Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Jussi Haaja

Opinnäytetyö

Windows-työasemien kirjautumiskomentosarjojen toteuttaminen Perl-ohjelmointikielellä

Työn ohjaaja: Ville Haapakangas, FM
Tilaaja: YH Länsi Oy, tietohallinto
Tampere 12/2008

Tampereen ammattikorkeakoulu
tietojenkäsittelyn koulutusohjelma, tietoverkkopalvelut

Tekijä: Jussi Haaja
Työn nimi: Windows-työasemien kirjautumiskomentosarjojen toteuttaminen Perl-ohjelmointikielellä
Sivumäärä: 43
Valmistumisaika: Joulukuu 2008
Työn ohjaaja: Ville Haapakangas
Työn tilaaja: YH Länsi Oy, tietohallinto

TIIVISTELMÄ

Kirjautumiskomentosarjat ovat olennainen osa lähes jokaisen suuremman Windows-verkkoympäristön päivittäistä toimintaa. Niiden avulla määritetään käyttäjille tarpeelliset palvelut sekä helpotetaan ylläpitäjien työtaakkaa automatisoimalla rutiinitehtäviä.

Tämän työn tarkoituksena on tutkia, millaisia menetelmiä käyttäen on helpointa toteuttaa Windowsin kirjautumiskomentosarjoja Perl-ohjelmointikiellä. Perl on alkujaan lähinnä UNIX-palvelinten ylläpitäjien työkalu, mutta on käytettävissä myös Windows-ympäristössä.

Työn teoriaosassa käsitellään yleisesti Windows-toimialueen rakennetta, Windowsin kirjautumisprosessia ja kirjautumiskomentosarjojen osuutta tässä prosessissa. Perl-ohjelmointikieltä esitellään erityisesti sen käyttöä Windows-ympäristössä silmälläpitäen, mutta myös kielen perusteet esitellään.

Työn teknisessä osassa käsitellään Perlin käyttämistä kirjautumiskomentosarjojen toteuttamisessa toimintokohtaisesti. Jokaisen toiminnon osalta esitellään siihen olennaisesti liittyvät Perl-moduulit sekä esimerkein niiden käyttö. Teknisessä toteutuksessa kommentoidaan myös vähemmän ilmeisiä ongelmia, joita työn tekemisen aikana kohdattiin, sekä ratkaisuehdotuksia näihin.

Avainsanat Windows, Perl, lähiverkot, kirjautumiskomentosarjat

TAMK University of Applied Sciences
Department of Business Information Systems

Writer: Jussi Haaja
Thesis: Implementing Windows logon scripts with Perl programming language
Pages: 43
Graduation time: December 2008
Thesis Supervisor: Ville Haapakangas
Co-operating company: YH Länsi Oy, IT Department

ABSTRACT

Logon scripts are an atomic part of almost every Windows network environment. Using them, an administrator is able to automate daily routines and implement necessary network services for users.

The goal of this thesis is to examine which methods are available for implementing Windows logon scripts using the Perl programming language. Perl has historically been mostly a tool for UNIX administrators, but is also usable in a Windows environment.

The theory part of the thesis concentrates on the structure of a Windows domain, and the process of logging on to a Windows network, including the role of logon scripts therein. Perl is discussed mostly from the point of view of a Windows environment, but also the basic syntax of the language is included.

The technical part of the thesis contains detailed instructions on how to use Perl to develop logon scripts for a Windows network. For every function the logon script does, the function is discussed and the modules used to implement the function are showcased, including coding examples. The technical part also covers the problems encountered during the implementation, and offers solutions and workarounds.

Keywords Windows, Perl, local area networks, logon scripts

Sisällysluettelo

Lyhenteiden ja termien luettelo

Active Directory	Microsoftin Windows Server-käyttöjärjestelmien mukana toimitettava hakemistopalvelu, jossa säilytetään mm. toimialueen käyttäjien tietoja.
DNS	Domain Name System, Internetin nimihakemistopalvelu, jonka avulla verkkoon liitetyt tietokoneet voivat paikantaa verkon palveluita ja muita tietokoneita.
LDAP	Lightweight Directory Access Protocol, protokolla, jota Active Directory käyttää hakemistosta tehtäviin hakuihin. LDAP on avoin standardi, joka on määritelty Internet Engineering Task Forcen Request for Comments (RFC) –dokumentissa RFC 4510.
Scripting Engine	COM-moduuli (Component Object Model), jonka avulla Windows Scripting Host voi suorittaa COM-moduulin tukemalla ohjelmointikielellä tehtyjä komentosarjoja.
SQL	Structured Query Language, erityisesti tietokantojen käsittelyyn tarkoitettu komentokieli, jota myös Windows Management Instrumentation käyttää tietojen noutamiseen.
XML	Extensible Markup Language, yleisluontoinen määritelmä tietorakenteiden tallentamiseen tekstimuodossa. Esimerkiksi Websivuissa käytetty XHTML on XML-muotoista tietoa.
XML Schema	XML-muotoinen määrittely, jossa määritellään tietyn tyyppisessä XML-tiedostossa sallitut rakenteet ja tietotyypit.
WSH	Windows Scripting Host, Microsoftin Windows-käyttöjärjestelmissä käyttämä ohjelmointikieliriippumaton automaatioteknologia, joka on suunniteltu erityisesti komentosarjoja silmälläpitäen
WMI	Windows Management Instrumentation, Microsoftin kehittämä hallintateknologia, jonka avulla komentosarjoilla voidaan hallita ja tarkkailla Windows-käyttöjärjestelmän resursseja ja palveluita

1 Johdanto

Lähes jokainen Windows-pohjainen työasemaverkko hyödyntää toiminnassaan kirjautumiskomentosarjoja (ammattikielessä kirjautumisskripti). Ei ole silti tavatonta, että verkon käyttäjä ei tiedä mitään tällaisen järjestelmän olemassaolosta. Siitä huolimatta kirjautumiskomentosarjat ovat tärkeä osa Windows-verkon ylläpitotyötä, koska ne mahdollistavat verkon peruspalvelujen (esimerkiksi levyjaot ja verkkotulostimet) helpon ja nopean saatavuuden heti verkkoonkirjautumishetkestä lähtien. Osaava ylläpitäjä voi hyödyntää kirjautumiskomentosarjoja lukuisiin muihinkin toimenpiteisiin, kuten pikakuvakkeiden luomiseen, ohjelmien perusasetusten asettamiseen sekä tiedotteiden esittämiseen käyttäjälle. Käytännössä mahdollisuuksia rajoittaa ainoastaan kirjautumiskomentosarjojen kirjoittamiseen käytetty ohjelmointikieli.

Ajatus kirjautumiskomentosarjoista ei suinkaan ole uusi. UNIX-käyttöjärjestelmissä on ollut olemassa jo varhaisista versioista lähtien ominaisuus, jossa järjestelmän ns. shell-ohjelma suorittaa tietyn komentojonotiedoston käynnistyessään. Windowsin kirjautumiskomentosarjat noudattavat tätä samaa periaatetta.

1.1 Taustaa

Työn toimeksiantajana toimi YH Länsi Oy:n tietohallinto. YH Länsi Oy:n Windows-verkko käsittää noin sata Windows XP –työasemaa useassa eri toimipisteessä. Kaikki työasemat käyttävät samaa kirjautumiskomentosarjaa. Ajatus työn tekemisestä syntyi heinäkuussa 2008, kun vanhoihin kirjautumiskomentosarjoihin tehtiin muutoksia. Vanhoissa KiXtart-kielellä toteutetuissa kirjautumiskomentosarjoissa huomattiin puutteita, erityisesti liittyen virhetilanteiden käsittelyyn. Lopulta päädyttiin siihen lopputulokseen, että komentosarjat olisi parasta toteuttaa uudelleen jollakin KiXtartia monipuolisemmalla ohjelmointikielellä.

Ensimmäinen tehtävä oli valita sopiva ohjelmointikieli komentosarjoja varten. Lyhyehkön tutkimustyön jälkeen vaihtoehdot oli rajattu kahteen ohjelmointikieleen: Pythoniin ja Perliin. Lopulta päädyttiin käyttämään Perliä, lähinnä paremman dokumentaation ansiosta. Saatavilla oleva informaatio koskien kirjautumiskomentosarjojen toteuttamista

Pythonilla oli työtä aloitettaessa varsin vähäistä. Perlin puolella tilanne oli merkittävästi parempi, joten se valittiin toteutuskieleksi.

1.2 Työn tavoite

Työn tavoitteena oli tutkia Windows-toimialueen kirjautumiskomentosarjojen toteuttamista Perl-ohjelmointikielellä. Työn tekemisen aikana tuli vastaan useasti tilanteita, joissa jonkin ongelman ratkaisemiseen oli olemassa lukuisia vaihtoehtoja. Iso osa varsinaista työtä olikin käydä läpi eri vaihtoehtoja ja pyrkiä valitsemaan niistä sopivin ja toimivin. Usein tarvittavia tiedon jyväsiä tarvitsi myös etsiä monesta eri lähteestä ja koostaa järkeväksi kokonaisuudeksi. Tässä työssä esitellään kerätyn tiedon pohjalta parhaaksi havaittuja menetelmiä.

Työn toisena, käytännönläheisempänä tavoitteena oli tuottaa kerätyn tiedon pohjalta toimivat ja käyttökelpoiset kirjautumiskomentosarjat YH Länsi Oy:lle. Viimeisessä luvussa palataan käytännön toimivuuden arviointiin lähemmin.

Luvussa 2 esitellään työn kannalta olennaista taustatietoa ja teoriaa Windows-ympäristöstä. Luvussa 3 esitellään Perliä ohjelmointikielenä niiltä osin, kuin kieltä on tarpeellista tuntea luvun 4 esimerkkejä ajatellen. Luku 4 on omistettu käytännön teknisen toteutuksen raportoinnille, ja siinä esitellään käytettyjä ratkaisuja ja yleisiä ongelmakohtia.

2 Windows-toimialue, kirjautuminen ja komentosarjat

Koska kirjautumiskomentosarjat on varsin tiukasti sidoksissa Windows-verkkoihin ja –toimialueisiin, on luontevaa aloittaa perusteista ja tarkastella hieman ympäristöä jossa kirjautumiskomentosarjoja hyödynnetään.

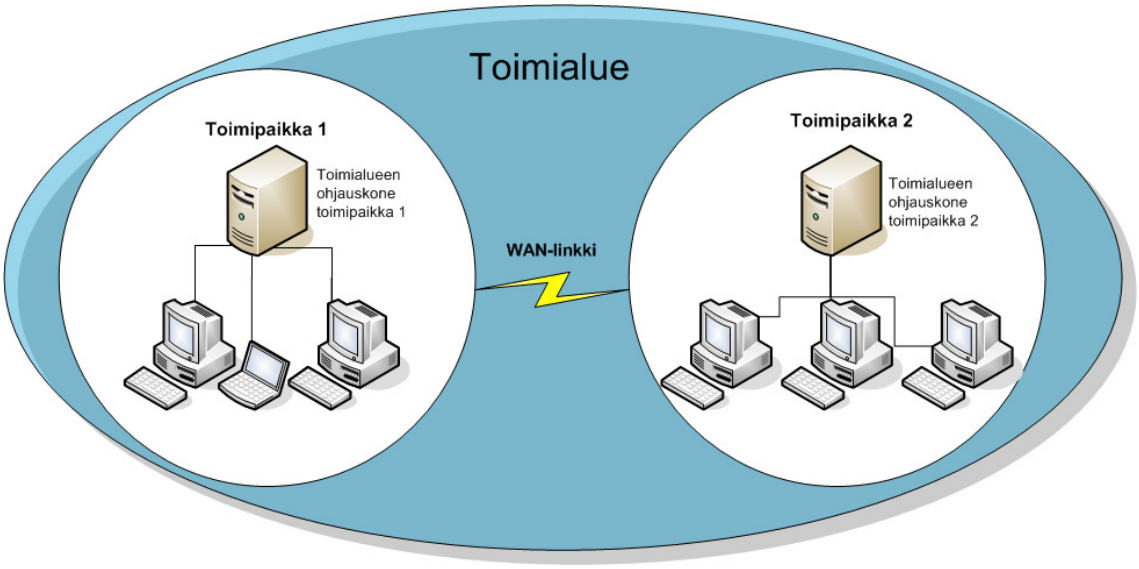
Seuraavassa käydään lävitse sekä Windows-toimialueen rakennetta, että siihen olennaisesti liittyvää kirjautumisprosessia. Lopuksi esitellään kirjautumiskomentosarjoihin ja niiden konfigurointiin liittyviä asioita.

2.1 Windows-toimialue

Windows-verkkoympäristöistä puhuttaessa toimialueella (domain) tarkoitetaan hallinnollisesti rajattua aluetta, joka koostuu yhdestä tai useammasta toimialueen ohjauskoneesta (domain controller) ja siihen liitetyistä asiakaskoneista. Keskeistä toimialueelle on myös ohjauskoneitten välinen replikaatio, eli tietojen kopioiminen ja synkronointi näiden laitteiden välillä. (Clines & Loughry, 2008, 12-13.)

Toinen keskeinen toimialueisiin liittyvä käsite on toimipaikka (site). Toimipaikalla tarkoitetaan yhtä tai useampaa toisiinsa nopealla verkkoyhteydellä liitettyä IP-aliverkkoa. Toimipaikkoja käytetään toimialueen fyysisen rakenteen määrittelyyn. Jokaisessa toimipaikassa on vähintään yksi toimialueen ohjauskone. (Clines & Loughry, 2008, 16.)

Windows-toimialueen käsite on vahvasti sidoksissa saman nimiseen DNS-käsitteeseen (Domain Name System). Windows-toimialueet yksilöidäänkin DNS-nimien perusteella (esimerkiksi yritys.local) (Clines & Loughry, 2008, 21). Kuvassa 1 esitetään kahdesta toimipaikasta koostuvan Windows-toimialueen looginen rakenne.



Kuva 1. Windows-toimialueen looginen rakenne

2.2 Kirjautumisprosessi

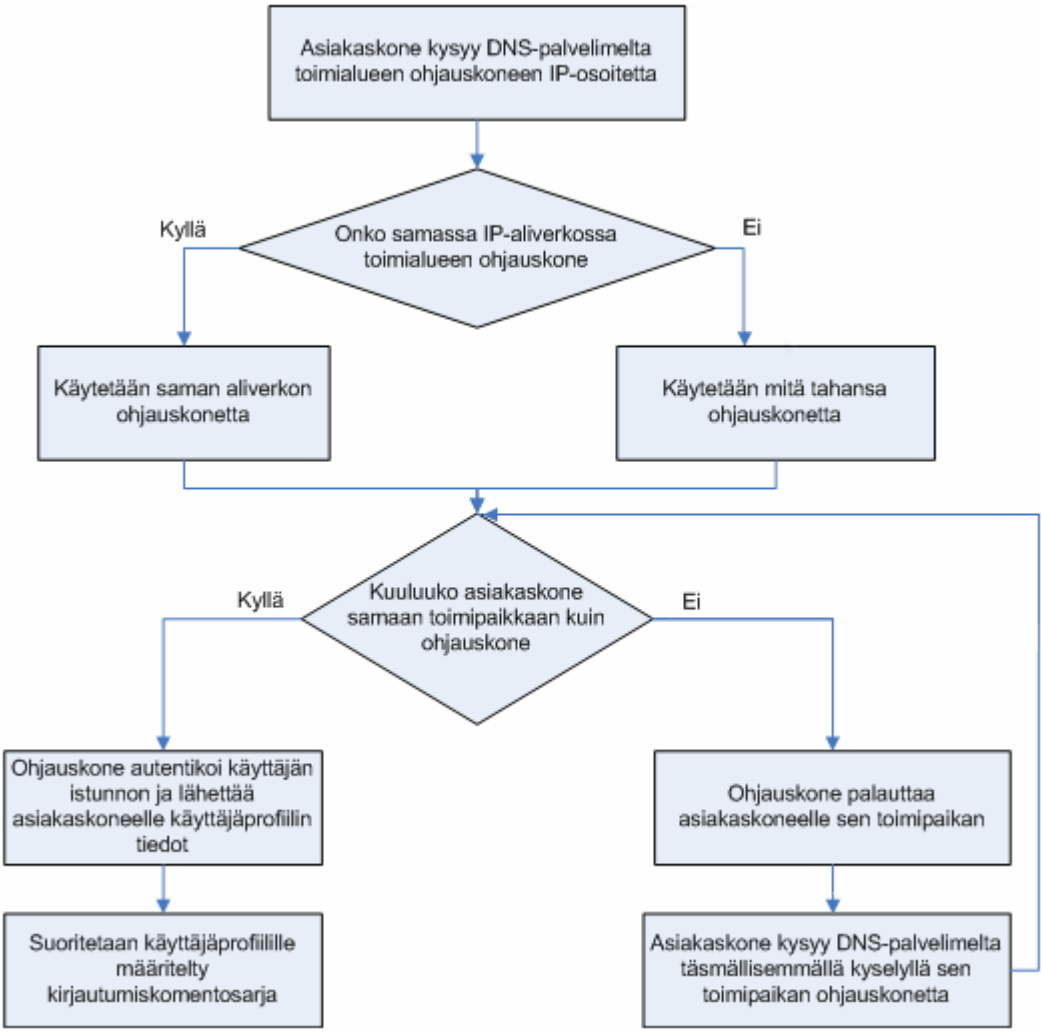
Kirjautuakseen toimialueelle asiakaskoneen täytyy paikantaa toimialueensa lähin ohjauskone. Tämä paikannus tapahtuu hyödyntämällä DNS-nimipalvelun SRV-tietueita. Asiakaskone lähettää DNS-palvelimelle kyselyn seuraavan muotoisesta tietueesta:

_LDAP._TCP.dc._msdcs.toimialue

Tällä kyselyllä yritetään paikantaa LDAP-palvelinta (Lightweight Directory Access Protocol) TCP-protokollalle toimialueella *toimialue*. (Microsoft Help and Support, 2002.)

Saatuaan DNS-palvelimelta listan toimialueensa LDAP-palvelimista asiakaskone kysyy DNS-palvelimelta LDAP-palvelimien IP-osoitteet. Asiakaskone pyrkii suosimaan samassa IP-aliverkossa olevaa palvelinta, mutta jos tällaista ei ole saatavilla, asiakaskone valitsee listasta jonkin muun palvelimen. Tämän jälkeen asiakaskone avaa yhteyden palvelimeen päästäkseen käsiksi toimialueen LDAP-hakemiston (Active Directory) tietoihin. Osana yhteyden käsittelyä ohjauskone päättelee asiakaskoneen IP-osoitteesta mihin toimipaikkaan asiakaskone kuuluu. Jos asiakaskone kommunikoi väärän (jonkin muun kuin lähimmän mahdollisen) ohjauskoneen kanssa, ohjauskone lähettää asiakkaalle tiedon asiakaskoneen toimipaikasta. Tämän jälkeen asiakaskone voi kysyä DNS-palvelimelta tarkempaa SRV-tietuetta, johon on liitetty myös toimipaikan nimi. (Microsoft Help and Support, 2002.)

Kun toimipaikan ohjauskone on paikallistettu, autentikoidaan käyttäjän istunto ja lopulta asiakaskone voi noutaa LDAP-palvelimelta käyttäjätiliin liittyvät tiedot, mukaan lukien suoritettavan kirjautumiskomentosarjan polun (Microsoft Help and Support, 2002). Kaaviossa 1 esitetään kirjautumisprosessi asiakaskoneen näkökulmasta siihen pisteeseen asti, jossa kirjautumiskomentosarjaa voidaan alkaa suorittaa.

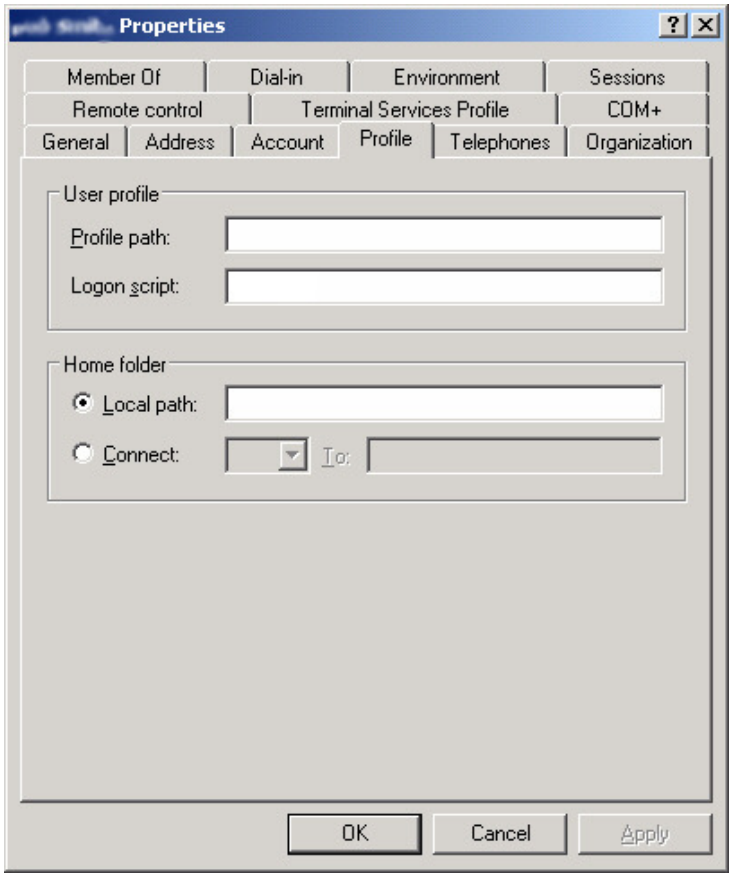


Kaavio 1. Windows-asiakkaan kirjautuminen toimialueelle

2.3 Komentosarjat

Kirjautumiskomentosarja voidaan määrittää joko ryhmäkäytännön (group policy) kautta, tai käyttämällä Active Directory Users and Computers –hallintakonsolia. Käytettäessä ryhmäkäytäntöjä kirjautumiskomentosarjojen määrittämiseen on tärkeää pitää mielessä, että Windows 2000:tta vanhemmat Windows-käyttöjärjestelmät eivät huomioi tällä tavalla asetettuja kirjautumiskomentosarjoja. Jos toimialueella on vielä käytössä

Windows 95-, 98-, ME- tai Windows NT –käyttöjärjestelmillä varustettuja tietokoneita, on kirjautumiskomentosarjat asetettava hallintakonsolin kautta käyttäjäkohtaisesti. (Mueller, 2007.) Käytettäessä hallintakonsolia kirjautumiskomentosarjan polku asetetaan kuvan 2 mukaisesti kohtaan ”Logon script”.



Kuva 2. Käyttäjän kirjautumiskomentosarjan määrittäminen hallintakonsolin kautta.

On myös huomioitava, että jos jostain syystä molempia tapoja kirjautumiskomentosarjojen määrittämiseen on käytetty, koneilla jotka tukevat ryhmäkäytäntöjä (Windows 2000- ja Windows XP –työasemat) suoritetaan sekä ryhmäkäytännön määrittämä komentosarja, että käyttäjäprofiilin määrittämä komentosarja. (Mueller, 2007.)

Käyttäjän kirjautumiskomentosarjaa voi myös muokata ohjelmallisesti muokkaamalla Active Directoryn käyttäjäkohtaista scriptPath-attribuuttia. Tämä on suositeltava menetelmä silloin, kun halutaan muokata kerralla suuren käyttäjämassan tietoja. (Mueller, 2007.)

Kirjautumiskomentosarjat varastoidaan toimialueen ohjauskoneille, hakemistoon %SystemRoot%\sysvol\sysvol\toimialue\scripts missä toimialue on toimialueen DNS-nimi. Tämä hakemisto näkyy ohjauspalvelimilla jaettuna hakemistona nimeltä NETLOGON. Oletuksena kaikilla toimialueen käyttäjillä on lukuoikeudet tähän levyjakoon. (Mueller, 2007.)

Kirjautumiskomentosarjan toteutuskieli riippuu käytössä olevista käyttöjärjestelmistä. Vain Windows 2000- ja Windows XP -työasemat tukevat kirjautumiskomentosarjojen toteuttamista Windows Scripting Host:n avulla. Windows Scripting Host mahdollistaa kirjautumiskomentosarjojen toteuttamisen suoraan millä tahansa ohjelmointikielellä, jolle on toteutettu WSH scripting engine. Vanhemmat Windows-perheen käyttöjärjestelmät tukevat vain .bat ja .exe –päätteisiä kirjautumiskomentosarjoja. (Mueller, 2007.)

3 Perl

Tässä luvussa esitellään Perl-ohjelmointikieltä luvun 4 suunnitteluratkaisujen ja koodiesimerkkien ymmärtämisen helpottamiseksi. Tämän luvun tarkoitus ei ole käsitellä syvällisesti Perlin syntaksia tai sisäistä toimintaa, vaan lähinnä tarjota riittävä viitekehys seuraavalle luvulle.

Luvussa esitellään myös muita Perliin olennaisesti liittyviä asioita, kuten laaja Perl-tietolähde Comprehensive Perl Archive Network sekä käytetyin Perlin Windows-jakelu ActiveState ActivePerl.

3.1 Perlin historia

Perlin historia alkaa vuodesta 1986, jolloin kielen isä Larry Wall työskenteli järjestelmäohjelmoijana Unisysillä. Työssään Larry törmäsi perinteisten UNIX-työkalujen awk:n ja sed:n silloisten versioitten rajoituksiin, ja päätti kehittää niille tehokkaamman korvaajan. Perlin ensimmäinen versio 1.0 julkaistiin 18. joulukuuta 1987. Perlin kehityksen kannalta seuraava keskeinen askel oli Programming Perlin ensimmäisen painoksen julkaisu vuonna 1991. (Wall, Christiansen, Orwant, 2000, 646-647.)

Perlin nykyinen pääversio Perl 5 julkaistiin lokakuussa 1994. Perlin tällä hetkellä uusin versio on 5.10.0, joka on julkaistu 18. joulukuuta 2007. Tässä opinnäytetyössä on käytetty kyseisestä versiosta johdettua ActiveStaten julkaisemaa jakelua.

3.2 Lyhyesti Perlin syntaksista ja tietorakenteista

Perlin syntaksi on lainannut paljon vaikutteita C-ohjelmointikielestä, sen johdannaisista sekä UNIXin komentotulkkiohjelmoinnista (esim. Linuxissa laajasti käytetystä bash-tulkista). Perlin syntaksi onkin varsin helppo omaksua, jos omaa aiempaa kokemusta ohjelmoinnista esimerkiksi C, C++, Java tai PHP-kielillä. (Wall, ym. 2000, 86-111.)

3.2.1 Syntaksi

Jokainen Perl-lauseke sisältää yhden tai useampia operaattoreita tai funktioita, ja päättyy aina puolipisteeseen. Esimerkiksi:

```
print ”Hei maailma!”;
$muuttuja = ”arvo”;
open(HANDLE, $tiedosto) or die “Cannot open $tiedosto”;
```

Kaikki ylläolevat ovat kelvollisia Perl-lausekkeita. Ensimmäinen tulostaa tekstin ”Hei maailma!” standardiulostuloon (yleensä komentoriville, josta ohjelma ajetaan). Seuraava sijoittaa tekstin ”arvo” muuttujaan muuttuja. Viimeinen yrittää avata muuttujan tiedosto määrittelemän tiedoston HANDLE-nimiseen tiedostokahvaan, ja pysäyttää komentosarjan suorituksen jos tämä ei onnistu.

3.2.2 Muuttujat ja tietotyypit

Perl on dynaamisesti tyypitetty ohjelmointikieli. Tämä tarkoittaa sitä, että toisin kuin esimerkiksi C-kielessä tai Javassa, muuttujan sisältämä arvo voi vaihtaa tyyppiä (esimerkiksi numerosta tekstiin) kesken ohjelman suorituksen. Perlissä on myös vain yhden tyyppisiä arvoja. Niistä käytetään Perl-termistössä nimeä scalar. (Wall ym. 2000, 51.)

Perlin perustietorakenne on niin sanottu scalar. Muuttuja sisältää yhden arvon, luvun tai merkkijonon. Perl tukee muuttujan ohella myös kahta muuta tietorakennetta. Ensimmäinen näistä on array. Array sisältää yhden tai useampia scalar-arvoja. Toinen keskeinen tietorakenne on hash. Hash sisältää yhden tai useampia scalar-avain/arvo-pareja. Sekä array- että hash-rakenteet voivat sisältää myös toisia hash- tai array-rakenteita ja niin edelleen. Näin voidaan luoda monimutkaisia, monitasoisia tietorakenteita. (Wall ym. 2000, 51.)

Perl-ohjelmakoodissa eri tietorakenteiden eteen merkitään ne toisistaan erottava merkki, $, @ tai % taulukon 1 mukaisesti.

Taulukko 1: Perlin muuttujatyyppien etuliitteet

Muuttujatyyppi	Etuliite
Yksittäinen arvo	$
Array	@
Hash	%

Asia voi vaikuttaa vähäpätöiseltä, mutta koodiesimerkkien ymmärtäminen ilman tätä seikkaa voi olla vaikeaa. Väärän etumerkin käyttäminen saattaa kaataa ohjelman, ja vähintäänkin saa ohjelman toimimaan jollakin ei-odotetulla tavalla.

3.2.3 Moduulit

Perl käyttää moduuleita helpottamaan koodin uudelleenkäyttöä. Perl-moduuli muodostuu mistä tahansa .pm-päätteisestä, Perl-koodia sisältävästä tiedostosta (vrt. Perlin normaali tiedostopääte .pl). Moduuleita voi tehdä helposti itse, mutta yhtä tärkeää, ellei jopa tärkeämpää on osata käyttää jo valmiita moduuleita, joita on helposti saatavilla tuhansia Comprehensive Perl Archive Networkistä. Yleisesti moduuli otetaan käyttöön seuraavanlaisella syntaksilla:

```
use moduuli;
```

missä moduuli on moduulin tiedostonimi ilman .pm-päätettä. (Wall ym. 2000, 299.)

Moduuleita etsitään hakemistoista, jotka on listattu Perlin sisäänrakenettussa @INC-arrayssa. Tyypillisesti kuitenkin moduulit sijoitetaan Perlin asennushakemiston (Windows-ympäristössä esim. C:\Perl) alihakemistoon site\lib. Moduulin nimessä mahdollisesti ilmenevät kaksi kaksoispistettä muunnetaan käyttöjärjestelmän hakemistoerottimeksi. Esimerkiksi moduulia Esimerkki::Moduuli::Yksi voitaisiin etsiä hakemistopolusta C:\Perl\site\lib\Esimerkki\Moduuli nimellä Yksi.pm. (Wall ym. 2000, 300.)

Moduulit voi sijoittaa myös johonkin muuhun hakemistopolkuun lisäämällä haluttu hakemistopolku @INC-arrayhin komennolla `use lib ”hakemistopolku”;`
Kun moduuli on use-komennolla tuotu käytettäväksi, voidaan kaikkia sen tarjoamia funktioita käyttää vapaasti komentosarjan sisällä. (Wall ym. 2000, 300.)

3.2.4 Oliot

Kuten monet muutkin modernit ohjelmointikielet, myös Perl tukee olio-ohjelmointia. Perlissä ei ole minkäänlaista erityissyntaksia olioiden käsittelyyn, vaan se kierrättää muussa käytössä olevaa syntaksia olioiden kanssa (Wall, ym. 2000, 310).

Yksittäistä oliota esittää aina yksi scalar-arvo, luokkaa moduuli, ja metodia alirutiini. Käytännössä Perl-olio on siis vain moduulin ”ilmentymä”, ja moduulin määrittelemät alirutiinit ja muuttujat ovat ilmentymän metodeita ja ilmentymämuuttujia. (Wall, ym. 2000, 310.)

Luokan metodeita voidaan kutsua käyttämällä nuoli-operaattoria (->) tai epäsuoraa objektia (Wall ym. 2000, 310). Esimerkkinä molemmista tekniikoista käytetään keksittyä oliota file, ja sen metodia read:

```
$file->read();    # nuoli-operaattori
read $file;       # epäsuora objekti
```

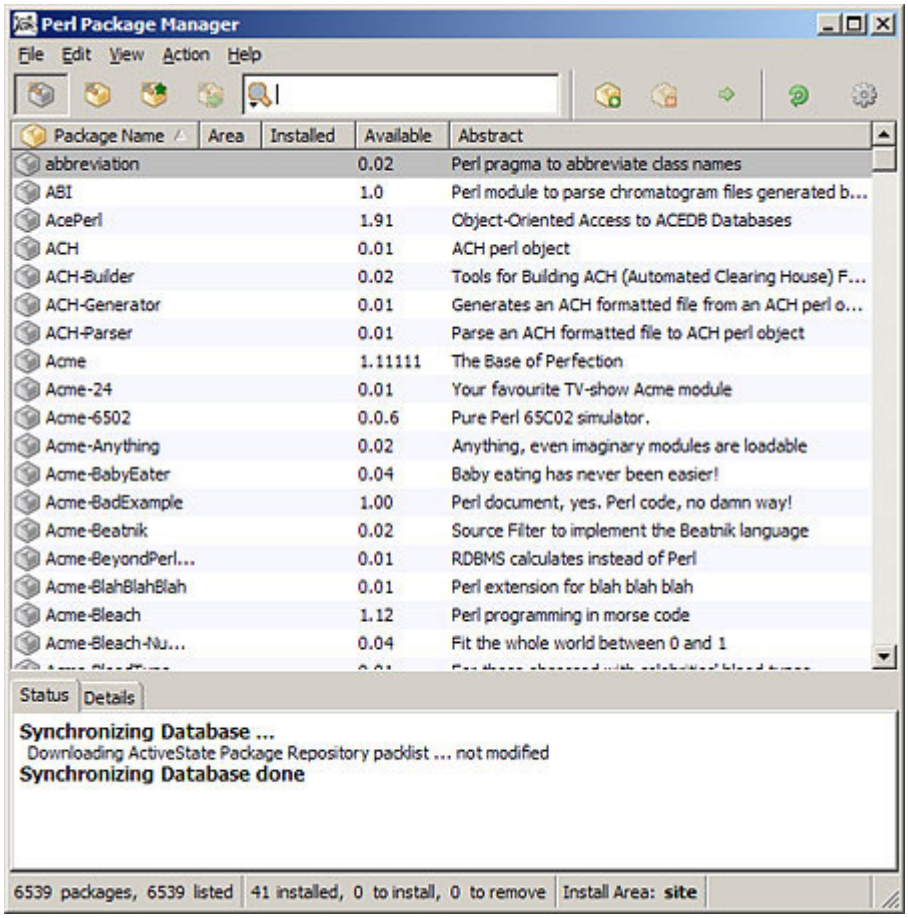
3.3 ActiveState ActivePerl

Perlin historia Windows-ympäristössä alkoi vuonna 1995. Tuolloin Microsoft palkkasi Dick Hardtin Hip Communicationsilta kehittämään Windows-yhteensopivaa Perl-versiota. Alun pitäen Perlin Windows-versio oli paikoin epäyhteensopiva Perlin ns. core-jakelun (UNIX-ympäristöissä käytetty Perl-versio) kanssa. Hip Communications (nykyisin nimeltään ActiveState) ja Perlin eri käyttöjärjestelmäversioista vastaava ryhmä Perl Porters pyrkivät kuitenkin saamaan virallisen Perl-jakelun kääntymään ja toimimaan Windows-alustalla. (Roth, 2002, 2.)

Perlin versiosta 5.005 lähtien ActiveState nimesi Perl-jakelunsa ActivePerliksi. Käytännössä tämä on Perlin core-jakelu, jonka päälle on lisätty vain Windows-alustalla toimivia lisämoduuleita sekä graafinen moduulien hallinta Perl Package Manager. Lähtien Perlin versiosta 5.6 sekä Perlin Windows-versio ActivePerl että core-jakelu ovat käyttäneet samaa koodikantaa. Tämä tarkoittaa sitä, että Perlin lähdekoodin voi sellaisenaan kääntää Windows-alustallakin. (Roth, 2000, 2.)

Tyypillisesti Perlin käyttöönottoon Windows-ympäristössä riittää ActiveStaten Active-Perlin MSI-asennuspaketin hakeminen ActiveStaten kotisivustolta ja sen suorittaminen. Asennusohjelma tekee tarpeelliset muutokset ympäristömuuttujiinkin automaattisesti, eli asentaminen on varsin suoraviivaista. (ActiveState, ActivePerl Installation Guide, 2008.)

ActivePerl ei suinkaan ole ainoa Perlin Windows-alustalle tarkoitettu binäärijakelu, mutta varmasti kaikkein käytetyin ja pisimmälle integroitu. ActivePerl sisältää myös valmiina runsaalti Win32-moduuleita sekä graafisen asennustyökalun lisämoduulien asentamista varten (Perl Package Manager), jonka käyttöliittymä on esitelty kuvassa 3.



Kuva 3. Perl Package Managerin käyttöliittymä

3.4 Perl Windows-ympäristössä

Optimitilanteessa Perl voitaisiin asentaa jokaiselle koneelle Windows-verkossa ja näin Perl-skriptit voitaisiin helposti suorittaa paikallisen binäärin avulla. Tämä voi tuottaa

ongelmia, jos käytössä on jo kymmeniä, satoja tai tuhansia koneita, joille kaikille Perlin asennus pitäisi suorittaa. On kuitenkin olemassa toinenkin mahdollisuus, Perlin suorittaminen verkkolevyltä. (Roth, 2001, 8.)

Paras vaihtoehto Perlin suorittamiseen verkkolevyltä on luoda tiedostopalvelimelle Perliä varten oma levyjako, ja lisätä tämän levyjaon UNC-polku (Universal Naming Convention) Windowsin PATH-ympäristömuuttujaan. Näin Perl-ohjelmat ovat helposti suoritettavissa. On tärkeää muistaa lisätä UNC-polku PATH-muuttujan loppuun, koska sen sijoittaminen muuttujan alkuun tai sen keskelle voi aiheuttaa pitkiä katkoja ohjelmien käynnistyessä silloin, kun levypalvelin ei ole saavutettavissa ja Windows odottaa pyynnön aikakatkaisua ennen PATH-muuttujassa etenemistä. (Roth, 2001, 8-9.)

Toinen PATH-muuttujaan liittyvä huomioinarvoinen seikka on se, että tämä UNC-polku tulisi sijoittaa *käyttäjän* PATH-ympäristömuuttujaan, eikä työasemalle. Tämä johtuu siitä, että käyttäjäkohtaiset PATH-asetukset lisätään työaseman muuttujan perään kirjautumisprosessin aikana. Jos UNC-polku sijaitsee työaseman muuttujassa, päätyy se jälleen keskelle muuttujaa. (Roth, 2001, 9.)

Viimeinen Perlin verkkolevyltä käyttöön liittyvä ongelma esiintyy lähinnä isoissa, useamman aliverkon kokoisissa Windows-verkoissa. Tilanteessa, jossa tätä aiemmin määriteltyä UNC-polkua käytetään useasta toimipisteestä, muodostuu kalliisiin WAN-linkkeihin turhan paljon liikennettä. Tähänkin ongelmaan on ratkaisu. Perlin tiedostot tulisi sijoittaa useampaan paikkaan lähiverkossa, mieluiten jokaiseen toimipisteeseen tai lähiverkkoon. Tällöin tarvitsee kuitenkin jotenkin määritellä, missä lähin Perl-levyjakoon sijaitsee, käyttäjähän voi kirjautua koneelle missä tahansa jolloin hän saattaisi käyttää epäoptimaalista binäärin sijaintia. Ongelman voi ratkaista määrittämällä jokaiselle työasemalle PERLDIR-ympäristömuuttujan, joka osoittaa lähimpään Perl-levyjakoon. Tämän jälkeen PERLDIR-muuttuja lisätään käyttäjien PATH-muuttujaan. Tämä on ideaali ratkaisu, jolla minimoidaan pitkiä matkoja verkossa kulkeva liikenne. (Roth, 2001, 9.)

Jos Perl-komentosarjaa käytetään toistuvasti verkon ylitse, voi olla hyvä ajatus muokata Windowsin oletustoiminnallisuutta ja estää Perlin ja sen kirjastojen poistaminen tietokoneen keskusmuistista. Näin tekemällä estetään Perliin liittyvien tiedostojen toistuva

lataaaminen verkkolevyltä, ja vähennetään verkon kuormaa. Perl voidaan merkitä säilytettäväksi keskusmuistissa käyttämällä Microsoft Visual C++:n mukana tulevaa editbin.exe-ohjelmaa. Ainakin perl.exe, ja mahdollisesti sen mukana tulevat dll-tiedostot tulisi käsitellä tällä ohjelmalla suorittamalla ohjelma komennolla editbin.exe /swaprun:net perl.exe. Muutoksen voi perua ajamalla ohjelma uudestaan niin, että ennen sanaa net valitsimeen lisätään huutomerkki (editbin.exe /swaprun:!net). (Roth, 2001, 11.)

Perl-ohjelmia on mahdollista suorittaa Windows-työasemalla ilman varsinaista Perl-asennusta. Tämä saadaan aikaiseksi kääntämällä Perl-ohjelma Windows-binääriksi jollakin siihen sopivalla ohjelmalla. Tällaisia ohjelmia ovat esimerkiksi kaupalliset IndigoSTAR Softwaren perl2exe ja ActiveStaten PerlApp sekä ilmainen PAR. Kaikilla näillä työkaluilla on mahdollista kääntää Perlillä kirjoitettu ohjelma itsekseen toimivaksi Windows-binääriksi, jolloin Perl-tulkin asennusta jokaiselle työasemalle ei tarvita. Tämä ratkaisu voi olla ainoa vaihtoehto, jos Perlin asentaminen jokaiselle työasemalle ei ole mahdollista.

3.5 Comprehensive Perl Archive Network

Comprehensive Perl Archive Network, CPAN, on keskeinen online-lähde kaikkeen, mikä liittyy Perliin. CPAN sisältää tuhansia eri tarkoituksiin tarkoitettuja Perl-moduuleita dokumentaatioineen. CPAN on yleensä paras paikka aloittaa etsiminen, jos on aikeissa tehdä Perlillä jotain. On olemassa erittäin suuri mahdollisuus, että joku on jo tehnyt puolet työstä luomalla käyttöä varten yleishyödyllisen moduulin, jota voi hyödyntää omassa ohjelmassaan. Windowsin kirjautumiskomentosarjoissa käytettäväksi soveltuvia moduulejakin on runsain mitoin.

Moduulien sisäänrakenettu POD-dokumentaatio (Plain Old Documentation) on myös luettavissa CPAN:ssa. Nämä sisältävät usein informaatiota, jota ei ole muualta saatavissa ja kirjoittaja on lähes poikkeuksetta sama henkilö, kuin moduulin kirjoittaja. CPANin tarjontaan tutustuminen on käytännöllisesti katsoen pakollista, jos suunnittelee toteuttavansa Perlillä mitään muutamaa kymmentä riviä pitempää ohjelmaa.

4 Tekninen toteutus

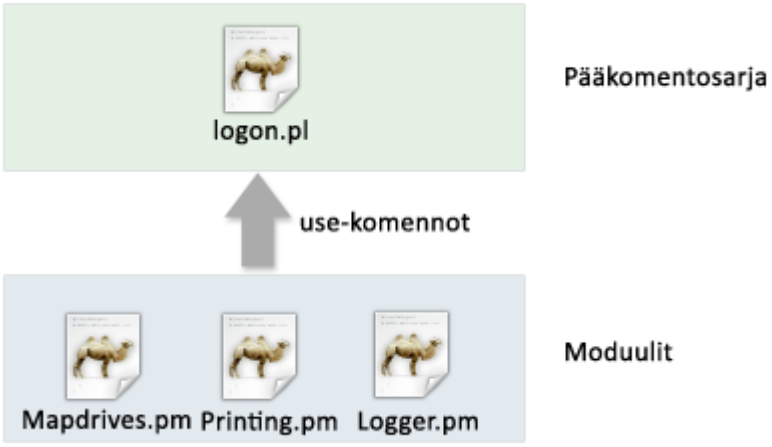
Tässä luvussa esitellään varsinaisen tekninen toteutus. Luku on pilkottu alilukuihin, joista jokaisessa esitellään jokin yksittäinen toiminnallisuus, ja siihen käytetyt moduulit. Mahdolliset ongelmakohdat ja niihin löydetyt ratkaisut käsitellään myös. Lopuksi esitellään menetelmä, jolla luodaan Perl-ohjelman kääntöympäristö käyttäen ilmaisia ohjelmistoja.

4.1 Suunnittelu

Kirjautumiskomentosarjojen suunnittelussa pyrittiin noudattamaan tiettyjä, itse valittuja ohjenuoria. Koska kirjautumiskomentosarjan tulevasta ylläpitäjästä ei ole varmuutta, ohjelmakoodin selkeys oli tärkeää. Pääasiallisesti tästä syystä pyrittiin suosimaan olioita hyödyntäviä moduuleja aina kun se oli mahdollista, koska näiden avulla monimutkaisten tietorakenteiden, kuten Windowsin rekisterin, esittäminen ohjelmakoodissa oli varsin havainnollista.

Perlin moduuleita pyrittiin hyödyntämään mahdollisimman tehokkaasti. Käytännössä tämä tehtiin niin, että kaikki selkeästi omat toiminnallisuutensa (kirjoittimien määrittäminen, verkkoasemien yhdistäminen) toteutettiin omina moduuleinaan. Myös valmiita moduuleita pyrittiin käyttämään aina, kun sellainen oli saatavilla.

Itse komentosarja toimii siten, että yhteen Perl-tiedostoon on use-komentoja käyttäen sisällytetty eri toimintoja suorittavat moduulit, joiden funktioita kutsutaan tästä ns. pääkomentosarjasta. Kuva 4 havainnollistaa komentosarjan rakennetta.



Kuva 4: .pm-päätteiset moduulit yhdistetään use-komennoilla pääkomentosarjaan

4.2 XML-konfiguraatio

Komentosarjojen hallinta on helpompaa, jos jokaista muutosta varten ei tarvitse tehdä muutoksia itse komentosarjaan. Tämän takia päädyttiin ottamaan käyttöön erillinen konfiguraatiotiedosto, jolla komentosarjan toimintaa voidaan helposti muokata ja ohjata. Kaikki muuttuvat asetukset, kuten yhdistettävät levyasemat, asennettavat kirjoittimet, lisättävät pikakuvakkeet, ylläpidon sähköpostiosoite ja niin edelleen määritellään erillisessä konfiguraatiotiedostossa.

Konfiguraatiotiedoston muodoksi valittiin XML (Extensible Markup Language), koska se hierarkisen rakenteensa vuoksi sopii hyvin tämän kaltaisen asetustiedoston tietojen säilyttämiseen. XML-formaatti tarjoaa myös mahdollisuuden tarkistaa tiedoston sisältämien tietojen oikeellisuus käyttämällä esimerkiksi Schema-määrittelyä. Näin vältyttäisiin mahdollisilta ylläpidon tekemiltä virheiltä, jotka puolestaan saattavat aiheuttaa vikatilanteita verkossa saavuttamattomien palvelujen muodossa.

XML-tiedosto jaotellaan päätasolla jokaisen moduulin nimiseen alilohkoon, jossa puolestaan määritellään moduulin ryhmäkohtaiset asetukset. Kuvio 1 havainnollistaa XML-tiedoston rakennetta.

```
<config>

    <mapdrives>
        <drives group="Marketing">
            <drive letter="X:" path="\\server1\marketing">
            <drive letter="Z:" path="\\server1\common_files">
        </drives>
    </mapdrives>

    <printing>
        <printgroup group="Marketing">
            <printer path="\\server1\hplaserjet">
        </printgroup>
    </printing>

    <logger>
        <mail>
            <address>admin@example.com</address>
        </mail>
    <logger>

</config>
```

Kuvio 1: XML-tiedoston hierarkinen rakenne

4.3 Komentosarjan ominaisuudet

Seuraavissa aliluvuissa esitellään toiminnoittain kirjautumiskomentosarjojen ominaisuudet sekä niiden toteuttamiseen käytetyt ohjelmointitekniset ratkaisut ja Perl-moduulit.

4.3.1 Konfiguraatiotiedoston lukeminen

Konfiguraatiotiedoston lukemiseen komentosarjan käytettäväksi käytettiin XML::Simple-moduulia. XML::Simple luo sille syötetystä XML-tiedostosta tietorakenteen, joka koostuu sisäkkäisistä hash- ja array-rakenteista. Näin XML-tiedoston tiedot saadaan ohjelmakoodin kannalta kohtalaisen luettavaan ja helposti käsiteltävään muotoon. XML::Simple-moduuli sisältää jonkin käyttökelpoisia optioita, joita voidaan käyttää avattaessa uutta XML-tiedostoa. Molemmat optiot syötetään luotaessa uutta XML-lukijaobjektia (new XML::Simple) Käyttämällä optiota KeyAttr voidaan lukija asettaa muuntamaan tietty attribuutti hash-avaimeksi array-arvon sijaan. Optiolla ForceArray

voidaan pakottaa yksielementtinen XML-puurakenne arrayksi. Oletusarvoisesti XML::Simple luo yksielementtisestä XML-puun osasta hash-rakenteen. XML::Simple moduulia voi käyttää myös ohjelmien, jotka tallettavat asetuksiaan XML-tiedostoihin, asetusten muokkaamiseen kirjautumisen yhteydessä.

4.3.2 Verkkoasemien yhdistäminen

Verkkoasemien yhdistämiseen voidaan käyttää Win32::OLE-moduulia, jolla luodaan WScript.Network-olio. Kyseessä on Windows Scripting Hostin OLE-olio (Object Linking and Embedding), joka tarjoaa funktiot MapNetworkDrive ja RemoveNetworkDrive verkkoasemien hallintaan. MapNetworkDrive-funktiolle syötetään argumentteina järjestyksessä seuraavat asiat: Kirjain, johon levyjako yhdistetään, levyjaon UNC-polku, persistenssi 0 tai 1 (säilyykö verkkoasema yhdistettynä istuntojen välissä), käyttäjätunnus ja salasana. Kolmea viimeistä ei tarvitse määritellä. Persistenssin oletusarvo on 0, eli verkkoasema ei säily yhdistettynä uloskirjautumisen jälkeen. RemoveNetworkDrive-funktiolle syötetään pelkästään paikallinen levyasemakirjain. Kuvion 2 esimerkkiohjelma yhdistää levyjaon \\server1\marketing asemakirjaimelle X, käyttäjätunnuksella tunnus ja salasanalla salasana, ilman persistenssiä. Tämän jälkeen ohjelma poistaa kyseisen levyjaon ja tulostaa virheilmoituksen, jos suorituksen aikana on tapahtunut virhe.

```
use Win32::OLE;

### Luo WScript.Network-olio
$wsh = Win32::OLE->new('WScript.Network');

### Yhdistä levyjako levyasematunnukselle X:
$wsh->MapNetworkDrive("X:", "\\server1\marketing", 0, "tunnus", "salasana");

### Jos yhdistämisessä tapahtui virhe, tulosta virheilmoitus
print Win32::OLE->LastError() if Win32::OLE->LastError();

### Poista yhdistetty levyjako
$wsh->RemoveNetworkDrive("X:");

### Jos poistamisessa tapahtui virhe, tulosta virheilmoitus
print Win32::OLE->LastError() if Win32::OLE->LastError();
```

Kuvio 2: Verkkoaseman yhdistäminen käyttämällä WScript.Network-oliota

Verkkoasemat voisi yhdistää myös esimerkiksi kutsumalla Perlistä ns. backtick-operaattorin avulla suoraan Windowsin NET USE-komentoa seuraavasti:

```
`NET USE X: \\palvelin\jako\`
```

On kuitenkin suositeltavampaa käyttää Win32::OLE-oliota, koska tällöin voidaan hyödyntää Win32::OLE-moduulin LastError-alirutiinia, jonka avulla saadaan yksityiskohtaista tietoa mahdollisista virhetilanteista, kuten siitä, jos haluttu levyasemakirjain on jo jonkin muun laitteen käytössä.

4.3.3 Verkkotulostimien yhdistäminen

Verkkotulostimien yhdistämiseen on myös helpointa käyttää Win32::OLE-moduulia, jonka avulla luodaan Windows Scripting Hostin Network-olio, joka levyjakoihin liittyvien funktioiden ohella tarjoaa myös funktioita tulostimien yhdistämiseen. Tulostimia voidaan yhdistää Network-olion AddWindowsPrinterConnection-metodilla ja niitä voidaan poistaa RemovePrinterConnection-metodilla. Lisäksi metodilla SetDefaultPrinter voidaan valita jokin tulostin työaseman oletustulostimeksi. (Microsoft Developer Network, WshNetwork Object). Kuvion 3 esimerkkiohjelma yhdistää, asettaa oletustulostimeksi ja poistaa verkkotulostimen \\server1\hplaserjet käyttäen Win32::OLE-oliota.

```
use Win32::OLE;

### Luo Windows Script Host Network-olio
$wsh = Win32::OLE->new('WScript.Network');

### Yhdistä tulostin työasemalle
$wsh->AddWindowsPrinterConnection("\\server1\hplaserjet");

### Jos yhdistämisessä tapahtui virheitä, tulosta ne
if(Win32::OLE->LastError != 0) {
    print Win32::OLE->LastError();
}

### Tee yhdistetystä tulostimesta oletustulostin
$wsh->SetDefaultPrinter("\\server1\\hplaserjet");

### Poista verkkotulostin työasemalta
$wsh->RemovePrinterConnection("\\server1\\hplaserjet");

if(Win32::OLE->LastError != 0) {
    print Win32::OLE->LastError();
}
```

Kuvio 3. Verkkotulostimen käsittely WScript.Network-olion avulla

Network-olio tarjoaa myös funktion AddPrinterConnection, mutta sitä on tarkoitus käyttää verkkotulostimien yhdistämiseen paikalliseen tulostinporttiin, kuten porttiin LPT1 (Microsoft Developer Network, AddPrinterConnection Method). Tästä on hyötyä

lähinnä käytettäessä vanhoja ohjelmia, jotka edellyttävät paikallisen tulostimen olemassaoloa.

4.3.4 Pikakuvakkeiden luominen

Pikakuvakkeiden luomiseen voidaan niin ikään hyödyntää Win32::OLE-moduulia. Tällöin luodaan Windows Scripting Hostin WScript.Shell-olio, joka tarjoaa funktion muun muassa pikakuvakkeiden luomiseen. Jos pikakuvakkeet halutaan lyödä käyttäjän työpöydälle, kannattaa käyttää WSCript.Shell-olion SpecialFolders-funktiota, joka palauttaa käyttäjän työpöytähakemiston sijainnin. Ongelmien välttämiseksi hakemistosta on suositeltavaa ottaa lyhyt, 8+3 merkkiä pitkä nimi käyttämällä Win32-moduulin GetShortPathName-funktiota, koska Win32::OLE-moduuli ei kaikkien funktioittensa osalta osaa käsitellä Unicode-merkkejä oikein. Kuvion 4 esimerkkiohjelma luo pikakuvakkeen Windowsin Muistio-ohjelmasta työpöydälle.

```
use Win32;
use Win32::OLE;

### Luo WScript.Shell-olio
$wsh = new Win32::OLE('WScript.Shell');

### Hae käyttäjän työpöytä-kansion absoluuttinen polku muuttujaan
$desktop = Win32::GetShortPathName($wsh->SpecialFolders('Desktop'));

### Luo pikakuvake-olio, jonka parametrina on kuvakkeen tiedostonimi
$shortcut = $wsh->CreateShortcut($desktop . '\\Notepad.lnk');

### Aseta pikakuvakkeen parametrit
$shortcut->{Description} = 'Notepad';
$shortcut->{IconLocation} = 'notepad.exe, 0';
$shortcut->{TargetPath} = 'C:\\Windows\\notepad.exe';

### Tallenna pikakuvake
$shortcut->Save();

### Tulosta virheilmoitus, jos kuvakkeen luonnissa tapahtui virhe
print Win32::OLE->LastError() if Win32::OLE->LastError();
```

Kuvio 4. Pikakuvakkeen luominen käyttäjän työpöydälle

Kirjoitettaessa Windows-hakemistopolkuja Perl-muuttujiin on tärkeää muistaa käyttää kaksinkertaisia kenoviivoja. Yksittäisen kenoviivan ja sitä seuraavan merkin Perl tulkitsee ns. escape-merkkinä, kuten esimerkiksi rivinvaihtona \n. Shortcut-objektille voi antaa myös argumentteja muokkaamalla sen muuttujaa Arguments ja muokata työhake-

mistoa muuttujalla WorkingDirectory. Työpöydällä pikakuvakkeen nimenä näkyy .lnk-päätteisen tiedoston nimi, ilman päätettä. Lopuksi tulostetaan mahdollinen ajonaikainen virheilmoitus, jos sellainen on sattunut.

4.3.5 Windowsin rekisterin lukeminen

Windowsin rekisterin lukemiseen Perl tarjoaa useitakin moduuleita. Vaihtoehtoina ovat Win32API::Registry, Win32::Registry ja Win32::TieRegistry. Näistä Win32API::Registry käyttää Win32 API (Application Programming Interface) –kutsuja WINREG.H-tiedostosta rekisterin lukemiseen. (CPAN, Win32API::Registry). Win32API::Registry ei ole oliopohjainen, eikä tuota API-sidonnaisuudestaan johtuen kovinkaan helppolukuista koodia. Win32::Registry on tästä jalostettu, oliopohjainen, mutta nykyään käytöstä poistettu versio. (CPAN, Win32::Registry).

Suositeltava tapa Windowsin rekisterin lukemiseen on siis käyttää Win32::TieRegistry-moduulia. TieRegistry on oliopohjainen ja luo halutusta rekisterin haarasta sisäkkäisiä hash-rakenteita hyödyntävän tietorakenteen. Tämä esitystapa on hyvin havainnollinen ja helppokäyttöinen. Kuvion 5 esimerkissä luetaan TieRegistry-moduulin avulla käyttäjän Käynnistä-valikon sijainti:

```
### Moduuli luo automaattisesti rekisterin juuriolion $Registry
use Win32::TieRegistry;

### Vaihda erottimeksi kauttaviiva kenoviivan sijaan
$Registry->Delimiter("/");

### Luo rekisterin haarasta uusi olio
$shellFolders = $Registry->{"CUser/Software/Microsoft/Windows" .
"/CurrentVersion/Explorer"}->{"Shell Folders"};

### Lue GetValue-metodilla arvo muuttujaan
$startMenu = $shellFolders->GetValue("Start Menu");
```

Kuvio 5: Rekisterin tietojen lukeminen käyttäen Win32::TieRegistry-moduulia

Moduulin käytössä on kaksi tärkeää ja huomionarvoista seikkaa. TieRegistry-moduuli yrittää aina avata kaikki rekisterihaarat sekä luku- että kirjoitusoikeuksin, eikä miten-kään ilmoita, jos tämä ei onnistu. Tämä tuottaa ongelmia tilanteissa, joissa komentosar-

jan pitäisi lukea tietoja esimerkiksi HKEY_LOCAL_MACHINE-haarasta, johon tavallisilla käyttäjillä on usein vain lukuoikeudet. Tällöin rekisterihaara on avattava erillisellä optiolla vain-luku moodissa kuvion 6 esimerkin mukaisesti.

```
use Win32::TieRegistry;

### Vaihda erottimeksi kauttaviiva kenoviivan sijaan
### Näin vältetään tarve käyttää kaksoiskenoviivoja
$Registry->Delimiter("/");

### Avaa rekisterin haara vain luku -oikeuksin
$winNT = $Registry->open("LMachine/Software/" .
"Microsoft/Windows NT", {Access=>"KEY_READ"});
```

Kuvio 6. Rekisterin haaran avaaminen vain luku -muodossa

Lisäksi on oltava tarkkana rekisterihaarojen nimien kanssa, jotka sisältävät välilyöntejä. Tällöin on muistettava käyttää välilyöntejä, tai haaran lukuyritys epäonnistuu harhaanjohtavaan virheilmoitukseen Can't locate object method "X" via package "Y" jossa X ja Y ovat välilyönnillä erotettuja sanoja.

4.3.6 Taustaprosessien käynnistys ja ohjelmien suorittaminen

Ohjelmien käynnistämiseen Perl-ohjelman sisältä on monia vaihtoehtoja. Perl itsessään sisältää sisäänrakennetut funktiot system() ja exec() sekä aiemminkin mainittu backtick-operaattori, joiden lisäksi Windows-ympäristössä on mahdollista käyttää Windows Scripting Hostin Shell-olion Exec-funktiota ja Win32::Process-moduulia. Kaksi jälkimmäistä tapaa ovat Windows-ympäristössä suositeltavimmat, koska ne tarjoavat enemmän mahdollisuuksia hallita prosessia, kuin Perlin sisäänrakennetut menetelmät. Erityisesti Win32::Process tarjoaa kattavan valikoiman asetuksia mm. prosessin prioriteetin valintaan. Molemmat mahdollistavat myös virheilmoitusten keräämisen funktioiden avulla. Win32::Process voi tosin olla työkaluna hieman liian järeä esimerkiksi pelkän kopiointikomennon suorittamiseen, jolloin voi olla järkevämpää käyttää WSH-oliota. Isompien taustaprosessien suorittamiseen Win32::Process on puolestaan parempi valinta. Kuvion 7 esimerkissä kopioidaan WScript.Shell-olion avulla tiedosto käyttäjän profiilihakemistoon, ja kuvio 8 esimerkissä käynnistetään taustaprosessi matalalla prioriteetilla.

```
use Win32::OLE;

### Luo uusi WScript.Shell-olio
$shell = Win32::OLE->new('WScript.Shell');

### Suorita komentotulkin kautta kopiointikomento
### Käytössä oleva komentotulkki on määritelty %COMSPEC% ympäristömuuttujassa
$shell->Exec("%COMSPEC% /C copy \\\\server1\\netlogon\\settings\\data.ini .
\"%USERPROFILE%\\data.ini\"");

### Tulosta virheilmoitus, jos kopiointi ei ole onnistunut
print Win32::OLE->LastError() if Win32::OLE->LastError();
```

Kuvio 7. Tiedoston kopiointi WScript.Shell-olion avulla.

Käytettäessä WSH-oliota, täytyy muistaa lisätä tavallisten, Windowsin komentorivin komentojen (copy, move jne.) eteen %COMPSPEC% /C jossa ympäristömuuttuja %COMSPEC% viittaa Windowsin komentotulkkiin, yleensä C:\Windows\system32\cmd.exe. /C-valitsimella varmistetaan, että komentotulkki sulkee itsensä komennon suorittamisen jälkeen. Ilman komentotulkin määrittämistä komennon suoritusyritys päättyy virheilmoitukseen.

```
use Win32;
use Win32::Process;

### Käynnistettävän prosessin asetukset

### Ohjelmabinäärin polku
$application = "\\server1\netlogon\bigprocess.exe";

### Ohjelmalle annettavat argumentit
$args = "";

### Peritäänkö avoimet tiedostokahvat kutsuvalta prosessilta
$iflags = 0;

### Prosessin prioriteetti
$cflags = IDLE_PRIORITY_CLASS;

### Prosessin työhakemisto, . tarkoittaa nykyistä hakemistoa
$workdir = ".";

### Luo prosessi edellä määritellyillä asetuksilla
Win32::Process::Create($myProcess, $application, $args,
$iflags, $cflags, $workdir);

### Tulosta virheilmoitus, jos prosessia ei voida käynnistää
### FormatMessage vaihtaa virheen muodon numerokoodista tekstiksi
print Win32::FormatMessage(Win32::GetLastError()) if Win32::GetLastError();
```

Kuvio 8. Prosessin käynnistäminen Win32::Process-moduulin avulla

Win32::Process::Create-funktio ottaa 6 argumenttia. Ensimmäinen näistä on täydellinen polku ohjelmaan, toinen ohjelman mahdolliset komentoriviargumentit. Kolmas määrittelee, peritäänkö kutsuvan prosessin avoimet tiedostokahvat. Käynnistyskomentosarjoissa tätä ei yleensä tarvita. Neljäs määrittelee prosessin luomisasetukset, kuten prioriteetin. Viimeinen argumentti määrittelee prosessin työhakemiston.

4.3.7 Ohjelmien automaattiohjaus

Komentosarjoja toteutettaessa huomattiin, että joitakin ohjelmia oli yksinkertaisesti mahdotonta ohjata millään muulla tavalla, kuin syöttämällä ohjelman ikkunalle peräkkäisiä näppäinkomentoja. Jos ohjelma ei tarjoa esimerkiksi minkäänlaista OLE-objektia jonka avulla sen toimintoja voisi ohjata, on tyydyttävä tähän menetelmään. Tähänkin toimintoon Perl tarjoaa useita moduuleita, mutta niistä toimivimmaksi ja helppokäyttöisimmäksi todettiin Win32::GuiTest-moduuli. Sen avulla voidaan aktivoida ikkunoita, ja syöttää ikkunan yksilöivään ns. window handleen näppäinkomentoja.

Win32::GuiTest tarjoaa myös joustavat mahdollisuudet etsiä ikkunoita niiden nimen perusteella hyödyntäen regular expressioneja. Tämä helpottaa oikean ikkunan löytämistä, koska ikkunan nimeä ei tarvitse tietää kokonaan sen löytämiseksi. Win32::GuiTest mahdollistaa myös käyttöliittymän painikkeiden painamisen automaattisesti, valikoiden selaamisen ja hiiren kursorin liikuttamisen. Moduulilla voi siis käytännöllisesti katsoen tehdä automaattisesti kaiken sen, mitä normaalistikin voisi tehdä näppäimistöllä ja hiirellä. Win32::GuiTestin tehokas käyttö vaatii jonkin verran tuntemusta Windowsin ikkunointijärjestelmän rakenteesta ja sisäisestä toiminnasta, mutta yksinkertaista automaatiota voi saada aikaiseksi vähemmälläkin osaamisella.

Kuvion 9 esimerkissä etsitään ikkuna, joka alkaa sanalla Citrix, painetaan sen käyttöliittymän ikkunassa ensin näppäintä F5 ja hetken päästä suljetaan ikkuna painamalla ALT-F4.

```
### :ALL tuo kaikki GuiTest-moduulin funktiot käytettäväksi
use Win32::GuiTest qw(:ALL);

### Odota 20 sekuntia ikkunaa, jonka otsikko alkaa sanalla Citrix
my @citrixWindows = WaitWindowLike(undef, "^Citrix", undef, un-def, undef, 20);

### Jos ikkuna löytyy
if(@citrixWindows)  {
    ### Näytä ikkuna
    ShowWindow($citrixWindows[0], SW_SHOW);
    ### Aseta ikkuna päällimmäiseksi
    SetForegroundWindow($citrixWindows[0]);
    ### Aktivoi ikkuna
    SetActiveWindow($citrixWindows[0]);
    ### Lähetä ikkunaan näppäinpainallus F5
    SendKeys("{F5}");
    ### Odota 1 sekunti
    sleep 1;
    ### Lähetä ikkunaan näppäinpainallus ALT+F4, % merkitsee ALT-painiketta
    SendKeys("%{F4}");
}
```

Kuvio 9. Ikkunan käsittely Win32::GuiTest-moduulin avulla

Esimerkissä käytetään funktiota WaitWindowLike ikkunan löytämiseen. Funktion argumentit, joille on annettu esimerkissä arvo undef, käsittelevät elementin luokkaan (painike, valikko ym.) liittyviä asioita. Näitä ei yksinkertaisessa automaatiossa juuri tarvita. Viimeinen argumentti, 20, tarkoittaa, että sanalla Citrix-alkavaa ikkunaa odotetaan 20 sekuntia. GuiTest-moduuli tarjoaa myös yksinkertaisemman, FindWindowLike-funktion, joka ei tarjoa mahdollisuutta aikakatkaisuun. Ensin mainittua kannattaa käyttää tilanteissa, joissa komentosarja itse käynnistää ensin ohjelman ja sen jälkeen syöttää siihen komentoja, jolloin ohjelmalla on argumentin määrittelemä aika aikaa ladata itsensä. Esimerkissä myös oletetaan, että vain yksi Citrix-alkuinen ikkuna löytyy. Muita arrayn elementtejä ei käsitellä ollenkaan. Joissain tilanteissa voi olla tarpeellista käydä esimerkiksi for-silmukan avulla kaikki tietyn tyyppiset ikkunat lävitse, esimerkiksi jos halutaan sulkea kaikki saman ohjelman esiintymät.

4.3.8 ODBC-tietokantayhteyksien luominen

ODBC-tietokantayhteyksiä voidaan luoda moduulilla Win32::ODBC. Tietokantayhteyksiä olisi mahdollista luoda myös kirjoittamalla niiden tiedot suoraan rekisteriin esimerkiksi käyttämällä aiemmin esiteltyä TieRegistry-moduulia, mutta Win32::ODBC-moduuli tarjoaa tyylillisesti siistimmän tavan saman asian saavuttamiseen. ODBC-

moduulia voi käyttää myös mm. tiedon lukemiseen ODBC-tietolähteestä, mutta tätä toiminnallisuutta ei työn toteuttamisessa testattu.

Tietokantayhteyksiä luodaan Win32::ODBC-moduulin funktiolla ConfigDSN. Kuvion 10 esimerkissä luodaan käyttäjälle DSN-yhteys (Data Source Name, tietolähde) MySQL-palvelimeen:

```
use Win32::ODBC;

### ODBC-yhteyden asetukset

### Tietolähteen nimi
$dsn = "Marketing Clients MySQL";

### Palvelimen nimi tai IP-osoite
$server = "mysql-server1";

### Tietokannan nimi
$database = "clients";

### Käyttäjätunnus
$username = "marketing";
### Salasana
$password  = "m4rk3tIng";

### Tietolähteen kuvaus
$description = "Marketing Clients";

### Yhdistä tietolähde funktiolla ConfigDSN
Win32::ODBC::ConfigDSN(ODBC_ADD_DSN, "MySQL ODBC 3.51 Driver",
("DSN=$dsn",
"SERVER=$server",
"DATABASE=$database",
"DESCRIPTION=$description",
"PWD=$password",
"UID=$username");
```

Kuvio 10. ODBC-tietolähteen lisääminen Win32::ODBC-moduulin avulla

ConfigDSN ottaa siis kolme argumenttia, toiminnon, ajurin ja attribuuttilistan. Samalla funktiolla voidaan siis myös poistaa tietolähteitä vaihtamalla ODBC_ADD_DSN ODBC_DEL_DSN:ksi tai muokata olemassa olevia ODBC_MODIFY_DSN:llä. Käytettävissä olevien ajurien nimet voi tarkistaa joko Windowsin hallintatyökalujen Tietolähteet (ODBC) –sovelluksella tai Win32::ODBC:n Drivers-funktiolla. Attribuutit ovat

array-muotoinen lista tietolähteen attribuuteista. Esimerkissä on katettu kaikki useimmin tarvittavat attribuutit. Poistettaessa tietolähdettä riittää ainoaksi attribuutiksi DSN. Muutettaessa tietolähdettä syötetään DSN sekä muutettavat attribuutit.

Käytettäessä moduulia on muistettava, että ODBC-ajurit, kuten esimerkin MySQL ODBC 3.51 Driver on asennettava jokaista eri tietokantatyyppiä varten erikseen asiakaskoneille. Tämä on tosin varsin helppoa, koska yleisimmistä ajureista on saatavilla Microsoft Installer (.msi) –paketit, jolloin toimialueympäristössä ajurit voidaan asentaa keskitetysti. Lisäksi moduulia testattaessa todettiin, että MySQL ODBC Connectorin versio 5.1 ei jostain syystä suostu toimimaan Win32::ODBC-moduulin kanssa yhteistyössä. Vanhempi versio 3.51 sen sijaan toimii erinomaisesti. Myös PostgreSQL:n ja Microsoft SQL Serverin ODBC-moduulit todettiin toimiviksi. Windowsin mukana toimitetaan ODBC-ajurit Access- ja Excel-pohjaisille tietokannoille.

4.3.9 Ohjelmien asetustiedostojen muokkaus

Monet Windows-sovellukset käyttävät edelleen tietojensa tallentamiseen rekisterin tai xml-tiedoston sijaan vanhempaa ini-tiedostotyyppiä. INI-tiedostot koostuvat osioista seuraavan esimerkin tapaan:

```
[settings]
username = jussi
homedirectory = \\server1\homes\jussi
```

INI-tiedostoja voisi käsitellä ilman erillisiä moduuleita esimerkiksi Perlin regular expressions –tuen avulla, mutta on helpompaa käyttää valmista moduulia tähän tarkoitukseen. CPAN:ssa on tarjolla lukuisia vaihtoehtoja, joista Config::Tiny päätyi tässä työssä käytettäväksi. Muut ehdokkaat olivat joko syntaksiltaan liian monimutkaisia tai toiminnoiltaan vajavaisia. Lisäksi Config::Tiny:n syntaksi ja sen käyttämät tietorakenteet muistuttavat XML::Simple-moduulin vastaavia, joten tämäkin puolti moduulin käyttöä. Kuvion 11 esimerkissä luetaan C:\Program\example.ini-tiedostosta, jonka sisältö on ylemmän esimerkin kaltainen käyttäjän kotihakemisto, ja kirjoitetaan uusi arvo settings-osioon.

```
use Config::Tiny

### Tiedoston nimi
$filename = "C:\\Program\\example.ini";

### Luo uusi Config::Tiny-olio
$inifile = Config::Tiny->new();

### Lue tiedosto
$inifile->read->($filename);

### Lue käyttäjätunnus muuttujaan
$username = $inifile->{settings}->{username};

### Lisää INI-tiedostoon arvo
$inifile->{settings}->{confirmexit} = "false";

### Tallenna muutos
$inifile->write($filename);
```

Kuvio 11: INI-tiedoston muokkaaminen Config::Tiny-moduulilla

Uusien arvojen ja osioiden lisäämiseen riittää siis vastaavien kohtien lisääminen tietorakenteeseen, ja lopuksi tiedoston tallentaminen moduulin write-funktiolla.

4.3.10 Ympäristön tietojen keruu

Kirjautumiskomentosarjoissa on usein tarpeellista kerätä kirjautuvasta työasemasta tietoja, kuten kirjautuva käyttäjä, tietokoneen isäntänimi, toimialue, kirjautumispalvelimen nimi, työaseman IP-osoite tai se, kuuluuko käyttäjä johonkin tiettyyn toimialueen käyttäjäryhmään.

Windows-perusmoduuli Win32 tarjoaa näistä tiedoista suurimman osan, mutta ip-osoitteen ja ryhmäjäsenyyksien keräämiseen tarvitaan muita moduuleja. Kuvion 12 esimerkissä kerätään Win32-moduulin avulla perustietoja työasemasta:

```
use Win32;
$username = Win32::LoginName(); ### Käyttäjätunnus
$hostname = Win32::NodeName();  ### Työaseman nimi
$domain = Win32::DomainName();  ### Toimialueen nimi
```

Kuvio 11. Kirjautuvan käyttäjän tietojen kerääminen Win32-moduulilla

Win32::NetAdmin-moduulia voidaan käyttää toimialueen ohjauskoneen nimen selvittämiseen, ja tätä taas puolestaan käyttäjän ryhmäjäsenyyksien tarkistamiseen. Kuvion 12 esimerkissä paikallistetaan käyttäjän sen hetkinen toimialueen ohjauskone, tarkistetaan ohjauskoneelta käyttäjän ryhmäjäsenyys ryhmälle Marketing ja tallennetaan kyselyn tulos muuttujaan $result.

```
use Win32;
### Tuo NetAdmin-moduulista funktiot GetAnyDomainController ja GroupIsMember
use Win32::NetAdmin qw(GetAnyDomainController GroupIsMember);

## Lue käyttäjän tunnus muuttujaan
$username = Win32::LoginName();

### Lue työaseman nimi muuttujaan
$hostname = Win32::NodeName();

### Lue toimialueen nimi muuttujaan
$domain = Win32::DomainName();

### Luo tyhjä muuttuja ohjauskoneen nimeä varten
$domainController = "";
### Funktio palauttaa kolmantena argumenttina olevaan muuttujaan
### toimialueen ohjauskoneen nimen
GetAnyDomainController($hostname, $domain, $domainController);

### Testaa käyttäjän jäsenyys ryhmään Marketing, kirjoita tulos muuttujaan
$result = GroupIsMember($domainController, "Marketing", $username);
```

Kuvio 12. Win32::NetAdmin-moduulin käyttö käyttäjän ryhmäjäsenyyden tarkistamiseen

NetAdmin-moduuli sisältää myös funktion GetDomainController, joka toimii muutoin samalla tavalla kuin GetAnyDomainController, mutta palauttaa aina toimialueen ensimmäisen ohjauskoneen nimen.

Työaseman IP-osoitteen saa helpoiten selville käyttämällä Socket-moduulia. Socket-moduulin käyttö voi tuottaa ongelmia käyttäjille, joilla on useampi IP-osoite käytössä yhtäaikaisesti, sillä seuraava algoritmi palauttaa vain ensimmäisen IP-osoitteen. (CPAN, Socket.pm). Kuvion 13 esimerkkiohjelma hakee käyttäjän IP-osoitteen käyttämällä Socket-moduulia.

```
use Win32;
use Socket;
### Lue työaseman nimi muuttujaan
$hostname = Win32::NodeName();
### Hae binäärimuotoinen IP-osoite muuttujaan
$packed_ip = gethostbyname($host);
### Pura binäärimuotoinen IP-osoite desimaalimuotoon
### ja kirjoita se muuttujaan
$ip = inet_ntoa($packed_ip);
```

Kuvio 13. IP-osoitteen lukeminen hyödyntäen Socket-moduulia

Socket-moduulia tarvitaan, koska Perlin sisäänrakennettu gethostbyname-funktio palauttaa IP-osoitteen C-ohjelmointikielen käyttämässä binäärimuodossa, jolloin funktiota inet_ntoa tarvitaan muuntamaan osoite takaisin ASCII-muotoon. (Christiansen & Torkington 2003, 721)

4.3.11 Sähköpostin lähettäminen

Virhetilanteista ilmoittamiseen ylläpidolle komentosarjat käyttävät Mail::Sendmail-moduulia. Kyseinen moduuli on toiminnaltaan ja syntaksiltaan varsin yksinkertainen. Moduulin käyttämisessä on huomioitava, että se tarvii toimiakseen olemassa olevan, postin lähettämiseen käytettävän SMTP-palvelimen. Palvelimen osoite täytyy syöttää moduulin asetuksiin, jotta postia voitaisiin lähettää. Moduuli ei myöskään tue SMTP AUTH –laajennusta (CPAN, Mail::Sendmail), eli postipalvelin ei saa vaatia käyttäjätunnusta ja salasanaa sähköpostin lähettämisen yhteydessä.

Lähtevän postin palvelimen osoite voidaan antaa moduulille kolmella tavalla: muokkaamalla Sendmail.pm-tiedoston riviä $mailcfg{smtp} ja syöttämällä siihen oikea postipalvelimen osoite, tai suoraan komentosarjasta joko komennolla:

```
unshift @{$Mail::Sendmail::mailcfg{'smtp'}} , 'mail.server';
```

Tai jokaisen sähköpostiviestin yhteydessä lisäämällä %mail-hashiin rivi

```
$mail{smtp} = 'mail.server';
```

Kokonaisuudessaan sähköpostia Mail::Sendmail-moduulin avulla lähetetään kuvion 14 esimerkin mukaisesti.

```
use Win32;
use Mail::Sendmail;

### Lue työaseman nimi muuttujaan
$hostname = Win32::NodeName();

### Luo sähköpostiviestistä hash-rakenne
%mail = (       To => 'admin@example.com',
                From => 'logon@example.com',
                Subject => 'Kirjautumisvirhe',
                Message => "Kirjautumisvirhe työasemalla $hostname"
        );
### Lähetä sähköpostiviesti
sendmail(%mail);
```

Kuvio 14. Sähköpostin lähettäminen Mail::Sendmail-moduulin avulla

Koko viesti rakennetaan otsikkotietoineen hash-rakenteeseen, joka syötetään argumenttina sendmail-funktiolle.

4.3.12 WMI

WMI (Windows Management Instrumentation) on Microsoftin implementaatio Web-based Enterprise Managementista (WBEM) jonka tarkoituksena on luoda standardi hallintatiedon käsittelyyn yritysympäristössä. (Microsoft Developer Network, About WMI)

Kirjautumiskomentosarjoissa WMI:tä voidaan hyödyntää keräämällä tietoa kirjautuvasta työasemasta. WMI:n avulla voidaan mm. tarkkailla vapaan levytilan määrää. Helpoin tapa käyttää WMI:tä Perl-komentosarjasta on luoda OLE-olio. Kuvion 15 esimerkissä kerätään kaikkien paikallisten levyasemien vapaat tilat array-rakenteeseen.

```
use Win32;
use Win32::OLE;

### Luo tyhjä array levytiloja varten
@drivespaces = ();

### Lue työaseman nimi muuttujaan
$hostname = Win32::NodeName();

### Luo uusi WMI-olio
$WMI = Win32::OLE->GetObject("winmgmts:\\\\$hostname\\root\\cimv2");

### Hae asemien tiedot WMI:n ExecQuery-metodin avulla
$drives = $WMI->ExecQuery("SELECT * FROM Win32_LogicalDisk");

### Käy läpi kaikki asemat, lisää jokaisen aseman vapaa tila
### drivespaces-arrayhin.
foreach my $drive (in $drives) {
    push(@drivespaces, $drive->FreeSpace());
}
```

Kuvio 15. Paikallisten kovalevyjen vapaan tilan tarkistus WMI-olion avulla

Kuten esimerkistä huomataan, WMI käyttää sisäisesti SQL-tyyppistä syntaksia tietojen tuomiseen eri luokistaan, jotka voidaan rinnastaa SQL-tauluihin. Lisäksi, WMI-olio ei luo keräämistään tiedoista tyypillistä Perl-tietorakennetta, vaan collection-tyyppisen rakenteen, jonka läpikäymiseen käytetään Win32::OLE-moduulin tarjoamaa in-funktiota.

4.4 Komentosarjan kääntäminen binääriksi

Kuten luvussa 3.4 jo todettiin, Perl-ohjelmia on mahdollista suorittaa ilman varsinaista Perl-asennusta, jos ohjelma käännetään binäärimuotoon jollakin ohjelmalla. Seuraavassa luvussa esitellään ilmainen, CPAN:sta saatavissa oleva PAR::Packer.

4.4.1 PAR::Packer

PAR::Packer on Perl-työkalu, jonka avulla on mahdollista luoda Perl-ohjelmasta ja sen käyttämistä moduuleista yksittäinen .par-tiedosto, joka sisältää kaikki ohjelman suorittamiseen vaadittavat tiedostot. Tätä voidaan jossain mielessä verrata (muutoinkin kuin nimensä osalta) Javan .jar-tiedostoihin. Lisäksi PAR::Packerin mukana tulee ohjelma pp, jonka avulla voidaan muuntaa luotu par-tiedosto Windowsin .exe-binääritiedostoksi.

Ainoa haittapuoli pp:n tuottamissa binääritiedostoissa on niiden jokseenkin suuri koko. Pienikin, vain muutamia moduuleita hyödyntävä Perl-ohjelma vie pp-binäärinä useita megatavuja. Koko voi nykymittapuulla vaikuttaa varsin mitättömältä, mutta jos kirjautumiskomentosarjaa käytetään esimerkiksi VPN-yhteyden yli hitaahkolla 3G-laajakaistaliittymällä, voi pelkästään kirjautumiskomentosarjan lataamiseen palvelimelta kulua aikaa useita minuutteja. Tässä työssä ongelma kierrettiin suorittamalla käynnistyskomentosarja paikallisesti. Komentosarjan päivittäminen työasemille järjestettiin asettamalla varsinaiseksi, profiileihin asetettavaksi käynnistyskomentosarjaksi yksinkertainen .BAT-tiedosto, joka tarvittaessa xcopy-ohjelman avulla päivittää paikallisen logon.exe:n uudempaan versioon. Kuvio 16 esittää .BAT-tiedoston sisältöä.

On myös syytä huomata, ettei PAR::Packer ole varsinaisesti kääntäjä, eli pp ei exe-tiedostoa luodessaan millään tavoin tarkista lähdekoodin oikeellisuutta. Tästä seuraa, että pp:llä on mahdollista luoda exe-tiedostoja, jotka eivät toimi. Ohjelman toiminta on aina ennen kääntämistä tarkistettava suorittamalla ohjelma Perl-tulkin avulla.

```
@echo off
@REM INIT.BAT FOR USE WITH PERL LOGON SCRIPTS (C) JUSSI HAAJA 2008
@XCOPY /D /Y /Q "%LOGONSERVER%\NETLOGON\LOGON.EXE" "%USERPROFILE%\LOGON.EXE"
@GOTO EXEC

:EXEC
@CLS
@"%USERPROFILE%\LOGON.EXE" "%LOGONSERVER%\NETLOGON\CONFIG.XML"
```

Kuvio 16. init.bat-tiedosto, jolla päivitetään paikallinen kopio käynnistyskomentosarjasta tarvittaessa.

4.4.2 Kääntöympäristön luominen

Varsinainen kääntöympäristö koostuu useammasta osasta, jotka täytyy kaikki yksi kerrallaan asentaa, jotta pp:tä voidaan käyttää exe-tiedoston luomiseen. Varsinaisen Perl-asennuksen lisäksi tarvitaan kääntäjä, jonka avulla varsinainen binääri luodaan, esimerkiksi Microsoftin Visual C++, tai ilmainen MinGW. Lisäksi tarvitaan Perlin käyttämien apuohjelma Perlin käyttämien ns. makefilejen käsittelyyn (esimerkiksi dmake tai nmake), sekä joitakin Perl-moduuleita CPANista. Yksityiskohtaiset ohjeet kääntöympäristön luomiseen on esitetty liitteessä 1.

5 Yhteenveto

Kaiken kaikkiaan Perl soveltuu mielestäni varsin hyvin Windowsin kirjautumiskomentosarjojen toteutuskieleksi. Erityisesti kattava online-dokumentaatio puoltaa Perlin valintaa. Myös laaja valikoima valmiita moduuleja helpottaa kehittäjän työtaakkaa suuresti, koska pyörää ei tarvitse lähteä keksimään uudelleen jokaista toimintoa varten. Pelkästään Windows-ympäristöön tarkoitettuja Win32-moduuleita on huikaiseva määrä. Perl mahdollistaa myös Windowsin omien automaatio- ja hallintatyökalujen OLE:n ja WMI:n helpon, oliopohjaisen käsittelyn.

Moduulien, dokumentaation ja ylipäätään Perliin liittyvän informaation suuren määrän takia kaikkein aikaavievin työvaihe oli tarvittavan tiedon koostaminen parhaan menetelmän valitsemiseksi. Työssä esitellyt menetelmät eivät toki ole ainoa mahdollinen tapa toteuttaa näitä toiminnallisuuksia, vaikkakin ovat toimiviksi testattuja ja hyviksi havaittuja.

Alkuperäisiin, KiXtart-skriptikielellä toteutettuihin komentosarjoihin nähden uudet, Perlillä toteutetut komentosarjat ovat ohjelmakoodiltaan paljon selkeämmät, ja helpommin ylläpidettävät. Myös virhetilanteista saatavilla oleva diagnostiikka on paljon yksityiskohtaisempaa ja helpommin saatavilla kuin aiemmin.

Täysin ongelmatonta Perlin käyttäminen Windows-ympäristössä ei toki ole. Erityisen paljon ongelmia tuotti ratkaista ongelmat, joita liittyi komentosarjan suorittamiseen asiakaskoneilla, kun Perlin asentaminen kaikille työasemille ei ollut järkevää tai mahdollista. Myös binäärimuunnoksen toiminta, nopeus ja luodun binäärin koko jättävät vielä varsin paljon parantamisen varaa. Sama ongelma ilmeisesti vaivaa myös muita tulkattavia ohjelmointikieliä, joiden tuottamat binäärit ovat samaa kokoluokkaa. Tämä selittyy osittain sillä, että jokaisen binääritiedostoon on sisällytetty kyseisen ohjelmointikielen tulkki, ja sen tarvitsemat peruskirjastot, jotka vievät yksinäänkin useita megatavuja.

Lähteet

Kirjallisuuslähteet

Christiansen Tom & Torkington, Nathan 2003. Perl Cookbook: Solutions and Examples for Perl Programmers, 2nd edition. Sebastopol: O'Reilly

Clines, Steve & Loughry, Marcia 2008. Active Directory For Dummies, Second Edition. Indianapolis: Wiley Publishing

Roth, Dave 2000. Win32 Perl Scripting: The Administrator's Handbook. Indianapolis: New Riders

Roth, Dave 2002. Win32 Perl Programming: The Standard Extensions, Second Edition. Indianapolis: New Riders

Wall, Orwant & Christiansen 2002. Programming Perl 3rd edition. Sebastopol: O'Reilly

Internet-lähteet

ActiveState, ActivePerl 5.10.0.1004 Installation Guide [online] [viitattu 24.11.2008] http://docs.activestate.com/activeperl/5.10/install.html

CPAN, Mail::Sendmail [online] [viitattu 8.11.2008]
http://search.cpan.org/~mivkovic/Mail-Sendmail-0.79/Sendmail.pm

CPAN, Socket.pm [online] [viitattu 8.11.2008]
http://search.cpan.org/~rgarcia/perl-5.10.0/ext/Socket/Socket.pm

CPAN, Win32API::Registry [online] [viitattu 4.11.2008]
http://search.cpan.org/~blm/Win32API-Registry-0.30/Registry.pm

CPAN, Win32::Registry [online] [viitattu 4.11.2008]
http://search.cpan.org/~jdb/Win32-Registry-0.10/Registry.pm

Microsoft Developer Network: AddPrinterConnection Method [online] [viitattu 13.11.2008]
http://msdn.microsoft.com/en-us/library/kxsdca3c(VS.85).aspx

Microsoft Developer Network: About WMI (Windows) [online] [viitattu 8.11.2008]
http://msdn.microsoft.com/en-us/library/aa384642(VS.85).aspx

Microsoft Developer Network: WshNetwork Object [online] [viitattu 13.11.2008] http://msdn.microsoft.com/en-us/library/s6wt333f(VS.85).aspx

Microsoft Windows Client TechCenter: Understanding Sites, Subnets and Site Links [online] [viitattu 4.10.2008] http://technet.microsoft.com/en-us/library/cc754697.aspx

Microsoft Help And Support: How Domain Controllers Are Located in Windows XP [online] [viitattu 4.10.2008] http://support.microsoft.com/kb/314861

McLean, Grant, CPAN: XML::Simple [online] [viitattu 7.10.2008] http://search.cpan.org/~grantm/XML-Simple-2.18/lib/XML/Simple.pm

Mueller, Richard L., Logon Script FAQ [online] [viitattu 4.10.2008] http://www.rlmueller.net/LogonScriptFAQ.htm

Liitteet

Liite 1: Perl-kääntöympäristön luominen

Ensimmäisenä työvaiheena asennetaan MinGW, jonka avulla varsinainen binääri luodaan. MinGW voidaan asentaa muutoin oletusasetuksin, mutta mukaan täytyy ottaa C++-kääntäjä g++.

Ellei ActivePerliä ole jo asennettu, asennetaan se seuraavaksi. ActivePerlin oletusasennus on riittävä. Seuraavaksi käytetään ActivePerlin mukana tullutta Perl Package Manager –ohjelmaa, jonka avulla asennetaan moduulit Get-opt::Argvfile, module::scandeps, par::dist ja parse::binary.

Kun ActivePerl ja vaaditut moduulit ovat paikallaan, asennetaan dmake (4.11 tai uudempi) osoitteesta http://search.cpan.org/~shay/dmake-4.11-20080107-SHAY/. Dmake asennetaan MinGW:n alihakemistoon dmake, eli jos MinGW:n asennuksessa on käytetty oletushakemistoa, asennetaan dmake hakemistoon C:\MinGW\dmake. Dmakea käytetään kolmen moduulin, jotka eivät ole PPM:n kautta saatavissa, asentamiseen. Dmaken asentamisen jälkeen luodaan ympäristöasetuksia luova .bat-tiedosto seuraavalla sisällöllä:

```
set MINGW_PATH=C:\MinGW
set PERL_PATH=C:\Perl
set
PATH=%MINGW_PATH%\bin;%MINGW_PATH%\mingw32\bin;%MINGW_PATH%\dm
ake;
set
PATH=%PATH%C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\system32\
WBEM;
set LIB=%MINGW_PATH%\lib;
set INCLUDE=%MINGW_PATH%\include;
set PATH=%PERL_PATH%\site\bin;%PERL_PATH%\bin;%PATH%
cmd /K
```

BAT-tiedostosta on hyvä tehdä pikakuvake, jolloin se on helpompi käynnistää. Seuraavaksi asennetaan paketit Win32::Exe, PAR ja PAR::Packer manuaalisesti. Kaikki löytyvät osoitteesta http://search.cpan.org/~smueller/ tai CPAN-haun avulla. Jokainen paketti puretaan omaan väliaikaishakemistoonsa. Kun paketit on purettu, käynnistetään komen-

totulkki edellä luodusta .bat-tiedostosta. Jokaisessa hakemistossa suoritetaan järjestyksessä seuraavat komennot:

```
perl Makefile.pl
dmake
dmake test
dmake install
```

Komento dmake test voi kirjoittaa joitakin virheilmoituksia konsoliin, mutta näistä ei tarvitse välittää. Asennus onnistuu todennäköisesti niistä huolimatta. Kun kaikki kolme pakettia on tällä tavoin asennettu, kääntöympäristö on valmis. Perl-ohjelmia voidaan käätää pp:n avulla exe-tiedoistoiksi seuraavalla komennolla:

```
pp –z 9 –o ohjelma.exe ohjelma.pl
```

Tämä luo Perl-ohjelmasta ohjelma.pl binäärin ohjelma.exe. –z 9 valitsin pakkaa exe-ohjelman mahdollisimman tehokkaasti, jotta säästettäisiin mahdollisimman paljon tilaa.