1.344 Ft

II. Évfolyam 5. szám

ISSN 15865185 05

9 771586 518005

tech.net

A Microsoft Magyarország szakmai magazinja

Tűzfal

Microsoft Internet Security and Acceleration Server 2000 (bemelegítés)

W2K **LDAP, LDIF**

Developer **Hailstorm**

Lifelong Learning

Tizedik kiadott lapszámunkhoz érkeztünk e májusi számmal, s az „ojság" *(ahogy a levelezési listákon említeni szokás)* igen szép statisztikákkal dicsekedhet átlagos olvasottságát illetően. Átlagos olvasottság alatt azt értem, hogy nagy átlagban minden cikk elolvasódik – de messze nem mindegyik hasznosul! Valahogy talán a rohanó világ az oka, hogy a szakembereknek nincs igazán idejük, vagy idegük el is játszadozni mindazokkal a lehetőségekkel, melyeket feltárunk. Természetesen mindenki 100%-ban befogadja saját szűkebb szakterületéhez tartozó cikkeinket, de ezzel csak céljaink 5%-át értük el. Az „ellenséges" cikkekről – mondjuk ki nyíltan – lepattannak olvasóink. Mit értek „ellenséges" alatt? Például egy rendszergazda számára „ellenséges" cikk egy, a Transact SQL mélységeiben bányászó sorozat. Egy fejlesztő számára „ellenséges" cikk az Exchange és az Active Directory kölcsönhatását feszegető iromány.

Célunk, hogy minden magyar informatikusnak megadhassuk azt, amit a napi robot nem képes megadni: a kitekintés, a felfedezés, a hatékony továbblépés lehetőségét.

Gyakran kérdezik tőlem, hogy nem hiányzik-e nekem az „igazi" munka, egy éles rendszer fenntartásának napi gyakorlata? Hogy-hogy van bőr az orcámon szaktanácsot adni olyanoknak, akik nap mint nap ugyanazt a témát nyűvik? Ki kell, hogy mondjam: az ember problémamegoldó képességét nem a naponta újraindított gépek, a naponta begépelt sorok, vagy a naponta visszadugott nyomtatókábelek számának növekedése fejleszti, hanem az újra, másra való nyitottság. Nemrégiben egy programozó ismerősömet sikerült meglepnem néhány fantasztikus újdonsággal a Visual Interdevben – pedig ő egy éve abban fejleszt, én meg még csak nem is tartom magam fejlesztőnek, csak bug-gyárosnak.

Hogyan tovább?

Megírtuk. Kiadtuk. Elolvasták. Az Önök elégedettségi szintje 100%-os. A mienk 5%. Hogyan lehetne megnövelni az „ellenséges" szakterületek iránti fogékonyságot? Például:

Hogyan lehetne rávenni egy Exchange gurut, hogy ugyan mélyedjen már el az XML-ben, vagy akár az LDAP-ban?

Az élet sodra rákényszeríthet erre a nem kívánatos lépésre, mert – fenti példánknál maradva – az Exchange 2000 például elképesztően sokrétűen használja ki az XML adta lehetőségeket *(OWA!)*, amelyhez előbb-utóbb hozzá kell férnünk, s az LDAP sem úszható meg hosszú távon, mert egyszer majd olyan Recipient Policyt kell faragni, amit szimpla varázslással nem lehet megalkotni *(lásd LDAP, LDIF cikkemet e lapszámban)*. Például ilyet:

```
(&(&(&(&(&(mailnickname=*)
(|(&(objectCategory=person)(objectClass=user)
(|(homeMDB=*)(msExchHomeServerName=*)))
(objectCategory=group))))(objectCategory=user)
(company=VelvetHammer)))
```

Nesze nekünk! És ki merné állítani, hogy ez a szörnyű LDAP nem gyűrűzik be előbb-utóbb mindenhova?

Hogyan lehetne rávenni egy rendszergazdát, hogy ugyan tanulja már meg a WMI-t, és soha többé nem lenne olyan kérdése, hogy hogyan kell egy távoli gépen éjjel egykor Disabled-be tenni a harmadik hálókártyát *(reggel ötkor meg vissza)*.

Ehhez már nem elég az élet sodra. A WMI roppant hatékony tud lenni, de sajnos a fordulatszám felvételéhez nem elég egy WMI lélekelemző cikk. *(Mint ahogy az SQL nyelvhez sem biztos, hogy eljut mindenki pusztán azáltal, hogy olvas a tranzakció-izolációs szintekről.)*

Valami kell még. Hmmm. Mi is?

Szép szó

Ez az. A szép, emberi beszéd! Az hiányzik! Milyen szép is lett volna, ha anno elmagyarázza nekem valaki a WMI mögötti objektummodell mögötti filozófia mögötti lényeget! A koncepciót! Hogy az egy más világ! Hogy a WMI objektummodellje úgy hierarchikus, hogy közben hálós! Hány felesleges órát megspórolhattam volna, ami arra ment el, hogy küszködtem, és nem értettem a nyomorultat, mert HÁLÓS!

Ugyanez elmondható az eddig példaként felhozott SQL nyelvről is. Akinek újdonság, hogy az SQL halmazorientált nyelv, az mindaddig lepattan az SQL-t fejtegető cikkekről, amíg a kezdő lökésen, a filozófiaváltáson túl nincs.

A megoldás

A megoldás az általunk eddig is dédelgetett Lifelong Learning *(LLL)* koncepcióban rejlik. Ősztől a NetAcademia Kft. több szinten, és több témakörben indít délutánonkénti oktatást, ahol szakavatott, tűzkeresztségen már átesett oktatók *(avagy nagyképűbben: mentorok)* segítik az önálló tanulást. Ezek között lesz tandíjas, tematikus tanfolyamsorozat *(MCP vizsgaelőkészítő tanfolyamok)*, de lesz olyan, amelynek belépti díját már kifizették Kedves Olvasóink az újságelőfizetési díj átutalásakor: ezek Mesterkurzus néven futnak majd. Tudom, rohamosan közeleg a nyár, de talán egy *(két)* próbaidőpontot elbír még a tavasz, május és június hó folyamán találkozzunk egy-egy

NetAcademia Mesterkurzus

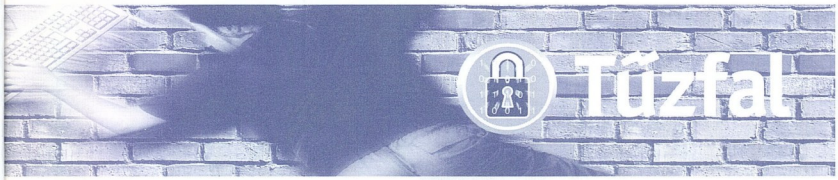
keretében, ahol a lepattanó cikkeket passzolja vissza mandínerből azok szerzője, hátha második körben már fejbetalál ;) Ezek az alkalmak minden előfizetőnk számára ingyenesek lesznek.

A NetAcademia Mesterkurzusokra szándékaink szerint minden hónap utolsó péntek délutánján, 2-től 6-ig kerülne sor, melyre szeretettel meghívjuk minden Kedves Olvasónkat. A Mesterkurzus ötlete oly friss, hogy e cikk megírásakor nem állt rendelkezésemre pontos helyszín, időpont és tematika sem, ezekről az újság borítékjában elhelyezett szórólapokon szerezhet tudomást.

A Lifelong Learning project keretében pedig ősszel indítunk MCP vizsgaelőkészítő esti kurzusokat azok számára, akiknek a munkanap nem áldozható fel a tanulás oltárán.

Fóti Marcell
marcellf@netacademia.net

2001 Május



tech.net

A Microsoft Magyarország Szakmai Magazinja

Szerkesztőség
Főszerkesztő: **Fóti Marcell**
marcellf@netacademia.net
Főszerkesztő-helyettes: **Fülöp Miklós**
mick@netacademia.net
Szerkesztőség címe:
1105 Budapest, Ihász utca 13.
Tel.: 263-2732
technet@netacademia.net
Nyilvános levelezési lista:
tech.net@lyris.netacademia.net

Kiadja és terjeszti
a **NetAcademia Kft.**
Terjesztési, előfizetési információ:
Tel.: 263-2732
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft
Példányszám: 3.000

Előfizethető megrendelőlevélben a szerkesztőségnél:
1105 Budapest, Ihász utca 13.
Fax: 261-7145
http://technet.netacademia.net/subs

Hirdetésfelvétel:
Bársonykalapács Marketing
Felelős: **Udvarev Rita**
Tel./Fax: 214-0923
info@velvethammer.hu
1027 Budapest, Fő utca 67. V. 1.
Grafikai tervezés, kivitelezés,
nyomdai előkészítés:
Bársonykalapács Marketing
Művészeti vezető: **Balogh Zoltán**

Nyomda:
Cerberus Kft.
1066 Budapest, Lovag u. 14.
Felelős vezető: **Schmidt Gábor**

ISSN 1586-5185

Nem lesz SP7

... legalábbis a Windows NT 4.0-hoz **[1]**. A hónap egyik legfontosabb híre, hogy a Microsoft felfüggesztette a Windows NT 4.0 Service Pack 7 fejlesztését. Az indoklás szerint az utóbbi időben a javítások száma folyamatosan csökkent *(no meg jön az új oprendszer – [mICK])*, ezért az újabb javítócsomagra már nem lesz szükség. A – most már biztosan utolsó – SP6a óta megjelent biztonsági javítások összecsomagolása mellett a felhasználók által várt két legfontosabb dolog a Directory Services Client **[2]**, valamint a nemzetközi High Encryption verziók **[3]** – ezek már külön-külön is elérhetők. Az első probléma nyitott maradt, ezért a harmadik negyedévben *(az SP7 eredetileg tervezett megjelenésekor)* megjelenik majd egy, az SP6a-ra telepíthető, biztonsági javításokat egyben tartalmazó upgrade csomag.

Windows 2002

A Whistler Server végleges neve Windows 2002 lesz – jelentette be a Microsoft április 30-án **[4]**. Bár a – legújabb hírek szerint – októberben megjelenő Windows XP és az őt valamikor 2001 első negyedévében követő kiszolgálócsalád, a Windows 2002 egy tőből fakad, a Microsoft továbbra is megkülönböztetni látszik a felhasználói és kiszolgálói oldalt. A Windows 2002 jelenleg – az XP-hez hasonlóan – béta 2 állapotban van, de az XP-vel ellentétben a Windows 2002 megjelenése előtt még egy béta 3 verzió készül majd.

Kirúgták az Office segédet

Az Office Segéd, más néven Clippy, a sokunk által *(nem)* kedvelt bajkeverő nyugdíjba vonul. Az Office 97-ben megjelent segéd eddig folyamatosan jótanácsaival látott el bennünket, mire végre megtaláltuk a módját, hogy eltüntessük az útból szegényt. *(Az Office 2000-ben ezen a területen fejlődött egyébként a legtöbbet a dolog)*. Az Office XP-ben *(eX-Paperclip)* a segédek már csak külön kívánságra bújnak elő, ha valaki mégis hiányolná őket. A Microsoft sokmillió dolláros kampányba kezdett **[5]**, ami Clippy kirúgásáról szól – a kampány eszmei mondanivalója az, hogy az Office XP használata már olyan egyszerű, hogy nincs többé szükség a Segédek közreműködésére.

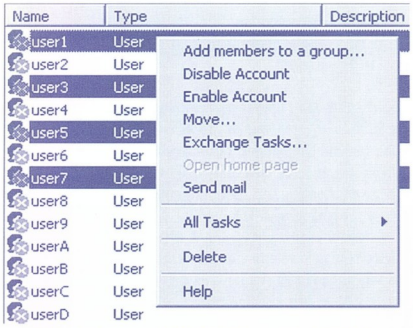
A cikkben található URL-ek:

[1] http://www.microsoft.com/ntserver/sp7.asp
[2] http://www.microsoft.com/windows2000/news/bulletins/adextension.asp
[3] http://www.microsoft.com/windows/ie/
[4] http://www.microsoft.com/presspass/features/2001/apr01/04-30w2k.asp
[5] http://www.officeclippy.com

LDAP, LDIF

Ezzel a cikkel nem kevesebbet vállaltam, minthogy egyszer s mindenkorra mindenkit átrugdalok az LDAP vizsgán, hisz az Active Directory már évek óta velünk van; nem rendszergazda a rendszergazda, ha nem ismeri saját rendszerét! Ráadásul amint egyre több címtáralapú alkalmazás születik *(Exchange 2000, ISA stb.)*, mindennapi munkánk során is egyre gyakrabban fogunk belebotlani az átkozott*(nak tűnő)* LDAP filterekbe. Lásd Exchange 2000 Recipient Policies. Van azonban ennél földhözragadtabb példám is...

Levelezési listáinkon merült fel egy igen érdekes igény a Windows 2000 Active Directoryval kapcsolatban: vajon hogyan lehet egynél több felhasználón egyszerre megváltoztatni valamilyen beállítást - mondjuk a home könyvtárat? A régi, bevált NT4-es módszer ugyanis *(sok felhasználó együttes kijelölése a User manager-ben, és a Properties menüpont kiválasztása)* nem működik az Active Directory Users and Computers eszközzel! Amint ugyanis egynél több felhasználót jelölünk ki, a Properties menüpont egyszerűen eltűnik.



Több objektum egyidejű kijelölése esetén NINCS Properties menüpont!

Hja kérem, a fejlődés megállíthatatlan. Más baj is van: ha nem egyazon OU-n belül szeretnénk válogatni, a hierarchikus felépítés miatt a csoportos kijelöléssel áthághatatlan egérakadályokba ütközünk.

Sok érdekes és előremutató megoldási javaslat elhangzott, írjunk ADSI script-et *(erről tavaly októberi számunkban olvashattak részletesen)*, vegyünk X dollárért külső szoftvert, de valahogy az egyik legkézenfekvőbb, legegyszerűbb, legszabványosabb, ingyenes módszer nem ismeretes a rendszergazdák előtt: export-import!

Export-import

Az Active Directory címtára minden további nélkül szövegfájlba exportálható, és onnan – a megfelelő módosítások elvégzése után – visszaimportálható. Erre kétféle eszköz is létezik: a CSVDE.EXE és az LDIFDE.EXE. Parancssori kapcsolóik *(szinte)* megegyeznek, ám a kettő nem egyenértékű, mindegyiknek megvan a maga szerepe.

CSVDE

A CSVDE.EXE nevéből is kitalálható módon *(Comma Separated Directory Export)* vesszővel határolt fájlokkal dolgozik, ami rendkívül hasznos, ha az adatokon adatbázisjellegű utófeldolgozást, módosítást szeretnénk végezni. Excel-lel kiválóan kezelhető, táblázatos kimenetet produkál, amit *(jobbára)* simán vissza lehet termelni a címtárba a módosítások után. Ugyanakkor csak új objektumok létrehozására és *(korlátozott)* módosítására alkalmas, törlésre és átszervezésre már nem jó, mivel a CSV fájlban csak és kizárólag adatok helyezkedhetnek el, parancsokat belevenni nem lehet. Azért írtam, hogy csak jó esetben lehet vele visszaimportálni az adatokat, mert ha nem vigyázunk, már az exportáláskor belekerül az outputba egy csomó olyan adat, amit visszaimportálni nem lehet, mert módosíthatatlan *(read only)*. Ilyen érték például a SID, és a replikációhoz tartozó USN értékek. Ezektől úgy lehet megszabadulni, hogy az exportáláskor használjuk a -m és -n kapcsolókat, melyek letiltják a bináris szemét fájlba írását. *(A két parancs kapcsolóinak teljes körét a cikk végén foglaltam össze.)*

LDIFDE

Az LDIFDE.EXE ezzel szemben speciális szövegfájlformátummal dolgozik, melynek felépítése a 2849-es RFC-t követi, s amelyben szépen elférnek egymás mellett adataink, és utasításaink. Az LDIF valójában az LDAP címtárak közötti replikáció szabványa, így nem csoda, ha mindenre képes: töröl, módosít, objektumot mozgat egyik szervezeti egységből a másikba, jelszót állít stb. Okos gyerekek tanulnak a University of Michiganen! *(Ott született az LDIF –a szerk.)* Talán emiatt is riadnak sokan vissza azonnal ettől a megoldástól, mert ha olyan sokat tud, akkor bizonyára halálosan bonyolult.

A félelem alaptalan, az LDIF fájlok tökéletesen érthetők. Annyiban azonban mégiscsak megérthető a viszolygás, hogy sajnos nemcsak egy szintaxist kell ismerni a teljes körű használathoz, hanem négyet:

- Az LDIF szintaxis ismerete szükséges a fájl elkészítéséhez, módosításához *(RFC 2849, **[2]**)*
- LDAP Distinguished Name segítségével állítjuk be az export/importot a hierarchia megfelelő pontjára *(RFC 2253, **[3]**)*
- LDAP keresőfeltétellel válogatjuk le exportáláskor a módosítani kívánt objektumokat *(lengyel jelölés, RFC 2254, **[4]**)*

És végül itt van még magának az LDIFDE.EXE-nek a sok ezer kapcsolója.

Hmmm. Hol is kezdjem? Legyünk sikerorientáltak, így elsőként valami működőképeset mutatok: az alábbi LDIF fájl létrehozza a „marketingesek" nevű szervezeti egységet a netacademia.net tartomány gyökerében:

```
dn: ou=marketingesek,dc=netacademia,dc=net
changetype: add
objectclass: organizationalUnit
ou: marketingesek
```

Ennyi. Pár sorral lejjebb el is magyarázom, de először lássuk a fájlt munka közben! Ha a fenti 4 sort begépeljük Notepad segítségével egy közönséges textfájlba, vagy leszedjük az LDIF példákat a **[1]** címről, már importálhatjuk is a címtárba. Mentsük el marketing.txt néven, hogy ezzel a névvel elejét vegyük annak, hogy bárki a jövőben esetleg belenézzen ebbe a fájlba, vagy akár le merészelje törölni ***(bűvös fájlnév!)***. Felhívnám Kedves Olvasóim figyelmét arra a könnyen belátható, ámde szomorú tényre, hogy nem minden vállalatnál netacademia.net a tartomány neve, így ezen hivatkozást ***(dc=netacademia,dc=net)*** a teljes siker érdekében érdemes átírni próbaképpen a vállalatnál használt tartomány valódi nevére ***(például dc=vallalat,dc=hu vagy ami éppen a valóság)***. Beimportálni így kell:

```
LDIFDE -i -f marketing.txt
```

Ezzel az aktuális bejelentkezett felhasználó nevében a legközelebbi tartományvezérlőhöz csatlakozva az LDIFDE.EXE import módban lefuttatja parancsfájlunkat. További kapcsolókkal meg lehetne változtatni a biztonsági jellemzőket ***(név+jelszó)***, illetve kiszolgálót lehet váltani, de ezekre később térjünk ki részletesen. Most valami ilyesmit válaszol a program:

```
E:\>ldifde -i -f a.txt
Connecting to "platan.netacademia.fm"
Logging in as current user using SSPI
Importing directory from file "a.txt"
Loading entries..
1 entry modified successfully.

The command has completed successfully
```

Ellenőrizzük, hogy igazat mond-e, nézzük meg az Active Directory Users and Computerssel, hogy létre jött-e „Marketingesek" nevű OU. Ha nem, annak a következő okai lehetnek:

1) Elgépelés. Megfelelően félregépelt „dn" segítségével akár még LDAP referralt is kaphatunk:

```
The server side error is "A referral was
returned from the server."
```

de könnyen előfordulhat ilyen válasz is:

```
There is a syntax error in the input file
Failed on token starting with 'd' on line 4
```

2) Jogosultsághiány. Az éppen aktuálisan bejelentkezett felhasználó erejével fut a fenti parancs!
3) DNS hiba. Minden Active Directory hiba oka a DNS-ben keresendő!

Most vizsgáljuk meg a parancsfájlt sorról sorra!

LDAP útvonalak

dn: ou=marketingesek,dc=netacademia,dc=net

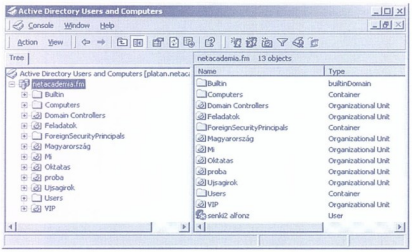
A "dn" az X.500 szabvány szerinti megkülönböztető név *(Distinguished Name, RFC 2253, **[3]**)* rövidítése, amely egy címtárobjektum teljes címtárazonosítója, olyasmi, mint a fájlrendszerben egy fájl teljes útvonala *(például C:\adatok\fontos.doc)*.

A fájlnévhasonlat több okból is telitalálat. Egyfelől a "dn"-nek éppúgy globálisan egyedinek kell lennie, mint egy teljes útvonalnak, másfelől éppúgy leválasztható róla az objektum saját neve *(rdn, Relative Distinguished Name)*, mint ahogy az útvonalból kiolvasható a fájlnév. Továbbá mindaddig lehetnek a címtár különböző pontjain azonos nevű objektumaink, amíg egy „könyvtárban" nincs két egyforma nevű *(mint ahogy ezer különböző könyvtárban lehet readme.txt nevű fájl, de egy adott könyvtárban egyszerre csak egyetlenegy fér meg)*. Magának a "dn" útvonalnak a szintaxisa sem túl bonyolult. Éppúgy balról jobbra dolgozunk vele, mint a DNS tartománynevekkel, jobbról balra a meghatározás egyre szűkebb. Az RFC szerint a következő útvonalazonosítók léteznek:

Azonosító	X.500 attribútum
CN	commonName
L	localityName
ST	stateOrProvinceName
O	organizationName
OU	organizationalUnitName
C	CountryName
STREET	streetAddress
DC	domainComponent
UID	userid

A következő gyakori útvonalazonosítókkal találkozunk a Windows 2000-ben *(a többivel én speciel még nem találkoztam)*:

- DC, Domain Component, tartományösszetevő. A tartományunk nevét adjuk meg ezzel a kulcsszóval, mégpedig úgy, hogy a tartomány DNS nevét tagonként felcímkézzük. Azaz ha a tartomány neve kukutyin.hu, akkor az X.500 útvonal így fest: DC=kukutyin,DC=hu. A DNS név és a DC hasonló felépítése nem véletlen, ez a szócska az átjáró az X.500 és a DNS világ között. Ha ugyanis címtárhozzáférésünk során nem adunk meg kiszolgálónevet *(ahogy a fenti példában sem)*, akkor a Windows a DC összetevők visszafejtése révén nyert DNS tartománynév alapján keresi meg a célba vett domaint – mégpedig DNS lekérdezéssel!
- OU, Organizational Unit, szervezeti egység. A tartománybelső hierarchiájának leírására való. Ha egymásba ágyazott szervezeti egységeink vannak, azokra így tudunk hivatkozni: OU=legalul,OU=kozepen,OU=legfelul. Érdekes jelenség, hogy az Active Directory telepítésekor megjelenő tárolók többsége nem OU, nem valódi, hanem álkonténer, így nem OU-ként hivatkozunk majd rájuk, hanem CN-nel. Az alábbi ábrán egy tartomány alapértelmezett, és utólag létrehozott konténereit láthatjuk. A valódi OU-k ikonja könyvecskés, az álkonténereké sima.



Amelyik OU ikonja nem „könyvecskés", az valójában nem is OU

CN, Common Name, objektumnév. Ez a szócska az úgynevezett Relative Distinguished Name megjelölésére szolgál. Ezzel azonosítjuk az objektumokat (*CN=Kis Pista, CN=Senki Alfonz stb.*). Az Active Directoryban felhasználók és egyéb hétköznapi objektumok mellett rendszeradatokat is találunk (*CN=Configurarion, CN=Schema és így tovább*), ezeken túlmenően az álkonténereket is hasonlóan címezzük (*CN=Users, CN=Computers stb.*).

Így a Falatrax cégnél dolgozó Kovács felhasználó, ha a Users tárolóban van:

```
CN=Kovács,CN=Users,DC=falatrax,DC=hu
```

azonban ha a marketingesek szervezeti egységben helyezkedik el:

```
CN=Kovács,OU=Marketingesek,DC=falatrax,DC=hu
```

LDIF módosítások

Egy lépessel közelebb kerülhetünk a végső célhoz, ha megkíséreljük módosítani egy meglévő felhasználó valamelyik adatát - mondjuk az Administrator fiók Description mezőjét. A következő LDIF fájl lesz a segítségünkre:

```
dn:CN=Administrator,CN=Users,DC=netacademia,
 DC=net
changetype: modify
replace: description
description: blablabla
-
```

Ez sem lényegesen bonyolultabb az előzőnél, s beimportálni is ugyanúgy kell. Azonban itt már látszik, hogy az LDIF szintaxist is tanulni kell. Az utolsó sorban a mínuszjel nem véletlenül van ott, hanem a parancs lényeges része. Az LDIF fájl általános felépítése a következő:

```
dn: útvonal
changetype: parancs
adatok
<üres sor>
dn: útvonal
changetype: parancs
adatok
<üres sor>
...
```

Az útvonal szerencsére már a markunkban van. A changetype sorban a következő parancsok szerepelhetnek:

```
add, delete, modify, modrnd, moddn
```

Ezek közül a két utolsó átnevezésre és átszervezésre való. A modify parancs kivételével az adatsorok folyamatosan követik egymást. Azonban a modify különleges, mert egyetlen módosítás során egy adott objektum több attribútumát egyidejűleg kezelhetjük: törölhetünk, felülírhatunk, vagy újat vehetünk fel. Ezeket az alparancsokat választja el a kötőjeles sor. Az alábbi példában egyszerre cserélem a description mezőt, törlöm a telefonszámokat és beállítom a dolgozó főnökét:

```
dn:CN=Valaki,OU=Marketingesek,DC=netacademia,DC=net
changetype: modify
replace: description
description: blablabla
-
delete: telephoneNumber
-
add: manager
manager:CN=senki,OU=Marketingesek,DCnetacademia,DC=net
-
```

Most próbáljuk ki a changetype:add parancsot, és hozzunk létre egy új felhasználót!

```
dn: CN=Hapsi,DC=netacademia,DC=net
changetype:add
objectclass: user
SamAccountName: Hapsi
```

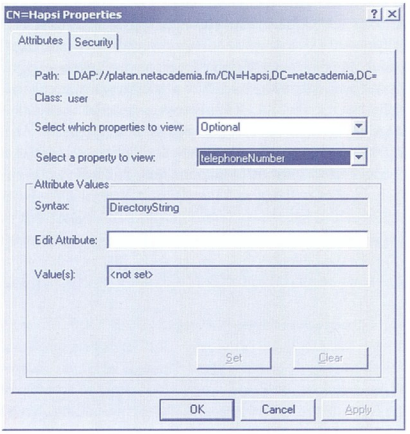
Nem igényel túl sok kötelező paramétert ugye? Se jelszó, se semmi! Ha leellenőrizzük Hapsit, láthatjuk, hogy létrejött a domain gyökerében, ám tiltott *(Account Disabled)* állapotban leledzik, erről árulkodik a piros jelecske: Hapsi User

Próbáljuk őt engedélyezni, vagy beállítani a homekönyvtárát, vagy bármely más attribútumát, természetesen LDIF-fel! Itt súlyos akadályba ütközünk: mindent tudunk a módosítás mikéntjéről, de vajon mi a kívánatos attribútum LDAP neve? Mit írjunk az LDIF fájlba?

A címtár felfedezése

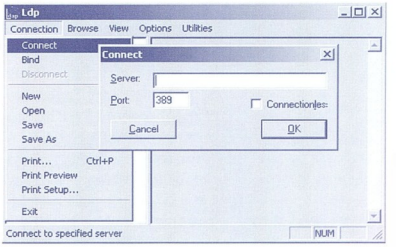
Természetesen az volna a legjobb, ha valamilyen kipróbált módszerünk lenne a címtár részletes kilistázására, ha lenne olyan eszköz, mely hasonló részletességgel tárja fel az Active Directoryt, mint ahogy a rendszerleíró adatbázist a regedt32.exe. Van ilyen eszköz, mégpedig ingyen! A Windows 2000 CD-n, a \Support\Tools könyvtárban található mini Resource Kit sok hasznos csecsebecse mellett tartalmaz nem is egy, hanem mindjárt két címtártúrót, az ADSIEditet és az LDP.EXE-t. E kettő közül az ADSIEdit felhasználói felülete a barátságosabb, ezért sokan azt hiszik, az a hasznosabb. A teljesen fapados LDP.EXE-nek azonban megvan az az óriási előnye, hogy sohasem hazudik. Az Active Directory rendkívül takarékosan bánik a tárolóhellyel: ha egy objektum egyik attribútumát *(például*

egy felhasználó telefonszám mezőjét) nem töltjük ki, akkor nem is tárol ilyen nevű mezőt, azaz a user-nek mindaddig egyáltalán nincs telefonszám-attribútuma az adatbázisban, amíg ezt a mezőt ki nem töltjük. Az LDP.EXE hűen tájékoztat a valóságról, míg az ADSIEdit az Active Directory séma alapján odahazudik egy üres attribútumot, amint az az alábbi ábrán is látszik. Emiatt az LDP-t választjuk.



Hapsinak egyáltalán nincs telephoneNumber attribútuma, az ADSIEdit mégis úgy mutatja, mintha lenne.

Fapados, puritán felhasználói felülete a gyakorlatlanabb rendszergazdákat elbizonytalanítja, nagyon-nagyon sokan képtelenek használni ezt a csodaeszközt, pedig milyen egyszerű! Ha ismerjük az LDAP protokoll működését, az LDP.EXE menüpontjai ismerősnek tűnnek. Nem más ez az eszköz, mint az LDAP protokollra épített példaalkalmazás, ezt az érzetet erősíti a kidolgozatlan felhasználói felületen kívül a tökéletes protokollhűség. A munka megkezdése előtt ugyanis kapcsolódnunk kell egy LDAP kiszolgálóhoz *(Connect)*, majd a megfelelő jogosultság megszerzéséhez létre kell hoznunk egy címtárkötést *(BindRequest)*, s ezután következhetnek a címtárműveletek *(például keresés, SearchRequest)*. Szerencsére e három lépésen az alábbi ábrák tanúsága szerint üres ablakokon keresztül vezet az út, s így a DNS-ből kiválasztott kiszolgálón, az éppen aktív felhasználó jogosultságaival a tartomány gyökerébe jutunk. Indítsuk el az LDP.EXE-t, és az alábbi módon NE töltsük ki egymás után a Connection/Connect, majd Connection/Bind, végül a View/Tree menüpontokra megjelenő ablakokat:

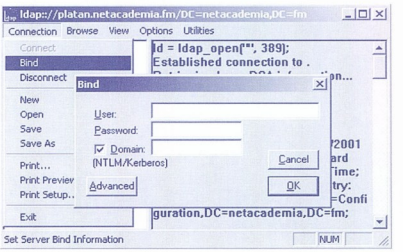


Ha nem adunk meg kiszolgálónevet, az eszköz DNS lekérdezéssel keres LDAP kiszolgálót

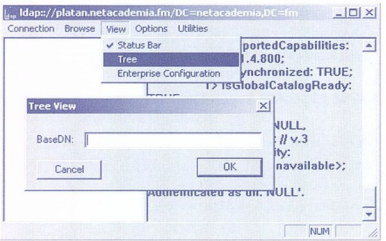
A fenti ábrán szépen látszik a fapados jelleg: a „Connectionless" felirat nem fért be az ablakba. Oda se neki! :) A sikeres kapcsolatfelvétel *(Figyelem! Még nem jelentkeztünk be!)* után a jobboldali panelen visszakapjuk az LDAP gyökér információit *(RootDSE)*, mely tájékoztat ezen címtár által kiszolgált névterek *(naming context)* elérési útvonaláról. Ismeretlen címtárak esetén ez az infó teszi lehetővé, hogy akkor is megtaláljuk az adatokat, ha esetleg a tartomány nevét sem ismerjük.

```
namingContexts: CN=Schema,CN=Configuration,
↳ DC=netacademia,DC=fm;
↳ CN=Configuration,DC=netacademia,DC=fm;
↳ DC=netacademia,DC=fm;
defaultNamingContext: DC=netacademia,DC=fm;
schemaNamingContext: CN=Schema,CN=Configuration,
↳ DC=netacademia,DC=fm;
configurationNamingContext: CN=Configuration,
↳ DC=netacademia,DC=fm;
rootDomainNamingContext: DC=netacademia,DC=fm;
```

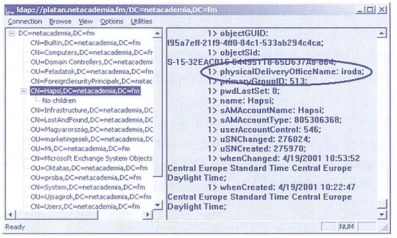
A következő két lépésben rákapcsolódunk az egyik névtérre, amihez viszont már be kell jelentkeznünk:



Ha nem adunk meg nevet és jelszót, az aktuális felhasználó nevében csatlakozunk



Ha nem adunk meg névteret, a tartomány gyökerébe jutunk

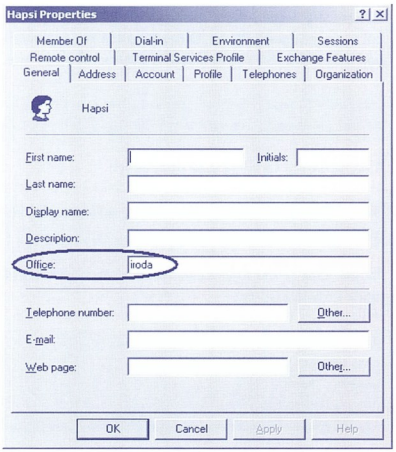


...az a valóságban physicalDeliveryOfficeName!

Három üres ablak, és máris miénk az AD!

Ezután már gyerekjáték megtalálni egy-egy objektum valódi nevét. Nézzük meg például kedvencem, a tulajdonságlapokon Office néven szereplő mező valódi nevét. Ehhez kitöltöm a mezőt az Active Directory Users and Computerssel *(mert ha nem tölteném ki, nem is látszana az LDP-vel)*, majd megkeresem ugyanezen objektumot LDP.EXE-vel, és a jobb panelen olvashatóvá válik az attribútum belső, scriptekben használatos neve:

```
physicalDeliveryOfficeName
```



Ami az AD-ben Office...

Hú, de sokat kellett gépelni! Vajon hogyan teszünk szert *(fél)*kész LDIF fájlra, mely már tartalmazza az objektumokat, hogy csak a módosítást kelljen megírni? Igen, exportálással, de ha nekem például csak a felhasználók kellenének egy ou-ból, hogyan válogatom ki? Gyenge megoldásnak tűnik, hogy minden objektumot kilistázunk, majd az ömlesztett LDIF fájlból kitöröljük, ami mégsem kell. Inkább ne is menjen ki a fájlba! Erre nyújtanak megoldást az LDAP filterek, melyek megszabják, hogy egy OU-n belül mely objektumokra vagyunk kíváncsiak. Ez lesz e cikk 23. szintaxisa, készüljenek, megint valami újdonság jön!

LDAP filterek, lengyel jelölés

Az LDAP szűrők szintaxisa még véletlenül sem hasonlít egyetlen nap mint nap használt lekérdezőnyelvre sem. Semmi SQL! Itt egy olyan őskövülettel találkozunk, melyhez hasonlót azok láthattak, aki még a '68-as diáklázadások előtt vásárolták elektronikus számológépet. Egyik ismerősöm édesapjának szomszédjának kollégája mesélte, hogy hallott olyan emberről, aki látott olyan embert, aki kezében fogott olyan „zseb" számológépet, amelyiken nem úgy kellett begépelni a világmegváltó atomfizikus ősegyenletet, hogy 1+1= hanem úgy, hogy először jöttek a feldolgozandó adatok *(1 és 1)*, majd a műveleti jel *(+)*. Ezek a gépek ugyanis közvetlenül a veremben dolgoztak, amibe be kellett tuszkolni *(push)* az adatokat, majd a műveleti jel beütésével azok onnan kipoppolódtak, kiszámolódtak, és a végeredmény visszakerült a sztekkre. Az ezen logikát követő jelölésmódot nevezzük FORDÍTOTT lengyel jelölésnek. Tehát *(2+3)*4* helyett 2, 3 + 4 *
Az LDAP filterekben ehhez hasonló, ám nem fordított jelölést használunk, ahol elöl áll a műveleti jel, s ezt követik az adatok. Filterről lévén szó természetesen logikai műveletekre kell

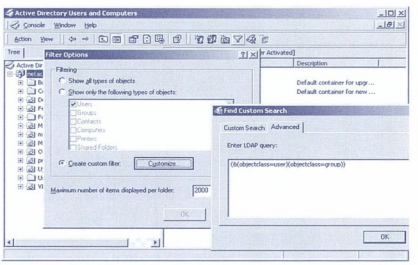
gondolnunk, van logikai ÉS *(jele: &)* VAGY *(jele:|)* és NOT *(jele: !)* műveletünk, továbbá egyenlőség, hasonlóság *(~=)*, kisebb, nagyobb. Ezekkel a jelekkel, és a megfelelő logikai kifejezésekkel válogathatjuk ki egy adott szervezeti egységből az összes user-t:

```
(objectclass=user)
```

az összes user-t és csoportot *(tehát más objektumokat, például névjegyeket nem)*:

```
(&(objectclass=user)(objectclass=group))
```

Bámulatos ugye? Próbáljuk ki újdonsült tudásunkat valami hétköznapi eszközzel – legyen mondjuk az... Active Directory Users and Computers!
Ennek filterét rajtunk kívül még soha emberfia nem használta. Egyszer ezen a teszten is át kell esnie az operációs rendszernek, nem igaz? ;)



Mostantól csak a felhasználók és csoportok lesznek láthatók

Most szűrjük ki az összes olyan csoportot, ahol a csoport neve „a" betűt tartalmaz:

```
(&(objectclass=group)(cn=*a*))
```

illetve azokat a felhasználókat és csoportokat, ahol a description mező ki van töltve:

```
(&(description=*)(|(objectclass=user)
↳ (objectclass=group)))
```

Ez utóbbi példa azért helyes, mert amelyik objektumnál nincs kitöltve a description mező, ott nincs is meg ez az attribútum, tehát tartalma sincs, így az nem egyenlő a csillaggal. Remélem ezen példák alapján bárki össze tud majd állítani tetszőleges keresési feltételt, például ki tudja majd válogatni az összes olyan névjegyet, ahol az email cím nincs kitöltve:

```
(&(objectclass=contact)(!(mail=*))
```

A filtereket természetesen bárhol felhasználhatjuk, ahol LDAP lekérdezést várnak tőlünk, például az LDIFDE.EXE parancssorában, ahol azonban fel kell készülnünk arra, hogy a DOS parancssor különleges karakternek tekinti a & és | jeleket, ezért mindegyik elé ^ *(kalap)* jelet kell tennünk, ez ugyanis mentesíti az átok alól az átkozottat. Lassan minden eszköz rendelkezésünkre áll egy fantasztikusan bonyolult export végrehajtásához, így itt az ideje, hogy a CSVDE.EXE és LDIFDE.EXE többi kapcsolójára is kitérjünk.

CSVDE és LDIFDE kapcsolók

```
-i: a importmód bekapcsolása
-f: az LDIF fájl neve
-s: kiszolgálónév (ha nem adjuk meg, sebaj)
-t: portszám (default = 389, SSL-lel 636)
-d: RootDN, az LDAP keresés gyökere
-r: LDAP szűrő (az alapértelmezés:
↳ "(objectClass=*)")
-p: a keresés mélysége (Base/OneLevel/Subtree)
-l: a kívánatos attribútumok listája
-o: a nemkívánatos attribútumok listája
-m: nem kérjük a SAM bináris szemetét (SID, stb.)
-n: semmilyen bináris szemét nem kell
-b UserName Domain [Password | *]
```

LDIF hardcore

Most néhány olyan példa következik, mely egyszerre használja az eddigi ismereteket, sőt tovább vezet minket az LDAP – LDIF páros alkotó módon történő történő felhasználásához. A gyengébb idegzetűek most hegyják abba a cikk olvasását!

Objektum mozgatása a címtárhierarchiában

Első példánk legyen Hapsi átmozgatása a tartomány gyökeréből a marketingesek OU-ba. Ehhez Hapsit ki kell exportálni, de úgy, hogy CSAK hapsit, és annak is CSAK a DN attribútumát:

```
ldifde -f hapsi.txt -d "DC=netacademia,DC=net"
↳ -r (cn=Hapsi) -l dn
```

A hapsi.txt fájl tartalma ezek után ennyi:

```
dn: CN=Hapsi,OU=marketingesek,DC=netacademia,DC=fm
changetype: add
```

Ezt már csak át kell írnunk olyan módosítássá, ami Hapsit átrepíti a kijelölt OU-ba:

```
dn: CN=Hapsi,DC=netacademia,DC=fm
changetype: moddn
newrdn: Hapsi
deleteoldrdn: 1
newsuperior: ou=marketingesek,dc=netacademia,dc=fm
```

Joggal vetődik fel a kérdés, hogy ezt meg honnan tudom. A válasz talán meglepő, de a 2849-es RFC-ből olvastam ki **[2]**

Jelszómódosítás LDIF-fel

Második példám igazán abszurd, ugyanis Hapsi jelszavát fogom átállítani LDIF-ből, amihez a puszta szintaxison kívül kell 1-2 dolog még, mert ezt a mezőt nem hagyja az Active Directory „csak úgy" babrálni. Sem kiolvasni, sem módosítani nem lehet, egyedül a Replace művelet engedélyezett, az is csak SSL-en keresztül, továbbá High Encryption Package kell hozzá, valamint a jelszót base64 kódolással kell beírni a fájlba:

```
dn: CN=Hapsi,DC=netacademia,DC=net
changetype: modify
replace: unicodePwd
unicodePwd::aGhhbGlobw==
```

Az SSL miatt a parancssor is változik:

```
Ldifde –i –f akarmi.txt –t 636
```

A jelszó átkódolására pedig ezt az agyament, viszont bravúros módszert találta nekem az MSDEV lista:

```
select cast('haliho' as varbinary) as pwd
for xml raw, binary base64
```

A cikkben szereplő példák az **[1]** címen találhatók meg, mégpedig egyetlen, ömlesztett fájlban, ahol a tartománynevet XXX és YYY karakterekkel helyettesítettem, hogy könnyen, gyorsan, egy mozdulattal ki lehessen cserélni a saját, otthoni nevekre. Ezután lehet az egyes részeket kicsi fájlocskákba kikopizni, és lefuttatni.

Utunk végére értünk, remélem szép, ismeretlen tájakat sikerült bemutatnom e kirándulással, legközelebb a WMI sötét sikátoraiba kalauzolom Önöket!

Fóti Marcell
marcellf@netacademia.net
MCT, MCSE, MCDBA

Nehéz volt? És mi ez NetAcademia Active Directory mélyvíz tanfolyamhoz képest!

A cikkben szereplő URL-ek:

[1] http://technet.netacademia.net/feladatok/ldap/ldif.txt
[2] RFC 2849 The LDAP Data Interchange Format (LDIF) – Technical Specification. http://www.ietf.org/rfc/rfc2849.txt
[3] RFC 2253 Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names. http://www.ietf.org/rfc/rfc2253.txt
[4] RFC 2254 The String Representation of LDAP Search Filters http://www.ietf.org/rfc/rfc2254.txt

Microsoft Network Monitor *(VI. rész)*

Vizsgáljuk meg kedvenc operációs rendszerünk rendszertöltés alatt kifejtett aktivitását, melyből igen sok érdekes, a Windows belső működését is megvilágító rendszerfolyamat működésére derülhet fény.

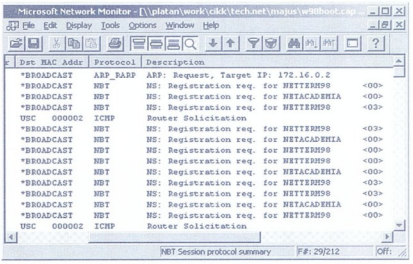
Kezdetben úgy gondoltam, hogy most már nem védem a kedves olvasót a valós hálózatok valós terhelésétől, hisz birtokunkban a szűrők összes változata, ám a webre az [1] címre elhelyezett capture fájljaim mégis mindenféle egyidejű hálózati forgalom zavaró hatásának kiszűrésével készültek, egyszerűen azért, hogy a fájl minél kisebb legyen. Hiszen ha összegyűjtöttem volna mindazt a hálózati forgalmat, ami egy vizsgált Windows 2000 több percnyi rendszertöltése alatt hálózatunkon lezajlott, több megabájtos fájlokat kellett volna az [1] címre mentenem. Annak ellenére, hogy már a forgalom elkapásakor dolgozott egy szűrő, mely csak a kiszemelt gép hálózati forgalmát engedte rögzíteni, a legnagyobb fájl így is 236 kilobájtos lett – ennyibe „kerül" egy Windows 2000 Advanced Server tagkiszolgáló bebootolása!

Három fájlunk van ebben a hónapban:

- A w98boot.cap egy Windows 98 rendszertöltésének forgalmát tartalmazza
- A w2kboot.cap a fent említett Windows 2000 Advanced Server tagkiszolgáló bootolásának folyamatát lesi meg
- A w2kshut.cap ugyanennek a gépnek a leállítását követte nyomon. Erre a cikk terjedelmi korlátai miatt nem térek ki, házi feladat!

A Windows 98 bootolása

Bár a Windows 98 már nem igazán tekinthető friss operációs rendszernek *(egyesek szerint operációs rendszernek sem)*, napjainkban fut még néhány millió a hálózatokon, így nem haszontalan megvizsgálni, mit is művel. A Windows ME hálózati forgalma kísértetiesen hasonló – lenne, ha ezen a gépen futott volna az Active Directory ügyfélszoftver és/vagy a gép bejelentkezett volna a tartományba. Az alábbi ábráról tulajdonképpen teljeskörűen leolvasható, hogy mit tesz a masina a hálózaton: ARP-vel keres egy IP címet, NetBIOS regisztrációt végez a NETTERM98 és NETACADEMIA nevekre, valamint ICMP Router Solicitation üzenetet küld. Ebből a néhány sorból ki is derül, hogy a gép nem DHCP ügyfél, hisz IP cím kérő hálózati forgalom nem ékelődik be a legelső ARP, és a közvetlenül ez után következő NetBIOS névregisztráció közé.



A Windows 98 bootolásának forgalma - részlet

Érdekes megfigyelni, hogy a legelső ARP mintha céltalan lenne: nem várunk a válaszra *(hanem azonnal usgyi Broadcastolni)*, és nem is jön válasz. Mi lehet ez?

Ez az úgynevezett Gratious ARP, melynek feladata az esetleges IP cím ütközések feltárása. Ebben a csomagban a Windows 98 a saját, registryből kiolvasott IP címére kérdez rá a nagyérdeműtől:

```
- Fiúk, kinek az IP címe … az én IP címem?
```

Ha bárki válaszol, akkor az adott IP cím foglalt. Mivel nem jött válasz, a gép használatba veszi a címet. Nem időznék ennél hosszabban az ARP protokollnál, hisz egy korábbi részben szinte bitszinten kielemeztük, ugorjunk a második csomagra, ahol a NETTERM98 NetBIOS név regisztrációja történik.

NetBIOS

A NetBIOS interface *(és NEM protokoll!)* egy viszonylag régi, elavulófélben lévő réteg a Windowsok rétegzett hálózati architektúrájában. Elsődleges feladata egy ember által is könnyen kezelhető névtér hozzárendelése az egyes gépekhez, hogy a kedves felhasználóknak például ne IP cím alapján kelljen távoli meghajtókra csatlakozniuk *(bár IP címmel is lehet. Try: \\12.34.56.78\share !)* Tisztán Windows 2000-es hálózatban nincs is rá szükség, de egyelőre viszonylag kevés ilyen hálózatról tudunk sajnos. A NetBIOS halála azért elkerülhetetlen, mert – ellentétben a DNS-sel – névtere kétdimenziós, azaz hierarchiába nem lehet felfűzni a gépeket.

A NetBIOS nevek 16 bájtosak, ebből 15 bájt van fenntartva a gép/tartomány/szolgáltatás nevének, a 16. bájt pedig az adott névhez tartozó szolgáltatás kódja. A fenti ábrán látható névregisztrációs utasításokon megfigyelhető a név, és a névhez tartozó szolgáltatáskód is, például:

```
NETACADEMIA     <00>
```

A teljesség igénye nélkül felsorolom a leggyakoribb szolgáltatáskódokat, melyekről a Microsoft Knowledge Base-ben olvashatunk bővebben:

00	Workstation Service *(ez a gép képes fájl- és nyomtatószolgáltatás igénybevételére)*
03	Messenger Service *(az adott névre üzenetet lehet küldeni Net Send paranccsal, vagy Winpopuppal)*
20	Server Service *(ez a gép képes fájl- és nyomtatószolgáltatás nyújtására. A 20-as rekord az alapértelmezett, ez nem írja ki külön a NetMon, lásd fenn.)*
1B	Domain Master Browser
1C	Domain Controller
BE	Network Monitor AGent

Egy elinduló gép sorban bejegyzi a WINS Serverbe *(vagy, mint a fenti ábrán, WINS hiányában Broadcasttal adja a többi gép tudtára)*, hogy milyen szolgáltatásokat érdemes nála keresni. *(Egy WINS adatbázis elég sok mindent elárul egy botcsinálta hackernek. A fenti rekordokon kívül ugyanis van még egy-kettő: az IIS is bejegyzi magát, az SQL Server sem tagadja le, hogy fent van egy adott gépen stb.)*

NetBIOS névfeloldásnak nevezzük azt a folyamatot, amikor a munkaállomás a felhasználó által megadott gépnévhez megpróbálja megkeresni a célgép IP címét, és a kért szolgáltatást. Ennek a folyamatnak sokféle változata képzelhető el, de mindig két alapeset kombinációjával állunk szemben: vagy broadcastot használunk, vagy NetBIOS Name Servert *(WINS)*. E kettő módszer mellett létezik persze nem hálózatos névfeloldás is az LMHOST fájl képében, de ez a NetMon szempontjából közömbös, mivel nem látszik a hálón, s így el sem tudjuk kapni. Az előző kettőt azonban igen! Mi több, meglepetéssel tapasztalhatjuk, hogy e kettő vegyesen szerepel a legtöbb hálózaton. Ennek az az oka, hogy a munkaállomás az úgynevezett Node Type beállítás függvényében képes egyszerre akár mindkét módszert használni. A következő Node Type állapotok léteznek:

- B Node: Broadcast üzemmód. Ekkor a munkaállomás kizárólag broadcast alapú névfeloldással próbálkozik, s ha ez nem sikerül, hibaüzenetet kapunk. Ez a típus használhatatlan több szegmensből álló, útválasztóval összekapcsolt hálózatok között, mivel minden jól nevelt router elnyeli a broadcastokat, így ezzel a módszerrel a túlsó hálózat gépei név szerint nem érhetők el. De hangsúlyoznám, hogy IP címmel igen!
- P Node: Peer üzemmód. Ebben az esetben a munkaállomás csak WINS lekérdezést használ, ami ugyan tetszőleges távolságban lévő WINS ügyfél megtalálására alkalmas – de a tőlem másfél méterre lévő gép mégsem lesz név szerint használható, ha az valamilyen okból nem WINS kliens, és nem regisztrálja be nevét a WINS kiszolgálóba.
- M Node: Mixed üzemmód. Az előző két módszer előnyeit egyesíti. A munkaállomás először broadcast-tal próbálkozik, majd ha az nem jött össze, a WINS-hez fordul. Tekintettel arra, hogy egy tipikus vállalati hálózatban majdnem minden gép WINS kliens, ez a stratégia sávszélességpazarló lehet.
- H Node: Hybrid üzemmód. Ilyenkor a munkaállomás először a WINS kiszolgálót kérdezi, majd ha negatív válasz érkezik, bánatában broadcast-ol egyet – hátha sikerül. Saját tapasztalatból mondom, hogy a Hybrid igen hasznos üzemmód, mert ha egy célgép nem WINS ügyfél, a broadcast még mindig elérhetővé teszi név szerint – legalábbis azonos alhálón.

Sajnos fantasztikus eszközünkkel, a Network Monitorral csak következtetni tudunk gépeink Node Type beállítására, a biztos módszer ugyanis a parancssorban lakozik: az IPCONFIG /ALL parancs szépen kiírja a képernyőre a Node Type beállítást. *(Win98 esetén WINIPCFG)*

Akármilyen úton-módon szerezte is meg gépünk a névhez tartozó IP címet, a választ egy helyen tárolja: a NetBIOS Cache-ben. Könnyedén meggyőződhetünk ennek tartalmáról az NBTSTAT -c paranccsal. Az így megjelenő listában nemcsak neveket, hanem szolgáltatáskódokat is találunk, a <0x20> például azt jelenti, hogy a feloldott nevű gépen van fájl-és nyomtatószolgáltatás, azaz fut a Server service.

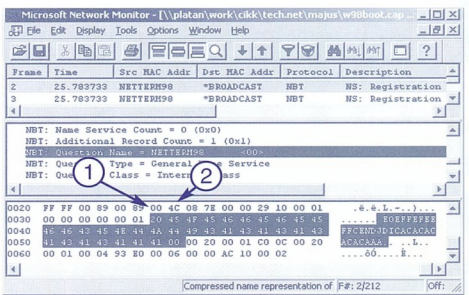
Távoli gépek NetBIOS cache-e is kilistázható, egyszerű módszert adva a kétbalkezes hacker kezébe, hogy megállapíthassa: vajon egy távoli gépen ki van bejelentkezve?

Az **[1]** címről letölthető w98boot.cap fájl 23. csomagjában megfigyelhetünk egy NetBIOS névkérésre érkező választ, melynek kérdésrésze valahova elveszett, eltűnt a Network Monitor árgus tekintete elől, ami nem jó jel. Úgy látszik, ez is csak szoftver...

A 29. csomagban NETTERM98 NetBIOS Session építési kérelmet küld PLATAN-nak, mégpedig a feladó Workstation <00> szolgáltatása a címzett Server szolgáltatására <20>. Erre egyszerre, egy időben két válasz érkezik *(30. és 31. csomag)*, mintha PLATAN dadogna, amit megint csak nem értek – ez is csak szoftver... Vagy a kannásbor miatt?

De térjünk vissza a második csomaghoz, nyissuk ki a NetBIOS névregisztrációt, s keressük meg benne a regisztrált gépnevet *(NETTERM98)*. Nem fogjuk megtalálni! Helyette ez „olvasható":

EOEFFEFEEFFCENDJDICACACACACA



A NETTERM98 név hexadecimálisan...

Ez nem egészen az ugye? Valami titkosítás, vagy átok ül rajta? Nem titkosítás, hanem átok. Úgy van lekódolva a név, hogy minden betűt két bájt tárol, de nem UNICODE, hanem BCD-jellegű, és még csak az sem. UUENCODE? Nem hinném! Az izgalom kedvéért fejtsük meg gyorsan a kódolást:

- Az ① jellel mutatott szám a string hosszbájtja *(0x20=16 bájt hosszú minden NetBIOS név)*
- A következő két bájt *(0x45, 0x4F)* lenne az „N" betű a NETTERM98 névből ②
- az „N" kódja az ASCII kódtáblában 78=0x4E.
- Ezt a 8 bitet vágjuk ketté, 4-4 bitre *(0x4 és 0xE)*
- Adjunk mindkettőhöz 0x41-et *(Az ABC „A" betűje)*, s voilá, előáll a 0x45, 0x4F

- És így tovább a többi karakterre. Ebből látszik, hogy a név és a legutolsó szolgáltatásbájt között az üres hely szóközökkel van feltöltve *(0x43, 0x41->0x20->space)*.

Hogy miért PONT így kódolják a neveket, tőlem akár ne is kérdezzék, fogalmam sincs! Ipartörténeti érdekesség.

ICMP Router Solicitation

A routerkeresésben talán a címzésmód érdemel kiemelt figyelmet. A címzett MAC Addresse *(01005E000002)* nem broadcast. Vajon kitől kéri le a Windows 98 a Default Gateway címét? Lássuk a cél IP címet: 224.0.0.2. Jesszus! Csak nem Amerikában keresi a routert? Nem-nem. Ez egy D osztályú Multicast cím, amely a Unicast és a Broadcast között helyezkedik el félúton: nem is egyetlenegy gépnek szól, de nem is mindnek, hanem csak azoknak, akik ugyanezzel a Multicast címmel rendelkeznek. A 224.0.0.2 cím a Multicast szabvány szerint: All IP Routers. Vagyis a gép bekiabál eme közismert Multicast címre, egy olyan csoportnak szólva, melynek tagja az összes, e szabványt ismerő útválasztó. Vagyis a Windows 98 MEGTANULJA, hol a Default Gateway!

A sok-sok NetBIOS regisztráció és routerfelfedezés után a Windows 98 rákapcsolódik a tartományvezérlő egyik könyvtárára. Minket a könyvtár neve és tartalma annyira nem is érdekel, az autentikáció sem fér e cikk kereteibe, így hát a fájl- és nyomtatószolgáltatás protokolljával, az SMB-vel ismerkedünk meg.

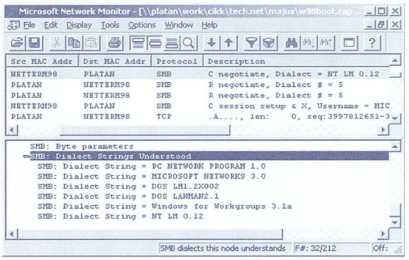
Server Message Block

A Windowsos fájlátvitel viszonylag magasszintű hálózati művelet, melynek során rendkívül sok finom részletet beszél meg egymással a munkaállomás és a kiszolgáló. A két gép között kiépülő TCP csatorna fölött NetBIOS réteg feszül, s e felett dolgozik a fájl- és nyomtatószolgáltatások alapvető protokollja, az SMB *(Server Message Block)*. Az SMB egyelőre nincs rajta a Microsoft halállistáján *(a MAPI, a NetBIOS és sok egyéb egyedi technológia mellé pedig odakívánkozna)*, de előbb-utóbb meghal. Inkább utóbb: a Whistler is SMB alapú. Amíg a WebDAV nem lesz képes ellátni az SMB által nyújtott összes szolgáltatást, az SMB marad.

A kapcsolatfelvétel során a részletek „megbeszélését" Negotiationnak, egyeztetésnek hívjuk. Többfajta egyezkedés zajlik a két gép között:

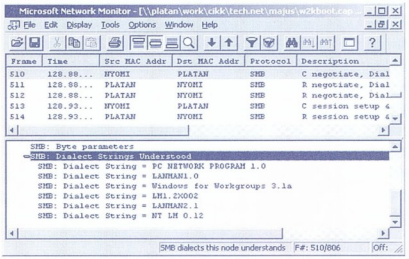
- Milyen authentikációs protokollban állapodjanak meg? A csoportos házirend függvényében ez a Clear Texttől *(hogy őslény Lan Manager 0.0 kiszolgálókkal is együtt tudjunk működni)* NTLM-en át *(hajrá LOPHTCrack!)* a Kerberosig terjedhet. Ezekkel most nem foglalkozunk.
- Melyik verziójú SMB-t beszéli a két fél? Ennek „letárgyalása" azért fontos, hogy kiderüljön, hogyan kell felépíteni az átküldött parancsot, s milyen információkat támogat a két fél a fájlok tartalmával, valamint a fájlnevekkel kapcsolatban? Csak a 8.3-as nevek mehetnek? Vagy hosszú fájlnevek is? Esetleg Unicode?
- NTFS spécik: Az NTFS-en a fájlok egyedi attribútumokkal rendelkezhetnek. Át lehet-e vinni ezeket? A tömörített fájlok tömörítve kerüljenek a hálóra, vagy ki kell előbb bontani, mert a túloldal nem ismeri a tömörítési algoritmust? És az EFS-sel titkosított fájlok vajon titkosítva menjenek-e át a hálón? Nyilván igen. Vagy mégse...?

Az egyezkedés gyönyörűen látszik a NetMonnal, a 32. csomagban a kapcsolatfelvételt kezdeményező NETTERM98 felsorolja, hogy az SMB mely dialektusait, „tájszólásait" beszéli. Az alábbi ábra arról tanúskodik, hogy eszméletlen régi SMB változatok is a palettán vannak. Ez teszi lehetővé, hogy olyan ócskaságokkal is szóba álljon, mint egy Lan Manager 0.0. A lista elemei felülről lefelé egye „erősebbek", tehát a Windows 98 által „beszélt" legújabb SMB az NT LM 0.12.



A Windows 98 által ismert SMB változatok

Az alábbi ábrán a w2kboot.cap fájl 510. csomagja látható, ahol egy Windows 2000 Advanced Server kezdeményez fájlátvitelt, ennek érdekében felsorolja az általa ismert SMB változatokat:



A Windows 2000 Advanced Server által ismert SMB változatok

Mint azt láthatjuk, a legfejlettebb SMB dialektus itt is az NT LM 0.12, ugyanaz, mint a Windows 98 esetében! A 33. csomagban a dialektuslistából egyébként PLATAN a legerősebbet, az ötödiket választja *(a sorszámozás nullától indul)*.

Emellett különböző flagbitek szolgálnak az operációs rendszer tudásszintjének pontosabb betájolására. A Windows 98 ezt tudja *(gyakorlatilag semmi különöset, mindenütt nulla áll)*:

```
SMB: Flags Summary = 128 (0x80)
.......0 = Lock & Read and Write & Unlock not
           supported
......0. = Send No Ack not supported
....0... = Using case sensitive pathnames
...0.... = No canonicalized pathnames
..0..... = No Opportunistic lock
.0...... = No Change Notify
SMB: flags2 Summary = 0 (0x0)
..............0 = Understands only
```

```
                    DOS 8.3 filenames
.............0. = Does not understand
                    extended attributes
...0.......... = No DFS namespace
..0........... = No paging of IO
.0............ = Using SMB status codes
0............. = Using ASCII strings
```

A nyomozás jelenlegi állása szerint nem tudnám megmondani, hogy a Windows 98 miért tagadta le a hosszú fájlneveket... Talán nem lenne minden tanulság nélküli a Windows 98 fenti bitgyűjteményének összevetése a Windows 2000 hasonló listájával. Íme a különbségek (ahol egyes állt a Windows 2000-nél):

```
SMB: Flags Summary = 24 (0x18)
....1... = Using caseless pathnames
...1.... = Canonicalized pathnames
SMB: flags2 Summary = 51283 (0xC853)
...............1 = Understands long filenames
.............1. = Understands extended attributes
.1.............. = Using NT status codes
1............... = Using UNICODE strings
```

Case-insensitive fájlnevek *(kis- és nagybetű különbözik)* és Extended NTFS attribútumok kezelése. Ennyi a különbség, se több, se kevesebb. Ebből az következik, hogy:

Az SMB protokoll mind a mai napig nem támogatja az NTFS speciális fájltárolását (az attribútumokat igen!).
1. Az NTFS tömörített fájljai kitömörítve utaznak a hálón még akkor is, ha mindkét fél Windows NT/2000.
2. Az EFS-sel titkosított fájlok kibontva, titkosítás nélkül utaznak a hálón, mivel az SMB nem tudja lekezelni a titkosítást.

Természetesen állításaim igazságáról bárki meggyőződhet saját hálózatán néhány spéci NTFS fájl lemásolásával. Ezzel megtaláltuk az okát annak, miért „írja elő" a marketingbrosúra IPSec alkalmazását még EFS-sel titkosított fájlok esetén is. Ugye megmondtam, hogy a NetMon mindenre jó?

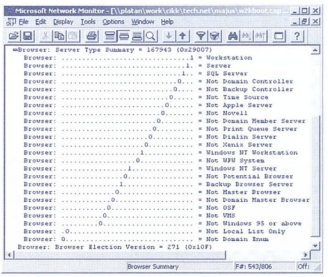
Browse Service

Külön hálózati „szolgáltatás" *(már ha lehet szolgáltatásnak nevezni)* a Browse Service, mely még manapság is alapvető fontosságú a hálózati erőforrások megtalálásában *(Network Neighborhood)*, annak ellenére, hogy számtalanszor bizonyította, nagy hálózatokon képtelen normálisan üzemelni. Hol kikapcsolt gépek virítanak a listájában, hol bekapcsolt gépek megjelenésére várunk - hiába. Ennek a megbízhatatlan működésnek az egyik alapvető oka, hogy amikor tallózunk, sohasem friss adatokat látunk *(akárhogy csapkod is a zseblámpa, NEM azon ügyködik, hogy körbekérdezze a létező gépeket)*, hanem az úgynevezett Master Browser majdnem friss, illetve nagyobb hálózatok esetén a Backup Browserek garantáltan elavult erőforráslistájában kattintgathatunk önfeledten. A másik oka a Browser által használt kapcsolatmentes UDP szolgáltatás, mely olyannyira nem garantálja a lentebb felsorolt csomagok célbajuttatását, hogy még az RFC is említ olyan eseteket, amikor az UDP befuccsol. Például ha a célgép MAC Addresse nincs benn a feladó ARP cachében, akkor az UDP elkiabál a levegőbe. *(Itt jegyzem meg, hogy igazi rendszergazda nem vacakol a Browse szolgáltatással, hanem egyszerűen megkerüli: ha UNC útvonalakkal [\\masina\printer], bemeppelt meghajtóval, vagy parancsikonon kattintva dolgozunk a hálón, és nem érintjük a Network Neigborhood ikont, elsiklunk a Browse mellett...)* A Browse szolgáltatás hálózati forgalmának valamelyes megértése sajnos még napjainkban is szükséges. Milyen csomagokkal találkozhatunk?

- Host Announcement: minden számítógép időről időre közlia Master Browserrel, hogy még mindig életben van. Emellett meglepő részletességgel közli az általa futtatott szolgáltatások listáját is. Pont, mint a WINS. A két szolgáltatás mégis különbözik, és nem is cserél adatot egymással.
- Browser Election: ezt a vicces csomagtípust most sajnos nem sikerült elcsípnem. A lényeg az, hogy minden alhálózaton, azon belül minden protokollon, azon belül minden frame type csoportban saját Master Browsernek kell lennie. Ez tipikusan a legerősebb gép *(nem hardverszempontból, hanem a futtatott operációs rendszer szerint mérjük az erősséget)*, de ha az ki van kapcsolva, akkor egy másik. Az erősebb gépek induláskor választást írnak ki, hogy ki legyen a Master Browser, és a választást azonnyomban meg is nyerik, hisz ők a legerősebbek. Tiszta politika :)
- Browser lekérdezések: a Network Neighborhood kattogtatása váltja ki ezt a műveletet.

Az alábbi táblázatban egy Announcement csomag azon részletét láthatjuk, mely a bejelentett szolgáltatásokat sorolja fel *(w2kboot.cap fájl 369.csomag)*:

Egy Browser Announcement bitjei



Pestiesen szólva: nem semmi. Hackerként megtudhatjuk például – ha eddiga NetBIOS cache-ből nem listáztuk ki – hogy a gépen nemcsak a Workstation és a Server service fut, hanem van rajta egy SQL Server is. Külön érdekesség, hogy ez a kutyát sem érdekli. *(Helyesbítek: az SQL Server Enterprise Managere ez alapján listázza ki a felvehető SQL Servereket.)* Viszont az is feketén-fehéren kiderül, hogy ez a Windows 2000 nem Novell Netware, nincs rajta Time Service *(de van, csak nem master)*, Browse szerepe pedig Backup Browser. És nem Windows for Workgroups. Hanem NT :-O

Fóti Marcell
marcellf@netacademia.net

A cikkben szereplő URL-ek:
[1] http://technet.netacademia.net/feladatok/netmon

A csoportos házirend

Amikor nagyméretű hálózatokkal dolgozunk, különösen fontos, hogy a felhasználók és számítógépek beállításait központilag tudjuk kezelni. E feladatra a Windows NT 4.0 operációs rendszer esetében a System Policy áll rendelkezésünkre. Az eszköz nagyon jól használható a regisztrációs adatbázis azon beállításainak megváltoztatására, amelyek számítógépre *(HKEY_LOCAL_MACHINE hive - HKLM)*, illetve a felhasználóra *(HKEY_CURRENT_USER hive - HKCU)* vonatkoznak. Az imént említett hive-ok alatt bármely bejegyzést felül tudunk írni, ha a regisztrációs adatbázis kulcsain beállított jogosultság ezt lehetővé teszi. De óvatosan kell eljárnunk, mert balszerencsés esetben – és az előzetes gondos tesztelés elmulasztásával – komoly fennakadások idézhetők elő a számítástechnikai rendszer működésében.

A System Policy révén a felhasználói felületen számos korlátozást tudunk érvényre juttatni. Ezek a tiltások azonban csak elrejtenek bizonyos funkciókat a felhasználók elől, nem tekinthetők valós biztonsági beállításoknak. Jó példa erre a házirenden keresztül megvalósítható korlátozás, amely megakadályozza a Registry Editor elindítását. Ez azonban nem jelent elegendő védelmet, hiszen a tiltás ellenére a regisztrációs adatbázis könnyen módosítható a megfelelő fájl *(.reg)* lefuttatásával. A System Policyval főként olyan korlátozások valósíthatók meg a leghatékonyabban, amelyekkel lényegesen csökkenthetjük a felhasználói hibákból fakadó működési zavarok kockázatát.

Ezzel szemben a Windows 2000 operációs rendszerben lévő csoportos házirend *(Group Policy)* segítségével a rendszergazdai feladatok sokkal szélesebb körét tudjuk központilag ellátni. A csoportos házirend esetében ugyanis a regisztrációs adatbázis bejegyzéseinek megváltoztatása csak egy funkció a sok közül. Ezen kívül már valós biztonsági beállításokat is megvalósíthatunk. Hiszen nem csak a fájlrendszerre, de a regisztrációs adatbázisra vonatkozó jogosultságot is kiosztathatunk. Mindezek mellett logon/logoff, startup/shutdown scripteket is kezelhetünk, sőt programtelepítésre is lehetőségünk van. A csoportos házirend - a System Policyhoz hasonlóan - ugyan lehetőséget ad arra, hogy a HKLM és a HKCU hive-ok alatt bárhol módosítsuk a bejegyzéseket, de alapértelmezésben nem támogatja ezt. A Windows 2000 regisztrációs adatbázisában előre meghatározott helyeken találhatóak a házirenddel kapcsolatos bejegyzések. Mind a felhasználóra, mind a számítógépre vonatkozó beállítások a megfelelő hive-hoz tartozó

```
Software\Policies
```

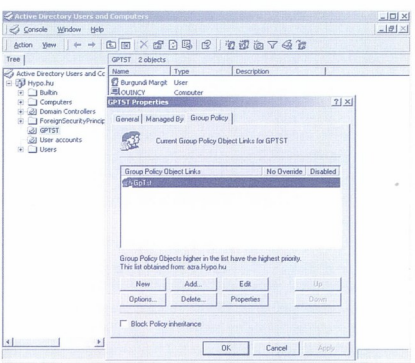
illetve a

```
Software\Microsoft\Windows\CurrentVersion\Policies
```

kulcsok alatt jelennek meg.

A csoportos házirend működése

A csoportos házirendet csak tiszta Windows 2000-es környezetben használhatjuk. A Group Policy ugyanis szorosan kötődik az Active Directoryhoz *(AD)*. A házirendbeállítások az AD felépítésének megfelelően objektumokban tárolódnak *(Group Policy Object – GPO)*, amelyeket az AD fő egységeihez *(telephely, tartomány, szervezeti egység)* tudunk hozzárendelni. Házirendobjektumot a Group Policy snap-in segítségével hozhatunk létre, amelyet önálló eszközként is használhatunk, de elérhetjük az Active Directory Users and Computers, valamint az Active Directory Site and Services kiterjesztéseként is. Ez utóbbi megoldás sokkal kényelmesebb, hiszen a GPO-kat, mint az imént már volt róla szó, valamelyik AD egységhez kell hozzárendelnünk. Természetesen az Active Directory Sites and Services használatakor csak telephelyhez tudunk GPO-t kapcsolni. Ha tartomány vagy szervezeti egység szintjén akarunk házirendet érvényesíteni, akkor az Active Directory Users and Computerst kell igénybe vennünk. Abban az esetben például, ha egy szervezeti egységhez szeretnénk kapcsolni egy házirendobjektumot, akkor a szervezeti egység tulajdonságait megjelenítő párbeszédablakban *(Action menü Properties parancs)* válasszuk ki a Group Policy fület.



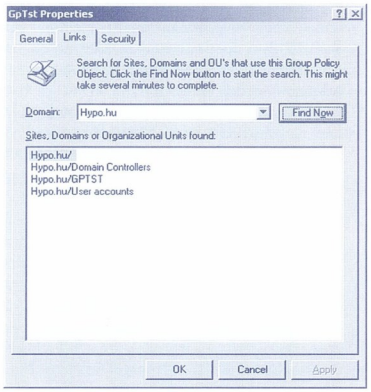
GPO létrehozása, törlése, prioritása, láncolása, öröklődés beállítása – minden egy helyen!

A képen látható felületen tudunk új GPO-t létrehozni az adott szervezeti egységen *(New nyomógomb)*. Ezen túlmenően már létező GPO-kat is hozzákapcsolhatunk *(Add nyomógomb)*. Természetesen lehetőségünk van a házirendobjektumok beállításainak módosítására is *(Edit nyomógomb)*. Vegyük figyelembe, hogy ha egy GPO-t több szervezeti egységhez is hozzákapcsolunk, akkor bármelyik helyen módosítjuk is a beállításokat, a változások minden olyan szervezeti egységnél éreztetni fogják a hatásukat, amelyhez a házirendobjektumot tár-

sítottuk. Ha több objektumot rendeltünk valamelyik szervezeti egységhez, akkor az Up és a Down gombbal tudjuk a sorrendjüket meghatározni. Ennek abban az esetben van jelentősége, ha a GPO-k egymást átfedő beállításokkal is rendelkeznek. Ilyen esetben a sorrendben előrébb lévő objektum beállításai érvényesülnek.

Az Options gomb megnyomásakor feltűnő ablakban egyrészt letilthatjuk a házirendobjektumot az adott tárolón *(Disabled jelölőnégyzet)*, másrészt megakadályozhatjuk a házirend felülbírálását *(No Override jelölőnégyzet)*. Ez utóbbi esetben az alsóbb szintű tárolókon érvényes házirend nem írja felül a szóban forgó GPO beállításait. Például ha tartományi szinten érvényes GPO-nál engedélyezzük a No Override jelölőnégyzetet, akkor az objektum által képviselt házirend az egész tartományban érvényes lesz, függetlenül attól, hogy a szervezeti egységeknél mit állítottunk be. Így lényegében az alapértelmezett öröklődési szabályokat változtatjuk meg. Ezen túlmenően módunk van arra is, hogy az öröklődést az AD-hierarchia bármely szintjén letiltsuk. Ezt az Options gomb alatt található Block Inheritance jelölőnégyzettel tudjuk megvalósítani. Érdemes azonban óvatosan bánni az öröklődési szabályok megváltoztatásának lehetőségével, mert nagyon megnehezítheti a hibaelhárítást.

Korábban már esett róla szó: egy GPO-t több tárolóhoz is hozzákapcsolhatunk. A házirend szerkesztésekor azonban nagyon fontos tudnunk, hogy a változtatások mely területeken érezhetik majd hatásukat. Szerencsére erről nem kell külön nyilvántartást vezetnünk, hiszen a házirendobjektum tulajdonságait megjelenítő párbeszédablakon *(Properties nyomógomb)* hozzáférhetők ezek az információk. A Links fülön található Find Now gombbal ugyanis megkereshetők azok az AD tárolók, amelyek a szóban forgó házirendobjektumot használják.



Egy GPO egyidejűleg sok helyen felhasználható

A Links mellett a Security fülről is érdemes szót ejteni. Itt lehet a GPO-hoz hozzáférési jogosultságot beállítani. A jogosultsági listában Apply Group Policy néven egy speciális jogosultság is fellelhető, amellyel az Authenticated Users csoport révén alapértelmezésben minden felhasználó rendelkezik. Ezzel a joggal szabályozhatjuk, hogy a beállítások mely felhasználókra, illetve felhasználói csoportokra legyenek érvényesek. Bár a szervezeti egység a legkisebb olyan objektum, amelyhez GPO-t rendelhetünk, a jogosultsági listán keresztül a csoportok szintjén is szabályozhatjuk a házirend végrehajtását. Ennek a megoldásnak előnye és hátránya is van. Előnye: sokkal differenciáltabban lehet meghatározni, hogy mely felhasználókra vonatkozzék az aktuális házirend. Hátránya: nagyon megnehezíti annak megállapítását, hogy melyik beállítás miért, vagy éppen miért nem jelenik meg az egyes felhasználóknál. Ha valamelyik csoporttól elvonjuk az Apply Group Policy jogot, akkor érdemes a Read jogot is letiltani, elsősorban a teljesítményt növelendő. Ellenkező esetben az operációs rendszer sorra veszi a GPO beállításait, bár nem érvényesíti őket.

Előfordulhat, hogy egy házirendobjektum csak felhasználóra vagy csak számítógépre vonatkozó beállításokat tartalmaz. Ilyenkor érdemes letiltani a házirend nem használt részét, mivel ezzel is csökkenthetjük a végrehajtás idejét. A letiltást a General fül Disable Computer Configuration settings, illetve Disable User Configuration settings elnevezésű jelölőnégyzettel tudjuk végrehajtani.

A Group Policy objektumokkal kapcsolatos információk tárolása

A csoportos házirendre vonatkozó információk tárolására a Group Policy Container és a Group Policy Template szolgál. Az előbbi az AD azon tárolója, amely a házirendobjektumok tulajdonságait *(verziószám, státusz stb.)* tartalmazza. Az utóbbi a fájlrendszerén lévő könyvtárstruktúrának az elnevezése, amelyben az Administrative Template-en alapuló házirendbeállítások, script fájlok *(logon/logoff, startup/shutdown)* stb. találhatók. A Group Policy Template mappaneve minden esetben megegyezik a GPO egyedi azonosítójával *(GUID)*, és

```
%SYSTEMROOT%\Sysvol\<SYSVOL_MEGOSZTAS>\<Tartomány-
név>\Policies\
```

elérési út alatt található a tartományvezérlőkön. A Sysvol megosztás tartalma a File Replication Service jóvoltából automatikusan replikálódik, így a módosítások mindegyik tartományvezérlőhöz eljutnak. *(A GPO-k tartományok között nem replikálódnak.)*

A Group Policy Template az alábbi alkönyvtárakat tartalmazhatja:

- Adm: A házirendobjektum beállításaihoz szükséges sablonfájlok *(.adm)* helye.
- Machine: Ebben a mappában a Registry.pol fájl található. Az állomány a regisztrációs adatbázis HKLM hive alatt lévő bejegyzéseit tartalmazza, amelyeket a házirenden keresztül módosítottunk *(Computer Configuration\ Administrative Templates)*.
- Machine\Applications: Tartalma attól függően változik, hogy mely alkalmazások telepítését rendeltük hozzá a számítógépekhez a csoportos házirend segítségével.
- Machine\Microsoft\WindowsNT\Secedit: Az itt fellelhető GptTmpl.inf fájl a tartományvezérlőn érvényes biztonsági beállításokat tartalmazza. Abban az esetben, ha a tartományvezérlőhöz kapcsolt GPO

```
Computer Configuration\Windows Settings\
Security Settings
```

beállításait módosítjuk, akkor GptTmpl.inf tartalma is módosul.

- Machine\Scripts\Shutdown: A számítógép leállításakor elinduló scriptek mappája.
- Machine\Scripts\Startup: A számítógép elindulásakor le futó scripteket kell ebben a könyvtárban elhelyeznünk.
- User: Ebben a mappában a Registry.pol fájl lelhető fel, amely a regisztrációs adatbázis HKCU hive alatt lévő, a házirenden keresztül módosított bejegyzéseit tartalmazza ***(User Configuration\ Administrative Templates)***.
- User\Applications: A felhasználók számára meghirdetett programtelepítés esetén ebben a könyvtárban létrejön egy .aas kiterjesztésű állomány, amelyet a Windows Installer használ.
- User\Documents & Settings: A speciális mappák ***(My Documents, My Pictures stb.)*** átirányításával kapcsolatos információkat hordozó Fdeploy.ini elnevezésű fájl kerül ebbe a mappába.
- User\Microsoft\IEAK: A GPO

```
User Configuration\Windows Settings\
Internet Explorer Maintenance
```

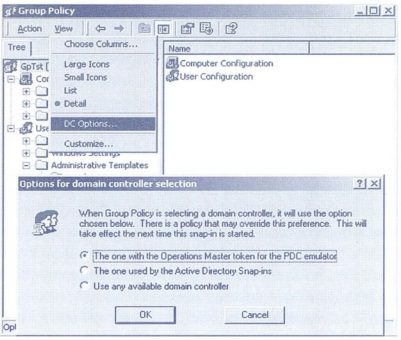
beállításaival kapcsolatos információk lelőhelye.

- User\Scripts\Logoff: A felhasználó kijelentkezésekor lefutó scripteket tárolja.
- User\Scripts\Logon: A felhasználó bejelentkezésekor lefutó scriptek mappája.
- User\Microsoft\RemoteInstall: A Windows 2000 szerver lehetőséget biztosít a rendszergazdák számára, hogy távoli gépekre Windows 2000 operációs rendszert telepítsenek a Remote Installation Services révén. Ezzel kapcsolatos beállításokat a RemoteInstall mappában levő OSCfilter.ini fájl tárolja.

A Group Policy Template mappában a gpt.ini nevű fájl is megtalálható. Ebben az állományban tárolódik a GPO verziószáma a következő formában: Version=<verziószám>. Egy nyolc számjegyes DWORD értékről van szó. Az első négy számjegy a felhasználóval, az utolsó négy a számítógéppel kapcsolatos beállítások verziószámát jelenti. Például a 0X00080006 arra utal, hogy a GPO User Settings részében nyolc, a Computer Settingsben pedig hat módosítás történt. Ez a szám decimális formában tárolódik. Ha az iménti példát vesszük alapul, akkor a gpt.ini fájlban a következőképpen jelenik meg a verziószám: Version=05240294. Ennek alapján az esetek többségében megállapítható, hogy melyik tartományvezérlőn legfrissebb a házirend. Előfordulhat azonban, hogy két tartományvezérlőn egyidőben ugyanazt a GPO-t módosítják a rendszergazdák. Ilyen esetben a replikáció során az egyik módosítás felülíródik.

A replikációs konfliktusok előfordulási valószínűségének csökkentése végett a Group Policy snap-in – függetlenül attól, hogy melyik számítógépről indítjuk el – a PDC emulátorral lép kapcsolatba.

Így a házirendobjektumokat mindig ugyanazon a tartományvezérlőn szerkeszthetjük. Ez a működési mód a Group Policy snap-in View menü DC Options parancsával megváltoztatható. *(Ez az utasítás csak akkor áll rendelkezésünkre, ha a Group Policy snap-in felhasználói felületének bal oldalán megjelenő fa gyökerét jelöljük ki.)*



Melyik házirendpéldányt szeretnénk szerkeszteni?

Ha a DC Options parancsra klikkelünk, az Options for domain controller selection elnevezésű párbeszédablak jelenik meg, amelyen háromféle választási lehetőségünk van:

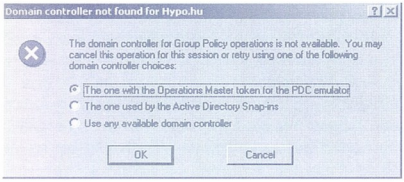
- The one with the Operations Master token for the PDC emulator: Ez az alapértelmezett beállítás. Számottevő előnye, hogy elkerülhetők a replikációs konfliktusból eredő problémák, hiszen a házirendobjektumokat mindig ugyanazon a tartományvezérlőn tudjuk szerkeszteni. Ebből következően ez az ajánlott beállítás, amelytől csak speciális esetben érdemes eltérni. Például akkor, ha a lassú hálózati kapcsolat miatt a PDC emulátor nehezen érhető el.
- The one used by the Active Directory Snap-ins: Azon a tartományvezérlőn fogjuk a GPO-kat szerkeszteni, amelyikkel az éppen használt Active Directory snap-in *(például Users and Computers)* kapcsolatban áll. Minden AD snap-in esetében meghatározhatjuk, hogy melyik tartományvezérlőt részesítse előnyben. A Users and Computers esetében például az Action menüben található Connect to Domain Controller paranccsal tudjuk ezt végrehajtani.
- Use any available domain controller: Lehetőséget ad a Group Policy snap-in számára, hogy bármely rendelkezésre álló tartományvezérlőt igénybe vegye a házirend szerkesztésekor.

A tartományvezérlő kiválasztásának imént ismertetett módjait a csoportos házirenden keresztül is be tudjuk állítani:

```
User Configuration\Administrative Templates\
System\Group Policy\Group Policy domain
controller selection
```

Ilyen esetben az operációs rendszer figyelmen kívül hagyja a Group Policy snap-in erre vonatkozó beállítását. Előfordulhat, hogy az előnyben részesített tartományvezérlő nem érhető el, ilyenkor hibaüzenetet kapunk. Ha a tartományvezérlő kiválasztását csak a Group Policy snap-inben határoztuk meg, ak-

kor az alábbi párbeszédablak jelenik meg, ahol lehetőségünk van megváltoztatni a korábbi beállítást.

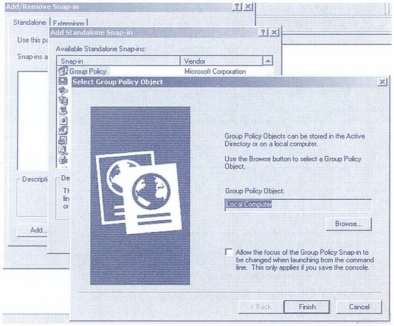


Ha a PDC emulátor nem érhető el, más GPO példányt is módosíthatunk

Ezzel szemben, ha a kiválasztással kapcsolatban házirend van érvényben, akkor a következő figyelmeztető üzenet tűnik fel: „Failed to find a domain controller. There may be a policy that prevents you from selecting another domain controller." Ebben az esetben nincs lehetőségünk módosításra.

Lokális csoportos házirend

Minden Windows 2000 Professional munkaállomáson lehetőségünk van beállítani helyi házirendet. A lokális Group Policy objektum *(LGPO)* a %SystemRoot%System32\GroupPolicy mappában helyezkedik el. Az ilyen típusú házirendobjektumnak értelemszerűen nincs Active Directorytól függő komponense.
Az LGPO jogosultsági listájában nincs Apply Group Policy jog, ezért ha az Administrators csoportot mentesíteni akarjuk a helyi házirend hatásától, akkor a Read jogot meg kell vonnunk tőle. Ez viszont azzal a hátránnyal jár, hogy olvasási jog nélkül a rendszergazdáknak nem lesz módjuk az LGPO tartalmát megtekinteni.
A helyi csoportos házirendet is a Microsoft Management Console segítségével tudjuk szerkeszteni. Az MMC elindítása után *(mmc.exe)* a Console menü Add/Remove snap-in parancsára kattintva nyílik lehetőségünk a Group Policy snap-int kiválasztására *(Standalone fül, Add nyomógomb)*. Ezt követően az MMC alapértelmezésben a lokális házirend szerkesztésének lehetőségét kínálja fel.



A helyi házirend módosítása MMC-vel

Ha a Finish nyomógombra kattintunk, akkor az aktuális munkaállomás házirendjét tudjuk szerkeszteni. Amennyiben egy távoli számítógép házirendjét szeretnénk módosítani, akkor a Browse gomb megnyomása után a Computers fülön választhatjuk ki a megfelelő munkaállomás nevét.

A házirend végrehajtásának folyamata

A számítógéphez rendelt házirend a gép elindítása, a felhasználói beállítások pedig a bejelentkezés során érvényesülnek. Először mindig a LGPO, ezt követően a telephelyhez, majd a tartományhoz, végül a szervezeti egységekhez tartozó GPO-k végrehajtására kerül sor. Az alapértelmezett végrehajtási sorrendet a korábban már említett Block Inheritance és a No Override parancsokkal megváltoztathatjuk. *(Ezeket az utasításokat a LGPO esetében nem használhatjuk.)*
Alapértelmezésben a beállítások végrehajtása szinkronizáltan történik. Ez a gyakorlatban azt jelenti, hogy a számítógépfüggő házirend a bejelentkezési ablak megjelenése, a felhasználói pedig a felhasználói felület betöltődése előtt jut érvényre. Ezt a működési módot a

```
Computer Configuration\Administrative Templates\
System\Group Policy
```

alatt megváltoztathatjuk. A változtatás azonban nem ajánlott, mivel számos probléma forrása lehet.
Újítás a Windows NT 4.0 System Policyjához képest, hogy a csoportos házirend beállításai bizonyos időközönként automatikusan frissülnek. Alapértelmezésben felhasználók és munkaállomások esetén ez az időtartam 90 perc, legfeljebb 30 perces eltolódással, a tartományvezérlőknél pedig 5 perc. A Computer, illetve a User Configuration

```
\Administrative Templates\
System\Group Policy
```

alatt lehetőségünk van ennek az értéknek a megváltoztatására. Óvakodjunk azonban attól, hogy ezt az értéket túl alacsonyra vegyük, hiszen ez növeli a hálózati forgalmat, valamint lelassíthatja a munkaállomás működését. Parancssorból is utasítást adhatunk a házirend frissítésére a

```
secedit.exe /refreshpolicy MACHINE_POLICY
[/enforce]
```

illetve a

```
secedit.exe /refreshpolicy USER_POLICY [/enforce]
```

utasítás révén. Az enforce kapcsolónak csak a Security, illetve az Encripted File System kiterjesztéseknél van hatása: a beállítások akkor is frissülnek, ha a házirendben nem történt változás.
Normális körülmények között a felhasználót követik a beállításai, azaz bármelyik munkaállomáson is jelentkezik be, ugyanaz a felhasználói házirend lesz érvényes rá. A csoportos házirend azonban lehetőséget kínál arra, hogy a felhasználói beállítások különbözzenek, attól függően, hogy melyik munkaállomáson jelentkezik be. Ezt nevezzük visszacsatolás *(loopback)* funkciónak, amelynek két fajtája van:

- Merge mode: A hagyományos működésnek megfelelően először a felhasználóra, majd a munkaállomásra vonatkozó beállítások hajtódnak végre. Ebből következően, ha átfedés van a kétféle házirend között, akkor a számítógéphez rendelt élvez elsőbbséget.
- Replace mode: Ebben az esetben csak a számítógépes objektumhoz tartozó házirend lesz érvényes a felhasználóra.

A visszacsatolás üzemmódot a

```
Computer Configuration\Administrative Template\
System\Group Policy\
```

alatt található „User Group Policy loopback processing mode" beállítással tudjuk engedélyezni.

A csoportos házirend működésének áttekintése után nézzünk meg egy-két újdonságot közelebbről is.

Programtelepítés Group Policyval

Ez a funkció nagyon megkönnyítheti a rendszergazdák munkáját, hiszen nem szükséges sok pénzért külön terméket vásárolni ahhoz, hogy a sok felhasználót érintő telepítéseket központilag el tudjuk végezni.

1) A programtelepítés akkor a leghatékonyabb, ha a Windows Installer Service-en keresztül történik. Ennek előfeltétele, hogy rendelkezzünk a telepítést elősegítő úgynevezett Windows Installer csomaggal *(.msi fájl)*. Abban az esetben, ha a program telepítőfájljai között nem találunk ilyen állományt, akkor két lehetőségünk van. Telepíthetjük a szoftvert a Windows Installer használata nélkül is. Ekkor azonban több hasznos szolgáltatásról le kell mondanunk. Például a szoftvert vagy a program egyik komponensét nem telepíthetjük az első használatkor. Ráadásul el kell készítenünk egy .zap kiterjesztésű text állományt, amelyben kódoljuk a telepítés főbb elemeit *(a setup.exe elérési útját, esetleges paramétereit stb.)*.
2) Ha nem ezt az utat választjuk, akkor el kell készítenünk a szoftverhez a megfelelő Windows Installer csomagot. Ezt a feladatot többféle alkalmazással is elvégezhetjük. Legkézenfekvőbb talán a Windows 2000 Server CD-n is megtalálható VERITAS WinInstall LE nevű programját igénybe venni. Használatához szükségünk lesz egy olyan számítógépre, amelyiken az operációs rendszeren és a megfelelő szervizcsomagon kívül mást nem telepítettünk. Erre a gépre kell majd felraknunk azt az alkalmazást, amit majd szeretnénk elterjeszteni a vállalatunknál. A szóban forgó alkalmazás telepítése előtt és után le kell futtatnunk a WinInstall Discover nevű segédprogramját, amely rögzíti a fájlrendszeren és a regisztrációs adatbázisban bekövetkező változásokat.

A WinInstall segítségével nemcsak új csomagokat hozhatunk létre, hanem a már meglévőket módosíthatjuk is. *(A VERITAS WinInstall LE leírása megtalálható a Microsoft honlapján* ***[1]****)*.

A programtelepítés fő lépései

Először is a telepítendő állományokat, illetve a Windows Installer csomagot másoljuk fel a kiszolgáló mindenki által elérhető *(megosztott)* mappájába *(Software Distribution Point)*. Ezután válasszuk a céljainknak legmegfelelőbb telepítési módot:

- Publish to Users: Lehetőségünk van meghirdetni a telepítést a felhasználók számára. Ebben az esetben a felhasználónak kell gondoskodnia a telepítés elindításáról a Control Panel => Add/Remove Programs => Add New Program segítségével. Ha olyan házirendobjektumnál engedélyezzük a szoftvertelepítést, amely már érvényben van a felhasználóknál, akkor nem kell a felhasználónak újra bejelentkeznie a telepítés indításához. Fontos tudni, hogy a felhasználók jogosultak a program eltávolítására is.
- Assign to Users: Az előző módszerhez képest, további szolgáltatás is rendelkezésünkre áll: a telepítés a program első használatakor is elindítható. A telepítendő program ikonja ugyanis a felhasználó bejelentkezését követően megjelenik a Start menüben.
- Assign to Computers: A telepítés a számítógép újraindításához kötött, és a felhasználó beavatkozása nélkül megy végbe. Ilyenkor csak a lokális rendszergazdai jogosultsággal rendelkezők tudják a programot eltávolítani.

A telepítési módokat a Group Policy snap-in Computer vagy User Configuration\Software Settings mappa alatt tudjuk kiválasztani.

Folytatjuk...

Tomasz Balázs
balazs.tomasz@hu.hypovereinsbank.com
pszichológus

A cikkben szereplő URL-ek

[1] http://www.microsoft.com/windows2000/library/ planning/management/veritas.asp

ISA Server – bemelegítés

Ez a cikk az első része annak a cikksorozatnak, melyben azt szeretném bemutatni, hogy hogyan is lehet az ISA Server felhasználásával biztonságos rendszert felépíteni. Mivel több hónapos cikksorozatról van szó ejtsünk néhány szót a tervezett témákról:

Alapvető tűzfalfunkciók megvalósítása:
- SecureNAT
- Server Publishing
- Szűrők, tiltólisták

A gyorsítótár:
- Láncok és tömbök

DMZ kialakítása:
- DMZ kialakításakor mit vegyünk figyelembe
- „két láb, vagy több láb"
- mail relay
- külső vagy belső webkiszolgálót telepítsünk

Virtuális magánhálózatok:
- ISA-ISA VPN
- Ügyfél-ISA VPN

Felhasználók azonosítása és hitelesítése
- tartomány
- LDAP
- Kerberos
- RADIUS

Együttműködés más gyártók termékeivel

Jelen cikkem tulajdonképpen bevezető, mely a tűzfalakkal és az Internetes biztonság általános kérdéseivel foglalkozik. Természetesen a témákkal kapcsolatos észrevételeket és javaslatokat szívesen fogadom *(ezek közlésének pedig legegyszerűbb módja a NetAcademia Tech.Net levelezési listája)*.

Sokakban felmerülhet a kérdés, hogy tulajdonképpen mi is a tűzfal. Jó kérdés! Mielőtt rátérnénk a tűzfalakra, tekintsük át egy kicsit *(tényleg csak egy kicsit)* az OSI rétegeket *(layer)*, és egy IP csomag szerkezetét. Hogy miért pont az IP csomagét? Egész egyszerűen azért, mert az Interneten folytatott kommunikáció legnagyobb része ilyen csomagok küldözgetéséből áll.

7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

Az OSI rétegek

Forrás MAC cím	Cél MAC cím	Forrás IP cím	Cél IP cím	Forrás port	Cél port	ADAT	CRC (ellenőrző összeg)	CRC (ellenőrző összeg)	CRC (ellenőrző összeg)

Egy IP csomag (némiképp leegyszerűsített) szerkezete

Amikor két számítógép IP kapcsolatot hoz létre, bizonyos információk beépülnek a hálózaton átmenő csomagokba. Ha a csomagokat egy erre szolgáló eszköz megállítja, megvizsgálja a benne levő adatokat, és ezek alapján eldönti, hogy mit kezdjen az adott csomaggal *(átengedi vagy eldobja)*, akkor ez az eszköz egy tűzfal.

Definíció: a tűzfal egy olyan hálózatok közti átjáró, amely rendelkezik egy szabályrendszerrel, és ezt a szabályrendszert kikényszeríti a rajta átmenő forgalmakon, ezzel megakadályozva a rendszerekhez való illetéktelen hozzáférést.

A fentiekből is látszik, hogy a tűzfal nem csodaszer. Nem képes például:
- A belső hálózaton található fájlkiszolgálónkat megvédeni a rosszindulatú belső felhasználóktól *(a rosszindulatú külső felhasználóktól viszont igen)*
- Valódi vírusszűrést végezni
- Önmagát beállítani, biztonságosabbá tenni *(ezért kell behatolás-érzékelő (IDS) eszközzel kiegészíteni)*
- Megvédeni a belső hálózatról modemező felhasználót *(hiszen ez a kapcsolat nem megy át a tűzfalon)*
- Kiégetni a rosszindulatért felelős idegsejteket a „hekker" agyából :-)

Viszont egy jó tűzfalszoftvernek mindenképpen képesnek kell lenni az alábbiakra:
- A rajta átmenő ÖSSZES forgalom ellenőrzése
- Felhasználók hitelesítése
- Tevékenységének ALAPOS naplózása *(és nem hátrány az sem, ha ezek az eseménynaplók könnyen elemezhetők)*
- A fenti tevékenységeket pedig a szabályrendszer alapján kell végeznie
- Nem nélkülözhetetlen, de egyre inkább előtérbe kerül az is, hogy a tűzfalnak képesnek kell lennie titkosított adatkapcsolatok kiépítésére, és azok fogadására.

Szóljunk néhány szót a tűzfalak előnyeiről is:
- Magasszintű biztonság érhető el használatukkal.
- Meglehetősen költséghatékony megoldást jelentenek, hiszen egyetlen tűzfallal akár több ezer gépet is megvédhetünk.
- Mivel a hálózat határán helyezzük el a tűzfalat, a belső hálózaton viszonylagos biztonságban érezhetjük magunkat.

Nyilvánvaló, hogy *(mint mindenhol)* a tűzfalak esetében is a biztonság és a kényelmes felhasználás ellentétesek egymással. Néhány alapelv, aminek alkalmazásával tűzfalunk biztonságosabbá tehető:

- Minél kevesebb dolgot engedélyezünk tűzfalunkon, annál kisebb annak az esélye, hogy biztonsági rést hagyunk rajta.
- A szabályrendszer legyen minél tömörebb, és *(a lehetőségekhez képest)* jól áttekinthető.
- Kissé elcsépelt már, de nem lehet elégszer mondani *(és a tűzfalnál különösen igaz)*: csak a feltétlenül szükséges szolgáltatások működjenek a tűzfalgépen. A tűzfal az tűzfal. Nem webkiszolgáló, nem fájlkiszolgáló, nem bármi más. CSAK tűzfal.

Most már sok mindent tudunk a tűzfalak alapelveiről, lássuk, hogy milyen konkrét megvalósítások léteznek!

A tűzfalaknak több típusa létezik, manapság már egyre ritkább az, hogy egy adott tűzfalszoftver tisztán az alábbi kategóriák valamelyikébe besorolható legyen:

- Csomagszűrők
- Állapottáblás csomagszűrő
- Proxy
- Címfordítók

Ezek a főbb műszaki megvalósítások, alapelv szerint pedig kétféle tűzfalat különböztetünk meg:

- Megengedő: ez gyakorlatilag azt jelenti, hogy ha a szabályrendszerében nincs az adott kapcsolatra megfelelő definíció, akkor azt átengedi magán, vagyis: „Mindent szabad, amit nem tilos."
- Tiltó: magától értetődően ez az előző ellentéte, tehát: „Mindent tilos, amit nem szabad." Ez az elterjedtebb alapfilozófia.

Csomagszűrő tűzfalak

Ezek a tűzfalak a csomagokat önálló egységként kezelik. Tulajdonképpen a szűrés megvalósításához ötféle információ áll rendelkezésre:

- Forrás IP cím
- Cél IP cím
- Forrás port
- Cél port
- A szolgáltatás típusa *(például TCP, UDP)*

Előnyeik:

- Viszonylag egyszerű eszköz, gyakorlatilag bármely útvonalválasztó *(router)* eszközzel megvalósítható
- A felhasználó számára észrevétlen *(persze csak addig, amíg nem ütközik valamilyen tiltó szabályba)*
- Gyors, mert a szűrés egyszerűsége miatt kicsi a teljesítményigény

Hátrányaik:

- Jól beállítani igen nehézkes *(egy szabályt körülbelül úgy kell elképzelni, hogy megadjuk a fenti öt információt minden szűrni kívánt forgalomhoz, és ezek alapján megmondjuk, hogy engedélyezzük-e az adott forgalmat, vagy nem)*, és a beállítások helyességéről nehéz megbizonyosodni.
- Mivel a cél és a forrás azonosítása az IP cím *(esetleg a fizikai cím)* alapján történik, a többi tűzfalhoz képest könnyebb kijátszani
- A csomagszűrők mai megjelenési formái jellemzően az útvonalválasztó eszközök. Ezek viszont nem arról híresek, hogy szinte korlátlan tárolókapacitással rendelkeznek, ezért a képződő eseménynaplókat el kell juttatni valamilyen másik eszközhöz. Ez a naplózást, és a naplók elemzését bonyolultabbá teszi.

Állapottáblás csomagszűrők *(stateful packet filter)*

Ez a tűzfaltípus tulajdonképpen a csomagszűrők továbbfejlesztett változata. Annyival tud többet, hogy a felépített kapcsolatokat állapottáblák *(state table)* használatával nyilvántartja. Ez a gyakorlatban azt jelenti, hogy alapállapotban a szabályrendszernek megfelelően az összes portot blokkolja. Amikor valaki kapcsolatot akar nyitni például egy webkiszolgáló felé *(most lényegtelen, hogy kifelé, vagy befelé irányuló kapcsolatról van szó)*, akkor – ha ez engedélyezve van – megnyitja a kapcsolatot, de a kiszolgálótól visszakapott csomagokat csak akkor engedi át, ha erre az állapottáblában megfelelő bejegyzést talál. A kapcsolat lezárása után a port ismét bezárul.

Előnyeik:

- A felhasználók számára észrevétlen
- Nemcsak csomag-, hanem kapcsolatszinten is képes vizsgálni a rajta átmenő forgalmat
- Jó teljesítmény
- Minden protokollt lehet szűrni vele *(hiszen ha ismerjük a protokoll jellemzőit, például azt hogy a kapcsolat kiépülése után a kiszolgáló milyen porton akar az ügyfél felé kapcsolatot nyitni, akkor azt biztosan tudjuk szűrni is)*
- Jól integrálható más megoldásokkal
- Az állapottábla használata
- A felhasználó azonosítása és hitelesítése is elvégezhető
- Könnyen bővíthető *(„csak" definiálni kell az új protokollt, és már működhet is)*
- Általában címfordítást végez *(lásd később)*, ezzel elrejti a belső hálózat felépítését

Hátrányaik:

- Nem „bontja" a kapcsolatot *(lásd a proxy tűzfalaknál)*

Igazából az egyetlen felemlíthető hátrány egyrészről a biztonság rovására megy, másrészről viszont jelentősen növeli a teljesítményt. Jellemzően ezt a megvalósítást sem használják önmagában, hanem a gyakran használt protokollokra *(főleg HTTP, SMTP, FTP)* proxy-kat alkalmaznak. Már sokat emlegettük a proxy-kat, lássuk hát mik is ezek.

Proxy tűzfalak:

A proxy tűzfalakat szokták application layer gateway-ként *(alkalmazás rétegben működő átjáró)* is emlegetni. Bármilyen meglepő is, ez az elnevezés nem véletlen. Ez a megoldás ugyanis az OSI rétegek közül a legfelsőben, az alkalmazásrétegben végez szűrést. Ez egyben azt is jelenti, hogy a szűrő az adott protokoll MINDEN jellemzőjét ismeri, és ha bármilyen szabálytalanságot észlel, azonnal elkaszálja az adott kapcsolatot. A megvalósítás másik fontos jellemzője az, hogy a proxy tűzfal bontja a kapcsolatot. Lássuk mit is jelent ez:

Maradva kedvenc webkiszolgálós példánknál, a belső hálózaton levő ügyfél meg akar nézni egy külső webkiszolgálón levő weblapot. Elmegy a tűzfalhoz. A tűzfal fogadja a kapcsolatot, és megjegyzi, hogy mit is kért az ügyfél. Ezután az ügyfél nevében eljárva, a saját IP címét használva megnyitja a kapcsolatot a webkiszolgáló felé, és megszerzi a kért adatokat. A kapott tartalmon teljeskörű protokoll-ellenőrzést végez *(mintha ő maga lenne az ügyfél)*, és ha mindent rendben talált, csak akkor továbbítja az ügyfél felé a kért adatokat.

Itt az ideje, hogy eloszlassak egy tévhitet *(tisztelet a kivételnek, vagyis annak, aki nem tévhitt :-))*. Sokan úgy gondolják, hogy a proxy az szükségszerűen gyorsítótár *(cache)* is. Ez ebben a formában nem igaz! A gyorsítótárat azért találták ki, mert a proxy tűzfalak egyik nagy hátránya éppen teljesítményigényük *(bár a mai gépek mellett ez egyre kevésbé jelent problémát)*. Könnyen spórolható meg jókora teljesítmény, ha a már átvizsgált, és még aktuális tartalmakat a tűzfalgép merevlemezén tároljuk, és a későbbi kéréseket innen szolgáljuk ki. Természetesen a gyorsítótár nagyon hasznos, és egyre fejlettebb megoldások állnak rendelkezésünkre, hogy gyorsítótárunkat, és ezzel együtt Internetkapcsolatunk sávszélességét minél jobban kihasználhassuk.

A proxy tűzfalak előnyei:

- A legmagasabb szintű biztonságot ez a megoldás garantálja
- A belső hálózatot elrejti
- A kifelé menő kérések a tűzfal IP címéről érkeznek, így egyetlen támadható IP címet mutatunk magunkból. Ha támadnak, így a tűzfalat támadják.
- Nemcsak magát a kapcsolatot, de annak tartalmát is vizsgálja
- Általában jár hozzá a gyorsítótár

A proxy tűzfalak hátrányai:

- A felhasználók számára nem átlátszó (általában)
- Használatához gyakran speciális ügyfélprogramra van szükség
- Nagy teljesítményigény *(a gyorsítótár használatával ez jelentősen csökkenthető)*
- Minden szolgáltatáshoz külön proxy kell
- Nehezen bővíthető, emiatt lehet, hogy nem tudja a legújabb protokollmegvalósításokat *(lásd SMTP és ESMTOP)*.

A proxy-k kétféle üzemmódban működhetnek, ez az üzemmód tulajdonképpen a felépítendő kapcsolatok irányától függ.

Forwarding proxy

Ez az üzemmód a „klasszikus" felállás. A tűzfal a hálózat határán helyezkedik el, és a belső hálózatról érkező kéréseket továbbítja a külvilágban található rendeltetési helyre.

Reverse proxy

Ez pedig, mint a neve is mutatja, fordított üzemmódban működik. Vagyis: itt az ügyfelek az Interneten vannak, és a saját kiszolgálók felé továbbítódnak a kérések. Nyilvánvaló előny, hogy magához a kiszolgálóhoz közvetlenül nem érkezhetnek kérések, így az igen jól védett környezetben működhet. Az ilyen megoldások az ügyfelek számára általában átlátszóak, az ügyfél nem is tudja, hogy nem közvetlenül a kiszolgálóval folytat kommunikációt

A címfordító *(NAT)* tűzfalak:

A címfordító eszközöket sokan nem is tekintik tűzfalnak *(viszont a tűzfalmegoldások technikai megvalósításánál muszáj megemlíteni a címfordítást, és ez itt nagyon jó helynek látszott :-))*. Miről is van szó? Tulajdonképpen arról, hogy a címfordítást végző eszköz az IP csomagban kicseréli valamelyik *(lásd később)* IP címet, és esetlegesen a portot is. Lássuk, milyen címfordításokat végezhetünk:

Dinamikus címfordítás

Ezt általában a belső hálózatról kifelé igyekvő ügyfelek esetén alkalmazzuk. Amikor dinamikus címfordítást végzünk, akkor a címfordító eszköz egyetlen érvényes IP cím mögé *(nem szükségszerűen a saját érvényes IP címe)* rejti el az ügyfelet. Jogos a kérdés, hogy hogyan is teszi ezt. Egészen egyszerűen kicseréli a forrás IP címet *(szükség esetén a forrás portot is)* a csomagokban, így a válaszok erre a megváltozott címre fognak érkezni. Természetesen az eszköznek meg kell oldania azt is, hogy ezek a csomagok visszajussanak az ügyfélhez, vagyis a kapcsolat idejére meg kell jegyeznie, hogy milyen címet *(esetleg portot)* mivel cserélt ki és hová irányult az adott kapcsolat.

Statikus célcím-fordítás

Mint talán nevéből is sejthető, mindig az egy adott címre érkező csomagokban levő célcímet cseréli ki, egy előre meghatározott címre. Mire is használható ez a gyakorlatban? Maradjunk a webkiszolgálós példánál, de azzal a változással, hogy legyen a kiszolgáló a belső hálózaton, és legyen privát címe. Nyilvánvaló, hogy csak akkor tudnak a webkiszolgáló felé kéréseket küldeni, ha van egy nyilvános címe is. Akkor most rakjuk ki az Internetre? A világért sem! Hiszen itt a kiváló tűzfalunk *(vagy legalábbis címfordító eszközünk)*! Akkor mit is teszünk? A DNS-ben megadunk a webkiszolgálónak egy nyilvános IP címet, és az erre a címre érkező kérések cél IP címeit pedig kicseréljük a webkiszolgálónk privát címére. És íme, a kérések már érkeznek is a webkiszolgálóhoz. *(Vigyázat! Ez a megoldás önmagában NEM FOKOZZA A BIZTONSÁGOT!!! Fontos, hogy a tűzfal szabályrendszerében ezekre a kapcsolatokra legyen valamilyen szűrés, vagy készítsük fel a kiszolgálót az esetleges támadások túlélésére.)* Szóval ott tartottunk, hogy a kérések megérkeznek a kiszolgálóhoz. De vajon a válaszok kimennek? Nem *(vagy ha igen, akkor majd az első helyesen beállított útvonalválasztó „elkaszálja" őket a privát IP cím miatt)*. Ezért van szükségünk a:

Statikus forráscím-fordításra

Ami pedig magától értetődően az előző ellentéte, vagyis az adott belső címről érkező csomagok forrás IP címeit kicseréljük az előre meghatározott külső IP címre. Most már mindenki megnyugodhat? Igen. A webkiszolgálónk megkapta a kéréseket, a külső ügyfél pedig a kért weblapokat. Hurrá! Mint az már az előzőekben is elhangzott, gyakorlatilag nincs olyan eszköz a piacon, amely az egyes funkciók *(csomagszűrés, állapottáblás csomagszűrés, proxy, címfordítás)* közül csak az egyiket tartalmazná. Jelenleg a legegyszerűbb tűzfaleszközök azok az útvonalválasztók, melyek rendelkeznek csomagszűrő képességgel, és tudnak címfordítást végezni. A fejlettebb, bonyolultabb, igazán tűzfalnak tekinthető rendszerek pedig általában a proxy-k, az állapottáblás csomagszűrők, és a címfordítás előnyeit kihasználva működnek.

Ki, és mi ellen is védekezünk?

Most már ismerjük a tűzfalmegoldások alapjait. Ha hatékonyan akarjuk megvédeni hálózatunkat, tudnunk kell azt is, hogy ki, és milyen támadástípusok ellen védekezünk. E cikk keretei nyilván nem elegendőek arra, hogy minden támadástípust részletesen ismertessünk, hiszen ezzel akár egy teljes

könyvet is meg lehetne tölteni, de azt mindenképpen szükségesnek érzem, hogy legalább a legfontosabbakat megemlítsem.

Ki ellen?
A hálózatokra támadókat sokféleképpen lehetne csoportosítani, én most a következő kategóriákat tartanám fontosnak *(szándékosan nem fordítom le a megnevezéseket, mert egyrészt nem igazán lehet visszaadni az eredeti szó jelentését, másrészt ezek már elég elterjedtek)*:

Hacker
Ő csak a kihívást keresi, általában nem akar, és ezért nem is okoz kárt, hiszen jól felkészült szakemberről van szó. Nagyon alaposan ismeri a védelmi rendszereket, a protokollokat, és ezek gyengeségeit. „Fegyvertárát" saját maga fejleszti. Ha bejutott a rendszerünkbe, általában ügyesen megsemmisíti ennek nyomait, esetleg külön felhívja a figyelmet „hőstettére".

Professzionális hacker
Ez egy kissé új keletű fogalom. A hacker-hez hasonló, viszont neki ez a „szakmája". Gyakorlatilag információkat lop, ezeket adja el. Ne tessék nevetni, ez létező dolog!

Cracker
Olyan mint a hacker, de ő az anarchista változat. Általában tevékenységének egyetlen célja a rombolás.

Script kiddie
Ők azok a hülyegyerekek *(már bocsánat)*, akik letöltenek az Internetről mindenféle „játékokat", majd ráeresztik azokat a kiszemelt rendszerre, vagy kiszolgálóra. Tudásuk nem túl nagy, éppen ezért gyakran azt sem tudják, hogy mivel is „játszanak". Talán ők a legveszélyesebbek, viszont őket lehet legjobban kiszűrni egy jól beállított, és naprakészen tartott tűzfallal.

Ártatlan áldozatok
Nem, ez nem tévedés. Bizony olyanok is támadhatnak ránk, akik maguk is ártatlan áldozatok. Ehhez mindössze annyi szükséges, hogy a gépükre bejusson egy trójai program, amely egy adott időben meghatározott tevékenységet hajt végre. A DDoS támadások „elkövetői" általában maguk is áldozatok *(a valódi elkövető pedig az, aki a trójai programot elkészítette)*.

Mi ellen?
Itt most csak érdekességképpen említenék néhány támadástípust a teljesség igénye nélkül. Akit ennél mélyebben érdekel a téma, annak érdemes részt venni a NetAcademia biztonsági tanfolyamain. Mindegyik támadástípusnál igyekeztem feltüntetni, hogy mi kivédésének legjobb módszere, ami azonban mindegyikre érvényes az az, hogy odafigyeléssel jelentősen csökkenthető a sikeres támadás esélye.

DoS *(Denial of Service – a szolgáltatás megszűnése)* támadás
Olyan támadási forma, melynek egyetlen célja az, hogy egy adott kiszolgáló ne tudjon szolgáltatást nyújtani. Elérhető például azzal, hogy egy operációs rendszer, protokoll, vagy az operációs rendszeren futó szolgáltatás egy gyengeségét kihasználva a processzor terhelése tartósan 100%-os lesz, vagy nemes egyszerűséggel „lefagy" a gép. De még ez sem feltétlen szükséges. Elég normális, szabályos kérésekkel bombáznunk a kiszolgálót, csak annyi a lényeg, hogy ebből sok legyen! Védekezni ez ellen a támadási forma ellen naprakészséggel lehet leginkább.

DDoS *(Distributed Denial of Service)*
A DoS támadás egy fajtája, azzal a különbséggel, hogy itt rengeteg „támadó" van *(ezért Distributed, vagyis elosztott)*. Egy ilyen megvalósításához nem is kell mást tenni, csak sok felhasználó gépére bejuttatni egy trójai programot, amely egy előre megadott időpontban elkezd kérésekkel bombázni egy adott webkiszolgálót. Ha elegendő helyre sikerült bejuttatni a trójait, szinte biztos a siker. A védekezés szinte lehetetlen.

Behatolás *(Intrusion)*
Most csak a hálózatba történő behatolásról beszélek. Ez önmagában nem okoz kárt, sőt, ha nem rosszindulatú a támadó, esetleg egy e-mail-ben felhívja figyelmünket rendszerünk gyengeségére. Az ilyen figyelmeztetéseket mindig vegyük nagyon komolyan! Rengeteg példa van ugyanis arra, hogy a második figyelmeztetés már komoly károkozással is együtt jár.

Információlopás
Ez lehet a behatolást követő lépés. Védekezni ellene csak úgy tudunk, ha a bizalmas adatainkat nagyon jól védjük.

Social engineering
Ennek sincs igazán jó magyar megfelelője. Itt igazából az emberi butaság/képzetlenség kihasználásáról van szó. Tipikus példája ennek, hogy felhívnak egy felhasználót, hogy ugyan már, tesztelési céllal árulja el a jelszavát, vagy csak egész egyszerűen leolvassák a jelszót a monitorra ragasztott kis sárga papírfecnikről. Védekezni ellene a felhasználók képzésével lehet.

Valaki más megszemélyesítése *(spoofing)*
Képzeljük el azt az esetet, amikor valaki egy általunk megbízhatónak ítélt webkiszolgáló helyett kezdi el működtetni saját kiszolgálóját. Rengeteg káros kódot juttathat be így hálózatunkba.
Ennyit az általánosságokról. Bár ezzel még korántsem teljes a kép, az ISA Server kiépítésének egyes lépéseinél ki fogok térni a még hiányzó dolgokra.
A jövő hónapban az ISA Server telepítéséről, és az alapvető funkciók *(kiszolgáló publikálása, SecureNAT, alapvető szűrések)* megvalósításáról fogok írni. Addig is javaslom mindenkinek, hogy olvassa a NetAcademia security levelezési listáját, hiszen itt az egyes problémák konkrét megoldása mellett naprakészen tájékozódhat a legújabb felderített biztonsági lyukakról, és „befoltozásukról".

Pb
MCSE

Az Exchange 2000 és az Active Directory

A ForestPrep és a DomainPrep

A Microsoft Exchange 2000 Server címtárként a Windows 2000 Active Directory-t használja. Az Active Directory adatbázisának szerkezetét a séma írja le, mely alaphelyzetben a Windows 2000 tartományvezérlő funkciók ellátásához szükséges adatok tárolását teszi lehetővé. Az Exchange 2000 telepítéséhez tehát elő kell készíteni az Active Directory címtárat. Ezt a legtöbb esetben a telepítéssel egy időben, maga a telepítőprogram elvégzi, de vannak bizonyos esetek, amikor az Exchange 2000 telepítése előtt kell felkészítenünk az Active Directory-t az Exchange fogadására. Erre szintén az Exchange 2000 kiszolgáló telepítőprogramját használhatjuk két parancssori kapcsolóval:

```
setup /ForestPrep
setup /DomainPrep
```

Most összefoglaljuk mindkét parancssori opció működését, valamint a futtatásukhoz szükséges jogosultságokat. Megismerjük, mire kell ügyelnünk a ForestPrep és a DomainPrep futtatása előtt.

Miért van szükség a ForestPrep és DomainPrep futtatására?

Általában a nagyvállalatok az üzenetküldő rendszereket kezelő rendszergazdáknak nem akarnak magasszintű tartományi vagy vállalati jogosultságokat adni, a legtöbb esetben az Exchange rendszergazdák egy külön csoportban dolgoznak a helyi hálózatot felügyelő és karbantartó kollégáiktól. A saját címtárral rendelkező Exchange 5.5-öt használó vállalatoknál könnyen szétválaszthatók a tartományi és az üzenetküldő rendszerrel kapcsolatos rendszergazdaszerepek, de az Exchange 2000 a Windows 2000 Active Directory címtárát használja, így körültekintően kell beállítani a kétfajta szerephez szükséges jogosultságokat egyazon adatbázisban.

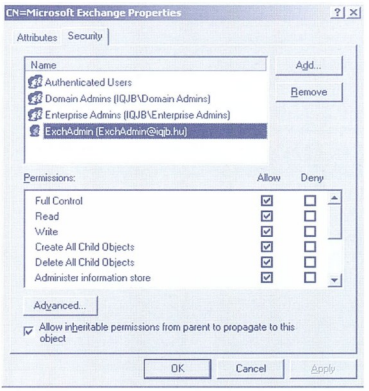
Ahol ez a szétválasztás fontos - a ForestPrep és a DomainPrep - el lehet különíteni a magasszintű hálózati jogosultságokat igénylő sémamódosítást és a címtárbeli jogosultságok beállítását az alacsonyabb szintű jogosultságot igénylő Exchange 2000 Server telepítésétől.

A Windows 2000 EnterpriseAdmin és SchemaAdmin jogokkal rendelkező rendszergazdák futtathatják a ForestPrep opcióval az Exchange 2000 telepítő programját, melynek során létrehoznak egy felhasználói fiókot az Exchange 2000 leendő rendszergazdája számára. Ez az Exchange rendszergazda a ForestPrep és a DomainPrep futtatása után elegendő tartományi jogosultságokkal rendelkezik az Exchange 2000 telepítéséhez, így már csak a helyi gépen kell rendszergazdának lennie.

A jogosultsági kérdéseken kívül egy másik praktikus ok, ami miatt külön kell futtatnunk a ForestPrep-et, hogy a sémát az Active Directory erdőben csak egy helyen, a Schema Master gépen módosíthatjuk. Így ha az első Exchange 2000 telepítését nem azon a telephelyen vagy tartományban végeznénk, ahol ez a Schema Master található, akkor nagy valószínűséggel a telepítő azt nem fogja tudni elérni vagy fizikai, vagy jogosultsági okokból, így a sémamódosítást nem tudjuk végrehajtani.

ForestPrep

Nézzük meg részletesen, hogy hogyan működik az Exchange telepítő ForestPrep opciója. A ForestPrep végrehajtja az összes olyan Exchange 2000 telepítési feladatot, amelyhez EnterpriseAdmin és SchemaAdmin jogosultságok, illetve a Schema Master gép elérése szükséges. Exchange-specifikus információk tárolásához szükséges osztályokkal, attribútumokkal és speciális jogosultságok definícióival bővíti az Active Directory sémáját, valamint objektumokat hoz létre az Active Directory konfigurációtárolójában *(Configuration Partition)*, és ezekhez az objektumokhoz a leendő Exchange rendszergazdai fiók számára megfelelő hozzáférési jogokat ad, így ez a rendszergazda most már elegendő jogosultsággal fog rendelkezni az első Exchange 2000 telepítéséhez.



Az Exchange rendszergazda jogosultságai az Exchange szervezet objektumán az AD konfigurációs partíciójában ForestPrep után

A ForestPrep által az Exchange 2000 rendszergazda számára létrehozott felhasználói fiók ugyanazokkal a jogosultságokkal rendelkezik, mint a majdani kész Exchange 2000-ben az Exchange Administration Delegation Wizard által szervezeti szinten létrehozott Exchange Full Administrator. A ForestPrep-et Windows 2000 erdőnként elég egyszer futtatni, ekkor a sémát – ami a teljes erdőre vonatkozik - módosítja, létrehozza, és bejegyzi az Exchange szervezet nevét és objektumát az Active Directory-ba.

A ForestPrep segítségével létrehozott Exchange rendszergazdának ugyan van joga telepíteni az Exchange 2000-et, de ezzel a felhasználói fiókkal alaphelyzetben nem lehet további felhasználói vagy levelezési fiókokat létrehozni. Ha ehhez szükséges jogokkal is fel kívánjuk ruházni az Exchange rendszergazdát, akkor több lehetőségünk van. Adhatunk rendszergazdai jogokat a Windows felhasználói fiókok létrehozásához és kezeléséhez ennek a fióknak, ha hozzáadjuk az Account Operators csoporthoz. Vagy használhatjuk a Windows 2000 Delegation of Control Wizard-ot, amellyel engedélyezhetjük az Exchange rendszergazdának a Users tároló objektumainak kezelését. Az is megoldás lehet, hogy egy új, speciális szervezeti egységet hozunk létre az Exchange felhasználók számára, és az Exchange rendszergazdának csak erre adunk teljes hozzáférési jogot.

A ForestPrep futtatása előtt

Néhány információt össze kell gyűjtenünk a ForestPrep futtatása előtt, mivel különböző kérdéseket fog feltenni attól függően, hogy ez egy új Exchange 2000 szervezet telepítése lesz, vagy egy létező Exchange 5.5 szervezetbe integráljuk.

Új telepítés

Az Exchange 2000 Server új szervezetbe történő telepítéséhez a hálózati rendszergazdának az alábbi információkra van szüksége a ForestPrep futtatása előtt:

- Az új Exchange 2000 szervezet neve
- Annak a felhasználói fióknak a neve, aki az Exchange 2000 kiszolgálót telepíteni fogja.

Telepítés Exchange 5.5 szervezetbe

Egy már meglévő Exchange 5.5 szervezetbe történő telepítéshez a hálózati rendszergazdának az alábbi információkra van szüksége a ForestPrep futtatása előtt:

- Egy olyan Exchange 5.5 kiszolgálónak a neve, ami azon Exchange telephelyen *(Site)* van, amelybe az Exchange 2000-et telepíteni szeretnénk
- Exchange 5.5 szolgáltatási fiók és jelszó

A megnevezett Exchange 5.5 kiszolgálónak üzemelnie kell és legalább a Service Pack 3-nak telepítve kell lennie rajta, valamint az Active Directory Connector *(ADC)* Exchange 2000 Server CD-n található változatának telepítve kell lennie az erdőben. Az Exchange 5.5 SP3 szolgáltatója a ForestPrep segédprogramnak az Exchange 5.5 szervezet információit.
Van némi átfedés a hálózati és az Exchange jogosultságok között az Exchange 5.5 telephelybe történő telepítés esetén, hiszen a ForestPrep-et futtató fióknak Admin jogosultsággal kell rendelkeznie mind az Exchange 5.5 telephelyhez, mind a telephely konfiguráció tárolóján *(Site Configuration Container)*. Mivel az Exchange 5.5 a Windows 2000 Active Directory-tól eltérő, saját címtárral rendelkezik, az EnterpriseAdmin jogosultság nem érvényes az Exchange 5.5 kiszolgálóra.

Milyen követelményei vannak a ForestPrep futtatásának?

A ForestPrep futtatásához az alábbi feltételeknek kell teljesülniük:

- A ForestPrep-nek a Schema Master-rel közös tartományban kell futnia *(alaphelyzetben ez a gyökértartomány – root domain)*, és el kell tudnia érni a Schema Master gépet
- A ForestPrep-et futtató személynek tagja kell lennie az Enterprise Admins és a Schema Admins csoportoknak
- Exchange 5.5 telephelybe történő telepítéskor a ForestPrep-et futtató fióknak Admin jogosultsággal kell rendelkeznie a telephelyobjektumon és a telephely-konfiguráció tárolóján is.

Mikor futtassuk a ForestPrep segédprogramot?

A legtöbb bevezetési forgatókönyvben szükség van a ForestPrep futtatására a sikeres Exchange 2000 telepítéshez. Akkor feltétlenül, ha az Exchange rendszergazdának nincs EnterpriseAdmin és SchemaAdmin jogosultsága, illetve a fenti követelmények közül bármelyik nem teljesül.
Az Exchange 2000 telepítéshez, az Exchange rendszergazdának rendszergazdai jogokkal kell rendelkeznie a helyi kiszolgálón és ezeket a jogokat engedélyeznie kell a ForestPrep számára is. Azonban ez teljesül, ha a rendszergazda a Domain Admins csoport tagja, mert e csoport tagjainak alaphelyzetben rendszergazdai jogosultságuk van a tartomány összes számítógépén.
Egy Exchange 2000 komponens telepítése – a Key Management Server - ehhez képest több jogot követel meg, itt nem elég helyi rendszergazdának lenni, hanem Domain Administrators csoport tagjának is.
Ha az Exchange 2000 telepítése majd gyermektartományban történik, akkor a ForestPrep segédprogramot először a szülőtartományban futtassuk. Ha ezt nem tesszük meg, a telepítő figyelmeztetni fog erre.

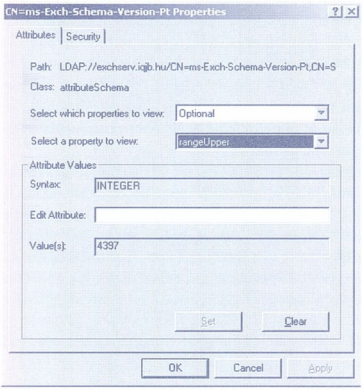
Mikor felesleges futtatni a ForestPrep segédprogramot?

A ForestPrep-et a vállalat felépítésétől függően, az első Exchange 2000 kiszolgáló telepítése előtt kell általában futtatni. Azonban van néhány kivétel *(általában a kisvállalatoknál)* amikor a ForestPrep használata nem szükséges.
A ForestPrep és a DomainPrep a telepítéskor automatikusan elindul, de csak akkor tud sikeresen lefutni, ha az Exchange rendszergazdai fiók a Schema Admins és az Enterprise Admins csoportok tagja és az első Exchange 2000 kiszolgáló ugyanabban a tartományban van, mint a Schema Master. Ha a feltételek nem teljesülnek, a telepítő hibajelzéssel megáll. Ebben az esetben nem szükséges kézzel elindítani a telepítést egyik előkészítő opcióval sem. Ilyenkor a telepítőt futtató felhasználói fiók lesz alaphelyzetben az Exchange 2000 rendszergazdai fiókja.

A replikációhoz szükséges idő

A ForestPrep futtatása után győződjünk meg arról, hogy eltelt-e elegendő idő az adatbázis séma replikációjára az egész tartományban és a teljes erdőben. A vállalat földrajzi elhelyezkedésétől és a Windows 2000 telephelyek *(Site)* és tartományok közötti hálózati kapcsolat sebességétől függően ez némi időt vehet igénybe. Csak abban az esetben futtassuk a DomainPrep segédprogramot, vagy a teljes telepítést, ha megbizonyosodtunk arról, hogy az Exchange-specifikus információk replikálása a teljes rendszerben megtörtént. A replikáció megtörténtéről úgy tudunk meggyőződni, hogy ellenőrizzük az 1575-ös esemény bekövetkeztét a címtárnaplóban, vagy megnézzük az ADSIEdit segédprogrammal a cn=ms-Exch-Schema-Version-Pt objektum rangeUpper pa-

raméterét. Ha ez kisebb, mint 4397, vagy nem létezik az objektum, akkor még nem futott le a séma replikációja.



Az AD séma verziójának ellenőrzése

DomainPrep

A DomainPrep végrehajtja azokat az Exchange telepítési feladatokat, amelyekhez DomainAdmin jogosultságra van szükség. A DomainPrep-et minden Exchange 2000 kiszolgálót, vagy Exchange felhasználókat tartalmazó tartományban csak egyszer kell lefuttatni. *(Azt az Exchange tartományt, amely levelezőfelhasználókat tartalmaz, de Exchange kiszolgálót nem, felhasználói tartománynak hívjuk.)* Ez a segédprogram létrehozza az Exchange kiszolgáló üzemeltetéséhez szükséges csoportokat és a felhasználói attribútumok olvasásához és módosításához szükséges jogosultságokat.

A DomainPrep két új csoportot hoz létre: az Exchange Domain Servers *(Windows 2000 globális biztonsági csoport)* és az Exchange Enterprise Servers *(Windows 2000 tartomány helyi biztonsági csoport)*.

A DomainPrep ezen kívül létrehoz egy Public Folder tárolót az Active Directory-ban. Míg a ForestPrep az egész erdőre kiterjedő konfiguráció-tárolóban dolgozik, a Public Folder objektum — a Microsoft Exchange System Objects — a tartománypartícióban található. A DomainPrep létrehozza ezt a tárolót minden tartományban, ahol lefuttatjuk.

A DomainPrep által létrehozott csoportok viszonya

Az Exchange Domain Servers csoport

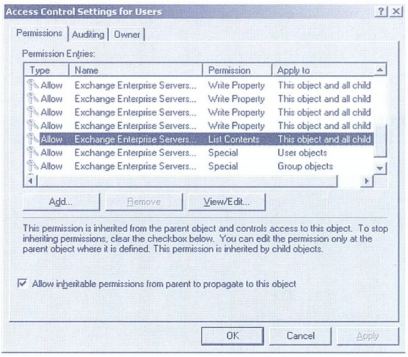
Az Exchange Domain Servers globális biztonsági csoport tartalmazza a tartomány összes Exchange kiszolgálójának számítógépfiókját. Bár ezt is a DomainPrep hozza létre, az Exchange Domain Servers csoport nem látható a tartomány első Exchange 2000 kiszolgálójának telepítéséig.

Az Exchange Domain Servers csoport szükséges a Recipient Update Service használatához, amire szükség van az Exchange szervezet minden egyes tartományában. Ebbe beletartoznak a felhasználói tartományok is, amelyek levelező felhasználókat tartalmaznak, de Exchange kiszolgálókat nem. A Recipient Update Service szolgáltatást az Exchange az alapértelmezett és egyedi címlisták, valamint a megváltoztatott címzett házirend *(Recipient Policy)* által előírt címváltozások előállítására és frissítésére használja.

Az Exchange Enterprise Servers csoport

Az Exchange Enterprise Servers csoport tartalmazza az összes Exchange Domain Servers csoportot a szervezeten belül. Más szóval, minden Exchange kiszolgálóval rendelkező tartomány Exchange Domain Servers csoportja, amelyben a DomainPrep lefutott és a Recipient Update Service szolgáltatás aktív, az Exchange Enterprise Servers csoport tagja.

Ez a csoport azonnal elérhető, amint a DomainPrep a tartományának Exchange Domain Servers csoportját hozzáadta. Az összes többi, aktív Recipient Update Service szolgáltatást futtató tartomány Exchange Domain Servers csoportját a Recipient Update Service adja hozzá.



Az Exchange kiszolgálók speciális jogai a felhasználói objektumokon

A DomainPrep futtatása előtt

A DomainPrep futtatásához semmilyen információra nincs szükségünk az Exchange szervezettel kapcsolatban, még Exchange 5.5 telephelybe történő telepítéskor sem.

Milyen követelményei vannak a DomainPrep futtatásának?

A DomainPrep segédprogram futtatásához az alábbi követelményeknek kell teljesülnie:

- A DomainPrep-et futtató felhasználói fióknak a Domain Admins csoport tagjának kell lennie

- A ForestPrep segédprogramnak már le kellett futnia a Windows 2000 erdőben
- A ForestPrep által az Active Directory számára készített sémamódosítások replikálásának meg kellett történnie a teljes szervezetben

Mikor futtassuk a DomainPrep segédprogramot?

A DomainPrep megfelelő működéséhez az alábbiak szükségesek:

- A ForestPrep futtatása, és az összes ForestPrep módosítás teljes szervezeti replikálása után
- Mielőtt az Exchange 2000 rendszergazda *(amit a ForestPrep hozott létre)*, a tartományban telepíti az első Exchange 2000 kiszolgálót
- Mielőtt egy felhasználói tartományban *(levelező felhasználók vannak, de Exchange kiszolgáló nincsen)* Recipient Update Service szolgáltatást szeretnénk telepíteni.

Mikor szükségtelen a DomainPrep futtatása?

Általában a tartomány első Exchange 2000 kiszolgálójának telepítése előtt futtatjuk a DomainPrep segédprogramot, de ez nem mindig szükséges. Abban az esetben például, ha a felhasználói fiók, amellyel telepítjük a tartomány első Exchange 2000 kiszolgálóját azon kívül, hogy Exchange Full Administrator még a Domain Admins csoport tagja is, nem szükséges futtatni a DomainPrep-et. Ugyanez érvényes, ha az Exchange-et telepítő felhasználó Enterprise Admin jogosultsággal rendelkezik.

Minkét esetben a DomainPrep az Exchange telepítésekor, rejtett összetevőként automatikusan lefut.

Soós Tibor (MCSE, MCT) az Exchange Server 4.0 megjelenésétől szakértője és oktatója az Exchange rendszereknek, az Exchange 2000 levelezőlista tanácsadója, az IQSOFT – John Bryce Oktatóközpont munkatársa. (soost@iqjb.hu)

Hailstorm

A probléma

Olyan régóta szolgáljuk ki a számítógépeket, hogy szinte már szükségszerűnek érezzük alárendeltségünket. Itt az ideje, hogy kiadjuk a jelszót: „Legyenek egyszerűbben használhatók a számítógépek!" Beszéljenek hozzánk, tegyenek meg nekünk mindent, biztosítsák a számunkra szükséges információkat, segítsenek másokkal együtt dolgozni és alkalmazkodjanak egyedi igényeinkhez. Csak így tehetnek minket produktívabbá és szolgálhatnak ki minket teljesen, ahelyett, hogy mi szolgálnánk ki őket.

– Michael L. Dertouzos, a MIT Laboratory for Computer Science igazgatója írta a The Unfinished Revolution című könyvében

Az elmúlt 25 évben az informatikai technológia fejlődése hihetetlen mértékben megkönnyítette a felhasználók és vállalatok munkáját, de még vannak problémák. Az egyedi alkalmazások és eszközök nem veszik figyelembe más területek igényeit. Arra kényszerülünk, hogy alkalmazkodjunk a technológiához, ahelyett, hogy a technológia alkalmazkodna hozzánk. Ez megnehezíti az emberek életét. Néha úgy tűnik, hogy minden programnak, minden webhelynek és minden eszköznek saját használati szabályai vannak. Ha például a PC-nkbe be akarjuk írni egy barátunk új telefonszámát, ahhoz a billentyűzet és az egér használata kell. Ahhoz, hogy ugyanezt az adatot a Palm Pilot-unkba, Pocket PC-nkbe vagy a mobiltelefonunkba írjuk be, teljesen más rendszer használatát kell megtanulnunk – mintha újra kellene tanulnunk a betűket!
Az emberek nem tudják irányítani az őket körülvevő technológiát. A fontos és személyes adataink több száz helyen szétszóródva, alkalmazásokba, termékek regisztrációs adatbázisaiba, cookie-kba és webhelyek adatbázisaiba zárva tárolódnak. Ha egy barátunk telefonszámát beírjuk a mobiltelefonunkba, az nem érhető el a PC email alkalmazásában: a két technológia képtelen az egymással való kommunikációra.
Ha elköltözünk, az új címet minden olyan webhelyen meg kell adnunk, amelynek szüksége van a címünkre – ha ezt elfelejtjük, az Internet "kényelmes" használata azonnal problémássá válik. Minden webhely egy-egy adatsziget, amely folyamatosan biztosítja, hogy ne legyünk képesek személyes információink felhasználásának irányítására. Nem tudjuk egyszerűen frissíteni a saját adatainkat, és nem tudjuk szabályozni, hogy mi történjen a megadott adatokkal – sok esetben meg sem tudjuk tekinteni, amit egyszer már megadtunk.
Az alkalmazások, webhelyek és szolgáltatások elszigeteltsége gyakorlatilag lehetetlenné teszi, hogy a technológiák együttműködjenek. Képzeljük el, hogy jegyet rendelünk egy online utazási helyfoglaló rendszerben, és azt szeretnénk, ha az útvonalterv automatikusan a naptárrendszerükhöz adódjon. A webhely és a naptáralkalmazás nem tud egymással kommunikálni – de ha mégis tudnak, egyik sem tud semmilyen módszerrel megbizonyosodni arról, hogy vajon ugyanarról a személyről van-e szó.
Mivel arra kényszerülünk, hogy mi alkalmazkodjunk a technológiához, ahelyett, hogy az alkalmazkodna hozzánk, az alkalmazások, webhelyek, és az eszközök csak korlátozottan tudják kiszolgálni az igényeinket. Ez nemcsak az új hardver- és szoftvertechnológiák elterjedését késlelteti, de korlátozza a hatékony, termelékeny és hasznos termékek és szolgáltatások kifejlesztését is.

Mi az a HailStorm?

A Microsoft .NET kezdeményezés részeként a Microsoft bevezeti a „HailStorm" felhasználóközpontú személyes adattár architektúrát és XML webszolgáltatás-készletet. A HailStorm lehetővé fogja tenni a jelenlegi sokezer adattárolóhely integrációját. A HailStorm szolgáltatások emberközpontúak, segítségükkel a felhasználók maguk kezelhetik a saját adataikat. A HailStorm szolgáltatások a .NET technológiát és architektúrát használják, amely lehetővé teszi az alkalmazások, eszközök és szolgáltatások együttműködését. E szolgáltatások lehetővé teszik, hogy a felhasználók határozzák meg, hogy ki férhet hozzá adataikhoz, mit tehet vele és mennyi időre kap hozzáférési jogokat.
A HailStorm *(amely a Passport felhasználói hitelesítési rendszeren alapul)* lehetővé teszi az alkalmazások és szolgáltatások együttműködését. A HailStorm szolgáltatásokkal például sokkal egyszerűbbé válik az online repülőjegy-foglalás, mivel a felhasználó hozzájárulásával az utazási iroda szolgáltatása automatikusan hozzáférhet a felhasználók beállításaihoz és bankszámlájához. Ha üzleti ügyben utazunk és vállalatunk kötelező utazási irányelveket ír elő, a személyes tagság és a vállalat HailStorm csoportbeállításai lehetővé teszik, hogy az utazási szolgáltatás kínálatában automatikusan csak azok a lehetőségek szerepeljenek, amelyek mind a személyes beállításoknak, mind a vállalati igényeknek megfelelnek. Miután kiválasztottuk a járatot, az utazási szolgáltatás *(ha engedélyezzük)* a HailStorm használatával meghatározza, hogy milyen naptárszolgáltatást használunk, és automatikusan ütemezi az útitervet, valamint automatikusan frissíti, és értesítést küld, ha a járat késni fog. A HailStorm-mal megoszthatjuk ezt az repülési útitervet azzal, akit meg akarunk látogatni, így ő is tudni fogja, hogy mikor és hova érkezünk. A HailStorm alapú naptár adatai a saját PC-nkkel, más PC-jével, telefonnal, PDA-val vagy bármilyen más intelligens eszközzel megtekinthetők.

Irányítás a HailStorm-mal

A HailStorm-mal a technológia a felhasználó érdekében és az ő irányítása alatt működik. Hasonlítsuk ezt össze a mai helyzettel, amelyben nekünk kell a technológiához alkalmazkodnunk, nekünk kell a híd szerepét betöltenünk a kü-

lönböző eszközök, alkalmazások és webhelyszínek között. A HailStorm-mal soha többé nem kell kézzel adatokat átmásolnunk egy szolgáltatásról egy másikra. Soha többé nem kell amiatt aggódnunk, hogy hogyan tudjuk a címünket átírni minden egyes helyen, ahol megadtuk.

A HailStorm a személyes adatok online védelmét is megoldja, használatával a felhasználó kezeli a személyes információkat, és ő dönti el, hogy kivel és milyen feltételek mellett osztja meg azokat. Az adatok tulajdonosa a felhasználó. Az adatok bármilyen hozzáféréséhez, módosításához, és azok bármilyen felhasználásához a felhasználó kifejezett jóváhagyása szükséges. A jóváhagyás korlátozott hatáskörben (milyen adatok hozzáférhetők?) és korlátozott időtartamra (mikor jár le az engedély?) érvényes. A HailStorm jogi és technikai módszerekkel tiltja le az adatok jogosulatlan használatát, és ez a korlátozás az adott tranzakción kívül is érvényes. A felhasználói irányítás hangsúlyozása szöges ellentétben áll a jelenlegi módszerrel, amelyben az alkalmazások és a vállalatok birtokolnak minden olyan adatot, amelyet a felhasználóról nyertek, és a felhasználó megkérdezése nélkül, gyakorlatilag korlátlanul használhatják azt.

A HailStorm felhasználóközpontú architektúra és szolgáltatáskészlet a .NET-hez, amely személyes információkat nyújt az Interneten egy felhasználó, egy futtatott szoftver vagy a felhasználó által használt eszközök számára. A HailStorm szolgáltatások a SOAP *(Simple Object Access Protocol)* és az XML *(eXtensible Markup Language)* használatával érhetők el, amelyek nyílt hozzáférési technológiák: minden olyan hálózatra csatlakoztatott eszközről meghívhatók, amely támogatja a SOAP-ot, függetlenül attól, hogy az egy operációs rendszer vagy egy Internetszolgáltató. A SOAP és az XML nyílt Internetszabvány, amelyeket a Microsoft a .NET első fázisában vezetett be. A HailStorm a következő lépés.

A HailStorm szolgáltatások

A HailStorm szolgáltatáskészlet segít az adatok kezelésében és védelmében, és az alkalmazások, eszközök és szolgáltatások közötti együttműködésben:

A HailStorm architektúra

A felhasználók a HailStorm-hoz alkalmazásaikkal, eszközeikkel és szolgáltatásaikkal *(a HailStorm végpontokkal)* férnek hozzá. A HailStorm alapú eszköz vagy alkalmazás a felhasználó jóváhagyásával automatikusan csatlakozik a megfelelő HailStorm szolgáltatáshoz. Mivel az általunk használt igen sok alkalmazás és eszköz egyetlen általunk kezelt, közös információkészlethez csatlakozik, képesek leszünk biztonságosan megosztani az adatokat a különböző technológiák, valamint emberek és szolgáltatások között.

A fejlesztők olyan alkalmazásokat és szolgáltatásokat hozhatnak létre, amelyek a HailStorm-ot használják a lehető legjobb eredmény elérésére. A HailStorm platform nyílt hozzáférési modellt használ, ami azt jelenti, hogy bármilyen eszközzel, alkalmazással és szolgáltatással elérhető, függetlenül a platformtól, operációs rendszertől, objektummodelltől, programozási nyelvtől vagy hálózatszolgáltatótól. A HailStorm szolgáltatások XML webszolgáltatások, amelyek a nyílt ipari szabványú XML-en és SOAP-on alapulnak; nincs szükség Microsoft környezetre vagy eszközökre a szolgáltatások eléréséhez. A Visual Studio.NET-ben található .NET infrastruktúra, a .NET Framework és a .NET Enterprise Servers természetesen teljesen támogatja a HailStorm-ot, hogy a fejlesztők számára a lehető legegyszerűbbé váljon a szolgáltatások használata.

Technikai szempontból a HailStorm alapja a Microsoft Passport hitelesítés. A HailStorm architektúra olyan azonosítási, adatvédelmi és adatmodelleket határoz meg, amelyek közösek a HailStorm szolgáltatásokban és így konzisztens fejlesztést és használatot tesznek lehetővé. A HailStorm egy elosztott rendszer, amely különböző alkalmazásokat, eszközöket és szolgáltatásokat tud kiszolgálni.

Az alapvető HailStorm szolgáltatások ezt az architektúrát használják a felhasználó alapadatainak *(pl. a naptár, a hely- és profiladatok)* kezelésére. A HailStorm-ot használó bármilyen megoldás használhatja ezeket az elemeket, így a felhasználónak nem kell újra megadnia és redundánsan tárolnia az adatokat, a fejlesztőnek pedig nem kell egyedi rendszert létrehoznia ezekhez.

A HailStorm-hoz szabványos XML webszolgáltatás készlettel férhetünk hozzá. A HailStorm alapú megoldások XML message interface-k *(XMI)* segítségével *(amelyek egyszerűen XML SOAP üzenetek készletei)* specifikus HailStorm eszközökkel működnek együtt.

A HailStorm szolgáltatáskészlete a következő:

- myAddress – elektronikus és földrajzi cím
- myProfile – név, becenév, személyes dátumok, kép
- myContacts – elektronikus kapcsolatok/címtár
- myLocation – elektronikus és földrajzi hely és találkozóhely
- myNotifications – értesítések kezelése és továbbítása
- myInbox – inboxelemek (email, hangos levél) a meglévő levelezési rendszerekkel való kapcsolattartáshoz
- myCalendar – időbeosztás és feladatkezelés
- myDocuments – dokumentumtár
- myApplicationSettings – alkalmazás-beállítások
- myFavoriteWebSites – kedvenc URL-ek és egyéb web-azonosítók
- myWallet – számlák, fizetőeszközök, kuponok és egyéb tranzakciós bejegyzések
- myDevices – eszközbeállítások
- myServices – személyes szolgáltatások
- myUsage – a fenti szolgáltatások használatának feljegyzései

A HailStorm architektúra konzisztens és rugalmasan bővíthető. Közös adat-, üzenet-, név-, helyszín-, biztonság-, adatmodellezés-, mérés- és hibakezelést biztosít az összes HailStorm szolgáltatásban. A HailStorm olyan, mint egy dinamikus, partícionált, sémázott XML tár. XML message interface-ekkel *(XMI)* érhető el, amelyek szabványos SOAP üzenetek, a visszanyert értékek XML-ek, és minden szolgáltatás támogatja a HTTP Post-ot üzenettovábbítási protokollként.

A HailStorm integrált adatvédelmi modellje Kerberos alapú hitelesítésre épül. A felhasználó szabályozza, hogy ki és milyen céllal férhet hozzá az adataihoz. A felhasználó vissza is vonhatja a hozzáférés engedélyezését. A felhasználók az adathozzáférést egy szolgáltatás vagy agent segítségével kezelhetik, ezek a szolgáltatások elég egyszerűek ahhoz, hogy használhatók legyenek.

A megbízhatóság igen fontos a HailStorm szolgáltatások sikeréhez. A Microsoft rengeteg jó *(és rossz)* tapasztalatot szerzett Internet helyszíneinek *(Hotmail web alapú email szolgáltatás, MSN, Microsoft.com és Passport)* működtetésekor, amelyek mindegyike a világ tíz legnagyobb webhelye között van. A Microsoft jelentős beruházásokat végez, hogy elérhetővé tegye azt a szolgáltatási szintet és megbízhatóságot, amit a HailStorm igényel.

A HailStorm üzlet

A dotcom összeomlás bizonyítja, hogy az internetes üzleti modellnek új alapokra van szüksége. Az ingyenes szolgáltatásnyújtás nem fenntartható módszer a vállalkozások számára. A Microsoft .NET lehetővé teszi olyan vállalkozások működését, amelyek az informatikát és a hálózati kapcsolatot felhasználva valódi értéket hozhatnak létre, olyan értéket, amiért az emberek hajlandóak fizetni.

A Microsoft a HailStorm szolgáltatást üzleti alapon fogja működtetni. A HailStorm szolgáltatások valódi működési költségekkel járnak, és ahelyett, hogy a felhasználó-központú modellt *(a kockázatosság miatt)* figyelmen kívül hagyva valaki mással, pl. hirdetőkkel fizettetnék meg a költségeket, az értéket kapó emberek – a végfelhasználók – fognak érte fizetni. A HailStorm segítségével az Interneten a felhasználók a kapott értékkel arányosan fizetnek. A Microsoft a fejlesztőktől is bevételekre számít, amelyek segítenek a szolgáltatások és termékek költségeinek fedezésében. Ezek kisebb összegek lesznek, hogy a lehető legtöbb fejlesztő dolgozzon a HailStorm-mal, de felszámolják a szokásos eszköz és támogatási költséget. Valamennyit az éles tesztkörnyezet használatáért is fizetni kell.

A szolgáltatáskezelők bizonyítványon alapuló licenckapcsolatban lesznek a Microsoft-tal, ez teszi lehetővé a HailStorm szolgáltatások használatát, és hogy egyetlen HailStorm-ot használó szolgáltatás se éljen vissza az erőforrásokkal. A bizonyítvány lehetővé teszi a szabályok megszegőinek kiszűrését a rendszerből. A bizonyítvány beszerzése és a HailStorm szolgáltatások használatának joga költséggel jár. A nagyobb támogatás, a szolgáltatásszintű szerződés és a nagyobb rendszerhasználat további költségeket jelenthet. Arra számítunk azonban, hogy ezek a költségek lényegesen kisebbek lesznek, mint egy hasonló, független szolgáltatás költségei.

A felhasználók, fejlesztők és szolgáltatáskezelők által fizetendő árak még nincsenek meghatározva.

A fejlesztők lehetőségei

A platform létrehozásának egyik nagy előnye, hogy lehetőséget biztosít más vállalatoknak arra, hogy a platformra üzleti modelleket hozzanak létre. A Microsoft .NET *(amelynek a HailStorm is része)* is biztosítani fogja ezeket a lehetőségeket. A fejlesztők számára a HailStorm kihasználására két jelentősebb lehetőség áll rendelkezésre:

- Létrehozhatnak olyan alkalmazásokat, eszközöket vagy szolgáltatásokat, amelyek a HailStorm szolgáltatásokat használják.
- Létrehozhatnak saját HailStorm kompatíbilis szolgáltatásokat.

Kihasználva, hogy a Microsoft jelentős beruházásokat végzett a HailStorm-ért, a fejlesztők képesek lesznek felhasználó-központú megoldásokat fejleszteni, hogy a technikai feltételek megteremtése helyett a saját területük fejlesztésére összpontosíthatnak. Ha például arról akarják értesíteni a felhasználót, hogy megérkezett egy áru amelyet megrendelt, a vállalatnak csak a SOAP és az XML létrehozásával kell törődnie, amely ahhoz szükséges, hogy kommunikálni tudjon a HailStorm myNotifications szolgáltatásával, amely szabványos SOAP és XML felülettel rendelkezik, függetlenül attól, hogy milyen alkalmazást használ a felhasználó. Nem kell azzal foglalkozniuk, hogy létrehozzanak egy olyan rendszert, amely azonosítja a felhasználót, vagy értesítést küld, és nem kell olyan alkalmazást sem létrehozniuk, amely fogadja ezeket az üzeneteket. Ehelyett arra koncentrálhatnak, hogy létrehozzák magát a szolgáltatást, amely gyorsabb, olcsóbb és sokkal egyszerűbben karbantartható.

A HailStorm szolgáltatások használatával a példában szereplő vállalat több embert tud elérni *(mivel nincs szükség egyedi alkalmazástelepítésre a szolgáltatás használatához, mert a felhasználók már rendelkeznek a szükséges szoftverekkel)* A sürgős üzeneteknek nem kell addig várakozniuk, amíg a felhasználó legközelebb valamilyen egyedi alkalmazást használ; ehelyett az értesítést azonnal megkapja, ha egy HailStorm eszközt használ. Az egy megoldásba összegyűjtött szolgáltatások sokkal hatékonyabban használhatók, mint az önálló megoldások. A HailStorm használatával az alkalmazásfejlesztők sokkal több potenciális ügyféllel léphetnek kapcsolatba, mint anélkül. Együttműködési alkalmazás használatához nincs szükség arra, hogy mindkét félnél egyidejűleg nyitva legyen az alkalmazás, csak arra, hogy mindketten egy intelligens HailStorm eszközt használjanak. Ha az egyik fél egy együttműködési alkalmazás felhasználója, a másik pedig nem, az együttműködés így is létrejöhet a Passport használatával; nem kell külön alkalmazás hozzá. Az alkalmazásfejlesztők számára ez azt jelenti, hogy a felhasználóik lényegesen több emberrel léphetnek kapcsolatba, akik megismerik az alkalmazásaikat.

A HailStorm elvei

A felhasználó irányít

A HailStorm lehetővé teszi a felhasználóknak, hogy ők kezeljék környezetüket és személyes adataikat. A többi Microsoft szolgáltatáshoz hasonlóan a HailStorm adatvédelmi modellje is megfelel a személyes adatok védelmére vonatkozó törvényeknek és meg fog felelni a Code of Fair Information Practices-nek is, ami több fogyasztóvédelmi program alapját fogja képezni, pl. az Online Privacy Alliance-ét, a TRUSTe-ét és a BBBOnLine-ét.

A HailStorm lehetővé teszi, hogy a vállalatok olyan ajánlatokat dolgozzanak ki, amelyek a felhasználó érdekében együttműködve hatékony, konzisztens és testreszabott szolgáltatásokat nyújtanak. A HailStorm egyik alapvető tervezési elve az volt, hogy saját adatait a felhasználó kezelje. Az adatvédelem alapvető pontja a HailStorm architektúrának.

A HailStorm modell a következő információs eljárásokra épül:

- Értesítés: a vevő értesítése az információ felhasználásáról;
- Választás: a személyes információk figyelmes összegyűjtése és elosztása;
- Hozzáférés: minden személyes információhoz;

- Biztonság: A beépített védelem miatt jóváhagyás nélkül senki sem férhet hozzá a személyes adatokhoz.

A személyes adatok védelme nagyon fontos tervezési elvárás a HailStorm architektúrával szemben. A HailStorm adatmodell része egy speciális biztonsági és hozzáférési modell, amely lehetővé teszi a felhasználók számára, hogy meghatározzák, hogy hogyan és kivel akarják megosztani a személyes adataikat. Ez az intelligens szoftver lehetővé teszi, hogy:

- Meghatározhassuk, hogy kik vagy milyen szolgáltatások férhessenek hozzá az adatainkhoz.
- Szükség szerint bárkivel megoszthassuk az adataikat. A HailStorm szigorú opt-in platformot alkalmaz a felhasználói adatokhoz.
- Szükség szerint visszavonhassuk a megosztási és hozzáférési jogokat. Ez az irányítási szint most általában nem elérhető a webhelyeken.
- A megosztás egy adott időre legyen érvényes: az adathozzáférés rendszer által felügyelt, időhöz kötött legyen.

A technikai lehetőségeken kívül az adatvédelemről a Microsoft adatgyűjtési és használati módszerei is gondoskodnak a HailStorm licenccel rendelkező felhasználói számára. A Microsoft szerződéses úton is kiköti a használat lehetőségeit, ez meghatározza, hogy mit lehet tenni a HailStorm forrásból származó felhasználói adatokkal. Ezenkívül a Microsoft elektronikusan és fizikai úton is védi a HailStormban kezelt adatokat, hogy megakadályozza a jogosulatlan hozzáférést és használatot.

A Microsoft a felhasználó kifejezett hozzájárulása nélkül nem kezel, ad el vagy tesz közzé semmilyen HailStorm felhasználói adatot. A felhasználó adataival való minden kölcsönhatás a jóváhagyásos opt-in modell alapján megy végbe: a személyes adatok csak az adat tulajdonosának kifejezett jóváhagyása esetén adhatók közre.

Nyílt hozzáférés

A HailStorm bármilyen olyan Internet kapcsolattal rendelkező eszközzel, szolgáltatással vagy alkalmazással elérhető, amely képes a felhasználó hitelesítésére és SOAP üzenetek küldésére és fogadására. A műveletek a platformtól, az operációs rendszertől, az objektummodelltől, a programozási nyelvtől, az alkalmazástól vagy online szolgáltatástól függetlenül szöveges SOAP üzenetek formájában továbbítódnak. A HailStorm ügyfél- és kiszolgálóoldalról is hozzáférhető, és egyik esetben sincs szükség Microsoft környezetre. A Microsoft már bemutatta, hogy a HailStorm szolgáltatások Windows®-ról, Macintosh-ról, Palm-ról, Pocket PC-ről és UNIX-ról is elérhetők.

Bővíthetőség

A HailStorm első kiadása a felhasználók és a fejlesztők számára feltételezhetően szükséges alapvető szolgáltatáskészletet nyújtja. Ezután az új szolgáltatások *(például myPhotos vagy myPortfolio)* és bővítmények a Microsoft Open Processzel a fejlesztők közösségének bevonásával vezethetők be. Egyetlen séma vonatkozik minden területre, hogy elkerülhetők legyenek a felhasználókra káros ütközések *(például, ha egyszerre van myTV és myFavoriteTVShows)* és hogy biztosítva legyen a konzisztens architektúra például a biztonsági modell és az adatkezelés esetében. A HailStorm bővítéseit a Microsoft speciális szakterületenként végzi.

A bevezetés

A HailStorm szolgáltatások kezdőkészlete várhatóan 2001 végén kerül fejlesztői béta állapotba, és 2002-ben jelenik meg a végleges változat. A további névterek és szolgáltatások a Microsoft Open Process során kerülhetnek online használatra. A HailStorm-ra kiegészítő szolgáltatások és bővítmények épülhetnek, ha az alapinfrastruktúra már működik.

A végpontok

A Microsoft támogatási programok segítségével jelenleg azon dolgozik, hogy minél több más gyártótól származó végpont legyen elérhető a HailStorm szolgáltatásokhoz.

A Microsoft természetesen biztosítani kívánja, hogy minden Microsoft eszköz jó HailStorm végpontként működjön. Ez azt jelenti, hogy az összes Microsoft alkalmazás *(a Microsoft Office-tól a Microsoft játékokig)* támogatni fogja a HailStormot. A szolgáltatások *(pl. a MSN és a bCentral™ kisvállalati portál)* HailStorm végpontként működnek, és a Microsoft szoftverrel működtetett eszközök, így a Xbox™ videojáték konzol, a Pocket PC és a „Stinger", a Microsoft intelligens telefonszoftver platformja is potenciális HailStorm végpontokká válnak. A Microsoft operációs rendszerei, például a Windows XP és a Windows CE önmaguk is HailStorm végpontokká válnak, és lehetővé teszik azt is, hogy a fejlesztők egyszerűen hozhassanak létre HailStorm alapú alkalmazásokat.

A Windows XP integrálja a Windows hitelesítési rendszert a Passport hitelesítési rendszerrel, így ha a felhasználó egyszer belép a Windows XP-be, ezzel a Passport-ba is belép, így további bejelentkezés nélkül is képessé válik a HailStorm szolgáltatások használatára. A Windows XP támogatja a programozott értesítéseket, így a HailStorm myNotifications szolgáltatás felhasználói egyszerűen megkaphatják a figyelmeztetéseiket a Windows XP-t futtató PC-n.

(Úgy érzem, erre a témára még vissza kell térnünk. Az összes valamirevaló anyagot felhasználtuk a cikkhez, de ez így is igencsak szűkszavú. Miközben olvastam, az alábbi kérdések merültek fel bennem:

1) *Fizikailag HOL is tárolódnak akkor az adatok? Kint a weben? De HOL?*
2) *Milyen titkosítással?*
3) *Hogyan működik az időzített jogosultság? Ha én például egy bankkártyaszámot az engedélyezett időm alatt kimásolok egy textfájlba, akkor az idő lejártával a leloptam.txt megsemmisíti önmagát? :)*
4) *Lehet-e piramist építeni a Hailstormokból, hasonlóan az egymásra épülő Certificate Serverekhez?*
5) *Ki fizet a szolgáltatásért, és mikor? Az adat tulajdonosa? Vagy a felhasználója?*

FM - a szerk.)

XML-gessünk *(II. rész)*

Bevezetés

Két hónappal ezelőtti számunkban elindultunk az XML-hez kapcsolódó technológiák felidézésével. Ebben a számban utánajárunk az XML dokumentumok strukturális leírására szolgáló sémáknak, majd rátérünk az Internet Explorer XML képességeire. Ezen belül elmélyedünk az XML dokumentumok formázásában az XSL és a CSS alkalmazására. Az itt leírt alapok megteremtik azt a tudást, amely segítségével már ügyfél és kiszolgáló oldalon is tudunk XML dokumentum alapú weboldalakat építeni – melynek részleteit a következő számban olvashatják.

XML Stratégia

Az XML technológia vívmányai, akár tetszik, akár nem, a nyakunkon vannak, és mielőbb alkalmaznunk kell tudni őket. Havonta 4-5 oldalban alig lehet valamit átadni ezekből a forradalmi módszerekből, mert ha alapozással akarnánk kezdeni, akkor minimum a következő 10 cikk csak fogalmak tisztázásával menne el, és még mindig nem tudnánk használni az XML-t, csak értenénk, hogy mit takarnak azok az X-szel kezdődő néhány betűs rövidítések. Emiatt a cikksorozatomban hibrid stratégiát vezetek be. Minden szám elején lesz körülbelül egy oldalnyi bevezető, amelynek célja néhány fogalom tisztázása az XML háza tájáról. A maradék oldalakat mindig valamilyen kézzelfogható, gyakorlatban azonnal alkalmazható példakódokkal fogom megtölteni. A két rész nem feltétlen fog szorosan összetartozni, de egy közös kapocs lesz közöttük: mindkettő XML-ről fog szólni. Emiatt előfordulhat, hogy lesz a cikkekben egy-két fogalom, amit előtte nem definiáltam, de ilyenkor mindig hivatkozni fogok egy-egy URL-re, ahol a fogalomról részletes információ kapható. Vágjuk hát bele!

Sémaleírás

Adott az XML, mint általános, egyszerű és hatékony adatleíró nyelv. Önleíró, azaz egy XML dokumentumot minden külső információ nélkül értelmezni lehet, be lehet járni a benne található csomópontokat, kiolvasva az elemek értékét és attribútumait. Azonban irányított kommunikáció esetén, akár gépek közötti információcserénél, akár gép-ember kapcsolatban sokszor szabályokat kell alkotnunk a küldendő illetve fogadandó XML dokumentum szerkezetére vonatkozóan. Ekkor már csak azokat az XML dokumentumokat fogadjuk el érvényesnek, amelyek szerkezete megfelel a formális leírásban szereplő feltételeknek. Például egy megrendelést leíró XML dokumentumra valószínűleg kikötnénk, hogy benne kell legyen a megrendelő neve, címe, adószáma satöbbi. Ha a kapott megrendelés.xml-ben nem szerepel minden kívánatos adat, akkor visszadobjuk a megrendelést, mert nem érvényes.

Az XML dokumentum szerkezetének, más néven sémájának leírására több módszer is a rendelkezésünkre áll, melyek fokozatosan, a használat során fejlődtek ki. Nézzük meg milyen hárombetűs rövidítések segítenek bennünket az XML dokumentumok sémájának leírásában:

- DTD, Document Type Definition: [1] a legelső eszköz volt az XML dokumentumok struktúrájának leírására. Több sebből is vérzik, de a legnagyobb problémája: nem XML formátumú. Ha már egyszer kitalálták az XML-t, nagyon gyors és hatékony feldolgozót *(parser)* írtak hozzá, akkor miért nem lehet az elemzendő XML dokumentum sémáját is XML-ben leírni? Emellett nincsenek benne adattípusok, mindent csak szövegként lehet definiálni, valamint nem támogatja a névtereket, amely nélkül szó sem lehet több forrásból összefűzött adatok ellenőrzésére. Ezek igen súlyos hiányosságok, amely miatt a DTD hamarosan ki fog halni, de egyenlőre sajnos egyedül ez az egyetlen, végleges, elfogadott sémaleíró nyelv.
- XDR, XML Data-Reduced: [2] a nagyreményű Microsoft DTD utód - volt. A Microsoft gyorsan lemondott a DTD használatáról, és gyors ütemben belekezdett egy XML formátumú sémaleíró nyelv kidolgozásába, amelyet a Word Wide Web konzorciumnak is elküldött szabványosításra. Megszületett az XDR. Amellett, hogy XML formátumú még jóval flexibilisebb is, mint a DTD. A DTD-ben leírt struktúrának maradéktalanul meg kell felelni egy XML dokumentumnak. Az XDR is tud ilyen szigorú lenni, de emellett elő lehet azt is írni, hogy az ellenőrizendő dokumentum egyes részeiben lehetnek további elemek is, amelyet a séma nem ír le. Például egy személyről szóló XML adatlapban kötelezővé tesszük a név, születési dátum és az anyja neve elemeket, de ezen felül megengedjük, hogy egy buzgó gazdi például a kutyája nevét és haja színét is beleszerkessze a dokumentumba. A hangsúly nem azon van, hogy előírhatunk opcionális elemeket, hanem azon hogy megengedhetünk olyan elemeket is, amelyekről a séma készítésekor még nem is tudtuk, hogy lesznek. Ehhez kapcsolódó szolgáltatás, hogy XDR segítségével le lehet szabályozni a dokumentum egy részét is, nemcsak a teljes egészet. DTD-vel természetesen csak az egész dokumentumra lehet szabályokat definiálni.

  Emellett az XDR bővíthető, azaz az igények megváltozásakor nem kell a sémát kidobni, csak egy másik névtér bevezetésével kiegészíteni a meglevőt. Utolsó, de nagyon fontos szolgáltatás az XDR-ben, hogy az elemek és attribútumoknak meg lehet adni a típusát *(egész szám, dátum satöbbi)*. A DTD-ben minden szövegként van deklarálva. A legtöbb, a közeli múltban fejlesztett Microsoft termék XDR-t használ sémaleírásra. Azonban már a kapuban dörömböl az
- XSD, XML Schema Definition [3], amely a cikk írásának pillanatában a leginkább aktuális sémaleíró nyelv. A Biztalk Server, illetve a .NET XML osztályok már tudják – tudni fogják ezt a sémaleírást is kezelni *(természete-*

sen az XDR mellett). A Visual Studio 7 egyik alapszolgáltatása az XSD sémák grafikus szerkesztése, konverziója XDR-ből XSD-be, XML dokumentumból XSD generálása stb. *(A Visual Studio 7 illetve a .NET stratégia fejlesztői szemmel nézve több, mint szenzációs. Újságunkban is rövidesen megkezdjük az alapok lefektetését.)*

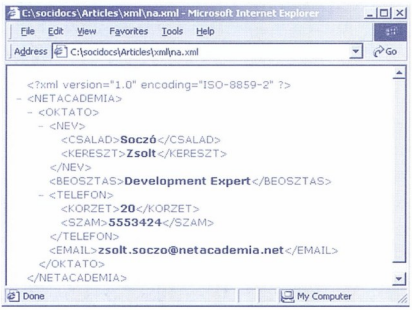
Az XML és a Web

A Microsoft nagy erőkkel ráállt az XML technológiában rejlő lehetőségekre, és ez az elkötelezettség meglátszik az utóbbi 3 év összes Microsoft termékén is. Az összes terméket áthatja az XML, legyen szó konfigurációs állományokról *(.NET-ben minden konfigurációs adat XML fájlokban van, hello registry – végre!)*, kiszolgáló termékekről, vagy az Internet Explorer-ről. Még azt sem tudtuk mi az az XML, de az IE4 már támogatta, persze az akkori szabványnak megfelelően még eléggé gyerekcipőben, de már működtek benne nagyon hasznos funkciók. A Microsoft Internet Explorer 5 már igen hatékony XML támogatást tartalmaz, amely segítségével az információk megjelenítéshez kapcsolódó funkciók jelentős részét ügyféloldalon meg lehet oldani, csökkentve ezzel a szerver terhelését. Ezt mondják Amerikában, ahol tényleg agyonterhelik a webkiszolgálókat a szörfözők. Ezzel szemben nálunk, ahol örül egy webszájt, ha van napi 5000 találata, inkább a lassú modemes kapcsolat miatti lecsökkent szerverhez fordulás az, ami miatt érdemes kihasználni az IE-ben rejlő lehetőségeket.

Persze általában az Internetes környezetben nem köthetjük ki a látogatóknak, hogy használjanak IE 5.5SP1-et, mert csak az tudja kihasználni az általunk megírt funkciót. Azonban heterogén böngészőkörnyezetben sem kell lemondani az XML kínálta előnyökről, csak buta böngésző esetén az XML-lel kapcsolatos funkciókat a kiszolgálón kell implementálni, és csak a kész HTML kódot leküldeni a böngészőknek. Okos böngészőnek megy az XML, butának a HTML. Az okos keveset fordul a webszerverhez, a buta sokat.

IE és XML, a két jó barát

Csapjunk bele az IE5 XML szolgáltatásaiba. Ha egy XML állományt adunk meg a böngészőnek, akkor ő azt közvetlenül megjeleníti.



Egy közönséges XML állomány az IE5-ben [4]

Figyeljük meg, az <?xml version="1.0" encoding="ISO-8859-2" ?> sort! Egyrészt leírja, hogy a dokumentum az XML1.0-s szabványra épül **[5]**, valamint azt, hogy a benne található karaktereket az ISO-8859-2-es kódtábla alapján kell értelmezni. Ha ezt nem tesszük meg, az összes értelmezőprogram, beleértve azt is, ami az IE mögött is van, kiborul az első ékezetes karakternél. Az IE5.5 így dohog:

```
An Invalid character was found in text content.
Line 6, Position 19
        <CSALAD>Socz?SALAD>
-------------------^
```

Jó, de ez így minden, csak nem felhasználóbarát. Alakítsuk át ezt HTML-é, amelyen keresztül már szépen, formázottan láttathatjuk az információkat. Két módszer kínálkozik erre, illetve egy harmadik, ami az első kettő keveréke.

1) Írjunk egy Cascaded Style Sheet-et *(CSS)* **[6]**, amiben leírjuk, hogy melyik elem hogyan jelenjen meg a böngészőben. Ez működő, de igen erősen behatárolt módszer, mert csak nagyon korlátozott lehetőségeink vannak a cél HTML dokumentum formátumának meghatározásában. Egyszerűen azért, mert a CSS nem erre van kitalálva. Arra nagyon jó, hogy leírjuk vele egy adott HTML elem megjelenését, de itt XML elemeket kellene transzformálni valamilyen, általunk definiált HTML formátumba, és erre a feladatra nem alkalmas a CSS. Viszont, erre van kitalálva az
2) Extensible Style Language, röviden XSL **[7]**. Az XSL pont arra van kitalálva, hogy tetszőlegesen bonyolult módon jelenítsünk meg egy XML dokumentumot, ami nevezhetnénk adatforrásnak is, kihangsúlyozva, hogy az nem hordoz semmiféle megjelenítési információt. XSL segítségével könnyedén megváltoztathatjuk az adatok, elemek eredeti sorrendjét, megszűrhetjük a benne található adatokat, ismétlődéseket vihetünk be, stb. Egyszóval XSL segítségével olyan HTML kimenetet generálunk, amilyet csak szeretnénk. Tulajdonképpen az eredeti XML dokumentumot bármi mássá is átkonvertálhatjuk vele, például egy másik XML dialektussá, azaz egy más struktúrájú XML dokumentummá, vagy éppen szöveges állománnyá! Amivé csak akarjuk. Nem véletlenül fejlődött ki az XSLT **[8]**, az XSL Transformations szabvány az XSL-ből. Az XSL esetén a kiinduló cél az XML adatok meg jelenítésének szabályozása volt - alternatíva a CSS mellé. Az XSLT már kifejezetten annak fényében készült, hogy az XML dokumentumokat valamilyen más formátumúra akarjuk átalakítani,transzformálni.

Van XSLT-nk, amivel pompásan át lehet alakítani a megjelenítendő XML dokumentumot HTML-é. Azonban egy bonyolultabb XSLT transzformáció igen áttekinthetetlen tud lenni, és ha még ebbe képeket, formázó parancsokat, stílusokat is beleszövünk, akkor egy olyan XML-HTML-formázó_parancsok spagettit kapunk, ami karbantarthatatlan lesz. Ekkor jön a nyerő ötlet. XSLT-vel transzformáljuk át a kívánatos HTML-é a forrás XML-t, és a formázásokra csak hivatkozunk benne, de ne fejtsük ki őket az XSLT-ben. Ehelyett az összes formázó utasítást rakjuk ki CSS-be, és máris ötvöztük az XSLT flexibilitását a CSS szuper formázási képességeivel. Mindeközben a grafikus bármikor nyugodtan átgyúrhatja a CSS-t, nem kell szembesülnie *(és elszaladnia :)* az XSLT-vel.

Az XSLT közbelép

Most, hogy kipletykáltuk a grafikust, készítsünk egy XSL dokumentumot, és adjuk meg a forrás XML-ünknek, hogy amikor a böngésző megjeleníti, használja az XSL-ünket. Ehhez az XML forrás elején megadjuk az XSL fájl elérését *(na.xsl)*:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet type="text/xsl" href="na.xsl"?>
<NETACADEMIA>
...
```

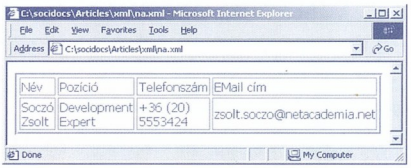
A böngésző az XML állomány betöltésekor megnézi, hogy van-e XSL hozzárendelve. Ha van, akkor letölti azt is, és végrehajtja az abban leírt műveleteket, majd megjeleníti a végeredményt. Nézzünk egy egyszerű transzformációt:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet xmlns:xsl="uri:xsl">
 <xsl:template match="/">                          ← 1
   <HTML>
   <HEAD>
   <link rel="stylesheet"
    type="text/css" href="na.css" />               ← 0
   </HEAD>
     <BODY>
       <TABLE CLASS="OktatokTabla">
         <TR>
           <TD>Név</TD>
           <TD>Pozíció</TD>
           <TD>Telefonszám</TD>
           <TD>EMail cím</TD>
         </TR>
         <xsl:for-each                             ← 3
              select="NETACADEMIA/OKTATÓ">
           <TR>
             <xsl:apply-templates/>                ← 4
           </TR>
         </xsl:for-each>
       </TABLE>
     </BODY>
   </HTML>
 </xsl:template>                                   ← 2

 <xsl:template match="NEV">                        ← 5
   <TD><xsl:value-of select="CSALAD"/>
   <xsl:value-of select="KERESZT"/></TD>
 </xsl:template>

 <xsl:template match="BEOSZTAS">                   ← 6
   <TD><xsl:value-of select="."/></TD>
 </xsl:template>

 <xsl:template match="TELEFON">                    ← 7
   <TD>+36 (<xsl:value-of select="KORZET"/>)
   <xsl:value-of select="SZAM"/></TD>              ← 9
 </xsl:template>

 <xsl:template match="EMAIL">                      ← 8
   <TD><xsl:value-of select="."/></TD>
 </xsl:template>

</xsl:stylesheet>
```

Hogyan hajtja vége a böngésző az imént felvázolt transzformációt? A ① azt jelzi, hogy a match-ben megjelölt elemre, ami jelen esetben a gyökér elem *(<NETACADEMIA>)* hajtsa végre a záró tagig ② leírt műveleteket. Mit jelent a végrehajtani? A nem XSL tagokat egyszerűen bemásolja a kimenetbe, így egészen a ③ pontig minden bekerül a végeredménybe. Jól látható, hogy itt egy HTML táblázatot hoztam létre, egy fejléccel. Az XML forrásból származó tartalom generálása a ③ körül történik. Az xsl:for-each arra utasítja az XSL értelmezőt, hogy a select után megadott XPath **[9]** kifejezés által kiválasztott elemeken menjen végig, és mindegyikre hajtsa végre az utasítás törzsében leírt parancsokat. A NETACADEMIA/OKTATO XPath kifejezés az összes, a NETACADEMIA elem alatt létrehozott OKTATO elemekre ad egyezést, azokat választja ki *(jelen esetben csak egyet, de ha több ismétlődő OKTATO elem lenne a forrásban, akkor mindegyiket)*. Mi történik a kiválasztott elemekkel? Generálódik hozzájuk egy <TR> </TR> páros, ami HTML-ben a táblázat egy sorát írja le. És a lényeg, a forrásadatok a ④ kifejezés jóvoltából kerülnek bele a kimenetbe. Az xsl:apply-templates elem azt jelenti, hogy az éppen feldolgozás alatt álló elemre nézze meg, hogy van-e egyező sablon létrehozva. Ha van, akkor azt végrehajtja, és annak a kimenete beleszövődik a ④ kifejezés helyébe. Esetünkben a ③ paranccsal lefúrtunk az XML forrásunk <OKTATO> elemében, így az ⑤-⑧ sablonok már erről a szintről indulnak, azaz a match attribútumokban megadott XPath kifejezések kiegészülnek így: /NETACADEMIA/OKTATO/TELEFON, mondjuk a ⑦ sablon esetén. Másképpen fogalmazva a forrás XML-ben a /NETACADEMIA/OKTATO/TELEFON elérési úttal jelzett elem elérésekor az XSL processzor meghívja a ⑦ sablont. A sablonon belül az xsl:value-of elem kiválasztja a select="KORZET" által kijelölt elem értékét, azaz a <KORZET></KORZET> közötti szöveget ⑨. Néhány egyéb formázó karakter és a SZAM elem értékének felolvasása után a sablon véget ér. Ezután az XSL motor megnézi, hogy van-e még más sablon, ami egyezést mutat valamelyik elemmel. Esetünkben mind a négy sablon szóhoz jut minden egyes oktatóhoz.

Miután ③ végigment az összes elemen, kiíródik a táblázat vége, és véget ér a végrehajtás a ② ponton.

Aki ezt az utat lelkiismeretesen végigkövette, az megérdemli, hogy megnézze a végeredményt:



Mitől lett ez ilyen díszes és tarka *(mi lesz ebből a szürke újságban?! :)* ? Hát attól, hogy az XSL által generált HTML

kimenet kapott még egy CSS-t is a ®-val megjelölt sorban. Anélkül csak egy fehér táblázatot látnánk. A teljesség kedvéért itt az na.css tartalma:

```
<STYLE>
BODY
{
    BACKGROUND-COLOR: snow
}
TABLE.OktatokTabla
{
    BORDER-RIGHT: groove;
    BORDER-TOP: groove;
    BORDER-LEFT: groove;
    COLOR: mediumblue;
    BORDER-BOTTOM: groove;
    FONT-FAMILY: Tahoma;
    BACKGROUND-COLOR: lightgoldenrodyellow
}
TD
{
    BORDER-RIGHT: thin groove;
    BORDER-TOP: thin groove;
    BORDER-LEFT: thin groove;
    BORDER-BOTTOM: thin groove
}
</STYLE>
```

Az előbbi XSL transzformáció egy nagyon egyszerű példa volt, ennek ellenére már ez is elég átláthatatlan és bonyolult lett. Az XML technológia bonyodalmaiba akkor kóstolunk bele igazán, amikor az XSLT nyelvet kezdjük el használni. A gond az, hogy az XSLT nem procedurális nyelv, hanem alapvetően sablonok egyeztetésén alapul. Ez a fajta gondolkodás igen szokatlan egy Basic vagy C nyelven felnőtt programozónak.
További nehézség, hogy mit tehetünk akkor, ha egy jó bonyolult XSLT transzformáció nem úgy működik, ahogy azt elvárjuk tőle? Nem formátumbeli hibákra gondolok, azt kiszűrik a transzformáló komponensek. Akkor leszünk meleg helyzetben, ha „csak" logikai hiba van az XSLT-nk működésében, és nem azt a transzformációt hajtja végre, amit mi módoltunk ki. Ilyenkor jönnek képbe a nyomkövető vagy debugger programok, amelyek segítségével lépésenként lehet végrehajtani a programunkat, ebben az esetben az XSL transzformációt. Ez egy kemény feladat, de szerencsére az Activestate cégnek - amely elsősorban a Win32-es Perl eszközeiről híres – megjelent a Visual XSLT 1.0 **[10]** programja, ami egy plug-in a Visual Studio.NET 7-hez, és amelynek segítségével szerkeszthetjük és nyomkövethetjük az XSL transzformációinkat. Szenzációs termék! A debugger ablakban láthatjuk a transzformáló XSLT-t, benne az éppen végrehajtás alatt álló XSL paranccsal. A Watch ablakban láthatjuk a forrás XML éppen feldolgozás alatt álló elemeit, az Output window-ban pedig a transzformált végeredményt.
A program jelenleg Beta 1 állapotban van, hasonlóan a Visual Studio.NET 7-hez. Ennek ellenére aki komolyan, a jelenben és a jövőben élve szeretne XML-lel foglalkozni, mindenképpen szerezzen be egy példányt a Visual Studio.NET 7-ből, és töltse le a Visual XSLT 1.0-t. Mindkettő Must Have!

Újdonságok

Az XML-t alkalmazó alkalmazásokat programozók legalapvetőbb fegyvere a parser, XML értelmező, amely az XML Document Object Model-en keresztül programozottan elérhetővé teszi az XML dokumentumok belső szerkezetét. A parser-ek lehetővé teszik a felolvasandó XML dokumentumok ellenőrzését egy megadott sémával szemben, amely jelen pillanatban – Microsoft-os platformon – DTD és XDR-el történhet. Az aktuális parser a Microsoft-tól a 3-as verziónál jár, ez az, amit éles környezetben is lehet használni.
Azonban nincs megállás. Áprilisban a Microsoft kiadta a Microsoft XML Parser *(MSXML)* 4.0 Technology Preview Release-t. Ezt még nem ajánlják produkciós környezetbe, de mindenképpen érdemes tanulmányozni. Miért? Azért, mert végre támogatja az XSD-t! A parser-ben implementált séma validáló kód a Wide Web Consortium *(W3C)* XML Schema, 2001. március 30-ikai ajánlásában leírt módon működik, azaz olyan friss, hogy még meleg! Mivel ez még mindig nem a végleges XSD szabvány, így várható, hogy az valamennyit még változni fog a végleges kiadásig. Az XML parser 4 természetesen követni fogja a változásokat, újabb és újabb kiadásokkal.
Mi van még a csomagban az XSD mellett? Megjelent benne egy új objektum, az MXHTMLWriter, amely segítségével a SAX API által felolvasott XML dokumentumból kapott adatfolyamot lehet HTML-lé transzformálni. Ez egy alternatív megoldás lesz az XML --> XML DOM --> XSLT --> HTML feldolgozásra, kifejezetten nagyteljesítményű webalkalmazásokhoz.
Az új verziót fel lehet telepíteni olyan gépre is, amin már van MSXML3-as parser, így az alkalmazáaok a verzió függő ProgID-k használatával tudják használni mind a 3-as, mind a 4-es változatot. Arra viszont vigyázzunk, hogy ne rakjuk fel produkciós szerverre, mert a verziófüggetlen objektumlétrehozást alkalmazó programok az új verzióból fognak egy példányt kapni, ami meglepetéseket okozhat.

Zárszó

Aki lelkiismeretesen végiglépkedett a példa XSL-en és XML-en, annak már van fogalma az XML alapú fejlesztések alapköveiről, látja a jéghegy csúcsát. De az igazi csemege még a víz alatt van. A következő számban a transzformációt már nem fogjuk az Internet Explorer-re bízni, hanem a kezünkbe vesszük az XML DOM-ot, és azzal hajttatjuk végre az XSL-ünket, mind kiszolgáló oldalon, mind ügyfél oldalon. És ha már ott vannak az adatok a böngészőben, miért ne lehetne rögtön szűrni illetve sorbarendezni őket, anélkül, hogy a szerverhez kellene fordulni.? Miért is ne? A júniusi számban megmutatom hogyan.

Soczó Zsolt MCSE, MCSD, MCDBA
Zsolt.Soczo@netacademia.net

A cikkben szereplő URL-ek:

[1]: DTD szabvány http://www.oasis-open.org/cover
[2]: XDR szabvány http://www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm
[3]: XSD http://www.w3.org/XML/Schema
[4]: Példakódok http://technet.netacademia.net/feladatok/xml
[5]: XML1.0 szabvány, magyarázattal! http://www.xml.com/axml/testaxml.htm
[6]: CSS szabvány http://www.w3.org/Style/CSS
[7]: XSL szabvány http://www.w3.org/Style/XSL
[8]: XSLT szabvány http://www.w3.org/TR/xslt
[9]: XPath szabvány http://www.w3.org/TR/xpath
[10]: Visual XSLT 1.0 Beta 1
http://www.activestate.com/ASPN/Downloads/VisualXSLT

Transact SQL *(VII. rész)*

Az árvíztűrő tükörfúrógép

Bevezetés

Egy hónap kihagyás után újra itt a Transact SQL sorozat, töretlen lendülettel! E havi részünkben azokkal a nyelvi finomságokkal és SQL Server 2000 újdonságokkal foglalkozom, amelyek nem kaptak akkora hírverést mint mondjuk az XML támogatás, de nagyon fontos apróságok, amelyek különösen magyar nyelvű szövegeket kezelő programok írásakor nagyon fontosak. Járjunk utána az ő ű betűk misztériumának!

Az alapprobléma

Ki ne bosszankodott volna azon, hogy egy lementett *(backup)*, magyar nyelvi beállításokkal működő adatbázist nem lehetett visszaállítani egy angol nyelvi beállításokkal telepített másik SQL Server 7-re, a nyelvi beállítások különbözősége miatt? Ennek az az egyszerű oka volt, hogy az SQL Server 7-ben a nyelvi beállítás globális, kiszolgálószintű volt, amely az összes adatbázisra, és azon belül az összes objektumra, adatra, oszlopra vonatkozott. Ez az információ benne volt a felhasználói adatbázisokban is, így visszaállításkor az SQL Server 7 észrevette a nyelvütközést, és nem engedte a visszaállítást.

De miért van egyáltalán szükség az egész nyelvi hókuszpókuszra? Miért volt ez annyira beleépülve a kiszolgálóba? A probléma alapja a nemzetek nyelveinek különbözőségében keresendő.

Ha egy karaktert egy bájton ábrázolunk, akkor 256 féle karaktert tudunk leírni. Ez bőven elég az angol abc leírásához, még sok egyéb extra karakter *(+!%...)* is belefér. Azonban a bőségből gyorsan „szűkség" lesz, ha elkezdjük felmérni és megpróbáljuk ábrázolni az összes nemzet valamennyi karakterét. 256 hely erre nem elég. Ezt feloldandó minden ország, helyesebben minden országcsoport, amelynek azonos karaktereik vannak az abc-ben kap egy karakter kódtáblát, kódlapot *(character set)*, ami az ő nyelveikben található betűket rendeli hozzá a 0...255-ös tartományhoz. A nem ékezetes betűk általában ugyanarra a kódra vannak leképezve a legtöbb kódlapban, így például a kis „a" betű a 97. pozícióra. Ellenőrzés:

```
-- Nyelvi beállítás: Latin1_General
-- (nyugati nyelvek)
SELECT ASCII('a')
97
```

Az ékezetes betűk helye már legtöbbször nyelvfüggő. Bizonyára mindenkinek ismerős a magyar „ő" és az „ű" betűk problémája. Ha egy számítógépen a nyelvi beállítások nem megfelelőek, akkor mindig ezzel a két karakterrel szokott baj lenni. Nézzünk csak ennek a körmére! Latin 1 *(nyugati nyelvek)* kódkészletet használva kérdezzük le az „ő" betű kódját, azaz azt, hogy ez a betű a Latin 1-es kódlap melyik pozícióján van értelmezve?

```
--Adatbázis beállítás: Latin1_General
SELECT ASCII('ő')
111
```

Ez egy picit gyanús, mert az ékezetes betűk általában a 128 feletti pozíción foglalnak helyet. Tegyünk egy ellenpróbát, az így kapott karakterkódot alakítassuk vissza az adott kódlapon érvényes karakterré:

```
SELECT CHAR(ASCII('ő'))
o !!!
```

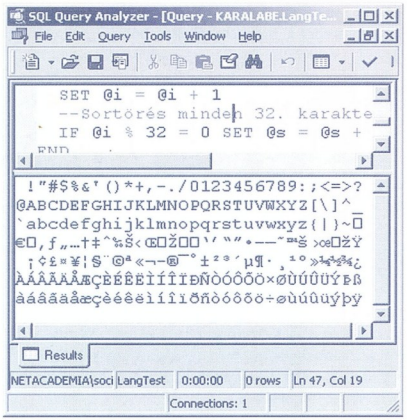
Hoppá, a kis „ő" betűnkből „o" lett! Nem véletlenül volt gyanús a 111-es kód. Az valójában az „o" betű kódja:

```
SELECT ASCII('o')
111
```

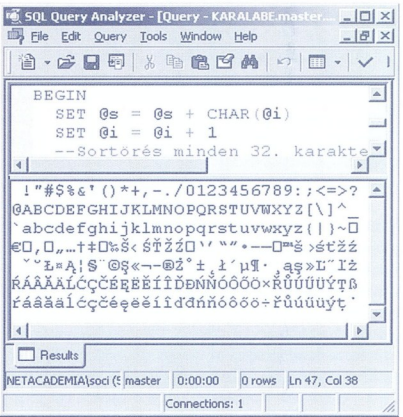
Mi volt itt a gond? Az, hogy a Latin1-es kódlapban, nincs helye a mi „ő" betűnknek! Nézzük csak meg milyen karaktereket ismer a Latin1:

```
DECLARE @i int
DECLARE @s varchar(2000)
SET @s = ''
SET @i = 32
WHILE @i < 256
BEGIN
   SET @s = @s + CHAR(@i)
   SET @i = @i + 1
   --Sortörés minden 32. karakter után
   IF @i % 32 = 0 SET @s = @s + CHAR(13)
END
PRINT @s
```

A lekérdezés kimenetét a nyomdai utómunkák okozta lehetséges karakter konverziók, torzulások miatt grafikusan mutatom meg. Hiába, a karakterkonverzió nem csak az SQL Serverben problémás pont. :-)



Karakterek a Latin1 kódlapban. Hová lettek az ő ű betűink?



Karakterek a magyar kódlapban. Itt már a helyükön vannak az ékezetes betűink.

Jól látható, hogy kétvesszős „ő" nincs ebben a készletben, csak kalapos, meg hullámos tetejű. Azaz, ha szeretnénk használni normális magyar karaktereket, akkor át kell kapcsolnunk az adatbázisunkat egy olyan kódkészletre, amely ismeri a rendes betűinket. Praktikusan magyarra. Lássuk, ekkor jól működnek-e a konverziók:

```
--Átkapcsolunk magyarra
ALTER DATABASE LangTest
COLLATE Hungarian_CI_AI

--Teszt 1: karakterből ASCII kód
SELECT ASCII('ó')
245

--Teszt 2: a kapott ASCII kódból karakter
SELECT CHAR(ASCII('ó'))
ó
```

Remekül megy! Azaz leszögezhetjük, hogy a magyar karakterek sikeres kezeléséhez vezető út első lépése a megfelelő karakterkészlet beállítása az adatbázisra.
Az összehasonlítás kedvéért nézzük még meg a magyar kódkészletet is:

Amikor két számítógép, vagy két adatbázis között mozgatunk nem UNICODE szöveges adatokat, akkor a célhelyen kódkonverzió történik. Ez a konverzió a célgép operációs rendszerének kódtábláiban van definiálva. Abban az esetben, ha a forrásszövegben van olyan karakter, ami a célszerver *(adatbázis)* kódtáblájában nincs definiálva, a konverzió sokszor az adatok megváltozásával jár. Ennek tipikus megjelenése, amikor egy adatbázis ügyfélalkalmazás nem jeleníti meg az „őű" betűket, hanem „ou"-t ír ki helyette. Például a képen látható Enterprise Manager ablakban megjelenítettem egy tábla tartalmát, amelyben helyesen, magyar kódlappal vannak eltárolva a karakterek. Csakhogy az Enterprise Managert futtató Windows 2000 System Local-je English-re volt állítva, így az ügyféladatbázis meghajtóprogramja átfordítja a kiszolgálóról letöltődő karaktereket angol kódlapra – megkérdőjelezhető sikerrel:

Hová lettek az őű betűűűűk?

Ha a System Local-t magyarra állítom, akkor mindjárt jól működnek az ékezetes karakterek is:

Ügyfél Windows 2000 System Locale: magyar. Megjöttek az őű betűk.

Nem állítom, hogy át kell állítani mind az SQL Server mind az ügyfél munkaállomások System Local-jét magyarra ahhoz, hogy ne történjék adatvesztés a konverzió során, de ha más ellenérv nem szól ellene, akkor érdemes ily módon beállítani őket. Ez nem feltétlenül szükséges, viszont elégséges feltétele a fájdalommentes, magyar szövegekkel operáló adatbázismüveleteknek.

Adatbázis collation

Lehet, hogy már sikerült meggyőzni a cégvezetést, és minden gépen magyar beállításokkal működik az NT vagy a Windows 2000. Biztos lehetek benne, hogy ezek után minden rendben lesz az ékezetekkel? Természetesen – nem! Az SQL Server 2000-ben adatbázisonként is megadható a nyelvi beállítás, így előfordulhat, hogy mégis gond lesz az adatmozgatások során. Nézzünk egy példát, ami jobban megvilágítja a jelenséget! Hozzunk létre két adatbázist, az egyiket állítsuk magyar collation-űre, a másikat Latin1-re. Szúrjuk be a cikk címében szereplő tesztszöveget a magyarra állított adatbázis egy táblájába, majd egy közönséges INSERT ... SELECT párossal másoljuk át a magyarba beszúrt sort az angol adatbázis táblájába. Figyeljük meg, mikor történik konverzió, és mikor nem! A példában a System Local magyar mind a kiszolgálón, mind a munkaállomáson *(egy gépen vannak :)*, tehát az nem okozhat problémát.

```
USE master
--Angol (Latin1) COLLATION-ű adatbázis létrehozása
CREATE DATABASE LangTestEng
ON
(
  NAME='LangTestEngPrimary',
  FILENAME='c:\temp\langtesteng.mdf'
) COLLATE Latin1_General_CI_AS

--Magyar COLLATION-ű adatbázis létrehozása
CREATE DATABASE LangTestHun
ON
(
  NAME='LangTestHunPrimary',
  FILENAME='c:\temp\langtesthun.mdf'
) COLLATE Hungarian_CI_AS

--Teszttáblák létrehozása mindkét adatbázisban
USE LangTestHun

CREATE TABLE Dumak (
  Szoveg varchar(50) )

USE LangTestEng

CREATE TABLE Dumak (
  Szoveg varchar(50) )

USE LangTestHun

--Beszúrás a "magyar" adatbázis táblájába
INSERT Dumak (Szoveg) VALUES (
'1. árvíztűrő tükörfúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP'
)

--Jól sikerült?
SELECT Szoveg FROM Dumak
```

Kimenet:

```
Szoveg
--------------------------------------------------
1. árvíztűrő tükörfúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP
```

Azaz a magyar beállításokkal rendelkező táblába sikerült helyesen beszúrni a szöveget. Mehet a másolás.

```
--Másoljuk át a magyarba beszúrt sort
--az angol collation-ű adatbázisba.
INSERT LangTestEng..Dumak SELECT Szoveg FROM Dumak

--Ellenőrizzük le, mi jelent meg benne!
SELECT Szoveg FROM LangTestEng..Dumak
```

Az másolatban bizony leestek a kedvenc kétvesszős magyar karaktereink:

```
Szoveg
--------------------------------------------------
1. árvízturo tükörfúrógép ÁRVÍZTURO TÜKÖRFÚRÓGÉP
```

Hasonlóan kiábrándító az eredmény, ha közvetlenül az angol nyelvű adatbázis táblájába próbálunk adatokat csempészni:

```
USE LangTestEng

-- Beszúrás közvetlenül az angol
-- adatbázis táblájába
INSERT Dumak (Szoveg) VALUES ('2. árvíztűrő tükörfúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP')

--Jól sikerült?
SELECT Szoveg FROM Dumak
```

Kimenet:

```
Szoveg
--------------------------------------------------
1. árvízturo tükörfúrógép ÁRVÍZTURO TÜKÖRFÚRÓGÉP
2. árvízturo tükörfúrógép ÁRVÍZTURO TÜKÖRFÚRÓGÉP
```

Oszlop collation

Ha létrehozunk egy táblát egy adatbázisban, akkor annak a szöveges oszlopai öröklik a szülő adatbázis collation-jét. Ha ez nekünk nem megfelelő, akkor a tábla létrehozásakor akár oszloponként megadhatunk különböző collation-öket. Így előállhat az a helyzet, hogy egy angol collation-ű adatbázisban létrehozunk egy magyar nyelvű oszloppal felvértezett táblát. Például:

```
USE LangTestEng

CREATE TABLE Dumak (
   Szoveg varchar(50) COLLATE Hungarian_CI_AS )
```

Ha a korábbi példában látott módon a magyarból másolunk az angolba, akkor most helyesen fognak átmenni az ékezetek, mert a céloszlop collation-je megegyezik a forrásával, így nincs szükség konverzióra.
Ami viszont nagyon érdekes, hogy a közvetlenül az angol nyelvű adatbázisba beszúrt adatokról elvesznek az ékezetek, annak ellenére, hogy a céloszlop magyar nyelvű:

```
USE LangTestEng
INSERT Dumak (Szoveg) VALUES ('3. árvíztűrő tükör-
fúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP')

SELECT Szoveg FROM Dumak
Szoveg
--------------------------------------------------
3. árvízturo tükörfúrógép ÁRVÍZTURO TÜKÖRFÚRÓGÉP
```

Ez még ügyféloldalon konvertálódik át „éktelenné", ami megelőzhető, ha a tesztszövegünket megjelöljük *(N a szöveg előtt)*, hogy az UNICODE formátumú, így az konverzió nélkül át tud utazni a kiszolgálóra. Igaz, hogy ott vissza kell konvertálni egybájtossá, de azt már az SQL Server helyesen, az oszlop típusának megfelelően teszi meg. Azaz:

```
USE LangTestEng

--Beszúrás közvetlenül az angol
-- adatbázis táblájába
INSERT Dumak (Szoveg) VALUES (N'3. árvíztűrő tükör-
fúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP')

SELECT Szoveg FROM Dumak

Szoveg
--------------------------------------------------
3. árvíztűrő tükörfúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP
```

Láthatjuk, hogy sok apró finomság állhat az ékezetek útjába. Mit tehetünk a siker érdekében? Ha van lehetőségünk homogén SQL Server-farm kialakítására, és nem szeretnénk csak magyar nyelvet, esetleg angolt *(részhalmaza a magyar kódlapnak, általában nincs vele probléma)* használni a táblákban, akkor állítsuk be mind az NT, 2000 kiszolgálókat, mind az SQL Servereket magyarra, és hagyjuk, hogy az adatbázisok és az oszlopok örököljék a magyar beállításokat. Ez különösen hálás lesz akkor, ha adatokat kell replikálni a szerverek között. A replikáció nem szereti a vegyesen beállított collation-öket, úgyhogy megéri még az elején konszolidálni a kiszolgálóparkot.

UNICODE, az univerzális megoldás?

Most mondhatná valaki, hogy miért kell ennyit problémázni a nemzeti karaktereken, amikor már ezer éve kitalálták a UNICODE-ot? Valóban, a UNICODE arról szól, hogy ne egy, hanem két byte-on írjuk le a karaktereket, így az eredő 65536-os tartományba majd csak belefér minden ország összes karaktere. Ez így igaz. Azonban nagy tömegű adathalmaznál ennek ára van – a kétszeres helyfoglalás. 100 MByte-nál ez nem kérdés, de pár tíz GByte felett ez már nagyon is számít.
Emellett a UNICODE túl nagyágyú, ha tudjuk, hogy soha nem fogunk ugyanabban az oszlopban többféle nyelvű szövegeket tárolni. Amennyiben ez a feladat, gondolkodás nélkül az UNICODE-hoz kell nyúlni. Egyes ázsiai nyelveknek több ezer karakteres betűkészlete van, ezeket csak UNICODE karakterekkel lehet leírni - ebbe is ritkán fut bele magyar ember. Ha SQL Serverek között mozgatunk adatokat, és nem azonos kódlapokat használnak a kommunikáló felek, akkor is érdemes UNICODE-ot használni, így várhatóan nem lesz gond a karakterek átvitelével, mert nincs szükség kódfordításra.
Azaz, ha teljesen biztosra akarunk menni, és kiszolgáló-, illetve adatbázisbeállítástól független módon akarunk magyar esetleg más nyelvű szövegeket tárolni, akkor használjunk UNICODE karaktereket:

```
USE LangTestEng

CREATE TABLE DumakUNI (
   Szoveg nvarchar(50) )
```

Ekkor nincs szükség karakterkonverzióra, hisz UNICODE esetén nincs szükség kódlapokra, a 65536-os tartományban jól megférnek egymás mellett a nemzetek karakterei. Egyre figyeljünk nagyon, használjuk az N prefixet a konstans szövegek előtt, jelezve, hogy UNICODE adatról van szó. Így egy angol nyelvű oszlopba, egy német alapértelmezett nyelvű kiszolgálón is hibátlanul át fognak menni a magyar ékezetes karakterek *(is)*.
Mindezen előnyök ellenére, ha nincs kényszerítő okunk a UNICODE használatára, érdemes maradni az egybájtos típusoknál, megfelelő kódlap kiválasztásával. Ez persze vitatható pont, ez csak az én személyes álláspontom.

Collation – részleteiben

Mit takar az az immáron többször is felbukkanó homályos fogalom, hogy nyelvi beállítás, vagy collation? Alapvetően a szöveges információk tárolását, leképezését *(kódlap)*, sorbarendezését és a karakterek viselkedését az összehasonlítások során határozza meg. Járjuk ezt egy kicsit körbe!
Kezdjük az összehasonlításokkal. „Alma" = „alma"? Ha Case Insensitive módon hasonlítjuk őket össze, akkor egyformának tekinthetjük őket, azaz a kis-nagybetű különbségeket nem vesszük figyelembe. Ha a beállítás Case Sensitive, akkor természetesen a kettő nem egyezik meg. Ez eddig nem volt nehéz.
Amit már sokkal kevesebben ismernek, az az Accent Sensitivity fogalma. Ez azt jelenti, hogy két szöveg, amelyek csak ékezetben különböznek egymástól egyformának tekinthető-e? Például „Eger" = „Egér"? Ha a nyelvi beállítás Accent Insensitive, akkor igen. Ha Accent Sensitive, akkor különböznek egymástól. Mi magyarok, akik ékezetes betűkkel írunk általában megkülönböztetjük az ékezetes betűket éktelen társaiktól.
Japán nyelvet kedvelőknek: ha az adott collation Kana-sensitive, akkor a Hiragana és Katakana karaktereket megkülönbözteti a kiszolgáló. Hurrá! Ez nagyon hasznos szolgáltatás a Japánoknak – na de miért kell ezzel nekünk törődnünk?

Azok, akik próbáltak már visszaállítani SQL Server 7-es adatbázist, tudják, hogy érdemes tudni a szerver Kana érzékenységének a beállított értékét, még akkor is, ha az a magyar nyelvre nem releváns. Ugyanis egy 7-es adatbázisban benne van az összes nyelvi beállítás aktuális értéke, beleértve a Kana-t is, valamint még egyet, amiről nem beszéltem, a Width érzékenységet is. Mit tennénk abban az esetben, ha kapunk egy SQL Server 7-es backup-ot, és azt mondják, hogy telepítsünk egy új SQL Servert, és állítsuk vissza rá a mentést? Ha tudjuk a Server eredeti nyelvi beállításait, beleérve kanna és vid pajtásokat is, akkor egyszerűen feltelepítjük a szervert, és visszaállítjuk a kért adatbázist. Viszont, ha nem, akkor maximum 16 féleképpen *(Case, Accent, Kana, Width kombinációi)* kell feltelepíteni, és amelyikre visszatöltődik az adatbázis, úgy volt beállítva az eredeti korábban. A rebuildm.exe meggyorsíthatja ezt a folyamatot, amivel az SQL Server teljes újratelepítése nélkül át tudjuk változtatni az alapértelmezett nyelvi beállításokat, de még így is elég gyilkos móka a restore.

Tipp: ha van a közelben egy SQL Server 2000, akkor arra probléma nélkül vissza lehet tölteni az SQL 7-es adatbázist, és meg lehet nézni az nyelvi beállítását.

Azaz látjuk, hogy SQL Server 7 esetén telepítéskor meg kellett adni a karakterkészletet, ami „beleégett" a szerverbe, és attól kezdve csakis azzal a kódkészlettel tudott dolgozni. Ez azt jelentette, hogy onnantól kezdve az összes, nem UNICODE oszlop a megadott kódlappal, az operátorok és sorbarendezések pedig a beállított nyelv logikája szerint működtek. Ha más nyelv kellett, újra kellett építeni a master adatbázist, ami hatásaiban egy SQL Server újratelepítéssel egyenértékű.
SQL Server 2000 esetén három szinten jelennek meg a nyelvi beállítások: szerverre alapértelmezetten, adatbázisszinten és a táblákban oszlopszinten. A Server alapértelmezett beállítása lesz érvényes az összes rendszeradatbázisra is: master, model, tempdb, msdb, és distribution. Ezt csak a telepítés folyamán lehet megadni, később már csak újratelepítéssel cserélhető le másra. Ez kihat az összes objektum nevének és egyéb tulajdonságának a kezelésére is. Így az oszlopnevekre is, amiből mókás hibák adódhatnak. Például tegyük fel, hogy a nyelvi beállítás Case Insensitive, és magyar. Legyen egy oszlop neve cString. Ekkor miért ne hivatkozhatnánk *(mondjuk egy scriptben)* az oszlopra, mint cstring, hisz kis-nagybetű nem számít? Hivatkozhatunk, de ekkor jön a hibaüzenet, hogy nem ismer ilyen oszlopot. Miért? Mert „cs" egyenlő csé, míg cS egyenlő cés – legalábbis a magyar nyelvben. És a kiszolgálóban az összehasonlítások is átalakultak magyarrá...
Ezek a tulajdonságok nagyon úgy hangzottak, mint az SQL Server 7-nél megismertek. Mi a különbség? Amikor létrehozunk egy új adatbázist, akkor az a model adatbázis másolataként jön létre, azaz indirekt módon örökölni fogja azokat a nyelvi beállításokat, amelyeket a szerverpéldányra adtunk meg *(hacsak nem módosítottuk a model-t)*. Viszont ettől – ellentétben a 7-es verzióval – eltérhetünk az adatbázis létrehozásakor, a COLLATE kulcsszó használatával. Azaz a globális beállításoktól függetlenül megadhatunk olyan nyelvet, amely alatt szeretnénk működtetni az adatbázisunkat.

Beállítási lehetőségek

A következőkben a LangTest nevű adatbázison keresztül bemutatom a különböző nyelvi beállítási lehetőségeket az SQL Server 2000-ben.
Az SQL Server példány alapértelmezett nyelvi beállításának lekérdezése:

```
SELECT SERVERPROPERTY ('Collation')
SQL_Hungarian_CP1250_CI_AS
```

Az adatbázis nyelvi beállításának lekérdezése:

```
SELECT
  DATABASEPROPERTYEX ('LangTest', 'Collation')
SQL_Hungarian_CP1250_CI_AS
```

Az adatbázis nyelvi beállításának megadása vagy megváltoztatása:

```
CREATE illetve ALTER DATABASE LangTest
COLLATE Hungarian_CI_AI
```

Tábla létrehozása, többféle nyelvű oszloppal:

```
CREATE TABLE VegyesDuma
(
  AngolSzoveg varchar(500)
  COLLATE Latin1_General_CI_AS,
  MagyarSzoveg varchar(500)
  COLLATE Hungarian_CI_AS,
  NemetSzoveg varchar(500)
  COLLATE German_PhoneBook_CI_AS,
  --Ez olyan collation-ű lesz, mint az adatbázis
  DBDefaultSzoveg varchar(500)
)
```

A lehetséges nyelvi konstansok listája:

```
SELECT * FROM ::fn_helpcollations()

name                 description
----                 -----------
...
Hindi_CS_AS_KS_WS Hindi, case-sensitive, accent...
Hungarian_BIN        Hungarian, binary sort
...
```

Csak a magyarok altípusai:

```
SELECT name FROM ::fn_helpcollations()
  WHERE name LIKE 'Hungarian%'

Hungarian_BIN
Hungarian_CI_AI
Hungarian_CI_AI_WS
Hungarian_CI_AI_KS
...
```

Egy kis magyarázat a fenti konstansokhoz. Az első rész a nyelvet vagy nyelvcsaládot jelenti. Utána a Case Sensiti-

vity jelzése *(S=Sensitive, I=Insensitive)*. Az „A" mint Accent, teljesen hasonlóan. A K és a W pedig a Kana ill. Width érzékenységet jelöli. Azaz, ha nekem olyan magyar beállítás kell, ami nem érzékeny a kis-nagybetűre, de megkülönbözteti az ékezeteket *(ez a tipikus beállítás)*, akkor a Hungarian_CI_AS a nyerő választás. Valójában kétféle nyelvi konstans csoport is van, az egyik a Windows 2000 által is biztosítottakkal azonos, a másik rész pedig csak az SQL Server által biztosítottak. Ezeket legtöbbször csak a régi SQL Serverekről történő frissítés során használjuk, az ajánlott *(és sokkal bővebb)* az első csoport.

Nyelvi csatározások

Mi van akkor, ha egy tábla különböző oszlopaiban különböző nyelveken írt szövegeket szeretnénk tárolni, az adott nyelvnek megfelelően sorbarendezni, kezelni? Az egyik oszlopban egy kis magyar szöveg, a következőben német, satöbbi. Természetesen ennek semmi akadálya, köszönhetően annak, hogy akár oszlopszinten megmondhatjuk a nyelvi beállításokat. Azonban ez jó kis kalamajkákhoz tud vezetni, amint valamilyen módon kapcsolatba kerülnek egymással.

Ha különböző collation-ű oszlopokat akarunk összehasonlítani *(<>=)*, akkor gondok lesznek, hisz melyiken értelmezett beállításokat, pl. kis-nagybetű érzékenységet vegye figyelembe az SQL Server? Ugyanez a kérdés jön elő különböző nyelvű szövegek összefűzésénél *(+)* is. Magyart a héberrel összehasonlítani semmi értelme, de pl. jól jöhet az, hogy egy oszlopban tárolt szöveg ékezetre érzékeny beállítású, míg egy másik nem, és ezeket szeretnénk összehasonlítani valamely módon, például ékezetek nélkül. Ami még ennél is gyakoribb, hogy az összehasonlítások során hol érzékenynek kell lenni a kis-nagybetűre, hol nem. Ilyenkor jön jól a COLLATE kulcsszó, amellyel igazságot lehet tenni. Ám előbb nézzük meg, hogy a különböző helyekről jövő nyelvi beállításoknak mekkora a prioritása egymással szemben.

Ha egy beépített függvény szöveget ad vissza *(pl. USER_NAME)*, akkor annak a collation-je az aktuális adatbázisra *(amiben a felhasználó van)* alapértelmezett beállítás lesz. Ez a leggyengébb prioritású.

Adatbázisoszlopra történő hivatkozásnál, az oszlop adatbázisból örökölt vagy a tábla létrahozásakor megadott nyelvi beállítás lesz az alap.

Ha egy kifejezésben *(például két szöveg összehasonlítása)* explicit megadunk egy collation-t valamelyik szövegre, akkor általában annak lesz a legnagyobb prioritása.

Nézzük meg az előbbieket egy példán keresztül! Készítsünk egy táblát, amelynek két oszlopa van, mindkettő magyar beállításokkal, de az egyik kis-nagybetű érzékeny, a másik nem.

```
CREATE TABLE Comp
(
  HCI varchar(50)
 COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
  HCS varchar(50)
  COLLATE SQL_Hungarian_CP1250_CI_AS NOT NULL
)
```

Szúrjunk be egy tesztsort, és próbáljuk meg összehasonlítani a két oszlopot!

```
INSERT Comp (HCI, HCS)
  VALUES ('Netacademia', 'NetAcademia')

--Teszt egyenlőségvizsgálat
SELECT
  CASE
    WHEN HCI = HCS THEN
    'Egyformák'
  ELSE
    'Különböznek'
  END,
  HCI,
  HCS
FROM
  Comp
```

Az összehasonlítás helyett egy csúnya hibaüzenetet kapunk:

```
Server: Msg 446, Level 16, State 9, Line 1
Cannot resolve collation conflict for equal to
operation.
```

Az a problémája az SQL Servernek, hogy össze akarunk hasonlítani két oszlopot, amelyeknek nem azonos a collation-je. Ez önmagában még nem volna probléma, csakhogy az egyenlőség mindkét oldalán oszlop-hivatkozás található, így a prioritások alapján nem lehet eldönteni az összehasonlítás kivitelezéséhez szükséges kis-nagybetű érzékenységet, mert a két oldal egyforma prioritású.

Mit lehet tenni? Egyrészt megmondhatnánk kiszolgálónak, hogy a jobboldali oszlop, amely egyébként kis-nagybetű érzékeny, ne legyen az. Így megszűnik a konfliktus oka, hisz a kifejezés mindkét fele azonos collation-ű lesz:

```
...
  CASE
  WHEN HCI =
       HCS COLLATE SQL_Hungarian_CP1250_CI_AS
...
```

Az eredmény az elvárt lesz, azaz a két oszlop megegyezik, mert kis-nagybetűre nem érzékeny collation-t választottunk közös nevezőnek:

```
             HCI          HCS
---------- ----------- -----------
Egyformák  Netacademia NetAcademia
```

Aki szereti a konfliktust, annak ajánlok egy másik megoldást. Melyik konstrukciónak is volt a legnagyobb prioritása? Hát az explicit COLLATE parancsnak. Mondjuk azt, hogy tudjuk, hogy a jobb oldali oszlop Case Sensitive, és ezt szeretnénk megerősíteni egy COLLATE-el is. Ekkor ez ellentmondás lesz a HCI oszlop által dirigált Case Insensitivity-vel szemben, de hát győz az erősebb:

```
...
  CASE
  WHEN HCI =
       HCS COLLATE SQL_Hungarian_CP1250_CS_AS
...

              HCI          HCS
------------ ------------ ------------
Különböznek Netacademia NetAcademia
```

A nyelvi beállítások miatti ütközéseknek, lehetséges konfliktusoknak most csak egy részét elemeztem ki. Részletes információt velük kapcsolatban a BOL-ban, a Collation Precedence címszó alatt találhat a Kedves Olvasó.

Zárszó

Az előző Tech.net számban kimaradt egy rész a Transact SQL sorozatból. Sok visszajelzést kaptam, amelyekben hiányolták az aktuális havi SQL Server evangéliumot. Köszönöm a dicsérő szavakat. Minden lelkes SQL Server rajongónak üzenem, hogy amíg lesz tinta a billentyűzetemben, addig lesz SQL sorozat is. A pozitív visszajelzések jót tesznek a töltőtollamnak, a kritikák pedig segítenek abban, hogy a sorozat még jobb legyen, és még inkább arról szóljon, ami segítik az Önök mindennapi munkáját.

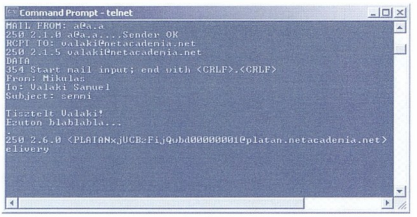
Soczó Zsolt MCSE, MCSD, MCDBA
Zsolt.Soczo@netacademia.net

Kérdezz, felelek!

K: *Hogyan lehet Telnet ablakból teljes értékű email írni? Csak kíváncsiságból...*
V: Aki eddigi cikkeinket elolvasta e tárgyban, az már majdnem tudja is. Így:

```
TELNET MAIL.AHOL.COM 25
```

(Az ahol.com tényleg SMTP Server, de ez csak példa!) Ezután az adott kiszolgáló üdvözlő üzenetét láthatjuk, és kezdhetjük a levél írását.



Email by hand

Először illedelmesen köszönünk:

```
EHLO
```

Azután megadjuk a feladót a borítékon:

```
MAIL FROM: a@a.a
```

Majd a címzett címe következik:

```
RCPT TO: valaki@netacademia.net
```

A DATA parancs után jön a levél:

```
DATA
```

A levélen belül ismét megadható a feladó és a címzett:

```
From: Mikulas
To: Valaki Samuel
Subject: semmi
```

Nagyon fontos a Subject utáni üres sor! Majd jöhet a szöveg. Legyen – mondjuk – plain text, mert mást úgysem tudunk kézzel írni:

```
Tisztelt Valaki!
Ezuton blablabla...
.
```

Az egyetlen pontot tartalmazó sor indítja be a levélküldés folymatát:

```
250 2.6.0
<PLATANxjUijQwbd00000001@platan.netacademia.net>
Queued mail for delivery
```

Vigyázat! Rontási lehetőség nincs, a BackSpace és a törlőgomb használata nem a kívánt hatással jár, hanem az adott billentyűnek megfelelő ASCII kód bekerül a parancsokba! Aki félregépelte, kezdheti elölről! *(Az Outlook kicsivel kénnyelmesebb nem?)*
Forrás: NetAcademia Exchange 2000 lista

K: *Hogyan lehetne olyan scriptet írni Exchange 5.5-re, ami nem aszinkron módon fut? Mert most az a baj, hogy mire lefut, már lehet hogy törölték az üzenetet, amin dolgoznia kellene...*
V: Upgrade to Exchange 2000. Részletesebben: szinkronműködésű scriptet az Exchange 5.5 nem tud.
Forrás: NetAcademia Exchange 2000 lista

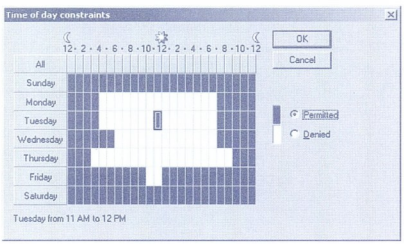
K: *Az a problémám, hogy egy tagkiszolgálót ki szerettem volna nevezni tartományvezérlővé, s ennek érdekében lefuttattam a DCPROMO.EXE-t, de sajnos az Active Directory telepítése nem sikerült. Ennél is szomorúbb, hogy most a gép nem tartományvezérlő, sem tagkiszolgáló többé, hanem félúton van. Nosza, ráeresztettem a DCPROMO-t mégegyszer, hogy visszavigyem a kiindulási állapotba – nem sikerül. Safe Mode sem segít. Az operációs rendszer újratelepítése sem túl jó ötlet. Most mi lesz?*
V: Ilyen esetekben a következőt kell tenni: egy ügyes regist-rymódosítással kell kibillenteni a gépet jelenlegi lehetetlen helyzetéből. Ha a

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Control\ProductOptions
```

kulcs alatti ProductType értékét átírjuk „LanmanNT"-ről „ServerNT"-re, akkor a következő újraindításnál az LSASS.EXE *(Local Security Authority)* nem húzza be az Active Directoryt a memóriába, hanem standalone üzemmódban indul. Ezzel már vissza is tértünk a tagkiszolgálóhoz. Már csak a WINNT\NTDS könyvtár törlése van hátra, és a DCPROMO indulhat is, ismét nulláról!
Forrás: NetAcademia Windows 2000 lista

K: *Van egy Win2000-es gép, amelyen azt szeretném beállítani, hogy a felhasználók ne tudjanak napközben ezen keresztül felcsatlakozni az Internetre, hanem csak a kedvezményes időszakban legyen erre lehetőségük. A legtökéletesebb megoldás az volna, ha a felhasználók közül néhányat ki tudnék jelölni, akikre nem érvényes ez a korlátozás.*

Jogosultságokkal el tudom érni ezt az állapotot, de sajnos nem tudom megoldani, hogy a kedvezményes időszak kezdetekor automatikusan engedjen mindenkit. Külön gond, hogy a kedvezményes időszakok másként alakulnak hétvégén és hétköznap. Milyen szoftver oldaná meg a problémámat?
V: A Windows 2000-be beépített RAS szolgáltatás pont erre való. A hozzá tartozó házirend *(policy)* kismillió beállítása között megtalálható a kapcsolatengedélyezés időablaka is, amelyet az alábbi ábrán rendesen „testreszabtam":



Kedden délelőtt 11-kor például szabad tárcsázni..

Ez még kevés lenne a boldogsághoz, mert személyenként különbözőképpen kellene beállítani. Ehhez további policykat kell felvenni, melyekben természetesen az érintett felhasználók mások lesznek, ezt különböző csoportok felvételével érhetjük el. Hab a tortán, hogy a házirend által engedélyezett kapcsolatokat a házirend profilja menet közben tudja vezérelni, adott esetben például megszakítani, vagy egynél több modemet egyidejűleg használni.
Forrás: NetAcademia Windows 2000 lista

K: *Adott egy tárolt eljárás, aminek a visszaadott eredményhalmazán, mint recordset-en további lekérdezéseket szeretnék futtatni. Van erre lehetőségem SQLServer 7 alatt?*
V: Első ránézésre a dolog SQL 2000 után kiált, de van lehetőség, mégpedig a következő trükkös módon: használd az OpenRowSet függvényt, mellyel a saját SQL Serveredhez kapcsolódsz, mintha az egy Linked Server lenne.

```
select *  from openrowset( 'SQLOLEDB', serv_name';
'sa'; '', 'exec sp_who' ) where status =
'sleeping'
```

Forrás: NetAcademia SQL 2000 lista

K: *Az AT parancsot használnám batch parancs futtatására, fájlok mentésére úgy, hogy közben lássam is, mi zajlik, de a /INTERACTIVE kapcsoló hol működik, hol nem. Mi a baj?*
V: A /INTERACTIVE kapcsoló trükkös nem működésének az az oka, hogy NEM MINDEGY, hogy az AT parancsba hova írjuk be, csak azon a pozíción működik, ahol az AT /? mondja! A többi esetben, pl. ha a a parancs végére írjuk, mindenféle hibaüzenet nélkül, csendben nem működik, és az időzített parancs nem jelenik meg a felhasználói felületen, hanem rejtve fut...
Forrás: NetAcademia Windows 2000 lista

K: *Mire jó a READ jog hiánya? Úgy vettem észre, az Active Directoryban a READ jog nem egészen úgy viselkedik, mint például az NTFS-en.*
V: Jogos észrevétel. NTFS esetén a READ jog hiánya nem tünteti el a szemem elől az érintett fájlokat és könyvtárakat *(pedig de jó lenne, ha így működne!)*, hanem csak a tartalma olvashatatlan. Ha megpróbálom megnyitni, jön az „Access Denied" móka.
Ám az Active Directoryban másképp van!
A READ jog hiánya nem feltétlenül „Access Denied" hatással jár, hanem egyéb funkciók ellátására is való! Példák:

1) Active Directory: ha nincs READ jogunk egy objektumon *(felhasználó, szervezeti egység stb.)*, akkor nem is látjuk az objektumot/konténert a címtárban *(á la novell)*. Itt a READ jog megvonása a láthatósággal van kapcsolatban.
2) Group Policy: ha nincs READ és APPLY joga egy felhasználónak egy GPO objektumon, akkor nem hibaüzenetet kap, hanem nem értékelődik ki rá a házirend. Vagyis a READ *(és/vagy APPLY)* jog megvonásával lehet kivételt képezni a házirend hatása alól.
3) Remote Installation Services: ha az Unattend.TXT fájlon nincs READ jogunk, ennek az lesz a hatása, hogy az adott telepítési lehetőség nem jelenik meg a RIS ügyfélképernyőjén a telepítési lehetőségek felsorolásában. Ebben az esetben a READ jog megvonása a felhasználók számára felkínált RIS telepítési lehetőségek, menüpontok korlátozására való.

Forrás: NetAcademia Windows 2000 lista

Pazarló világban élünk, és a pazarlás bűn. Különösen az egy olyan világban, ahol az a szűkös erőforrásokat *(például a sávszélességet, a tárkapacitást, a processzoridőt, a mi drága időnket stb.)* érinti. Az egymást gerjesztő hardver- és szoftvervilág megalomániáját látva a példákat végeláthatatlanul sorolhatnám. Ehelyett inkább hadd ragadjak meg egyetlen példát, a weboldalak ékesítésének szinte már nélkülözhetetlen eszközét: a képet.

Egy digitális kép parányi négyzetekből, képpontokból áll, melyek mindegyikéhez egy véges halmazból választható szín. Szorozzuk össze a vízszintes és a függőleges felbontást a színmélységgel! Az eredmény a kép tárolásához szükséges hely. Bocsánat, hadd javítsam ki magam: az eredmény a kép tárolásához elégséges hely. Nem túl brilliáns módszer, ugye? Biztosan mindenki hallotta már a következő rövidítéseket: JPG, GIF, TIF, PXC, PNG hogy csak a legismertebbeket említsem. Nincs varázslat, csak tömörítés. A különféle módszerek hatékonyságát megítélni nem egyszerű feladat, mivel a legtöbbnél nemcsak a méret változik, hanem a minőség is *(ellentétben azokkal a képtömörítő eljárásokkal, amelyek kizárólag a redundancián alapulnak)*.

Az Internet egyik nagy kedvence a Joint Photographic Experts Group által kifejlesztett JPG *(a 8.3-as fájlnévkorszak hanyatlása után egyre gyakrabban JPEG)*, amely nagyon jó színhűséggel ábrázolja az eredeti képet, kis helyen. A trükk az, hogy a képet 8x8 képpont nagyságú négyzetekre bontják, amiből a túl sok helyet foglaló, de a minőség szempontjából kevésbé lényeges színinformációt egy DCT nevű transzformációval kiszűrik. *(Ha nem hiszi, töltsön be egy JPEG képet, nagyítsa fel a kétszeresére vagy a négyszeresére, és meglátja a normál méretnél szinte észrevehetetlen négyzethálót. Ha a program alkalmas rá, a négyzetek méretét is ellenőrizheti.)* A DCT a képeknek elsősorban azon részeit érinti hátrányosan, ahol a színek hirtelen változnak. A kép darabos lesz, viszont kárpótlásul nem kell sokat várni, hogy a grafikai elemek megjelenjenek a letöltött weboldalakon. Mondhatjuk, hogy a JPEG megfelel a célnak? Mondhatnánk. Én viszont azt mondom, hogy a JPEG pazarlás. Miért? Mert van jobb!

A tömörítési algoritmusok fejlődése csodákra képes. Jobb minőség kisebb helyen – akár egy marketing doksiban. A Microsoft jelfeldolgozási csoportjának vezetője, Henrique „Rico" Malvar olyan kódolási szabványt dolgozott ki, amely a JPEG-nél gyorsabb és hatékonyabb tömörítést tesz lehetővé. Az algoritmus neve Progressive Wavelet Codec *(PWC)*. A Wavelet Codec kifejezés egy sor kódolási megoldást takar, de a PWC kitűnik közülük egyszerűségével, gyorsaságával és skálázhatóságával.

A PWC kép minősége önmagában nem csúcs, de a JPG-vel összehasonlítva óriási előrelépés. Szebb vonalvezetése annak köszönhető, hogy a képet a valódi kontúrokat követő, különféle méretű ívek építik fel, nem pedig a szem érzékenységéhez képest nagy négyzetek. Míg a JPG esetében az éles kontúroknál nagyon „bolyhos" lesz a kép, ugyanennek a PWC kódolású változata sokkal kellemesebb látványt nyújt.

A PWC tömörítés rugalmas és robusztus képkezelést biztosít. Másképpen fogalmazva: egyrészt veszteség nélkül tömörít – legalábbis ami a színeket illeti, – másrészt adott felbontás és élethűség mellett tömörített kép dekódolása kisebb felbontásnál és élethűségnél is elvégezhető. Nincs szükség tehát arra, hogy az átméretezésnél először az eredeti paraméterekkel dekódoljunk, majd az újakkal ismét kódoljunk, az új beállítások azonnal érvényesíthetők a dekódolási folyamatban. Azok, akik készítettek már weboldalt, biztosan tudják, milyen nehézkes tud lenni a JPEG képek kezelése. A PWC többek között náluk is nagy sikerre számíthat. Képzeljünk el egy 10 Mbps-es helyi hálózatot, amelyhez modemmel is lehet kapcsolódni a nyilvános telefon-hálózaton keresztül. Az utóbbi esetben szerényebb átviteli sebességgel, mondjuk 33,6 Kbps-sel kell beérnünk. Ki tölti le hamarabb ugyanazt a hagyományos képekkel teletűzdelt weboldalt? Időm se nagyon lenne kimondani, hogy rajt, és az első versenyző már célba is ért. Ez a küzdelem nem felelt meg a fair play szabályainak. De fog, ha segít a PWC első betűje mögé rejtőző progressive *(fokozatos)* jelző.

A fokozatos kódolás a képet ún. rétegekre bontja, amelyből mindenki a maga erejének megfelelő mennyiséget töltheti le. Magától értetődő, hogy a LAN-os versenyző képei szebbek lesznek, de nem lepődnék meg, ha a modemes résztvevő azt mondaná: nem annyival, amennyivel a letöltést gyorsabban befejeztem!

A PWC ennek a technikának valójában egy sajátos és különösen hasznos formáját, a beágyazott kódolást alkalmazza. Az előzőhöz képest többletet az jelenti, hogy a képet leíró bitsorozatokból elegendő csak akkora, az erőforrásainknak megfelelő előtagot figyelembe venni *(letölteni)*, amekkorát adott felbontás, illetve élethűség megkíván.

A kutatók – és most már valószínűleg az olvasók is – bíznak benne, hogy a PWC kiterjesztést hamarosan egyre több képszerkesztő program és böngésző ismeri majd fel. Az eredmény: takarékosság a sávszélességgel, a tárhellyel, javul a weboldalak letöltésének sebessége, és ami legalább ennyire fontos, mindez nem a képek minőségének a rovására, hanem épp ellenkezőleg, a vizuális élvezet fokozásával válik valóra. Úgy legyen!

zacco

KAPCSOLJON!

A „**tech.net magazin Brainstorm**" a Dupla KV rovathoz hasonló, ám a személyes kérdésfelvetést és vitát is lehetővé tevő rendezvény, melynek célja

- **Az elsőre talán ismeretlen technológiák élő bemutatása**
- **A cikkekhez kapcsolódó kódok megírása/kipróbálása**
- **A terjedelmi okokból kimaradt információk átadása**

E magazinnal együtt a legelső rendezvényre érvényes belépőjegyet minden előfizetőnkhöz eljuttattuk.
További információk a belépőjegyen olvashatók.

NETACADEMIA

A legjobbakat tanítjuk.

Várjuk önöket a NetAcademia Mesterkurzusokon!

(Bár belépőjegyet adtunk, ez a tény önmagában nem biztosítja helyét a rendezvényen. A jegy célja előfizetőink elsődlegességének biztosítása, de a korlátozott résztvevői létszám (100 fő) miatt a regisztráció kötelező. Jelentkezzen, amíg nem késő!)

http://technet.netacademia.net/brainstorm

(folyt. köv.)