Path for the role containing the alias definitions |
| `/NODE` | Servers the alias will operate against |
| `/IMPLEVEL` | Client impersonation level |
| `/AUTHLEVEL` | Client authentication level |
| `/LOCALE` | Language ID the client should use |
| `/PRIVILEGES` | Enables or disables all privileges |
| `/TRACE` | Outputs debugging information to stderr |
| `/RECORD` | Logs all input commands and output |
| `/INTERACTIVE` | Sets or resets the interactive mode |
| `/USER` | User to be used during the session |
| `/PASSWORD` | Password to be used for session login |
| `/?[:<BRIEF\|FULL>` | Usage information |

**Tablica 3:** globalni prekidači u WMIC okruženju

| Alias | Opis |
|---|---|
| `BASEBOARD` | Provides access to the baseboard (also known as a motherboard or system board) |
| `BIOS` | Provides access to the attributes of the computer system's basic input/output services (BIOS) that are installed on the computer |
| `BOOTCONFIG` | Provides access to the boot configuration of a system |
| `CDROM` | Provides access to the CD-ROM drives on a computer system |
| `COMPUTERSYSTEM` | Provides access to the computer system operating in a user environment |
| `CSPRODUCT` | Corresponds to software and hardware products used on a computer system |
| `DEVICEMEMORYADDRESS` | Provides access to the device memory addresses on a system |
| `DIRECTORY` | Provides access to the directory entries on a computer system |
| `DISKDRIVE` | Describes a physical disk drive as seen by a computer |
| `DMACHANNEL` | Provides information on the direct memory access (DMA) channel on a computer system |
| `DRIVERVXD` | Provides access to the virtual device driver/s on a computer system |
| `ENVIRONMENT` | Provides access to the system environment settings on a computer system. |
| `GROUP` | Provides access to data about a group account |
| `IRQRESOURCE` | Provides access to the interrupt request line (IRQ) number on a computer system |

| Alias | Opis |
|---|---|
| LOADORDER | Provides access to the group of system services that define execution dependencies |
| LOGICALDISK | Provides access to the data source that resolves to an actual local storage device on a system |
| LOGICALMEMORY | Allows access to the configuration layout and examination of the availabe memory on a system |
| NETADAPTER | Provides access to the network adapters installed on a system |
| NETADAPTERCONFIG | Provides access to and allows changing the attributes and behaviors of a network adapter |
| NETCONNECTION | Provides access to the active network connection in an environment |
| NETLOGIN | Provides access to the network login information of a particular user on a system |
| NETPROTOCOL | Provides access to the protocols and their network characteristics on a computer system |
| NTEVENTLOG | Allows access to the NT eventlog file |
| ONBOARDDEVICE | Describes common adapter devices built into the motherboard (system board) |
| OS | Provides access to the operating system/s installed on a computer |
| PAGEFILE | Provides access to the files used for handling virtual memory file swapping on a system |
| PAGEFILESET | Provides access to the settings of a page file |
| PARTITION | Provides access to the capabilities and management capacity of a partitioned area of a physical disk on a system |
| PHYSICALMEMORY | Allows access to details about the computer system's physical memory |
| PORT | Provides access to the I/O ports on a computer system |
| PORTCONNECTOR | Provides access to the physical connection ports |
| PRINTER | Provides access to all printer devices connected to a computer system |
| PRINTERCONFIG | Allows review of the configuration for a printer device |
| PRINTJOB | Provides access to the print jobs generated by an application |
| PROCESS | Provides access to the sequence of events on a system |
| PRODUCT | Correlates tasks to a single installation package |
| RECOVEROS | Provides access to the types of information that will be gathered from memory when the operating system fails |
| REGISTRY | Provides access to the computer system registry |
| SCHEDULEDJOB | Provides access to the jobs scheduled using the schedule service |
| SERVER | Provides access to the server information |
| SERVICE | Provides access to the service applications on a computer system |
| SHARE | Allows access to the shared resources on a system |
| SOFTWAREELEMENT | Provides access to the elements of a software product installed on a system |
| SOFTWAREFEATURE | Provides access to and allows changing of the software product subsets of SoftwareElements |
| STARTUP | Allows access to the command that runs automatically when a user logs onto the computer system |
| SYSACCOUNT | Allows access to the system accounts |
| SYSDRIVER | Provides access to the system driver for a base service |
| SYSTEMENCLOSURE | Provides access to the properties associated with a physical system enclosure |

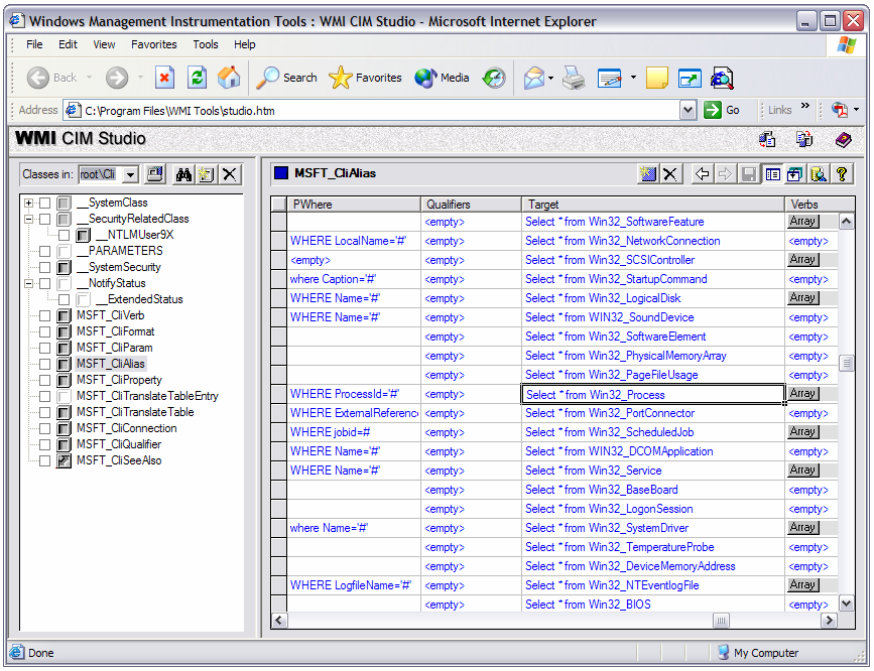
| Alias | Opis |
|---|---|
| SYSTEMSLOT | Provides access to the physical connection points including ports, slots and peripherals, and proprietary connections points |
| TAPEDRIVE | Describes a tape drive on a computer |
| TEMPERATURE | Allows access to the properties of a temperature sensor (electronic thermometer) |
| TIMEZONE | Provides access to the time-zone information for a system |
| UPS | Provides access to the capabilities and management capacity of an uninterruptible power supply (UPS) |
| USERACCOUNT | Provides access to information about a user account on a system |
| VOLTAGE | Allows access to the properties of a voltage sensor (electronic voltmeter) |
| WMISET | Provides access to and allows changes to be made to the operational parameters for the WMI service |

**Tablica 4:** Podržani ALIAS-i u WMIC-u

Na sljedećem primjeru pokazano je koliko je dohvaćanje i upravljanje informacijama jednostavno putem WMIC klijenta. Primjer će biti uspoređen sa postupkom izvršavanja istih zadataka putem ranije opisanih WQL jezika i `wbemtest` programa. Za ovaj zadatak u `wbemtest` programu potrebno je bilo povezati se na udaljeni ili lokalni WMI provider, obaviti enumeraciju, otkriti koji imenski prostor i klasa je zadužena za upravljanje procesima te na kraju kreirati i izvršiti WQL upit. U `wmic` klijentu ovaj proces sveden je na najjednostavniji mogući oblik. Isti zadatak u `wmic`-u može se obaviti izvršavanjem sljedećih naredbi:

```
wmic /node:"wmi-client" process
```

Pozivanjem `wmic` programa na ovaj način, program neće ući u interaktivni način rada zbog navedenih naredbi kao argumenata, već će program ispisati sve instance razreda `Win32_Process` (i njihova svojstva) koji se izvršavaju na udaljenom računalu. `/node` parametar WMI-u govori na koje računalo da se poveže (`wmi-client`) i da dohvati sve procese uporabom aliasa **process**. Aliasi su apstraktni sloj koji skriva izvorni način dohvaćanja informacije putem WQL. Svakom aliasu odgovara jedan WQL upit. Korisno je znati kako su aliasi pohranjeni u WMI skladištu kao instance *MSFT_CliAlias* razreda u **root/cli** imenskom prostoru (Slika 11.)

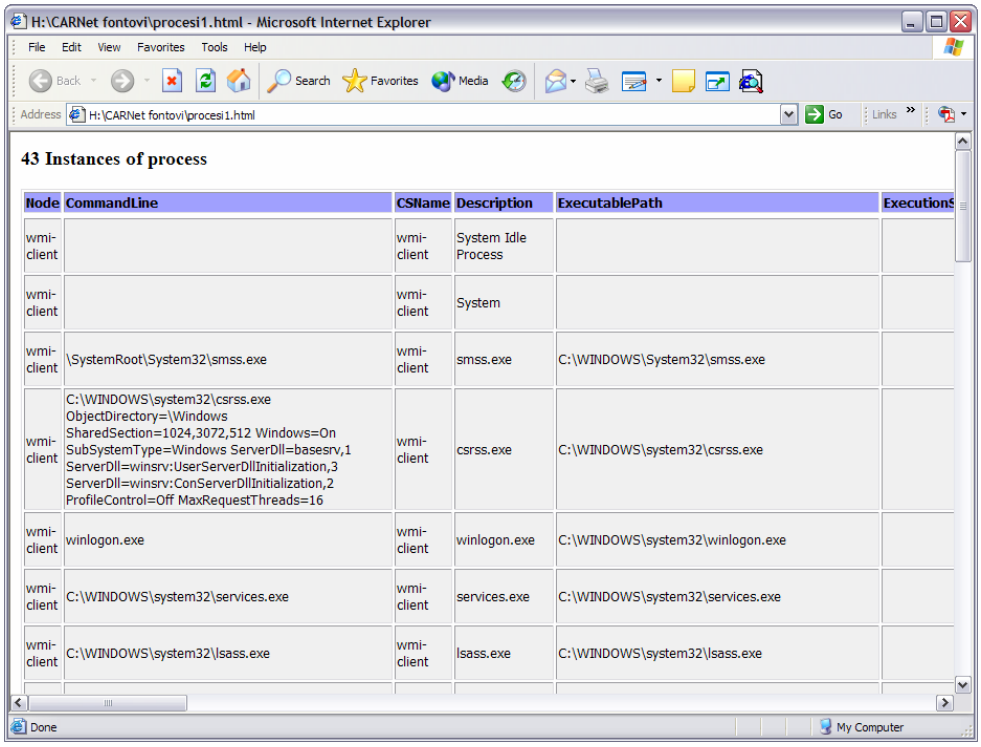



**Slika 11:** Prikaz Aliasa u WMI repozitoriju pomoću WMI CIM studia

`Wmic` klijent podržava još brojne prekidače i naredbe, što ga čini izuzetno moćnim alatom osobito pri pisanju skripti. Kasnije u dokumentu, na primjeru *batch* skripti biti će detaljnije opisan ovaj koncept, a u nastavku slijedi opis parametara `Wmic` klijneta koje omogućuju izvoz podataka u html (pored html-a podatke je još moguće izvoziti u XML i MOF formate). WMIC interno izvozi podatke u XML, no uporabom odgovarajućih XSLT shema vrši se pretvorba u HTML. Na raspolaganju nam je desetak XSLT shema no moguće je kreirati i druge vlastite. Izvorne sheme nalaze se u direktoriju: `\%SystemRoot%\System32\WBEM`. Izvoz se obavlja naredbom koja umjesto ispisivanja podataka na standardni izlaz, generira jednostavan XML oblik podataka (`/translate:basicxml`), formatira ih koristeći shemu `test.xsl` (`/format:test.xsl`), ili stvara html datoteku zadanog formata – (`/Output:izlaz.html`). Npr.

```
wmic /output:ServiceList.HTM /node:"wmi-client" process list
/translate:basicxml /format:htable.xsl
```

Rezultat izvršavanja prikazane naredbe dan je na slici (Slika 12).
WMIC je jednostavan i moćan alat, koji se može iskoristiti u brojne svrhe, gotovo i bez poznavanja WQL-a i WSH-a. U nastavku su prikazana dva primjera u kojima se WMIC može pokazati iznimno korisnim alatom.

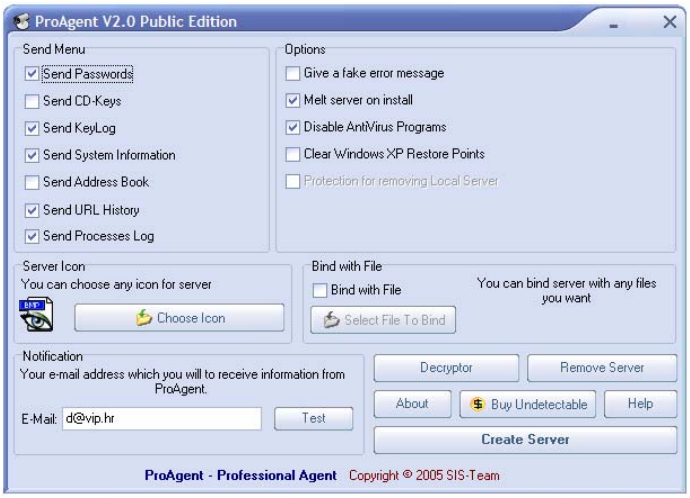



**Slika 12:** Generirani html dokument s listom procesa i svim svojstvima

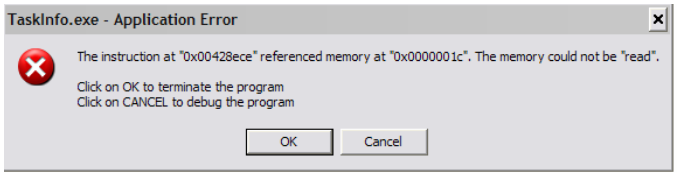
### 4.1.1. WMIC u ulozi rootkit revealera

Danas su maliciozni programi sve napredniji i sve ih je teže otkriti. Sve se češće koriste rootkit tehnike koje ujedno onemogućuju vatrozidnu, odnosno antivirusnu zaštitu. Jedan od takvih *rootkit* trojanskih konja je SIS Teamov ProAgent (Slika 13, Slika 14) dostupan na adresi http://www.spyinstructors.com. Ovaj trojanski konj koristi *rootkit* tehnike kako bi se sakrio od korisnika i pri tome slao povjerljive informacije. Neke tehnike koje koristi su "DLL injection" u proces Internet Explorera Web preglednika koji se ne vidi, skrivanje od **Task Managera** zaobilaženjem *ntdll.NtQuerySystemInformation* poziva (Phrack Volume 0x0b, Issue 0x3e), onemogućavanje antivirusne zaštite. Ovaj trojanski konj nije moguće detektirati ni na koji način, jer čak onemogućuje i tzv. *third-party task managere* koji ne koriste spornu funkciju za prikaz procesa (npr. taskinfo – http://www.iarsn.com). Ukoliko pri ruci nije

dostupan neki od naprednijih programa za otkrivanje rootkit programa (eng. *rootkit revealer*), WMI u ovom pogledu može znatno pomoći.




**Slika 13:** Proagent – trojanski konj koji koristi napredne tehnike prikrivanja...




**Slika 14**: ...i koji onemogućuje programe (taskinfo) koji ga mogu otkriti

Gornji primjer u kojem je pokazan postupak dohvaćanje liste procesa koji se izvršavaju na sustavu, primijenit će se i u ovom slučaju. U listi procesa moguće je primijetiti jednu instancu Internet Explorer Web preglednika, koja nije vidljiva nigdje osim u Windows Management Instrumentation konzoli. Ovo je sasvim dovoljna indicija da na sustavu nešto nije u redu i da se poduzmu daljnje detekcijske mjere. Nekoliko je mjera koje je moguće poduzeti, a za sve je korištena mogućnost izvršavanja metoda putem `Wmic` klijenta. Otprije je poznato kako se uz pojedinu instancu nekog razreda vežu i pripadajuće metode kojima je moguće izvršavati određene akcije nad željenim instancama. U danom primjeru dvije su metode koje su primjenjive, a to su *AttachDebugger* i *Terminate* metode. Na sljedećem primjeru prikazana je mogućnost gašenja sumnjivog procesa. U naredbeni redak potrebno je upisati sljedeću naredbu:

```
wmic process where name="iexplore.exe" terminate
```

Nakon izvršavanja naredbe dobit ćemo obavijest o uspješnom terminiranju procesa.

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>wmic process where name="iexplore.exe" terminate
Executing(\\wmi-client\ROOT\CIMV2:Win32_Process.Handle="2520")-
>startservice()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ReturnValue = 0;
};
```

Ovaj aspekt uporabe WMI-a pokazuje da WMI ima ogroman potencijal ne samo u pisanju automatiziranih skripti za obavljanje administrativnih poslova, već i definiranju sigurnosne politike mreže, mrežnih servisa i korisnika. Bolji primjeri navedenih funkcionalnosti dani su u idućem poglavlju.

### 4.1.2. Integracija u *batch* skripte

U nastavku je pokazan još jedan primjer upotrebe `Wmic` klijenta i to uključivanjem u *batch* skripte. Ovakav način uporabe omogućuje jednostavno obavljanje složenijih zadataka zahvaljujući mogućnosti definicije složenijih naredbi i upita koristeći skripte. Također, na ovaj način nije potrebno prilikom svakog zadatka ponovno upisivati isti niz naredbi.
WMIC *batch* skripte mogu imati i definirane varijable, te nije potrebno izravno u skriptu unositi imena računala ili neke druge vrijednosti kada se one mogu zadati i prilikom izvršavanja same skripte. Varijable se definiraju u formatu %1, %2 itd., i predstavljaju argumente koji se predaju skripti prilikom njenog izvršavanja. Alternativno, omogućeno je i kreiranje tekstualne datoteke koja sadrži CSV (eng. *Comma Separated Values*) listu varijabli, koja se zatim u skripti poziva dodavanjem simbola @. Kada su definirana imena računala *switch* **/node** naredba shvaća simbol **@** kao pokazivač na datoteku, a ne kao ime računala.

```
//kill.bat

@echo off
if "%1"=="" goto msg
if "%2"=="" goto msg

wmic  /output:"%1_processlist.txt"  /node:"%1"  process  get
name /format:textvaluelist.xsl

@echo on
wmic /node:"%1" process where name="%2" call terminate
goto end

:msg
echo Nedovoljan broj argumenata
echo Uporaba: kill.bat IME_RACUNALA IME_PROCESA
:end
```

Prikazana skripta kao argumente prima ime udaljenog računala i ime procesa koji će biti terminiran, ukoliko postoji u listi aktivnih. Prije terminiranja procesa, *batch* skripta stvara tekstualnu datoteku u kojoj se nalazi popis svih procesa na zadanom računalu. Ovaj jednostavan primjer pokazuje način na koji je moguće znatno ubrzati i olakšati brojne svakodnevne zadatke.

## 4.2. Skriptne tehnike u WMI okruženju

Prije opisa alata i razvojnog okruženja za pisanje skripti, slijedi kratki pregled skripti i ključnih komponenti u interakciji s WMI-em. Pisanje skripti sposobnih za interakciju s WMI-em uključuje uporabu WMI COM objekata. Skripte koriste dva tipa objekata: objekte sadržane u definiciji svakog razreda i objekte koje definira WMI API (COM objekti). Za uspješno pisanje skripti koje koriste COM apstrakcijski sloj nužno je razumijevanje objektnog modela, ne samo u WMI CIM objektnom modelu nego i u WMI COM modelu. Prijašnja poglavlja bavila su se upravo WMI CIM objektnim modelom, dok će u nastavku biti riječi o WMI COM modelu. Pokazano je kako je putem ovog API-ja moguće koristiti objekte iz WMI CIM repozitorija i na koji način je moguće dohvaćati informacije o WMI razredima.
WMI API (eng. *Application Programming Interface*) temelji se na COM objektima koji se mogu koristiti u sklopu brojnih programskih jezika (Visual Basic, C++, jezici .NET obitelji), ali i u sklopu skriptnih jezika kao što su VBScript, Jscript, Perl i sl. Prvi korak je instanciranje samih COM objekata, a pored

njih i razredi iz WMI skladišta također moraju biti instancirani kao objekti, kako bi omogućili rad s upravljivim WMI entitetima.
Dva su različita načina kako instancirati objekte u WMI-u:

1. Uporabom objekata s COM ProgID oznakom. U ovisnosti o programskom jeziku programski kod je: CreateObject za VBScript, new ActiveXObject za JScript, te XML <object> tag u ASP ili Windows Script File jeziku.
2. Uprabom **GetObject** naredbe za VBSrcript ili JScript

```
SWBemlocator = new ActiveXObject("WbemScripting.SWBemlocator");
```

Za upravljanje entitetom iz stvarnog svijeta potrebno se je povezati s objektom koji predstavlja dotični entitet u WMI objektnom modelu. Za uspostavu konekcije s WMI providerom potreban je objekt **objWMIServices,** koji se stvara putem WMI API poziva **ConnectServer** funkcije **SWbemLocator** objekta. Funkcija **ConnectServer** vraća objekt tipa **SWbemServices**. Zanimljivo svojstvo **SWbemLocator** objekata je mogućnost specificiranja dodatnih parametara kao što su drugi username i password ili lokalizacijski znakovni niz. **SwbemLocator** svojstva navedena su u tablici (Tablica 5).

| **SWbemLocator** | | |
|---|---|---|
| *Svojstva (Properties)* | Security_ | Čitanje ili promjena sigurnosnih postavki |
| *Postupci (Methods)* | ConnectServer [ strServer = "" ],<br>[ strNameSpace = "" ],<br>[ strUser = "" ],<br>[ strPassword = "" ],<br>[ strLocale = "" ],<br>[ strAuthority = "" ],<br>[ intSecurityFlags = 0 ],<br>[ objwbemNamedValueSet = null ] | Spajanje na WMI provider na zadanom računalu |

***Tablica 5:*** *Svojstva SWbemLocator objekta*

Npr.

```
objWMIService =
SWBemlocator.ConnectServer(strComputer,"/root/CIMV2");
```

Nakon povezivanja `/root/CIMV2` imenski prostor udaljenog računala potrebno je dohvatiti željene podatke. Nakon toga moguće je instancirati WMI razred tako da predstavlja stvarni entitet, a to čini objekt tipa **SWbemObject**. WMI objekti koji mogu stvoriti ovaj objekt su **SWbemServices, SWbemObjectSet** i **SWbemEventSource.** Za dohvaćanje WMI podataka potreban je SWbemServices objekt koji je korišten u sljedećem primjeru. **SWbemServices** objekt posjeduje veliki broj metoda, no u navedenom primjeru korištena je samo ExecQuery metoda (Tablica 6.)

| **SWbemServices** | | |
|---|---|---|
| *Svojstva (Properties)* | Security_ | Čitanje ili promjena sigurnosnih postavki |
| *Postupci (Methods)* | ExecQuery strQuery,<br>[ strQueryLanguage = "WQL" ],<br>[ intFlags = wbemFlagReturnImmediately ],<br>[ objWbemNamedValueSet = null ] | Izvršava WQL upit |

***Tablica 6:*** *Jedno od brojnih svojstava SWbemServices objekta*

Npr.

```
colItems = objWMIService.ExecQuery("Select * from
Win32_Process");
```

### 4.2.1. Skripte kao process firewall

Skripta koja koristi navedene mehanizme zamišljena je kao jednostavan *process firewall*. Skripta ispituje izvršava li se na udaljenom računalu neki proces koji ne bi smio. Ukoliko je to slučaj, WMI će ugasiti zabranjeni proces. Ovakvu provjeru moguće je periodički izvršavati, a preciznom definicijom pravila moguće je onemogućiti izvršavanje nepoželjnih procesa. Dobar primjer nepoželjnih procesa su popularne P2P aplikacije, koje stvaraju probleme brojnim sistem administratorima.
Ukoliko ne postoji mogućnost nabave nekog *statefull inspection* vatrozida, kontrola izvršavanja procesa putem WMI-a je itekako dobra alternativa.

```
// Simple_process_firewall.js

var strComputer;
var objWMIService;
var colItems;
var propEnum;
var objItem;

var SWBemlocator;

try {
       strComputer = "wmi-client";
       SWBemlocator = new ActiveXObject("WbemScripting.SWbemLocator");
       objWMIService=SWBemlocator.ConnectServer(strComputer,"/root/CIMV2");
       colItems = objWMIService.ExecQuery("Select * from Win32_Process");
       propEnum = new Enumerator(colItems);

       for (;!propEnum.atEnd();propEnum.moveNext()) {
              objItem = propEnum.item();

              if(objItem.Name!= null)
              {
                     WScript.Echo("Name: " + objItem.Name);
              }
              else
              {
                     WScript.Echo("Name: ");
              }
              if (objItem.Name == "emule.exe")
              {
                     objItem.Terminate();
                     WScript.Echo("Process " + objItem.Name + " terminated");
              }
       }
}
catch(e)  {
       WScript.Echo("catch caught " + e.number + " " + e.description);
}
```

Ukoliko ime procesa odgovara imenu *emule.exe,* proces će biti terminiran. Rezultat izvršavanja skripte dan je u nastavku:

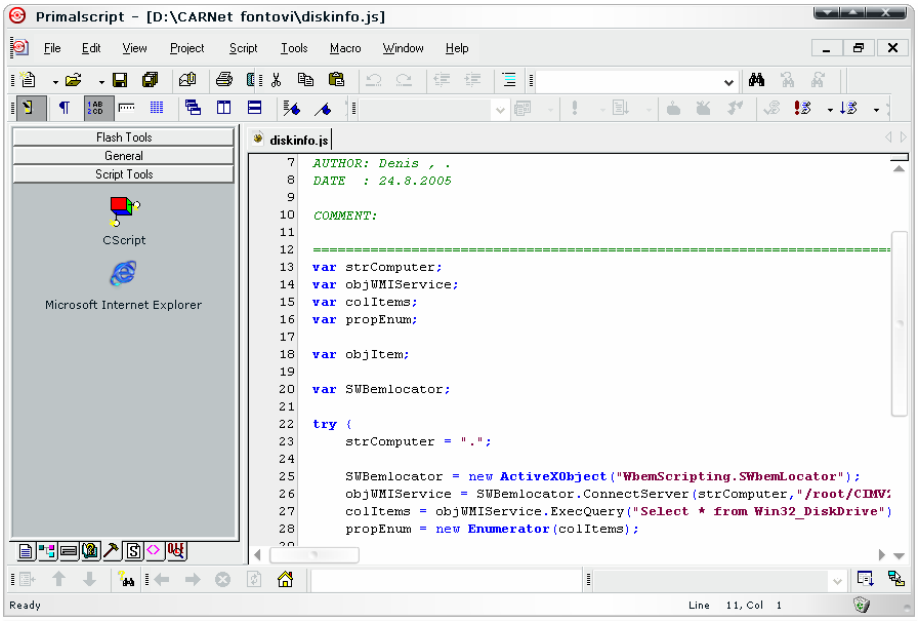
```
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Name: System Idle Process
Name: System
Name: smss.exe
Name: csrss.exe
Name: winlogon.exe
Name: services.exe
Name: lsass.exe
Name: svchost.exe
Name: svchost.exe
...
```

```
...
Name: PrimalScript.exe
Name: wmiprvse.exe
Name: emule.exe
Process emule.exe terminated
Name: cscript.exe
Exit code: 0 , 0000h
```

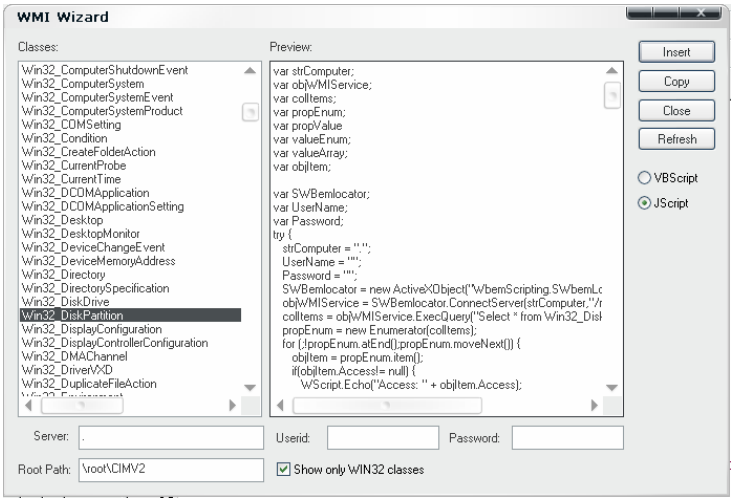
### 4.2.2. PrimalScript i Scriptomatic

Navedena skripta, kao i sve ostale, izrađene su u PrimalScript (www.sapien.com) editoru. Činjenica je kako na tržištu postoji izuzetno mali broj IDE-a za WMI, PrimalScript se smatra najboljim i biti će ukratko opisan u nastavku. PrimalScript (Slika 15) nije predviđen samo za JScript/VBScript, no ovdje je razmatran samo taj aspekt. Sučelje je pregledno i intuitivno, a integrirane su funkcije za izvršavanje i uklanjanje grešaka (engl. *debugger*) skripti, pa čak i za digitalno potpisivanje istih. Također, prisutno je i bojanje sintakse, a i predikcija naredbi funkcionalnost je koja olakšava korištenje alata. Za pisanje većeg broja skripti PrimalScript je pravi izbor.




**Slika 15:** PrimalScript – razvojno okruženje za WMI skripte

PrimalScript pored navedenih svojstava također omogućuje automatsko generiranje WMI skripti, nalik onome kod Scriptomatic alata opisanog u nastavku (Slika 17). Do WMI čarobnjaka se dolazi izborom **WMI Wizard** opcije u **Script** izborniku (Slika 16). Čarobnjak omogućuje izbor računala na koji se spaja, izbor imenskog prostora, iako je podrazumijevani `\root\cimv2`. Na lijevoj strani se nalazi popis svih razreda, a pritiskom na neki od njih u desnom dijelu se prikazuje gotova skripta koju je pritiskom na gumb **Insert** moguće ubaciti u razvojno sučelje Primalscripta. Mogući izbor skriptnih jezika je: JScript i VBScript.

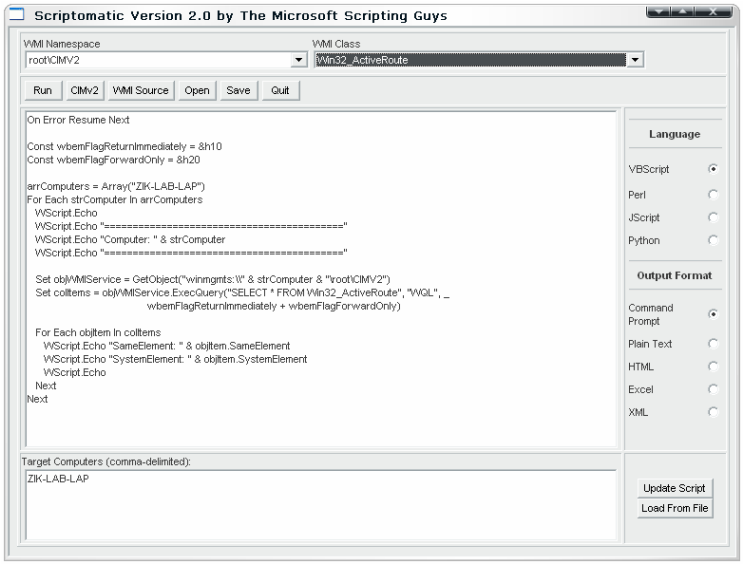



**Slika 16:** WMI Wizard – jednostavno generiranje WMI skripti

Pored ovoga *editora,* uvijek nam ostaje Visual Studio, no veliki problem VS-a je nepostojanje IntelISence podrške za skriptne jezike što znatno umanjuje njegovu funkcionalnost. Za ovo naravno postoji rješenje koje dolazi u vidu dodatka za Visual Studio, a zove se VisualAssistX (www.wholetomato.com) VisualAssistX je dodatak koji u velikoj mjeri povećava produktivnost svakog programera te nudi brojne napredne opcije u dopuni i predikciji koda. Ovo je savršena kombinacija, no cjenovno nije opravdano uspoređivati nju s Primalscriptom.

Drugi alat koji olakšava posao pisanja i skripti i pomaže manje iskusnim korisnicima je besplatni Microsoftov alat Scriptomatic trenutne inačice 2.0. Sličan je Primalscriptovom WMI wizardu no nešto je moćniji od njega. Alat se može dohvatiti s adrese http://www.microsoft.com/downloads/details.aspx?familyid=09DFC342-648B-4119-B7EB-783B0F7D1178&displaylang=en, a dolazi u vidu `hta` aplikacije.

Svrha ovoga alata je generiranje osnovnih skripti koje sadrže sva svojstva odabrane klase.Alat može raditi samo sa svojstvima razreda, ne i sa metodama, te radi isključivo s WMI-em (nema podrške za SNMP primjerice). Iako ima svojih ograničenja, koristan je u postupku učenja i u trenutcima kada smo ograničeni s vremenom. Omogućuje generiranje skripti u čak 4 jezika: Jscript, VBScript, Perl, Python, posjeduje konfiguraciju izlaza, koji može biti, standardni izlaz u naredbenom retku, običan tekst, HTML, XML, ili Excel format. Prednost je što možemo navesti imena računala na koje se želimo spojiti u donjem dijelu sučelja u vidu CSV liste, a dostupan je i standardan izbor imenskog prostora kao i sadržanih razreda.




**Slika 17:** Scriptomatic – alat za generiranja skripti

## 4.3. WMI događaji

Nakon upoznavanja s WQL-om i WMI API skriptnim tehnikama, ovdje će biti prikazano još jedno napredno svojstvo WMI-a. Riječ je o događajima (eng. *events*). Do ovog trenutka svaka skripta koja je bila temeljena na WMI-u obavljala je neke zadatke nad određenim entitetom objektnog modela. Ovo se ne može smatrati u potpunosti ispravnim nadgledanjem, te je zbog toga WMI uveo obavijest o događajima (eng. *event notification*). Upravljanje nekom okolinom ne znači samo obavljanje zadataka nad entitetima već i mogućnost reagiranja na promjene u toj okolini. Ovo dakako implicira kako tehnologija koja se koristi u zadacima za upravljanje može detektirati te promjene. Upravo te promjene WMI naziva događajima i oni čine vjerojatno najmoćniji dio WMI-a. Jasno je kako je vjerojatnost pojave različitih događaja u jednom velikom sustavu velika, stoga je bitno pravilno ocijeniti koji će događaji biti praćeni. Pored ovoga kada je neki događaj detektiran WMI mora znati kome će ga dostaviti te naposljetku kakve akcije će se izvršiti.

Za detekciju događaja WMI koristi posebne dobavljače koji se nazivaju *event providers*, npr. SNMP *event provider*, NT Event Log *event provider*, *Registry event provider* i drugi. Za prihvaćanje-detekciju događaja WMI klijenti moraju biti "pretplaćeni" na WMI događaje, u postupku "pretplate" WMI klijent dostavlja dvije stvari: događaje za koje se želi pretplatiti te akcije koje će se izvršiti kada se taj događaj dogodi. Takvi klijenti mogu biti aplikacije kao primjerice Wbemtest, WMI skripte ili neka druga aplikacija koja koristi WMI API. WMI pretplatnik definira događaje za koje se želi pretplatiti uporabom WQL jezika, sljedeći redak pokazuje jedan takav primjer:

```
SELECT * FROM RegistryTreeChangeEvent WHERE Hive=
'HKEY_LOCAL_MACHINE' AND RootPath=''
```

Kako se definiraju akcije koje će se izvršiti tijekom detekcije događaja biti će opisano u poglavlju koje slijedi.

Svaki *event provider* sadrži pripadni WQL upit koji je pohranjen u EventQueryList opisniku. Najzanimljiviji *event provideri* dani su u tablici (Tablica 7).

| Namespace | Event Provider | Event Query lIst |
|---|---|---|
| Root/CIMV2 | MS_Power_Management_Event_Provider | select * from Win32_PowerManagementEvent |
| | SystemConfigurationChangeEvents | select * from Win32_SystemConfigurationChangeEvent |
| | MS_Shutdown_Event_Provider | select * from |

| Namespace | Event Provider | Event Query lIst |
|---|---|---|
| | | Win32_ComputerShutdownEvent |
| | VolumeChangeEvents | select * from Win32_VolumeChangeEvent |
| | Microsoft WMI Forwarding Event Provider | select * from MSFT_ForwardedEvent |
| | RouteEventProvider | select * from Win32_IP4RouteTableEvent |
| | WMI Kernel Trace Event Provider | select * from Win32_ProcessStartTrace |
| | | select * from Win32_ProcessStopTrace |
| | | select * from Win32_ThreadStartTrace |
| | | select * from Win32_ThreadStopTrace |
| | | select * from Win32_ModuleLoadTrace |
| | MS_NT_EVENTLOG_EVENT_PROVIDER | select * from __InstanceCreationEvent where TargetInstance isa "Win32_NTLogEvent" |
| | Win32ClockProvider | select * from __InstanceModificationEvent where TargetInstance isa "Win32_CurrentTime" |
| Root/DEFAULT | RegistryEventProvider | select * from RegistryEvent |
| Root/MicrosoftCluster | Cluster Event Provider | select * from MicrosoftCluster_Event |
| Root/snmp/localhost | MS_SNMP_REFERENT_EVENT_PROVIDER | select * from SnmpExtendedNotification |
| | MS_SNMP_ENCAPSULATED_EVENT_PROVIDER | select * from SnmpNotification |
| Root/subscription | Microsoft WMI Template Event Provider | select * from __InstanceOperationEvent WHERE TargetInstance ISA "MSFT_TemplateBase" |
| | Microsoft WMI Transient Event Provider | select * from MSFT_TransientEggTimerEvent |
| | | select * from __InstanceOperationEvent where TargetInstance isa "MSFT_TransientStateBase" |
| | Microsoft WMI Forwarding Consumer Trace Event Provider | select * from MSFT_FCTraceEventBase |
| | Microsoft WMI Transient Reboot Event Provider | select * from MSFT_TransientRebootEvent |
| | Microsoft WMI Updating Consumer Event Provider | select * from MSFT_UCTraceEventBase |
| | | select * from MSFT_UCEventBase |
| Root/WMI | WMIEventProv | select * from WMIEvent |

**Tablica 7**: Lista najzanimljivijih dobavljača događaja

### 4.3.1. Hvatanje WMI događaja u svrhu nadgledanja registry datoteke

Za primjer uporabe WMI događaja, s fokusom na sigurnosne aspekte, izabrana je *registry* datoteka, pošto brojni maliciozni programi upravo modificiraju ključeve upravo u *registry* datoteci. Ukoliko nismo u mogućnosti u potpunosti zabraniti promjenu *registry* vrijednosti, rješenje donosi nadgledanje promjena vrijednosti u *registry* bazi. WMI događaji upravo su predviđeni za definiranje ovakve sigurnosne politike, a primjer jednostavne skripte koja nadgleda promjene u HKEY_LOCAL_MACHINE dijelu pojasnit će detaljnije skriptnu tehniku koja koristi ove napredne mehanizme.

```
//registry monitoring.js

var strComputer = "wmi-client";
```

```
var wmiServices =
GetObject("winmgmts:{impersonationLevel=impersonate}!//" +
strComputer + "/root/default");

var wmiSink = WScript.CreateObject("WbemScripting.SWbemSink",
"SINK_");


wmiServices.ExecNotificationQueryAsync (wmiSink,
    "SELECT * FROM RegistryTreeChangeEvent WHERE Hive= " +
     "'HKEY_LOCAL_MACHINE' AND RootPath=''") ;

WScript.Echo ("Listening for Registry Change Events...");
while(true)
{
    WScript.Sleep(1000);
}

function SINK_OnObjectReady(wmiObject, wmiAsyncContext)
{    WScript.Echo ("Received Registry Change Event" + ":\n" +
                "------------------------------" +
                wmiObject.GetObjectText_());
}
```

U navedenoj skripti se mogu naći dvije bitne stvari koje nisu bile pokazane do sada, a to su `impersonationLevel` sigurnosna opcija u pozivu GetObject() funkcije te asinkrono izvršavanje akcija tijekom detekcije definiranog događaja. *Impersonationlevel = impersonate* znači poziv i izvršavanje funkcije na udaljenom stroju u ime izvršitelja skripte (u ovom slučaju administratora, s obzirom da se pod njegovim ovlastima skripta izvršava na poslužitelju). Druge moguće vrijednosti *Impersonationlevel* opcije su: *anonymous, identify* i *delegate.* (za detalje vidjeti WMI referencu). Akcije koje se izvršavaju tijekom nekog događaja definirane su u `SINK_OnObjectReady` funkciji. Asinkrono izvršavanje omogućuje paralelno istovremeno upravljanje primljenim događajima. U nastavku je prikazan izlaz skripte.

```
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights
reserved.

Listening for Registry Change Events...
Received Registry Change Event:
------------------------------
instance of RegistryTreeChangeEvent
{
      Hive = "HKEY_LOCAL_MACHINE";
      RootPath = "";
      TIME_CREATED = "127709524965526160";
};

Received Registry Change Event:
------------------------------
instance of RegistryTreeChangeEvent
{
      Hive = "HKEY_LOCAL_MACHINE";
      RootPath = "";
      TIME_CREATED = "127709525298104384";
};

Exit code: 1 , 0001h
```

Svaka promjena ključa u HKEY_LOCAL_MACHINE dijelu registryja izaziva izvršavanje gore spomenute funkcije te ispis vremena kada je do promjene došlo.

# 5. Zaključak

Ovaj dokument pokušao je objasniti osnovne koncepte WMI-a. Prikazan je rad s alatima koji ne zahtijevaju programsko iskustvo i koji su ilustrirali moć WMI-a. Bitno je bilo shvatiti osnovni koncept objektnog modela i razreda te način dohvaćanja informacija iz CIM skladišta. Prikazani su alati koji su većim dijelom dio operativnog sustava i kroz koje je prikazano kako dohvaćati informacije iz WMI skladišta, iz ove specijalizirane baze entiteta. Uz uporabu WQL moguće je postavljati bilo kakve upite što je osnova za pisanje skripti. U drugom dijelu opisane su skriptne tehnike kojima se mogu obavljati i automatizirati brojni zadaci, te su prikazani neki alati čija je svrha ubrzati i olakšati proces pisanja skripti. WMI je se pokazao kao sustav koji je dorastao svakom zadatku vezanom za upravljanje objektima u informacijskim sustavima, bez obzira bila to samo provjera hardwareskih komponenti pojedinih računala u mreži ili implementacija sigurnosne politike kao što je kontrola pristupa, *process firewall* ili zaštita sustava od malicioznih programa. Ova izuzetno moćna i robusna tehnologija pruža zaista puno a granica iskoristivosti leži samo u našoj osobnoj kreativnosti.

# 6. Reference

[1] http://msdn.microsoft.com/library/en-us/wmisdk/wmi/wmi_reference.asp
[2] MSDN Traininig - Scripting Microsoft® Windows® Management Instrumentation – Course Number 2439A
[3] Jscript 5.5 Reference, MS Press, 2000.
[4] Microsoft Windows 2000 Scripting Guide, MS Press, 2002.
[5] Leveraging WMI Scripting: Using Windows Management Instrumentation to Solve Windows Management Problems, Digital Press, 2003.