tech.net

working with windows

III./05.szám
1344 Ft

Metadirectory

13. oldal

ISSN 15865185 05

9 771586 518005

A LEGENDÁS ÖT KILENCES. KÖZELEBB VAN MINT GONDOLNÁ!

A szerver operációs rendszereknél, az a bizonyos öt kilences a megbízhatóság legfelsőbb fokát jelzi. Gyakorlati nyelvre lefordítva, ez a teljes rendelkezésre állás mellett, mindössze 5 perc kiesést jelent évente!* A 99,999%-os megbízhatósági mutató már nem csak laboratóriumi körülmények között érhető el. Már nem csak olvasni lehet róla. Már nem csak a méregdrága számítástechnikai rendszerek kiváltsága. Közelebb van mint gondolná! Képzelje el, hogy már az Ön cége számára is elérhető ez a szédületesen stabil szerver háttér! Képzelje el, hogy mindez kedvező áron megvásárolható! Képzelje el, hogy nem kell többé elképzelnie, mert a 99,999%-os megbízhatóság az Ön cége számára is valósággá válhat!

Ha a **Microsoft Windows 2000 Server** családot választja, akkor a megbízhatóság csúcsát választja.

Microsoft Windows 2000 Server Family

*Ez az adat függ az operációs rendszeren kívüli körülményektől is, mint például a hardvereszközök milyenségétől, illetve más, az operációs rendszertől független szoftverektől.

99999 A legendás öt kilences: 99,999%-os rendelkezésre állás.

http://www.microsoft.com/hun/products/windows.htm
http://www.microsoft.com/hun

Microsoft Ügyfélszolgálat: 2MSINFO [267-4636]
Microsoft Forródrót: 2MSSUGO [267-7846]

Microsoft

A MesterQrzus visszatér

tech.net
working with windows

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@technetklub.hu

Kiadja és terjeszti a
NetAcademia
A LEGJOBBAKAT TANÍTJUK.

NetAcademia Kft.
Terjesztési, előfizetési információ:
Tel.: 263-2732
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft
Példányszám: 4.000

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

Hirdetésfelvétel:
BÁRSONYKALAPÁCS
MARKETING
VELVET HAMMER MARKETING
Az áttörő siker kovácsa.
Felelős: **Sallai Eszter**
Tel.: 489-4661
Fax: 489-4660
info@velvethammer.hu
1027 Budapest, Fő utca 67. V. 1.
Grafikai tervezés, kivitelezés, nyomdai előkészítés:
Bársonykalapács Marketing
Művészeti vezető: **Balogh Zoltán**

Nyomda:
Hieron Kft.
2120 Dunakeszi, Tamási A. u. 11/a.
Felelős vezető: **Török Andrea**

ISSN 1586-5185

December-január fordulóján hirtelen annyi munkánk lett, hogy egyre nagyobb erőfeszítésembe került az általunk indított levelezési listák forgalmának nyomon követése. Bárhogy küzdöttem, az olvasatlan levelek száma egyre csak szaporodott, mígnem feladtam a küzdelmet: beszüntettem szeretett listáim olvasását. Most visszanéztem rá: összesen mintegy hatezer olvasatlan levelem keletkezett. Szép szám! De mit kezdjek ezzel a levéltömeggel?

A levlista olyan kicsiben, mint az Internet nagyban: az ömlesztett „információ"- jelentős százaléka zaj, a maradék pedig koherenciamentes infopép. Aki ebből tanulni akar, igen erős alapokkal kell, hogy rendelkezzen. Egy megfelelően felkészült szakember ugyanakkor katasztrofális jel/zaj-arányt érzékel, és mint megszűrhetetlent, kihajítja az egészet. Tudják, mi a reménytelen? Aranyat mosni a fürdőkád vizében.

Most, hogy ismét elkezdtem levlistázni *(de most már nem a válaszadás belső kényszerével, hanem külső szemlélőként, olvasgatva követem)*, úgy látom, egyesek pontszerű kérdéseire mások pontszerű válaszokat adnak. Mintha egy pointillista festményen odaböknénk egy pontra, és megkérdeznénk: ez milyen színű? Sárga. És amott? Ott kék. De miért? Ha távolabbról nézzük, akkor áll össze a teljes kép, és az emberen úrrá lesz az „Aha!"-érzés. Többé már nem is kell rákérdeznie az egyedi pontocskák színére. Vagy talán még jobb példa a kotta. Kodály országában mindenki tanul zenét, tán még a violinkulcs szerepe is világos - de hiába nézzük a kottabogyókat *(c, e, c, e, g, g)*, nem szólal meg bennünk a zene.

Remélem, a tech.net magazin rendszeresen nyújt „Aha" élményt. Ha nem, az sem baj: mindenkinek tanulnia kell a teljes egész felismerését. Van, akinek elég a kotta, és megszólal fejében a szimfónia. Többségünk nem ilyen. Ismerhetjük ugyan a kottajeleket, de a koncert nyújtja az igazi élményt.

Ebben a hasonlatban a tech.net magazin a kotta, s a MesterQrzus előadás a koncert. Ősztől újraindítjuk korábban megismert MesterQrzus előadásainkat. A jól bevált célokon és felépítésen nem változtatunk: havonta egy délután szeretettel várjuk olvasóinkat a tech.net magazinhoz kapcsolódó előadásokon. Változik viszont az időpont, valamint a helyszín – de ezeket a lényegtelen apróságokat a hamarosan postázandó belépőjegyeken is feltüntetjük. Sokkal fontosabb az a változás, hogy – mivel az évek során meglehetősen nagy mennyiségű előadásanyagot halmoztunk fel – rendelkezünk vázlatos előadástervvel, így ki-ki idejekorán eldöntheti, melyik előadás érdekli. Íme a majdnem véglegesnek mondható őszi kínálat:

Golyóálló webkiszolgáló	Windows Scripting Host	Active Directory Services Interface
Újdonságok a .NET Serverben	Group Policy	Remote Installation Services
.NET Framework bevezető	Objektumorientált alapelvek +DP	C#

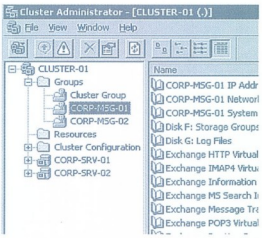
Fóti Marcell
marcellf@netacademia.net

2002 Május

Metadirectory

A legutóbbi Tech.net magazinban áttekintettük, hogy általában milyen elvárásaink lehetnek egy metacímtár szolgáltatástól. Most konkrétan a Microsoft Metadirectory Service *(MMS 2.2)* működésével ismerkedhetünk meg.

13. oldal



Farkasokkal táncoló

(VII. rész) – Cluster a gyakorlatban

Az infrastruktúra területén tett kirándulásunk után visszatérünk az erőforrások világába, és megnézzünk, hogyan lehet fontos alkalmazásszoftvereknek magas rendelkezésre állást biztosítani.

5. oldal

Windows .NET

Server Beta 3

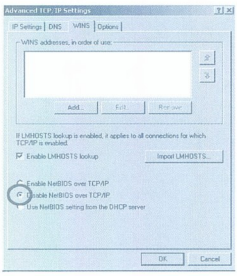
Folytatjuk az előzetes szemezgetést a Windows.NET Server új funkcióiból, egyelőre még mindig a béta3 verziót használva – no és persze ismét teszünk egy kis kitérőt a belső fejlesztési folyamatok rejtelmei felé...

9. oldal

Resszkessenek a hackerek?

Az a szakértő, aki tudásával, ismereteivel segíti a hackert, ugyancsak bűncselekmény elkövetőjévé válik, és két évig terjedő szabadságvesztéssel számolhat.

18. oldal

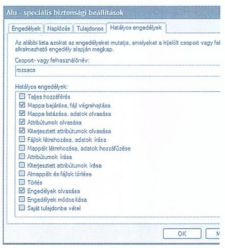


Támadás az IPSec ellen

Amire az IPSEC csomagszűrőnél figyelni kell

A tech.net magazin III. évfolyamának első számában olvashattuk, hogyan használható a Windows 2000 IPSec implementációja, mint egyszerű csomagszűrő. Sajnos ez a tulajdonság csak mellékterméke, tervezéskor egyáltalán nem szerepelt a célok között. Ezt hallva az olvasó rögtön gyanút foghat, vajon nem származik-e ebből valami alapvető probléma.

20. oldal



DumpSec

NTFS jogosultság listázása

Több alkalommal is felmerült a NetAcademia listáin a kérdés, vajon hogyan lehetne megállapítani, hogy egy felhasználónak milyen tényleges jogai vannak egy könyvtárban, illetve milyen eszközzel lehetne akár egy egész könyvtárstruktúra aktuális hozzáférési szabályairól kimutatást készíteni. Nos, egy beépített eszköz és egy szabadon használható szoftver megoldást jelenthet a problémával küzdőknek.

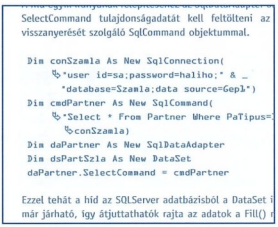
24. oldal

XMLgessünk

Az XML Shema (II. rész)

Az előző részben láthattuk, hogyan kell közvetlen egymásba ágyazással, referenciákkal és típusok definiálásával egyszerűbb sémákat szerkeszteni. Részletesen megnéztük hogyan lehet egyszerű típusokat létrehozni a már meglévő típusokból. Ebben a részben a komplex típusokat tekintjük át. Megnézzük, hogy összetettebb típusok létrehozására milyen fejlettebb nyelvi elemek állnak rendelkezésre.

28. oldal

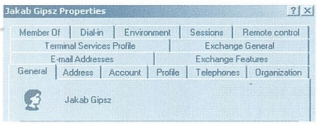


Adathíd

1+3 sávos aszfaltozási munkálata

Egy memóriában tárolt adatbázis óriási kincs, de csak akkor, ha stabil technológiai kapoccsal összeköthető a meglévő adatbázisszerverekkel és a helyi gépeken tárolt XML állományokkal. Az alábbi cikk a két kapcsolódási felület közül az elsőt elemzi, és az SQLServer adatbázis és a DataSet objektum közötti híd „aszfaltozási munkálataiba" avatja be az Olvasót.

30. oldal



Exchange 2000

Felhasználók, csoportok, címlisták

Az előző szám végén ott tartottunk, hogy a felhasználók beállításai jönnek. Jöjjenek, és nemcsak a felhasználók, hanem az e-mail címmel rendelkező egyéb objektumok – tehát a kapcsolatok *(Contacts)* és az e-mail címmel rendelkező csoportok - tulajdonságait is megnézegetjük ebben a részben.

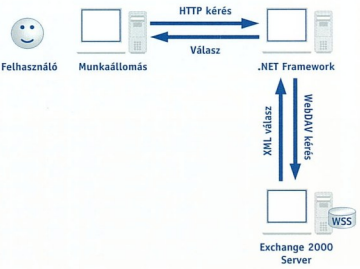
33. oldal

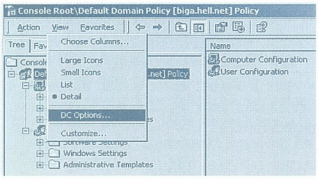
Exchange 2000

Server és web storage system

Korábban bemutattuk, hogyan lehet az Exchange2000 Server Web Storage System-ét *(WSS)* elérni ADO-n *(ActiveX Database Objects)* keresztül. Most a .NET keretrendszer segítségével állunk neki a feladatnak. Egyelőre azonban – sajnálatos módon előre láthatatlan ideig – ADO.NET-ben nincs támogatás az Exchange2000 Server és a WSS elérésére. Ezért más, .NET alatt is használható megoldások után kell néznünk.

37. oldal





Dupla KV

41. oldal

Farsite

Úgy tíz évvel ezelőtt egy Deltában láttam, amint japán kutatók bemutatták, hogyan tudnak szemétből gyémántot előállítani, mondhatnám varázsolni. Önkéntelenül ez jutott eszembe, amikor először hallottam a Farsite-ról, mert ezzel temérdek ócskavasból *(azaz idejétmúlt, de még nem teljesen használhatatlan számítógépből)* gigászi tárhelyet, amolyan internetes bőségszarut építhetünk.

44. oldal

Farkasokkal táncoló

(VII. rész) – Cluster a gyakorlatban

Az infrastruktúra területén tett kirándulásunk után visszatérünk az erőforrások világába, és megnézzünk, hogyan lehet fontos alkalmazásszoftvereknek magas rendelkezésre állást biztosítani.

Látni kell, hogy itt már nem pusztán egy szolgáltatást hozunk létre, hanem egy teljes alkalmazást helyezünk el fürtözött környezetben, ez pedig minden eddiginél nagyobb figyelmet követel tőlünk. A következőkben két Microsoft-terméket és egy harmadik, más gyártótól származó szoftvert vizsgálunk meg.

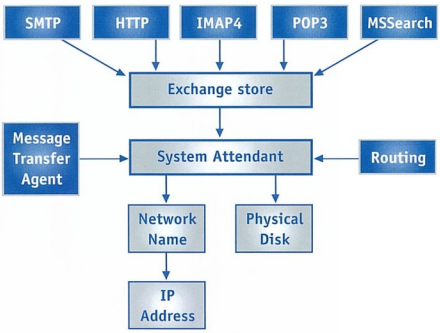
A Microsoft Exchange fürtkörnyezetben

Az Exchange Server az 5.5-ös verzió óta támogatja a clustertechnológiát, mi azonban most csak a Windows 2000 – Exchange 2000 párost vizsgáljuk. Ebből sem valamennyi verziót: csak az Exchange 2000 Enterprise Edition változat képes együttműködni a fürtszolgáltatással, vagyis ezen változat megléte követelmény a telepítéshez.

A Microsoft üzenetkezelő-alkalmazása natív módon támogatja a fürtkörnyezetet. Ez azt jelenti, hogy saját erőforrás DLL-ekkel rendelkezik, amelyek az Exchange-komponensek és a fürt közötti kommunikációt biztosítják. Pontosan ismeri azokat a fogalmakat, amelyekre a Microsoft fürtrendszer épül –a **csoport**, az **erőforrás**, a **függőség** és a **quorum** fogalmát. Többféle konfigurációt is létrehozhatunk *(aktív-aktív, aktív-passzív)*, sőt datacenter környezetben négy állomáson akár 3 aktív-1 passzív felállást is építhetünk. A négy aktív állomás egyelőre nem támogatott.

Az alkalmazás tervezőmérnökei az „Exchange Virtual Server" fogalmát úgy valósították meg, hogy az egybeesik a fürt „csoport"-objektumával, ha a következő feltételek teljesülnek: a csoportnak van fizikai lemez, IP-cím és hálózati név erőforrása, továbbá a teljes fürtre vonatkozóan már létezik a „System Attendant" nevű, az Exchange 2000 részeként szállított erőforrás.

Az Exchange egyes komponensei pontos függőségi viszonyban vannak egymással, ahogy ezt az alábbi ábra mutatja:



Az Exchange-komponensek függőségi rendszere

A levelezőt jól ismerő Olvasónak feltűnhet, hogy korántsem teljes ez a lista. Valóban. Sajnos az Exchange 2000 bizonyos komponensei nem fürtözhetők. Íme a lista:

- Microsoft Active Directory™ Connector *(ADC)*
- Chat Service
- Microsoft® Exchange 2000 Conferencing Server
- Instant Messaging
- Key Management Service
- Exchange Calendar Connector
- Exchange Connector for Lotus cc:Mail
- Exchange Connector for Lotus Notes
- Exchange Connector for Microsoft Mail
- Exchange Connector for Novell GroupWise
- Event Service
- Network News Transfer Protocol *(NNTP)*
- Site Replication Service *(SRS)*

Eltérően a korábbi verziótól, akkor sincs mód a komponenst önállóan az állomásra telepíteni, ha maga esetleg önálló alkalmazás lenne *(csevegés, azonnali üzenetküldés, konferenciaszerver)*, mert a protokollinformációk a fürtben találhatók. Ha tehát szükség van ezekre az alkalmazásokra, akkor újabb, önálló Exchange Servert kell telepítenünk. Azért nem olyan vészes a helyzet: az ADC és az SRS csak átmenetileg szükséges *(amíg 5.5-ös rendszerünkkel kell együttélnünk)*, a konnektorok pedig nem fürtözött szolgáltatáshoz kapcsolódnak – így valószínűleg nem **keletkezik** Achilles sarok attól, hogy a csatolószoftver nem fürtözhető.

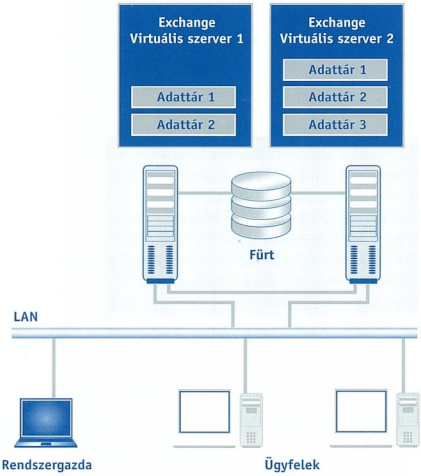
Nem korlát, de jó, ha tudjuk: az MTA erőforrás mindig aktív-passzív, másképp fogalmazva, egy fürtben egyszerre csak egy MTA erőforrás lehet aktív, még datacenter esetében is. Az MTA mindig az első virtuális szerveren jön létre, és ha később törölnénk, új csoportba költözik mindaddig, amíg legalább egy Exchange virtuális kiszolgáló létezik a fürtön.

Az Exchange 2000 telepítés előkészületei

Járjuk körül egy kicsit pontosabban, mit is jelent az aktív-aktív, illetve az aktív-passzív konfiguráció! Azt a korábbi cikkekből már tudjuk, hogy a Microsoft fürt-implementációja nem nyújt terhelésmegosztást, nem erre tervezték. Minden egyes létrehozott erőforrás számára a rendelkezésre állás növelését kínálja. Aktív-passzív beállításnál ez egyértelmű. Az egyik állomás helyet ad az Exchange virtuális kiszolgálónak, vagyis a fürt egy csoportjának, ha pedig valamilyen hiba lép fel, ez a csoport átköltözik a másik szerverre, amely eddig csak magában dohogott. Aktív-aktív rendszernél sincs ez másképp, a különbség az, hogy a második állomáson is van egy virtuális Exchange kiszolgáló. Milyen funkciót lát el? Bármi mást, mint az első. Ha például az egyik fürtcsoport

a felhasználók postaládáit kezeli, a másik állomáson a másik Exchange a nyilvános mappákat. Az aktív-aktív jelző tehát a teljes fürtre értendő, nem az egyes virtuális szerverekre, azok ugyanis mindig csak az egyik node-on futnak.

A tervezésnek más fontos momentuma is van. Egy Exchange 2000 kiszolgálónak legfeljebb négy ún. adattárcsoportja *(storage group)* lehet. *(Ez nem keverendő össze a fürtcsoporttal. Az adattárcsoport olyan, mint egy adatbázis az 5.5-ös verzióban, persze számos új tulajdonsággal.)* Itt szándékosan nem virtuális kiszolgálóról beszélünk, hanem valódiról. Tehát egy fürtállomáson is legfeljebb négy adattárcsoport lehet. Ebből viszont következik, hogy a teljes fürtben nem érdemes négynél több adattárat definiálni. Ha az alábbi ábrán a második virtuális kiszolgálónak át kellene költöznie az első állomásra, a három adattárcsoportjából csak kettő indulna el:



Az adattárak száma nem lehet több négynél egy Advanced Server fürtben

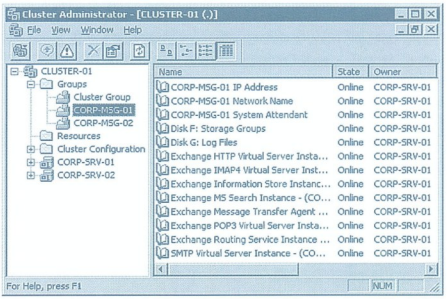
A telepítésnek vannak olyan feltételei, amelyek nemcsak a fürtökre, hanem minden Exchange kiszolgálóra érvényesek, *(pl.: DNS, AD, GC telephelyek stb.)*. Ez azonban nem témája vizsgálódásunknak. Vannak viszont a fürttel kapcsolatos egyedi feltételek is, amelyeket be kell tartani. Ezek:

- Egyszerre nem szabad telepítést végezni mindkét állomáson. Az idővel nem így kell takarékoskodni. Különösen igaz ez, ha tartományvezérlőkre telepítjük a rendszerünket.
- A fürtszolgáltatás fiókja legyen tagja az „Exchange Full Administrators" csoportnak! Figyelem! Ezt mindkét állomáson érvényre is kell juttatni, vagyis a fürtszolgáltatást legalább egyszer újra kell indítani, vagy meg kell várni, amíg a Kerberos jegyet megújítja a szolgáltatás *(8 órán belül)*.
- A telepítést a két állomáson szimmetrikusan kell elvégezni – ugyanazokat a meghajtókat kell használni. Az alkalmazást nem kell közös lemezre tenni, használható a „C:\Program files\Exchgsrv" mappa.
- Kötelező felrakni legalább a „Messaging and Collaboration" és a „Microsoft Exchange System Management Tools" komponenseket.

A továbbiakban feltételezem, hogy az erdő- és tartományelőkészítés már megtörtént. A fürtre telepítés a következő lépésekből áll:

1. Az első állomáson az Exchange 2000-et telepítsük az operációs rendszert tartalmazó lemezre! Ide kerülhetnek az adatbázisok is.
2. Végezzük el a telepítést az elsővel azonos módon a második állomáson is!
3. Hozzunk létre egy csoportot a Cluster Administrator segítségével! Adjuk meg azokat az állomásokat, amelyekre a csoport átköltözhet! A csoport neve legyen a majdani Exchange 2000 virtuális szerver neve.
4. Hozzunk létre IP-címet, hálózati nevet és lemezerőforrásokat! A hálózati név lesz az Exchange 2000 szerverünk neve.
5. A fenti csoportban hozzunk létre az erőforrás-varázsló segítségével egy Exchange System Attendant erőforrást! Tegyük függővé a hálózati névtől és a fizikai lemeztől!
6. Ellenőrizzük le, hogy a „Data Directory" párbeszédpanelen a fizikai lemez erőforrást adtuk-e meg!
7. Indítsuk el a „System Attendant" erőforrást!

Ha mindent jól végeztünk, akkor a „System Attendant" elkezd dolgozni, és az erőforráscsoportot további erőforrásokkal népesíti be.



A System Attendant elvégezte a munkáját

Az Exchange 2000 fürt immár munkára kész. Ha aktív-aktív rendszert szeretnénk építeni, akkor a másik állomáson ugyanezeket a lépéseket végrehajtva új virtuális szervert hozhatunk létre.

A telepítéskor még egy fontos dolgot kell tudni. Ha már létezik Exchange 5.5 rendszerünk, akkor a fürt nem lehet az első Exchange 2000 szerver az adott telephelyen. A telepítés ugyan hiba nélkül lezajlik, de az Exchange konfigurációs információk másolása az Exchange 5.5 kiszolgálóról nem történik meg. Ezért ugyanis a „Site Replication Service" szolgáltatás a felelős – az viszont nem működik fürtön. Azt javaslom, hogy ilyen esetben először egy nem fürtözött Exchange 2000 kiszolgálót keltsünk életre, azon automatikusan létrejön az SRS szolgáltatás. Az Exchange 5.5 eltűnésével a SRS szerepe megszűnik – így nem lesz akadálya az azonnali fürtre telepítésnek sem.

A Microsoft SQL 2000 fürtkiszolgálókon

Az SQL Servernél több szempontból is szerencsénk van az Exchange 2000-hez képest. A termék, jellegénél fogva, nem elsősorban szerverszolgáltatások elosztott hálózata *(habár a szerverek közti replikáció képességével rendelkezik)*, sokkal inkább beszélhetünk önmagában álló kiszolgálókról, amelyek mindenekelőtt a felhasználókkal kommunikálnak, esetleg más – nem SQL – szolgáltatások adatkezelését támogatják. Ráadásul a termék címtárszim-

biózisa sem olyan mértékű, mint az Exchange 2000-nél, inkább lehetőség, mint kötelesség. Így aztán nem kell csodálkozni, hogy műszaki értelemben jóval egyszerűbb feladat a fürtözése, mint a levelező-kiszolgálóé volt. Ha még azt is hozzátesszük, hogy egy SQL Server általában előbb válik üzletileg kritikus alkalmazássá, mint egy Exchange, akkor már szinte dupla a haszon. Nézzük tehát, mit kell tudni a fürtbe állításhoz.

Az SQL fürtözési technikával való együttműködése sokat fejlődött, mára már természetes a fürt natív támogatása *(saját erőforrás DLL állományokkal rendelkezik)*. Olyan szép ez a házasság, hogy nincs is szükség külön beállítgatásokra, az adatbáziskezelő fürtre települ, ha azt mondják neki, és a fürtből való eltávolítása is csak a telepítő „Eltávolítás" funkciójával valósítható meg. Akár 16 erőforráscsoportban 16 SQL szervert futtathatunk - ennek persze nincs sok értelme, viszont az aktív-aktív rendszer jól jöhet. Itt az aktivitást ugyanúgy kell értelmezni, mint az Exchange 2000 esetén. Az SQL további simulékonyságát bizonyítja, hogy az egyállomásos fürttől a négyállomásos adatközpontig mindenütt otthon érzi magát a clusteren. Nemcsak a fogalmak azonosak, hanem a szolgáltatások igénybevétele is szabványos. Az SQL fürt pontosan úgy vizsgálja önmaga egészségét, ahogy az elvárható: „looks alive" és „is alive" vizsgálatokat végez rendszeresen, ez utóbbit SELECT @@VERSIONSQL lekérdezéssel. Egy szó mint száz, ezt a két alkalmazást egymásnak teremtették.

Tiszta haszon, hogy a fürt és az adatbáziskezelő így ismerik egymást. Az SQL szerver tulajdonképpen egy erőforráscsoport lesz, ahogy azt vártuk. Az adatbázisok majd a közös lemezerőforráson helyezkednek el, és újra szükségünk lesz egy hálózati név meg egy IP-cím erőforrásra. No meg egy Distributed Transaction Coordinatorra is *(a cikksorozat ötödik részében már megismerkedtünk vele)*, fürtönként egy dukál ebből. Ugyancsak egyetlen példány lesz fürtönként a Microsoft Message Queuing *(MSMQ)* és a „Full Text Search" komponensekből.

Talán sehol sem olyan fontos a kiszolgáló méretezése, mint az adatbáziskezelőnél, ezért javaslom, hogy nagyon körültekintően becsüljük meg a felhasználók számát, a tranzakciók és a tárolt adatok mennyiségét, a szükséges memória méretét. Ebben a vonatkozásban egy fürt éppúgy viselkedik, mint egy hétköznapi kiszolgáló, annyi kiegészítéssel, hogy állomásonként egynél több aktív SQL kiszolgáló példány megnehezíti a dolgunkat, mert a rendszer memóriáját mindkét szerver magánál szeretné tudni. Egy alkalmazás egyszerű esetben Windows 2000 alatt legfeljebb 2 GB memóriához juthat hozzá, ami a 32 bites memóriacímzésből fakad. 2^{32}=4.294.967.296 vagyis 4 GB, ám ebből 2 GB az operációs rendszeré. NT4 alatt ugyan megjelent a BOOT.INI-ben egy /3GB kapcsoló, ám ez nem oldotta meg a problémát, csupán az operációs rendszert korlátozta 1 GB-os címtartományba, így engedve az alkalmazásoknak 3 GB-ot.

A Windows 2000 ennél sokkal elegánsabb megoldást is hozott, ez pedig az AWE, azaz Address Windowing Extensions. A funkciót alkalmazás-hozzáférési csatolókkal *(API)* lehet elérni, vagyis az alkalmazásnak ismernie kell a lehetőséget és a csatolófelületet. Szerencsére az SQL 2000 ezt ismeri, így Advanced Server fürtkörnyezetben hozzávetőlegesen 8 GB memóriát használhat, datacenter környezetben ez 64 GB is lehet. Fontos, hogy az AWE-támogatást be kell kapcsolni, mert az adatbáziskezelő kezdetben nem használja. Az AWE-támogatás példányonként értendő, vagyis ha több erőforráscsoportban több virtuális SQL kiszolgálót hozunk létre, akkor mindegyiknél külön-külön el kell végezni a bekapcsolást. Mindenképpen azt tanácsolom, hogy az AWE használatát alapos teszt előzze meg. Az SQL többi hardverrel kapcsolatos követelményéhez már nem a fürt, hanem az adatbáziskezelő kézikönyveit kell búnyi.

Az SQL telepítése fürtre

Az ismétlés elkerülése érdekében a telepítést úgy kezdjük, hogy már feltételezzük az MSDTC meglétét *(lásd: Farkasokkal táncoló V.)*.

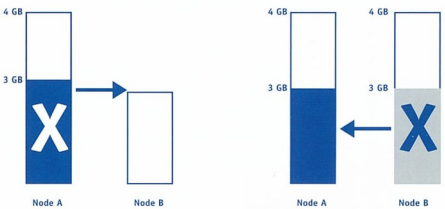
1. Indítsuk el a telepítőt!
2. Az „Installation options" képernyőn válasszuk az „Advanced option"-t!
3. A következő képernyőn válasszuk a „Maintain a virtual server for failover clustering" lehetőséget!
4. Itt adhatjuk meg a fürtből már ismert fogalmak tartalmát: virtuális kiszolgáló nevet választhatunk, IP-címet és alhálózatot adhatunk meg.
5. A „Cluster Management" képernyőn megtekinthetjük a fürt konfigurációját. Telepítéskor a szerver mindkét állomásra fel fogja másolni az SQL állományokat. *(Ezen a ponton lényeges eltérés tapasztalható az SQL Server és az Exchange között: az SQL mindkét állomást „megdolgozza", az Exchange Server viszont nem.)*
6. Végezetül meg kell adnunk a lemezerőforrásokat, amelyeket használni szeretnénk. Ha véletlenül a Quorum lemezét adjuk meg, akkor az SQL erőteljesen tiltakozik. Ez érthető, többször említettük a Quorum és a clustercsoport sérthetetlenségének elvét. Ne telepítsünk semmit a Quorumba!

Ezzel végeztünk is. Aktív-aktív konfigurációnál hasonló módon hozhatunk létre újabb virtuális szervereket.

Már a tervezés során igen fontos, a használat közben pedig kritikus lehet a házirend arról, hogyan költözzön az SQL virtuális szerver az egyik állomásról a másikra, s éppilyen fontos, hogyan költözzön vissza. Az át- és visszaköltözés ugyanis megszakítja a meglévő kapcsolatokat, azokat tehát újra fel kell építeni. Mivel az ügyfelek oldalán általában egy üzleti alkalmazás fut, nem mindegy, hogy miként valósították meg a fejlesztők a kiszolgálótól való elszakadást. Ha a kapcsolat nem épül fel automatikusan, esetleg újra kell indítani az ügyfélprogramokat. Ilyenkor érdemes a kevésbé terhelt időszakra időzíteni az erőforráscsoport visszatérését eredeti helyére, hogy ne okozzon újabb szolgáltatástkiesést a normális üzemre történő visszaállás. Ennek a megoldásnak is lehet hátránya, ahogy azt mindjárt látni fogjuk.

Az átköltözési szabályokat befolyásolhatja az eltérő memóriamennyiség vagy memóriaigény a különböző állomásokon.

Az átköltözési szabályokat befolyásolhatja az eltérő memóriamennyiség vagy memóriaigény a különböző állomásokon.



Rosszul tervezett memóriahasználat SQL fürtözésnél

Az bal oldali ábrán azt látjuk, hogy az egyik állomáson bekapcsolták a 3 GB támogatást, a másikon nem. Az átköltözéssel erőteljesen vissza fog esni a teljesítmény, mert a második állomáson nem „fér el" az SQL Server. A jobb oldali ábrán mindkét állomáson bekapcsolták a 3 GB kapcsolót, ám az aktív-aktív konfiguráció állomásainak teljes fizikai memóriája csak 4-4 GB, tehát egy átköltözés ismét teljesítménycsökkenést okoz majd. A tisztánlátás kedvéért: nem arról van szó, hogy az átköltözés nem történik meg, hanem arról, hogy visszaesik a teljesítmény, a rendszer intenzív lapozásba kezd. Még több virtuális SQL Servernél a probléma hatványozottan jelentkezik. Ez arra sarkallhatja a rendszergazdákat, hogy minél előbb állítsák helyre az eredeti állapotot, még az ügyfelek ismételt leszakadása árán is. Hosszú távon persze csak az alapos tervezés és előregondolkodás segíthet.

Túlzás lenne azt állítani, hogy kimerítettük az összes tudnivalót az SQL és a fürt kapcsolatáról, de egy jó alapozást végeztünk. Befejezésül egy olyan alkalmazást veszünk górcső alá, amelyet nem is a Microsoft gyártott, mégis fürtre kész, és bizonyos helyzetekben nélkülözhetetlen.

Az ArcServe 2000 fürtözött környezetben

Az ArcServe 2000 a Computer Associates által gyártott mentési rendszer. Azt mondhatjuk, hogy az ArcServe 2000 ma *(majdnem)* mindent tud, amit Windows környezetben egyáltalán tudni lehet, beleértve az olyan nyalánkságokat is, mint a szalagos könyvtárak, a RAID szalagok és a fürtök támogatása. Ez utóbbi nemcsak azt jelenti, hogy a szoftver lementi a fürthöz tartozó beállítások halmazát *(a Quorum lemezt és a regisztrációs adatbázis cluster ágát)*, hanem azt is, hogy az ArcServe maga is fürtözhető, így a mentési feladatok minden körülmények között lefutnak, függetlenül attól, hogy épp melyik állomáson aktív egy erőforráscsoport.

Ez a rövid leírás nem lesz elegendő ahhoz, hogy az ArcServe 2000 minden tulajdonságát megismerjük, amelyet fürtözött környezetben magáról mutat. Annyit teszünk csupán, hogy a szoftvert a fürtre telepítjük, illetve megvizsgáljuk, milyen előnyök származnak a szolgáltatás megvalósításából.

Az ArcServe 2000 fürtözése alapvetően három célt szolgál:

1. A hálózaton található ArcServe ügynökök számára nagy rendelkezésre állású ArcServe kiszolgálót biztosít.
2. Szűk keresztmetszet esetén egy második ArcServe 2000 szervert lehet üzembe helyezni *(aktív-aktív fürt)*.
3. Meghibásodás miatt félbeszakadt mentési munkafolyamatokat automatikusan újra lehet indítani.

Az ArcServe 2000-nek saját erőforrás DLL-je van, ám valami oknál fogva azt nem regisztrálja magától, tehát nekünk kell megtenni ezt a lépést. Ezen kívül végképp nincs segítségünk, az alapszoftver telepítése után ránk vár a fürtbe állítás feladata. A munka menete a következő:

1. Telepítsük az ArcServe 2000-et az első állomáson a közös lemezre!
2. Telepítsük a Tape Library opciót!
3. Állítsuk be a közös mentési eszközt az állomáson!
4. Telepítsük az ArcServe 2000-et a második node-on a közös lemezre!
5. Ide is telepítsük a Tape Library opciót!
6. Állítsuk be a közös mentési eszközt az állomáson!
7. Ha Fibre channel környezetben dolgozunk,akkor mindkét állomáson állítsuk be az alábbi regisztrációs kulcsot:

```
HKLM\Software\...\Tapeengine\Config
Érték: EnableSharedDevices: DWORD: 1
```

8. Adjuk ki a következő parancsot mindkét állomáson:

```
cluster resourcetype "ARCserve"/create
↳/dllname:ascluster.dll
```

9. Hozzunk létre egy erőforráscsoportot, vagy jelöljünk ki egyet, és készítsünk egy ArcServe erőforrást például így:

```
cluster resource "My ARCserve" /create /group:
↳"Csoportneve" /type:"ARCserve"
```

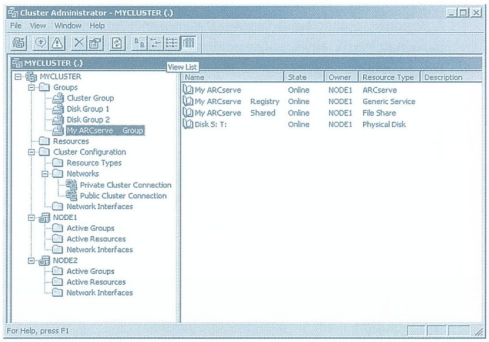
10. Tegyük függővé az erőforrást a fizikai lemeztől!
11. Hozzunk létre egy általános szolgáltatást:

```
cluster resource "My ARCserve Registry" /create
↳/group:"Csoportneve" /type:"Generic Service"
```

12. Adjuk meg a szolgáltatás nevét: astapeengine!
13. Adjuk meg az alábbi regisztrációs kulcsot:

```
"SOFTWARE\ComputerAssociates\ARCserveIT\Base".
```

14. Hozzunk létre egy megosztáserőforrást az alábbi paraméterekkel: Megosztásnév "ARCserve$"!
Elérési út: "S:\Program Files\ComputerAssociates\ARCserve".
Ezzel elkészült a fürtözött ArcServe szolgáltatás.



Az ArcServe 2000 fürtkörnyezetben

Véget ért az erőforrásokkal való ismerkedés. Legközelebb hibaelhárítással és egyéb kacifántos helyzetekkel foglalkozunk majd.

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

Q292757 Exchange 2000 Setup and Installation Top Support Issues
Q239758 SRS Can't Run in Native Exchange 2000 Environment
Q192708 Installation Order, Cluster Server Support for SQL or MSMQ
Q260758 Frequently Asked Questions - SQL 2000 - Failover Clustering
Q225092 Evicting a Node in a Cluster Can Cause Problems in SQL
Q239885 How to Change Service Accounts on a SQL Virtual Server
Q243218 Installation Order for SQL 2000 Enterprise Edition
Q254321 Clustered SQL Server Do's, Don'ts and Basic Warnings

Windows.NET Server Beta 3

Folytatjuk az előzetes szemezgetést a Windows.NET Server új funkcióiból, egyelőre még mindig a béta3 verziót használva – no és persze ismét teszünk egy kis kitérőt a belső fejlesztési folyamatok rejtelmei felé...

Az ütemezés

A legutóbbi tech.net magazinban jól körüljártam a Windows.Net Server korábbi kódnevének hiteles eredetét – éppen csak a legfontosabbról feledkeztem meg: nem derült ki, hogy hogyan is állunk az idővel, milyen fázisban van a fejlesztés, és mi az új termék megjelenésének várható ütemezése.

A Windows.NET Server 2001. november 16. óta van béta3 állapotban, ez a 3590-es számú build. Ez az utolsó béta változat, ezután a tervek szerint két Release Candidate, az RC1 és az RC2 fog megjelenni, majd a végleges, RTM *(Release to Manufacturing, gyártásba kerülhet)* termékkód. A tervek szerint az RC1 2002 második negyedévében, az RC2 a harmadik negyedévben, a végleges kód pedig ez év második felében várható. Valamikor.

A Windows csapat belül természetesen ennél konkrétabb dátumokkal dolgozik. Nekik egész pontosan ki van adva, hogy az RC1-nek *(tegyük fel, valós adatok hiányában pusztán példaként, kitalált időpontokkal szemléltetve)* június elejére el kell készülnie, az RC2 fejlesztésének mondjuk szeptember elején be kell fejeződnie, és lehet, hogy a Köztársaság kikiáltását ősszel itthon már egy végleges Windows.NET Server kóddal ünnepelhetjük. Lehet. Nem tudhatjuk. Ennek *(és a fentieknek)* a megértéséhez ismerjük meg egy kicsit jobban a fejlesztés menetét.

A kutyaeledel

Egyáltalán, mi is az a 3590-es build? Nos, a Windows fejlesztők napi ciklusban dolgoznak *(jó példával elöljárva az MSF, a Microsoft Solution Framework ajánlásait követik betű szerint)*. Minden nap lefordítják, összelinkelik az aznapi termést, ebből lesz az ún. napi build, azaz tulajdonképpen a Windows új változata. Másnap reggel minden tesztelő *(hiszen külön ilyen csapatok is vannak)* gépére az előző napi build kerül feltelepítésre, ezt kezdik nyúzni stressztesztekkel. Maguk a fejlesztők persze több gépen is dolgoznak, nekik nem kötelező a napi buildre átállni, és mindig azon dolgozni *(pedig ez elég motiváló erő lenne ahhoz, hogy ne kövessenek el túlzottan nagy hibákat...)*.

Hibák természetesen keletkeznek *(ezt azért látjuk kívülről is)*. Nem minden napi build stabil, a funkciók fejlődése mellett viszont ezeket a hibákat és azok javításának állapotát is nyomonkövetik, dokumentálják, sőt előre tervezik, mit, mikor, melyik fázisban, milyen más funkciók hibáival együtt javítanak majd *(priorizálva, egy változáskezelési módszert használva, melyről szintén az MSF-ben olvashatunk)*.

Pár napos, esetleg pár hetes időközönként egy-egy build-ről kimondatik, hogy az stabil - ezeket az oprendszer változatokat attól függően hívják ún. IDW vagy IDS buildeknek, hogy mennyire gondolják őket stabilnak. A rövidítés az Internal Development Workstation/Server kifejezést takarja, amiből látszik is, hogy ez az a belső kutyaeledel amit a fejlesztőknek saját maguknak kell megenniük *(eat your own dogfood)*. A Workstation jelentése itt nem az oprendszer típusára vonatkozik. Egy olyan „alacsonyabb" minőségi szintű változatot jelöl, ami elég stabil ahhoz, hogy a belső Microsoft hálózaton saját számítógépen lehessen használni. Az IDS változatban a Server szó magasabb minőségre, stabilitásra utal, amely ahhoz szükséges, hogy ezt a verziót az emberfia nyugodt lélekkel fel merje tenni az éles üzemben működő kiszolgálójára. A legfrissebb IDW vagy IDS build tehát nem a legfrissebb napi változat, hanem mindig a legutóbbi stabil build, amihez, mint kályhához vissza lehet menni. Későbbi, nem közbülső buildekkel is kísérletezgethetnek a vállalkozóbb szelleműek, azonban esetleg érdekes jelenségekkel kell számolniuk – ezek neve éppen ezért TST, azaz teszt build. *(Egyébként meglehetősen leegyszerűsítettem a buildfolyamatot, ennél azért komplexebb a dolog...)* Ahogy írtam, éles környezetben senkinek sem kötelező az átállás a legfrissebb stabil buildekre, de a korábban már említett „együk meg azt a kutyaeledelt, amit főztünk" jelmondatot követve sok fejlesztő használ a belső szlengben ténylegesen „dogfood"-nak nevezett munkaállomásokat és szervereket. Az IDW, IDS buildek nem nyilvánosak *(mint arra az Internal szóból is következtethetünk)*, a külső bétatesztelők sem érhetik el vagy tölthetik le ezeket. Az egyéb napi buildekhez a Microsoft belső hálózatán is csak azok juthatnak hozzá, akiknek az ún. buildszerverekhez hozzáférése van *(a technikai embereknek ezt általában megadják)*. Jelen sorok írásakor egyébként a legfrissebb Windows.NET Server TST build száma 3629, a legfrissebb IDW a 3628-as, az IDS pedig a 3621-es.

> ***...lehet, hogy a Köztársaság kikiáltását ősszel itthon már egy végleges kóddal ünnepelhetjük.***

A béta

Mit jelent akkor tulajdonképpen a béta? Nagyobb mérföldkő a fejlesztés menetében. Mérföldkőhöz akkor érkezünk, ha a projektdokumentum szerint valamilyen követelmények teljesülnek – ez bármi lehet, általában a funkcionalitás valahány százalékának elérését, bizonyos konkrét funkciók megírását jelenti, teljesen projektfüggő. Az első bétaváltozatnál szoktak elindulni az ún. technikai bétateszt programok, amikor a szokásos belső tesztelés mellett már külső tesztelőket is bevon a Microsoft. Külső technikai bétatesztelőnek bárki jelentkezhet, ám a bekerüléshez elég sok feltételnek eleget kell tenni. Valójában a technikai bétatesztelő cím nagyon sok kötelezettséggel is jár. Ha valaki ilyenre jelentkezik, akkor vállalnia kell, hogy egy-egy funkcióterületet szisztematikusan tesztel, és a hibákról *(reprodukálható környezettel és hiba-*

leírással) részletes beszámolót ír *(a beépített error reporting már sokat segít ebben, azonban nem mindenre terjed ki a működése)*.
Sokszor felmerül a kérdés, hogyan lehet valaki hivatalos bétatesztelő – nos, a fentiek fényében rávilágítanék arra, hogy a Kedves Kollégák általában nem tesztelni akarnak, mivel ez elég nagy kötöttséggel jár, a legtöbbször pusztán a bétaváltozatokhoz szeretnének hozzájutni, és próbálgatni, tanulgatni az új dolgokat. Minderre van lehetőségünk, a technikai bétatesztelés mellett egy idő után a bétaprogram nyilvános lesz *(általában inkább már a Release Candidate-ek megjelenésekor)*. Ilyenkor a Microsoft sokszor fűnek-fának osztogatja az időbombás béta vagy RC változatokat, gyakran magazinok CD-mellékletekként is megjelennek *(akár nálunk is, ha emlékszünk a Windows 2000 esetére)*.
Attól a lehetőségtől ugyan elesünk, hogy a közbülső IDW/IDS buildekhez hozzájussunk, ám általában jobb a bétákat használni, mivel azok érezhetően stabilabbak, mint a közbülső stabil buildek. Az utóbbi években már megszokott lett, hogy kb. a béta3-as Windows változatok már olyan mértékben stabilak, hogy éles környezetben is használhatók. A Microsoft hálózatán belül már minden tartományvezérlőn megtörtént az áttérés Windows.NET Server Beta3-ra, ezen kívül az ún. Joint Development Programban *(JDP – az áprilisi Whistler cikkben erről már esett szó)* szereplő vállalatoknál, szervezeteknél is működnek éles környezetben .NET Server Beta3-ak – sőt, ilyenre nálunk is volt példa a Windows 2000 idején, egy hazai távközlési cég működő számlázási rendszerében futottak Windows 2000 Beta 3 Serverek.
Hogyan jussunk hozzá tehát a bétákhoz? Ebben a TechNet *(nem a magazin, hanem a CD!)* vagy MSDN előfizetés szokott segíteni, hiszen ezekkel a termékbétákat is megkapjuk. Vigyázzunk, nem mindegyik típussal! A TechNet esetében csak a TechNet Plus előfizetés esetén jönnek a béták, az MSDN weboldalon pedig egy frappáns összefoglalót találunk arról, melyik előfizetés-típussal pontosan mit is kapunk meg **[1]**.

A Release Candidate

Már csak egy kérdésre kell válaszolnunk: mit takar a Release Candidate kifejezés? Végül is a béta egy-egy nagyobb mérföldkövet jelent, akkor vajon az RC egy még nagyobb mérföldkő lenne? Nos, valahogy így van. Korábban, 4-5 évvel ezelőtt nem létezett a Release Candidate kifejezés, csak alfák és béták voltak. Az RC neve már sokat elárul arról, hogy mit is jelent a fogalom – magyarul valahogy úgy fordíthatnánk, hogy ez a változat jelölt, esélyes arra, hogy végül is kimondasson róla: ő a végleges, kész változat, az ún. Release. *(Érdekes, hogy bár a release szó is az MSF-ből jön, és szó szerint termékkibocsátást jelent, az Internetes szlengben is meglehetősen elterjedt: a cracker, ripper csapatok is ezt használják a feltört szoftverek, videók, zenei anyagok kiadására. Honosított és igésített változata sem idegen már a magyar fülnek, teljesen hétköznapinak számít a „warezt", „gamezt" (összefoglaló néven stuff-ot, stuffát azaz stuffezt) „rilizelni" kifejezés. Örvendetes, hogy ilyen helyekre is eljut a Solutions Framework terminológiája...* 😊*)*.
Az RC-t az is megkülönbözteti a bétáktól, hogy RC fázisban már csak hibajavítás folyik, újabb funkciók már nem kerülnek be a termékbe. Kifelé viszont még ebben a fázisban is kerülhetnek. Bétában ez teljesen természetes, akár technikai, akár marketing-okokból kieshetnek funkciók belőle. *(Lehet pl., hogy az adott funkciókészlet annyira hasznos, hogy érdemesebb külön termékként, plusz pénzért árulni. Ez történt pl. a COM+ terhelésmegosztással, mely a farmkezelési funkciókkal együtt még benne volt a Windows 2000 Advanced Server Beta 3-ban, de időközben kikerült az oprendszerből, és külön szervertermékként került forgalomba: ez az Application Center Server 2000)*.
Éppen ezért hangsúlyozom folyton a cikkekben, hogy egyáltalán nem biztos, hogy a most kezünkben lévő, szemünkkel tapasztalt funkciók a végleges termékben is benne lesznek. Hasonló okok *(de inkább a minőségi mércék)* az időzítést is befolyásolhatják. Brian Valentine *(szintén ismerős személy az áprilisi tech.net számból)* már a Windows 2000 idején is széles *(belső)* plénum előtt közölte, hogy addig nem adja ki a kezéből a kódot, amíg nincs meggyőződve a stabilitásáról – bármekkora nyomást is gyakorol rá a marketing, a sajtó és a közvélemény. Ez azért valahol – valljuk be őszintén – dicséretes.
Egy-egy hiba kijavítása az RC fázisban is csak több napi build ciklusa alatt zajlik le, így ha több kiemelt hiba is összehangoltan javításra kerül egy buildben, az eredmény újabb mérföldkő, azaz újabb RC lesz – ez történik majd a .NET Server RC1 és RC2 változatában is. A végleges kód belső neve az ún. RTM, a Release To Manufacturing kifejezés rövidítése. A fejlesztőcsapat ekkor adja hivatalosan gyártásba a legfrissebb IDS buildet *(most már értjük, mit jelent – ekkora már RTM szóval is illetik)*, tehát ebből indulnak meg a lemezprések. A Windows XP RTM idején ez volt az a kicsit túlspirázott pillanat, amikor helikopterekkel vitték iziben, melegen az arany CD lemezre kiírt végleges kód mesterpéldányát a nagy OEM gyártóknak *(Compaq, HP, Dell, IBM, stb,)* hogy beindulhassanak a gyártósorok. A bolti megjelenés *(az ún. dobozos termék elkészülte)* ilyenkor még akár 4-6 hetet is várat magára, mert a doboz összeállításhoz és gyártásához *(dokumentáció, csomagolás, CD)*, illetve a terítéshez, szállításhoz ennyi időre van szükség. A nagy, népünnepély szintű termékbejelentő eseményekkel éppen ezért mindig megvárja a Microsoft a doboz elkészültét is, hogy a bejelentésen legyen valami materiális reprezentációja annak, hogy íme, itt a termék, elkészült, megérkezett *(ha ekkor feltartjuk a magasba az új Windows dobozt, azért ügyeljünk arra, hogy ne fordítva tegyük, ez sokat tud rontani a hatáson...* 😊*) (Tapasztalat? – a szerk.)*
Térjünk át lassan ismét az új funkciókra – szokás szerint kezdjünk megint az Active Directoryval.

A Release Candidate kifejezés rövidítése.

A küzdelem

Egy tipikus hollywoodi forgatókönyvben a legyőzött ellenfélnek illendő kevéssel a film vége előtt, avagy a folytatásban feltámadnia, hogy újra le lehessen győzni. A Windows 2000 Active Directory is produkálhat elvileg ilyen horrorisztikus jeleneteket – egy objektum törlésénél ugyanis *(mint az ismeretes)* nem fizikailag töröljük az adott objektumot az adatbázisból, csupán egy sírkövet *(tombstone)* replikálunk szerte a világba, melyek majd a mögöttük lévő objektumokkal együtt a garbage collection metódus később szépen kitakarít, ahogy azt már az Exchange 5.5 óta megszoktuk. A sírköveknek élettartamuk van *(TTL – Time To Live)*, azaz egy idő után eltűnnek, maguk után rántva a takart objektumot. Ha ez az időtartam jóval nagyobb, mint teljes rendszerünk legnagyobb replikációs késleltetése *(azaz előbb-utóbb a legtávolabbi helyre is biztosan eljut ez a sírkő, mielőtt kihalna)*, akkor minden egyes tartományvezérlőről valóban törlődik az összes objektumpéldány. Gond csak akkor van, ha túl kicsire vesszük ezt a paramétert *(tombstone lifetime)*. Ez viszont már a mi hibánk, és egyébként is csak hangolási kérdés.

Mi történik azonban akkor, ha egy tartományvezérlő huzamosabb ideig ki van valahol kapcsolva *(vagy ami a szempontunkból ugyanaz – jó hosszú ideig nincs hálózati kapcsolata egyetlen más DC-vel sem)*? Lehetséges, hogy ezalatt a kieső idő alatt a többi DC-n törölt objektumok sírköveinek élettartama már lejárt, eltűntek, így nincs mit replikálni. Ilyen esetben fordulhat elő, hogy a semmiből egyszer csak régen törölt objektumok jönnek elő. Feltámadnak, újra le kell győznie őket a rendszergazdának. Feltéve, ha észreveszi őket. Nos, igen, ez elég komoly biztonsági rés lehet. A .NET Serverben a repadmin eszköz megoldást ad erre a problémára: azonosíthatjuk és törölhetjük az ilyen semmiből újra feltámadt objektumokat.

Windows NT4 Domain Controller emuláció

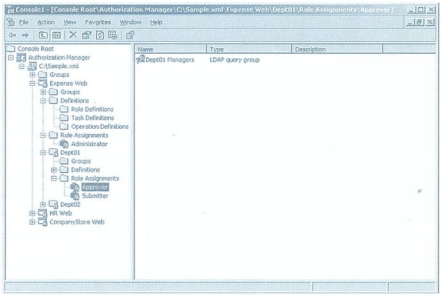
Már hallom is a Kedves Olvasót, amint felkiált: „Hiszen ezt ismerem! Ez már Windows 2000-ben is volt!"
Higgadjunk le, kérem. Nem a PDC Emulátorról van szó.
Korántsem. A probléma a következő *(Q298713)*:
Egyelőre még NT4-es tartományunk van *(végre egy izig-vérig magyar forgatókönyv) (nem valami Amerika... – a szerk. gyenge megjegyz.)*. Kliensoldalon *(az újonnan vásárolt gépeken)* viszont már elkezdtünk Windows 2000-re, vagy éppenséggel Windows XP-re átállni.
Csakhogy, amint elhatározzuk, hogy átállunk Active Directoryra, és annak rendje-módja szerint Windows 2000-re frissítjük az NT4 PDC-t, abban a szent pillanatban az összes Windows 2000 Professional SP2 és Windows XP Pro kliens mindenfajta ügyesbajos címtárral kapcsolatos forgalma az egyetlen Windows 2000 DC-hez fog becsapódni. Bármennyi NT4-es BDC legyen is még a láthatáron *(akár a helyi LAN-on, ugyanazon a telephelyen)*, a 2000/XP kliensek ezután csak a Win2000-es *(vagy .NET Serveres)* DC-vel hajlandóak szóba állni. Ha nincs ilyen, akkor jó az NT4 is. Ha viszont akár egyetlen is van a közelben *(telephelyen)*, akkor csak ő jó. Ezek a kliensek már csak ilyenek. By design.
Ha bírja ez az egy DC a terhelést *(no itt lesz valami amerikai a történet – Magyarországon bírni fogja...)*, semmi gond. Ám ha nem bírná, rábírhatjuk a .NET Servert, hogy *(legalábbis DC szempontból)* tegyen úgy, mintha NT4 lenne. PDC vagy BDC? – jöhet a kérdés. Ez most mindegy, mert nem a replikáció, hanem egyéb címtárműveletek *(hitelesítés, LDAP lekérdezés, stb.)* szempontjából fog NT4 akcentussal beszélni. Mindezt a registryben lehet beállítani, és csak kétállású a kapcsoló *(ki és be – tehát nincs PDC és BDC, erre továbbra is van PDC emulátor FSMO szerepkör)*. Ez a lehetőség egyébként SP2-től kezdve a Windows 2000 Serverben is használható. Ha félünk a túlterheléstől, minden frissített DC-n bekapcsolhatjuk az NT4 emulációt – a terhelés pedig megoszlik. Egész addig így hagyjuk a rendszert, amíg kettő vagy több szervert nem frissítünk, s amikor már úgy gondoljuk, hogy bírni fogják a DC-k a kliensek rohamát, kikapcsoljuk az NT4 emulációt.
Ahogy említettem, ekkora terhelés nem valószínű hazai környezetekben, egy telephelyen. Miért lehet ez mégis hasznos hazánkban? Szerte az országot járva hallani híreket olyan egyedi fejlesztésű, *(de akár még gyári)* eredetileg NT4-re írt vállalati alkalmazásokról, amelyek valami *(teljesen ismeretlen)* fejlesztői hóbort miatt NT4 PDC-t követelnek.

Az API például olyan új függvényeket is tartalmaz, amelyekkel az egyes felhasználókhoz (szerepkörhöz) tartozó jogosultságok hatékonyabban összegyűjthetők

Windows 2000/.NET Serveren persze van PDC Emulator szerepkörünk, de csak egy darab. *(Ezért hívják single operations masternak, ugyebár.)* Ha több telephelyen *(de egy tartományban)* szeretnénk az alkalmazást használni, akkor azok mindenütt igényelnének egy NT4 PDC-nek tűnő dolgot. Persze az ntdsutil seize parancsával elvileg csinálhatunk PDC Emulator-ból is többet, sokat, bármennyit. *(Ezt hívják hackelésnek. Már a gondolat is halálos! Felejtsük is gyorsan el!)*
Windows 2000 SP2-vel és .NET Serverrel viszont több DC-n is bekapcsolhatjuk az NT4 emulációt, és ki tudja, lehet, hogy ennyi elég is az alkalmazásnak, ezzel működni fog. Nem kell várni, míg a fejlesztők átírják, tesztelik, ami jó időbe telik *(és általában meg is akadályozza erre az időre a vállalaton belül a Windows 2000 Serverre való átállást, hiszen ilyen egyedi fejlesztéseknél általában kulcsfontosságú üzleti alkalmazásokról van szó)*.

Biztonság

A Windows.NET Serverben a biztonsági szolgáltatások területén is sok apró újdonság, továbbfejlesztés található. Az újdonságokról már olvashatunk is egy külön műszaki tanulmányt **[2]**.
Roppant érdekes újdonság az ún. Authorization Manager MMC Snap-in *(érdekességét az is fokozza, hogy a béta3-ban ezt még nem találjuk meg, későbbi buildekbe került csak bele...)*. Ez egy olyan felület, mellyel a jogosultságokat ténylegesen a céges működésre leképezve, üzleti folyamatok szerint szabályozhatjuk. Szerepköröket használhatunk, melyek a felhasználók valós életben betöltött munkakörével azonosak *(pl. főnök, beosztott, pénzügyes, marketinges)*, műveleteket definiálhatunk a szerepkörökhöz, melyeket engedélyezünk vagy tiltunk. Mindezt persze eddig is megoldhattuk csoportokkal, de a leghasznosabb funkció az, hogy az erőforrásokat, objektumokat többféle halmazba, ún. scope-ba rendezhetjük. Ezeknek a halmazoknak összevontan kezelhetjük a hozzáférésért és a megfelelő szerepkörökhöz jogosultságokat rendelhetünk.



Az Autorization Manager MMC Snap-In felülete (amit a béta3-ban még hiába keresünk...)

Mielőtt ráfognánk az új MMC beépülő modulra, hogy csak marketing-csingilingi, tudnunk kell, hogy mögötte egy ún. Authorization API készlet is megjelent, melynek feladata, hogy közbenső rétegként eltakarja a hozzáférés-vezérlési listák *(ACL-ek)* és egyes rendszererőforrások objektumszerkezetének működését olyan absztrakciós felületet képezve, ami pl. a fenti szerepköröket és objektum scope-okat biztosítja. Az API például olyan új függvényeket is tartalmaz, amelyekkel az egyes felhasználókhoz *(szerepkörökhöz)* tartozó jogosultságok hatékonyabban összegyűjthetők – így régi nagy álmunk is teljesül, egyetlen rendszerhívással megkaphatjuk, hogy adott felhasználónak az NTFS fájlrendszeren milyen jogai vannak – anélkül, hogy olyan szkriptet

vagy programot kellene írnunk, ami végigfut az összes könyvtáron és fájlon, hogy megnézze az ACL-eket. *(Ilyen programról olvashatnak a 24. oldalon, a Szerszámosláda rovatban.)*

Headless Server

Pasquale DeMaio egy nagyon szerény francia srác, akinek őszintén szólva nem irigylem a helyzetét. Ő a Microsoftón belül a .NET Server távoli felügyeleti technológiáinak, az ún. Headless Server fejlesztésnek *(tech.net magazin első Whistler előzetes)* felelős Program Managere. Főnökei, kollégái, beosztottjai, de még a barátai is csak úgy emlegetik – a Headless Program Manager...

Itt talán nem is az a legfontosabb fejlesztés, hogy monitor, billetnyűzet, egér nélkül használhatjuk a rendszert. A távoli felügyeletnek két nagy területe van: az ún. in-band és az out-of-theband, azaz a sávszélességen belüli és kívüli felügyelet. Az esetek és az idő 98%-ában sávszélességen belül akarunk felügyelni, erre ismerjük az eszközöket – terminálszolgáltatás, MMC modulok távoli csatlakozása, távoli registryszerkesztés, stb. Az érdekes és új dolog a sávszélességen kívüli, ún. Emergency Management System, EMS. Számos hardver akkor is támogatja már a távoli felügyeletet *(egy külön szervízprocesszorral, külön hálókártyával, vagy éppen modemes behívás lehetőségével magába a PC vasba)*, ha a hálózat és így a gépen lévő oprendszer nem elérhető. A Windows.NET Server szabványos felületen *(ez az EMS)* hozzáférhetővé teszi az oprendszert akkor is, amikor egyébként hálózatról még, vagy már nem érnénk el – boot folyamat alatt, a telepítés karakteres képernyő részénél, RIS telepítés alatt, blue screen képernyőnél. A hangsúly itt a szabványosságon van. Egyes gyártók *(pl. Compaq, HP)* eddig is árultak olyan szervízkártyákat, amelyek speciális eszközillesztőivel szintén elérhető volt távolról grafikus felület, ezek azonban egyedi, gyártó- *(és szervervas-)* specifikus megoldások voltak, és így meglehetősen sokba kerültek.

Az EMS a szabványos VT-UTF8 terminálprotokollt használja, mely visszamenőlegesen kompatíbilis a jól ismert VT100-zal *(színeket, nem angolszász karakterkészleteket és egyéb nyalánkságokat támogat még pluszban a VT100-hoz képest)*. Ha tehát van egy VT100 terminálunk *(Valljuk be, sokan őrzünk ilyet otthon nosztalgiából csak azért, hogy megmutathassuk a gyerekeinknek, hogy a Mátrix c. filmben micsoda anakronisztikus ellentét a VT-100 monitorhoz csatlakoztatott Microsoft Natural Keyboard. A terminál az terminál. Annak a billentyűzet is része. Szentségtörés ilyen erőszakot tenni vele...)*, vagy egy VT100 illetve VT-UTF8 terminálemulátor-szoftverünk, soros porton keresztül – hardvertámogatás esetén egy másik hálókártyán vagy modemen keresztül – távolról elérhetjük a gépünket, és matathatunk akár a BIOS-ban is. Amennyiben a hardver kiadja ezt a terminálkimenetre *(ehhez a .NET Server-nek sok köze nincs)*... A VT-UTF8 csatlakozáson keresztül dolgozhatunk a korábban szintén említett SAC konzollal *(Specialty Administration Console)*, mely hasonlít a Recovery Console-hoz, ám sokkal kevesebb *(kb. 8-10)* utasítást támogat. Elindíthatunk viszont más alkalmazásokat belőle *(pl. cmd.exe-t, normál parancssori felületet)*, amivel a helyzet rögtön sokkal rózsásabbá válik, hiszen akár szkripteket is futtathatunk így távolról egy olyan gépen, amit egyébként a hálózaton keresztül nem érünk el.

Vállalati felhasználóknak *(zárt szerverszobákhoz)* és kisvállalatoknak *(Internetszolgáltatónál hosztolt gépek)* is hasznos ez a funkció. A Netacademia munkatársainak – hogy egy valós életbeli példával szolgáljak – sokszor gondot okoz, hogy a BIX-re *(a hazai Internet főgerincre)* közvetlenül csatlakozó www.netacademia.net szerveren *(ez a jó tulajdonság az itt működő hivatalos magyar Microsoft letöltőközpont sebességén is tükröződik)* a két felhasználós admin módú terminálszolgáltatásban „bentragadnak" a felhasználók, és két kapcsolat után már többen nem csatlakozhatnak. *(Ha már a konkrétumoknál tartunk: nem a termék a hibás, hanem jelen cikk szerzője szokta magát bennfelejteni – a szerk.)* A terminál MMC, amivel ki lehet dobni az embereket, RPC-vel kapcsolódik, ami persze nem megy át a tűzfalakon. Ilyenkor tehát trükkös módszereket kell használni a kiléptetéshez *(vagy csak a kicsatlakoztatáshoz)*, hogy egyáltalán távolról felügyelni lehessen a gépet. Más esetben ez a patthelyzet odáig is fajulhat, hogy nincs más lehetőség, újra kell indítani a gépet, hogy visszakapjuk a távoli kontrollt felette. Munkaidőben persze oda lehet telefonálni a szolgáltatónak, hogy *(megfelelő előszó alapú hitelesítési protokoll használata után)* nyomja meg a reset gombot – éjszaka és hétvégén ez persze nehézkes.

Piaci forgalomban elérhetőek egyébként olyan ún. terminálkoncentrátorok, amelyekbe modemen keresztül betárcsázva megkapjuk a VT-UTF8 konzolt. A koncentrátor soros porton több géphez is csatlakozhat, melyek között váltogathatjuk a modemes terminálkapcsolatot. A soros vonalon kívül a koncentrátor a 220-as vezetékeket is koncentrálja magában, és minden ilyen tápkábelhez gépenként megszakítót is tartalmaz. A megszakítót egy VT-UTF8 konzol paranccsal vezérelhetjük, azaz ha minden szakad, távoli emberi segítség nélkül telefonon keresztül adhatunk hard resetet a gépnek.

Ha már a terminálszolgáltatásnál járunk, a Windows 2000-es Terminal Connection Manager MMC-t nem igazán tudjuk lebeszélni arról, hogy indítás után az összes tartomány összes gépét felsorolja nekünk csatlakozási lehetőségként a bal oldalon. Ha viszont egy konkrét szerverhez akarunk kapcsolódni, azt nem tudjuk így megadni *(csak ha terminál ablakból indítjuk az MMC-t)*. Én pórul is jártam emiatt a minap, mert nem tudtam a fenti okokból bentragadt admin kapcsolatokat kilőni *(vagy csak lecsatlakoztatni)* RAS-on keresztül – a távoli gépem ugyanis nem volt tagja egy tartománynak sem, munkacsoportban ment. Nekem is trükköznöm kellett. Mindenesetre nem grafikus felületről oldottam meg a problémát *(nekem legalább volt RPC-m, mivel közvetlenül a céges hálózatba tárcsáztam be)*.

Mindezt azért említem, hogy tudjuk értékelni: a .NET Serverben a terminal MMC alapértelmezésben nem sorolja fel a tartományokat és gépeiket, illetve bármikor megadhatunk NETBIOS, FQDN néven vagy IP-címen egy terminálszervert, amit felügyelni akarunk.

Folytatás júniusban – maradt még téma bőven, és részben még az előző számban igértekkel is adós maradtam...

Horváth Tamás
MCSE2000
Microsoft Magyarország

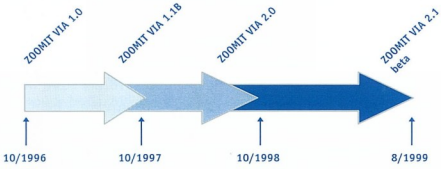
A cikkben szereplő URL-ek:
[1]http://msdn.microsoft.com/subscriptions/prodinfo/levels.asp
[2]http://www.microsoft.com/windowsxp/pro/techinfo/planning/pkiwinxp/default.asp

Microsoft Metadirectory Services *(II. rész)*

A legutóbbi Tech.net magazinban áttekintettük, hogy általában milyen elvárásaink lehetnek egy metacímtár szolgáltatástól. Most konkrétan a Microsoft Metadirectory Service *(MMS 2.2)* működésével ismerkedhetünk meg.

Zoomit és Microsoft

A Windows 2000-ben megjelent Active Directory nagyvállalati címtárszolgáltatás belső kifejlesztése mellett a Microsoft a metacímtár-szolgáltatásra, a korábban tárgyalt identitáskezelési problémákra is megoldást kívánt adni. Ezen megfontolásból 1999 júliusában megvásárolta a ZOOMIT céget, mely akkor a metadirectory megoldások területén piacvezetőnek számított. Az akvizíció eredményeként a Microsoft, Windows 2000 Server platformon a címtárszolgáltatással kapcsolatos biztonsági, nagyvállalati és metacímtár szolgáltatásokat egyaránt biztosítani tudja. Idővel persze a rendszer *(Microsoft Metadirectory Services néven)* beépül majd a Windows elosztott rendszereinek sorába. Az MMS tehát egy, a piacra már bevezetett termék. Hosszú élettörténettel és kiterjedt nagyvállalati vásárlói körrel rendelkezik – ez megnyugtató. A továbbiakban a jelenlegi változatról, az MMS 2.2-ről lesz szó.

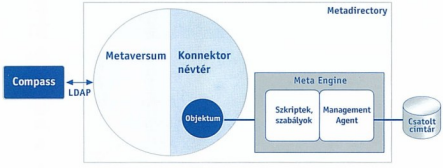


A Zoomit cég VIA termékének története

Microsoft Metadirectory Services 2.2

Az alábbi ábrán koncepcionális szinten láthatók a metacímtár-szolgáltatás komponensei:

- csatolt *(forrás)* címtár *(connected directory)*,
- Meta Engine,
- konnektor névtér *(connector namespace)*,
- metaverzum *(metaverse)*,
- kliens *(client)*.



A Microsoft Metadirectory Services felépítése

A fenti illusztrációban a metacímtárt LDAP-on keresztül elérő ügyfél az MMS-ben lévő Compass nevű kliensszoftver, de az MMS a HTTP *(tehát web böngészővel történő)* elérést is támogatja.

A metacímtár névterei

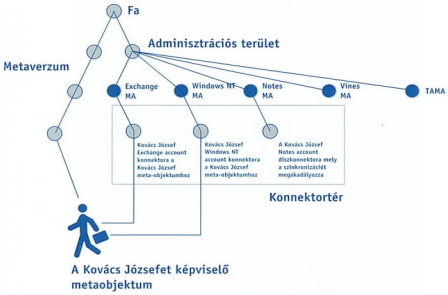
A metacímtár két névtérre oszlik:

A konnektortér az a hely, ahová a csatolt címtárból importált elemek megérkeznek. Minden egyes csatolt címtár saját területtel rendelkezik a konnektortérben. A konnektortér speciális objektumok, ún. konnektorok és diszkonnektorok gyűjteménye. Ezek utóbbiak közt az a különbség, hogy a konnektor rendelkezik egy attribútummal, mely a hozzá kapcsolt metaverzum-objektum *(metaobjektum)* Distinguished Name *(egyedi, megkülönböztető név)* paraméterét tartalmazza. A diszkonnektornak ugyanez a paramétere nincs kitöltve, így valójában csak helyet foglal egy bejegyzés számára a csatolt címtárban – a hivatkozott metaobjektum lehet hogy létezik, lehet hogy nem. A konnektorok teremtik meg a kapcsolatot a metaobjektumok és a csatolt címtár objektumai között, lehetővé téve köztük a szinkronizációt és az attribútumok áramlását. A diszkonnektorok az ilyen jellegű kommunikációt megállítják. A konnektortér objektumai a metacímtár-hierarchiában mindig a Management Agents bejegyzés alatt jelennek meg. Általában kevés attribútumuk van kitöltve, elsősorban közvetítőként működnek a metaverzum és a csatolt címtár között.

A metaverzum a címtárnak az a része, mely a különböző címtárakból az egyesítés során létrejött objektumok közös képét tárolja.

A metaverzum a címtárnak az a része, mely a különböző címtárakból az egyesítés során létrejött objektumok közös képét tárolja. A metaverzum tartalmának legnagyobb része a csatolt címtárakból származik, de lehetőség van a metaverzumba való közvetlen adatbevitelre az objektumhoz kapcsolódó címtár megadása nélkül is.

Nézzük az alábbi ábrán szereplő névteret:



A névtér (Namespace)

Az ábrán a Kovács Józsefet reprezentáló metaobjektum egyaránt tartalmaz tulajdonságokat a Microsoft Exchange-ben és a Windows NT-ben található, Kovács József objektumból. Valamikor a Kovács Józsefet reprezentáló Lotus Notes-béli objektum szintén kapcsolódott a metaverzumban lévő reprezentációhoz, ezt azonban mára szétkapcsolták. Kovács József nem szerepel a Banyan Vines címtárban és nem vesz részt a TAMA *(Together Administration Management Agent)* által megvalósítható felvétel/távozás szituációban sem *(a TAMA-t később részletesen tárgyaljuk – ne keverjük a SAMA-val, ami egy exkluzív szemüvegkeret márka www.samaeyewear.net – a szerk. megj.)*

A következő ábrán látható Compass kliens képernyőképek a két névteret egymás mellett mutatják. Az elsőn a metaverzum képe látható, mely a hierarchia csúcsával *(The Known Universe)* kezdődik, és a fa Daniel Penn bejegyzésig vezető ágait mutatja. A képen látható utolsó bejegyzés az mdserver, mely a metacímtár szervert jelképezi – ennek szerkezetét mutatja a második képernyőkép. Itt láthatóak a Management Agentek *(MA-k)* és azok konnektorterei. Szintén itt található a metadirectory számára létrehozott séma, mely a replikációs és időzítési információkat tartalmazza, és ahol telepítéskor a teljes metadirectoryt kezelő Administrator bejegyzés létrejön.

A metaverzum és a konnektor névterek

Az illusztráción látható, hogy az Autók *(Autos)* objektum *(mely egy osztály)* és a Daniel Penn objektum *(egy személy)* a konnektortérben és a metaverzumban egyaránt léteznek. A konnektortérben található példányok konnektorok *(meglepő módon)*, a metaverzumban található reprezentációjuktól egy ikon különbözteti meg őket. A csatolt címtárat kezelő konnektortér a „Humongous Insurance HR" nevű MA, vagyis egy ügynök, azaz management agent. Más MA-k szintén tartalmazhatnak konnektorokat a saját címterükben létező Daniel Penn reprezentációkra. A metaverzumban található Daniel Penn objektum ezekből mind tartalmazhat bizonyos attribútumokat.

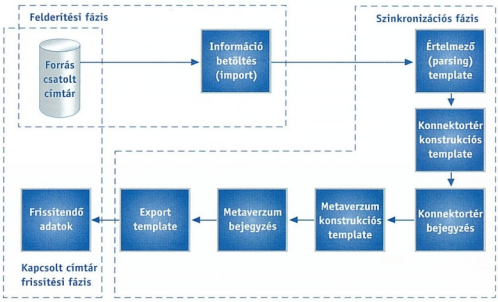
Management Agent-ek *(MA-k)*

A Meta Engine *(metamotor)* vezérli a metacímtár és a csatolt címtárak közti kommunikációt. Minden információt tartalmaz, mely az objektumok létrehozásához, törléséhez, tulajdonságaik integritásának és történetüknek fenntartásához szükséges, és megvalósítja a tulajdonságok birtoklását. A tulajdonságok leírásához szükséges Meta Engine utasításokat a Management Agentek tartalmazzák. Ez utóbbiak speciális objektumok, melyek a csatolt címtárak integrációjához szükséges paramétereket, szkripteket, átalakítási szabályokat, attribútumbirtoklási és áramlási szabályokat írják le.

Az MA-k kezelik a metadirectory kapcsolónévtere, a csatolt címtárak és a metaverzum közti relációkat mind az objektumok, mind az attribútumok szintjén. Az MA-k helyileg az MMS szerveren leledzenek, és *(csatolt)* címtárspecifikusak. Egy MA belső konfigurációja tehát minden csatolt címtár esetében különböző. Fontos megjegyezni, hogy az MMS semmilyen szoftverkomponens telepítését nem igényli a csatolt címtárakat futtató rendszereken.

A szinkronizációs ciklus

Az MA belső megvalósítást tekintve a címtár egy objektuma és egy szolgáltatás *(service)*, mely a címtárszinkronizációt valósítja meg. Meghatározza, hogy a szinkronizációt hogyan kell elvégezni, és végre is hajtja azt. Az MA működésének három fázisát egy szkript írja le, ezek a felderítés *(discovery)*, szinkronizáció, és a frissítés *(update)*:



A Management Agent szinkronizációs fázisai

A felderítési és a frissítési fázisok általában közös kódot használnak a csatolt címtár elérésére és a címtáradatok olvasására/írására. Az MA lelke a szinkronizációs fázis, mely import és export mintákat *(template-eket, sablonokat)* és attribútumáramlási szabályokat *(melyeket az MA objektum tulajdonságaként tárol)* használ a változások jellegének megállapítására, mely alapján a metacímtár és a csatolt címtár frissíthető.

Az MMS beépítetten tartalmazza a legelterjedtebb identitáskezelő rendszerekhez szükséges MA-kat *(ügynököket)*, melyek a következők:

- Active Directory Management Agent
- Banyan Vines Management Agent
- Generic Management Agent
- Lotus cc:Mail Management Agent
- Lotus Notes Management Agent

- LDAP alapú Exchange Management Agent
- MAPI alapú Exchange Management Agent
- Microsoft Windows NT Management Agent
- Netscape LDAP Management Agent
- Novell Groupwise Management Agent
- LDAP alapú Novell NDS Management Agent
- Novell Netware Management Agent
- Report *(jelentéskészítő)* Management Agent
- Together Administration Management Agent

A támogatott MA-k alapesetben a legtöbb, az adott gyártó címtárában gyakran használt attribútumot kezelik. A Lotus Notes MA például a Notes OfficeTelephoneNumber attribútumát az LDAP telephoneNumber attribútumára képezi le, illetve a Notes szervezeti felépítés alapján hozza létre a kezdeti hierarchiát.

Az eredeti MA-k szkriptjeinek és sablonjainak módosításával a rendszer működése az egyedi implementációkhoz hangolható. Az Exchange például lehetőséget ad ún. Custom Attribute, azaz egyedi mezők használatára, melyekben tetszőleges, cégspecifikus információ tárolható. Az MA-k szkriptje könnyen módosítható úgy, hogy a MA az Exchange Custom-Attribute-1 paraméterét specialTelephoneNumber-re képezze le a metacímtárban.

A Management Agent Toolkit-ben leírt információk alapján lehetőség van új MA típusok készítésére is. Amennyiben egy rendszer *(pl. egy adatbázis)* képes adatokat fájlba exportálni, könnyen készíthető olyan MA, mely e fájlt feldolgozza, és az adott adattár és a metacímtár tartalmát szinkronizálja. Adatbázisalapú, UNIX, VMS, OS/400 vagy egyéb nem-Windows platformon futó alkalmazások ezzel a módszerrel könnyen integrálhatók a metacímtárba.

Változó adatok kezelése

Vajon ki tartja karban azokat az adatokat, melyek több címtárban is szerepelnek? Ha a metacímtárban is és másutt is változtatjuk az adatokat, azok fokozatosan kiesnek a szinkronból. Az MMS-ben meghatározhatjuk, hogy objektumok mely címtárban hozhatók létre, hol törölhetők, sőt azt is, hogy az egyes attribútumok hol változtathatók.

Az MA-k rendszeresen, időzítetten összehasonlítják az csatolt címtárak és a metacímtár tartalmát. Ha az adatok eltérnek, az MA szinkronizálja azokat. Az eltérések kétfélék lehetnek: valamely címtárban szereplő objektum egy másikban még nem létezik, vagy egy mindkét címtárban szereplő objektum bizonyos attribútumai különböznek. Az MA ezeket az eltéréseket a benne megfogalmazott konfigurációs és szinkronizációs szabályok alapján oldja fel.

Az MMS nem igényel semmilyen szoftverkomponens-telepítést

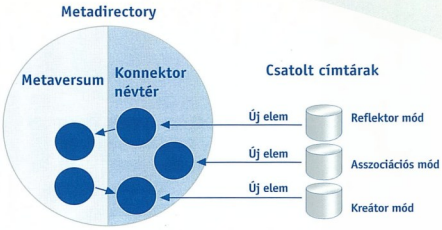
Objektumok kezelése

Az MA üzemmódja *(operating mode)* határozza meg, hogy a metacímtár objektumainak létrehozása, illetve törlése hol végezhető – vagy a metacímtárban *(centralizált menedzsment)*, vagy a csatolt címtárban *(helyi vagy elosztott menedzsment)*. A működési módok a következők lehetnek *(lásd a következő ábrán)*:

Reflector *(reflektor, követő)*: a csatolt címtárban elvégzett beszúrások és törlések tükröződnek a névtérben és a metaverzumban.

Creator *(kreátor)*: a névtérben és a metaverzumban elvégzett törlések és beszúrások tükröződnek a csatolt címtárban.

Association *(asszociáció, társítás)*: a csatolt címtárban elvégzett beszúrások és törlések tükröződnek a névtérben de nem görgetődnek be a metaverzumba.



A Management Agent üzemmódjai

Helyi *(elosztott)* menedzsment

Az MA reflektormódú működése esetén a metadirectory egyszerűen követi a csatolt címtárban elvégzett törléseket és beszúrásokat. Az asszociációs mód gyakorlatilag a reflektor módnak egy esete, melyben a két címtárat megfeleltetjük ugyan egymásnak, de nem integráljuk őket. Ez a mód általában átmeneti állapotot képvisel a reflektor mód felé, mivel lehetővé teszi az importálandó adatok áttekintését, mielőtt azok a metaverzumba kerülnek.

Központosított menedzsment

Ha az MA kreátor módban működik, objektumokat csak a metacímtárban lehet törölni vagy létrehozni, mely műveletek azután végrehajtódnak a csatolt címtárakban is. Ha a csatolt címtár kiesik a szinkronból, a MA automatikusan helyrehozza a kapcsolatot, szükség szerint véve fel vagy törölve elemeket a csatolt címtárból.

Attribútumok kezelése

Az MA üzemmódja kizárólag az objektumok törlésének és létrehozásának helyét határozza meg. A létező objektumok attribútumai akár a metacímtárban, akár a csatolt címtárban módosíthatók, függetlenül az MA üzemmódjától. Eltérő attribútumok esetén az attribútumáramlási *(attribute-flow)* szabályok feladata a megőrzendő változat kiválasztása. Az attribútumáramlási szabályok alkalmazásához azonban a metaobjektumot és a csatolt címtárobjektumot össze kell kötni *(a megfelelő konnektortér-bejegyzés segítségével)*.

Az illesztés *(join)*

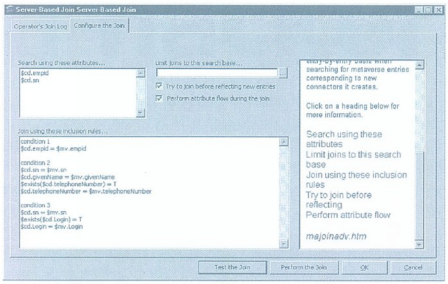
Az illesztés egy metaobjektum és egy adott, konnektortér-beli objektum között teremt kapcsolatot, ezáltal pedig közvetetten a metaobjektumot magához a csatolt címtárbeli objektumhoz kapcsolja. A kapcsolat létrehozható automatikusan, valamiféle szabály alapján, vagy interaktívan.

Változatos illesztési lehetőségekre több okból is szükség van: ha bizonyos objektumok több csatolt címtárban is szerepelnek, de nem akarjuk őket egyetlen metaobjektummá egyesíteni, vagy ha nem dönthető el automatikusan, hogy egy adott metaobjektum illetve egy konnektortér-beli objektum azonos entitást képvisel-e.

Az illesztést háromféleképp végezhetjük el:

A Compass kliens Join Action funkciójával automatikus, kötegelt illesztés hozható létre valamilyen feltétel alapján, melyet aztán tetszőleges alkalommal lefuttathatunk. Általában a csatolt címtárak első betöltésekor használjuk. Az illesztési szabály által nem kezelt kivételek esetében nem konnektorok, hanem diszkonnektorok keletkeznek, melyeket a különálló Account Joiner alkalmazással oldhatunk fel, manuálisan választva ki egy létező metaobjektumot vagy pedig létrehozva egy újat. Az időközben létrejövő,

új objektumok kezelésére az MA beállítható.
Az illesztés automatikus elvégzéséhez fel kell állítani annak szabályait *(a join kritériumokat)*. Erre a Configure the Join *(illesztés beállítása)* felület használható, melyet a Join Action és az MA join funkciója egyaránt tartalmaz.

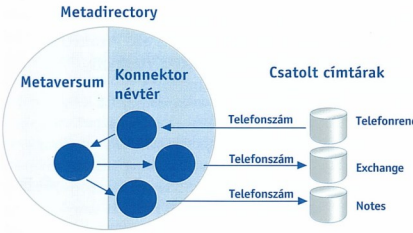


Az illesztési szabályok beállítása

A Join Action a konnektortér valamennyi diszkonnektorát megvizsgálja, a beállított szabályoknak megfelelő párt keresve hozzájuk a metaverzumban. A keresés eredményeként esetleg több lehetséges párosítás is adódik. A Join Action ezután a beszámítási *(inclusion)* szabályok alapján dönti el, hogy ezek közül melyeket tartsa meg. Ha adódik megfelelő illesztés, akkor a két bejegyzés között létrejön a kapcsolat.
A Try to join before reflecting new entries kapcsoló arra utasítja a reflektormódban működő MA-t, hogy az új bejegyzések létrehozása előtt próbáljon illesztést keresni a már meglévő bejegyzésekkel. Az Account Joiner segítségével lehetőségünk van arra, hogy különböző szabályokkal kísérletezzünk, amíg egy megfelelően illeszthető objektumot találunk, vagy újat hozunk létre. Az így kipróbált szabályokat azután beépíthetjük az MA által használt automatizmusok folyamatába.

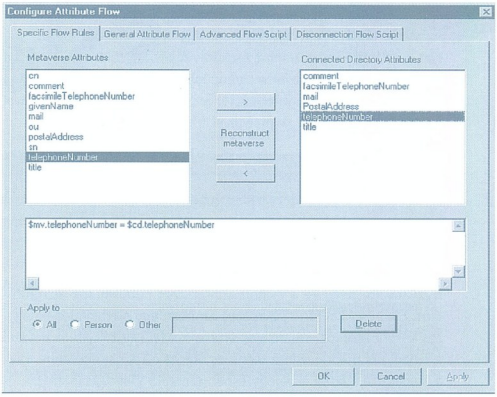
Attribútumáramlási szabályok

Ha egy metaobjektumot több MA is frissít, előfordulhat, hogy felülírják a másik által módosított attribútumokat. Ennek elkerülésére az MA-k attribútumáramlási szabályainak meg kell határozniuk, hogy az egyes attribútumoknak mely csatolt címtár a hiteles forrása.



Attribútumáramlás a metacímtárban

A fenti ábrán a telefonszámok hiteles forrása a telefonrendszer, az MMS-t tehát úgy állítjuk be, hogy a telefonszámok módosítását csak innen fogadja el. Az itt végzett módosítások eljutnak valamennyi rendszerbe, felülírva a Notes vagy az Exchange rendszerben *(esetleg jogosulatlanul módosított)* értékeket.
Az attribútumáramlás ezen szabályai a következő ábrán látható felületen adhatók meg:



Az attribútumáramlás beállítása

A kérdéses attribútumot kiválasztva, és valamely irány-gombot megnyomva létrehozható egy egyszerű szabály a telefonrendszer MA-jében.
$mv.telephoneNumber = $cd.telephoneNumber
Ez a szabály írja le azt, hogy a metaverzum telefonszám-attribútuma a telefonrendszer csatolt címtárában található telefonszámra állítandó. Az Exchange és Notes MA-jeiben a szabály ehhez hasonlít, de ellenkező irányú:
$cd.telephoneNumber = $mv.telephoneNumber
A fenti ábrán látható Advanced Flow Script lehetőséget ad összetettebb, szkriptjellegű szabályok megfogalmazására is. Az ellenőrzésnek e szintje adja a metacímtár elosztott jellegét lehetőséget biztosítva arra, hogy az egyes attribútumokat abban a rendszerben tartsuk karban, ahol az a legkézenfekvőbb: globális adatok *(pl. alkalmazotti azonosítók)* központilag, lokális adatok *(pl. telefonszámok)* helyben kezelhetők.

Adat-transzformáció

Az MA-k bizonyos feldolgozás után adatokat írnak vissza a csatolt címtárakba, az attribútumok egyszerű átadása azonban az alábbi gyakori esetekben nem elegendő:

- Ha valamely csatolt címtárbeli attribútumnak nincs pontos megfelelője a metacímtában.
- A metacímtárban és a csatolt címtárban lévő adatok formátuma eltérő.
- A kapcsolat valamelyik oldalán az információ több mező összekapcsolásából áll csak össze.
- Ha egy mező kiegészítésre szorul annak érdekében, hogy a metacímtárban az egyediséget fenntartsuk.
- Ha a csatolt címtár vagy a metacímtár olyan objektumot tartalmaz, melyet a kapcsolaton nem kívánunk átvinni.

Az MA-k template-ek *(minták, sablonok)* alapján határozzák meg az attribútumok beolvasásának és frissítésének módját. A sablonokat egy magasszintű, interpretált nyelven fogalmazzuk meg, melyet az import program értelmez és az adatátvitel során felhasznál.
Az MA templatenyelv az alábbi lehetőségeket tartalmazza:

- Egyszerű, közvetlen módosítások elvégzése.
- Beépített függvények alkalmazása.

- Más metacímtár objektumok adatainak felhasználása.
- Utasítások feltételes végrehajtása.
- A címtár-frissítésekben érintett *(bevont vagy kizárt)* objektumok meghatározása.

Az alábbi táblázat bal oldali oszlopa egy olyan rekordot mutat, melyet a cc:Mail Export/Import eszközével készítettünk a metacímtár importeljárások előkészítése során. Jobboldalt az a beolvasó sablon látható, mely az előző információt a csatolt címtár attribútumaival és ideiglenes változók segítségével írja le.

File	Contents Template
Name: Dunn, Matt	Name: $v_surname,$v_givenname
Locn: L	Locn: $cd.zcCcLocation
Addr: ccmP0	Addr: $cd.zcCcPostOffice
Cmts:	Cmts: $cd.description

Nyilvánvaló, hogy ez a csatolt címtár nem publikál sok attribútumot. Ennél sokkal több szükséges azonban egy metacímtár-bejegyzés létrehozásához. Valamilyen értékekkel fel kell töltenünk például a bejegyzés Distinguished Name és object class paramétereit. Ezek a kiegészítő adatok a bejegyzésben szereplő identitásadatokból és a MA által tárolt egyéb információkból együttesen állíthatók elő. Ennek megvalósításához a beolvasó *(parsing)* sablonok kiegészítéseként minden MA tartalmaz néhány konstrukciós sablont is.
A példában szereplő konstrukciós sablonon szemléltetjük ennek működését:

```
If $cd.zcCcLocation = P      (ha ez Postafiók bejegy-
zés)
then
        $mv.zcoc = organizationalUnit      (object class)
        $cs.zcoc =
        zcCcMailPostOffice,zcAliasThing,Top
        $mv.organizationalUnitName =
        $v_surname(,$v_givenName)
        . . .
else
        $mv.zcoc = zcPerson
        $cs.zcoc = zcCcMailBox,zcAliasThing,Top
        $mv.commonName = $get_substring("$v_givenName\
        (^ $v_surname)", "", "@"
        . . .
endIf
```

A példa teljes mélységű megértése természetesen nem szükséges, jól szemlélteti azonban, hogyan használható a sablon az információ átadására.
Érthető, hogy a reflektor MA a konstrukciós sablont használja egy új objektum létrehozása során. Mi a helyzet viszont akkor, ha a megfelelő metaverzum bejegyzés már létezik, és a konnektorhoz már illesztettük? Mi történik az attribútumáramlási szabályokkal? A válasz az, hogy a konstrukciós sablonok csak a beállítandó adatok értékét állítják elő, de nem feltétlenül jegyzik be azt – ez már az egyes MA-k attribútumáramlási szabályai alapján történik.

A felvétel/távozás megvalósítása

A metacímtár egyik legfontosabb szolgáltatása az egyes vállalati rendszerekhez való hozzáférés biztosítása az alkalmazotti ciklus folyamán. Amikor az illetőt felveszik, szüksége lesz bizonyos erőforrásokra *(pl. fájlok, nyomtatók)*, szolgáltatásokra *(pl. e-mail)*. Egy-egy előléptetés alkalmával ezen jogosultságok, szolgáltatások módosulhatnak.
Az MMS ezt a funkcionalitást a TAMA *(Together Administration Management Agent)* segítségével valósítja meg. A TAMA egy speciális MA, mely más MA-k működését képes koordinálni, képes arra, hogy a csatolt névterekben konnektorokat, ezáltal pedig a csatolt címtárakban objektumokat hozzon létre. Ezt a működést általában egy új bejegyzés létrehozása váltja ki. A TAMA ezután más MA-kban konnektorokat hoz létre, melyek aztán a csatolt címtárak objektumait képviselik. Például egy új bejegyzés létrehozása a HR konnektortérben eredményezheti egy új Windows NT fiók, egy Exchange postafiók és egy Notes azonosító létrehozását. A TAMA az MA-k által alkalmazott szkriptnyelvet használja az új fiókok létrehozási módjának pontos leírására. Hasonlóan, ha valaki távozik a cégtől *(törlik a személyügyi alkalmazásból)*, a TAMA gondoskodhat a csatolt címtárakban hozzá tartozó accountok felszámolásáról vagy letiltásáról.

Metadirectory Services és az Active Directory

Az MMS 2.2 a következő szolgáltatásokkal egészíti ki az Active Directoryt:

- Több, egymással összekapcsolt címtár szinkronizációja csillagtopológia mentén.
- Egy objektum különböző képeinek automatikus egyesítése egyetlen, központi objektummá. Bár nem reális azt feltételezni, hogy egy szervezet valamennyi identitásadata 100%-ban egyesíthető, az MMS rugalmassága nagyban segíti ennek elérését.
- Valamely címtárnak egy-egy attribútum hiteles forrásaként való kijelölése.
- Az üzleti folyamatok támogatása a felvétel/távozás folyamatok megvalósításával.

A cikk az alábbi Microsoft műszaki tanulmány alapján készült:
Microsoft Metadirectory Services Concepts and Architecture
http://www.microsoft.com/windows2000/techinfo/Ahowitworks/activedirectory/MMSintro.asp

Reszkessenek a hackerek?

Április 1.-től az eddiginél jóval nehezebb helyzetbe kerülhetnek azok, akik az Internetet ártó szándékkal használják. Ha valamilyen, a Büntető Törvénykönyvben meghatározott tényállást valósítanak meg, akkor adott esetben sokéves börtönbüntetéssel is számolhatnak. A korábban ismertek mellett új elkövetési alakzatok is megjelentek a törvényben, ezzel korábban bűncselekménynek nem minősülő magatartások is e körbe kerültek.

A magántitok védelme eddig is fontos célja volt a jogalkotásnak, és annak kifürkészése és rögzítése már alapesetben is öt évig terjedő szabadságvesztéssel volt büntethető. Lényeges hiányossága volt azonban a Büntető Törvénykönyvünknek *(közismert rövidítéssel Btk.-nak)*, hogy a jogvédelmet nem terjesztette ki sem a világhálóra, sem a belső hálózatokra. Ez történt meg a Btk. legutóbbi módosításával. Bűncselekményt tehát az követ el, aki a más részére hírközlő berendezés útján vagy számítástechnikai rendszeren másnak továbbított közleményt, adatot kifürkész, és az észlelteket technikai eszközzel rögzíti. A közlemény kifürkészésén kell érteni minden olyan magatartást melynek célja a közlemény tartalmának jogosulatlan megismerése úgy, hogy másnak a titkát aprólékos gonddal felderíti.

Az e-mail jogosulatlan elolvasása tehát önmagában még nem bűncselekmény, csupán szabálysértés. Ellenben ha annak tartalmát fájlba elmentjük, bizony hosszas hűsölésre számíthatunk a rácsok mögött. Aki azonban az e-mail tartalmához úgy fér hozzá, hogy ehhez valamilyen számítástechnikai rendszer védelmét biztosító intézkedést játszik ki, bűncselekmény elkövetőjévé válhat.

Az Európa Tanács keretében előkészített, és Budapesten, 2001. novemberében elfogadott Informatikai bűnözés elleni Egyezmény égisze alatt terjesztette ki a Btk. a bűncselekmények körét a hackerekre. Új tényállás került be a törvénybe a 300/C. § alatt számítástechnikai rendszer és adatok elleni bűncselekmény címen. A bűncselekmény elkövetője bárki lehet, aki a rendszer védelmét szolgáló intézkedés megsértésével vagy kijátszásával oda jogosulatlanul belép, vagy belépési jogosultsága kereteit túllépve, illetőleg azt megsértve bent marad. Az e-mail jogosulatlan kifürkészése önmagában tehát nem biztos, hogy bűncselekmény, ám aki más levelezőrendszerébe jogosulatlanul behatol, még akkor is bűncselekményt követ el *(és egy évig terjedő szabadságvesztéssel számolhat)*, ha nem olvassa el az ott található üzeneteket.

Felmerülhet a kérdés: vajon bűncselekményt követ-e el az a kedves munkatárs vagy netán főnök, aki távollétünkben leül a gépünk elé, és azt bekapcsolva szépen végigolvassa az e-mailjeinket? Hiszen ez esetben szó sincs a védelmi intézkedés kijátszásáról. Jó hírem van azok számára, akiknek az elképzelés kapcsán már borzolódott a szőr a hátukon: igen, ez is bűncselekmény, bár csupán vétség, és így egy évig terjedő szabadságvesztés lehet a maximum, amit ezért kedves kollegánk kaphat.

Az igazi hackerek azonban nem elégszenek meg azzal, hogy egy adott rendszerbe behatoljanak, ott ennek nyomát is akarják hagyni. Hiszen ügyességüknek, tudásuknak ez lenne fényes bizonyítéka. Nos, aki illetéktelen behatolás során a rendszerben valamilyen adatot jogosulatlanul megváltoztat, töröl, vagy hozzáférhetetlenné tesz, vagy akár a rendszer működését jogosulatlanul akadályozza, már két év hűsöléssel számolhat. Aki pedig mindezt nemcsak a saját ügyességének tanúsítására, hanem bankszámlája hizlalásra is fel kívánja használni, az okozott kár nagyságától függően akár tíz évet is kaphat.

Az a szakértő, aki tudásával, ismereteivel segíti azt a hackert, aki önmaga nem elég ügyes, vagy netán a számítástechnikai rendszerbe való belépést lehetővé tevő kódot, jelszót elkészíti, esetleg megszerzi, vagy a feltörést biztosító kódot árusítja, ugyancsak bűncselekmény elkövetőjévé válik, és két évig terjedő szabadságvesztéssel számolhat. *(Mostantól nem tarthatunk security tanfolyamot? Az IPSec cikk szerzője már csomagolhat is, megy a dutyiba! – a szerk.)*

Az említett, Informatikai bűnözés elleni Egyezmény 9. Cikkében foglaltaknak megfelelően a Btk. a tiltott pornográf felvétel vagy felvételek vonatkozásában új elkövetési magatartásokat nyilvánít bűncselekménnyé. Az Egyezmény kiindulópontja a gyermekekről készült pornográf felvételekkel kapcsolatos magatartások *(készítés, megszerzés, átadás, tartás, forgalomba hozatal)* átfogó, teljes tilalma. A törvény is ennek megfelelően rendeli büntetni az ilyen cselekményeket, és új elkövetési magatartásokat iktat be.

> **Az a szakértő, aki tudásával, ismereteivel segíti a hackert, ugyancsak bűncselekmény elkövetőjévé válik, és két évig terjedő szabadságvesztéssel számolhat.**

Tiltott pornográf felvételek megszerzését és tartását három évig terjedő szabadságvesztéssel rendeli büntetni a törvény. Ezzel tehát a felhasználó is büntethetővé vált, ha nem tesz mást, mint a világhálón található pedofil tartalmú oldalakat letölti saját gépére és ott tárolja azokat. Súlyosabb minősítést von maga után egyetlen pornográf képfelvétel akár ingyenes, akár ellenszolgáltatás fejében meghatározott személy részére történő átadása, illetőleg kínálása is. Vagyis vicből se küldjünk ilyeneket!

A törvény nem változtat a hatályos törvény által is büntetni rendelt elkövetési magatartások miatt megállapítható büntetési tételen: a tiltott pornográf felvétel készítését, forgalomba hozatalát, ilyen felvétellel történő kereskedelmet, valamit annak nagy nyilvánosság számára hozzáférhetővé tételét továbbra is két évtől nyolc évig terjedő szabadságvesztéssel rendeli büntetni.

Fontos megjegyezni, hogy bűncselekménynek csak a gyermekpornográfiát tartalmazó felvételekkel kapcsolatos magatartások minősülnek. Az egyéb erotikus, vagy éppen pornóképek megszerzése, átadása, akár az azokkal való kereskedelem még akkor sem bűncselekmény, ha éppen durva, hard pornóról van szó.

A most ismertetetett rendelkezések csak a Btk. új, április 1.-től hatályos szabályai. Természetesen nem kerülhetett sor a teljesség igényével mindazoknak a cselekményeknek a bemutatására, amelyeket a Btk. bűncselekménnyé nyilvánít, és amelyeket jellemzően nagy számban éppen az Interneten lehet elkövetni.

Dr. Mayer Erika
mayer@nadas-mayer.hu

Támadás az IPSec ellen

Amire az IPSEC csomagszűrőnél figyelni kell

A tech.net magazin III. évfolyamának első számában olvashattuk, hogyan használható a Windows 2000 IPSec implementációja, mint egyszerű csomagszűrő. Sajnos ez a tulajdonság csak melléktermék, tervezéskor egyáltalán nem szerepelt a célok között. Ezt hallva az olvasó rögtön gyanút foghat, vajon nem származik-e ebből valami alapvető probléma.

A gyanút fogott olvasónak igaza van, ugyanis az IPSec szűrők alkalmazásánál vannak kivételek, vagyis egyes csomagokkal – még ha meg felelnek is a szűrőfeltételnek – nem foglalkozik az IPSec. A cikk további részében először ezeket a kivételeket sorolom fel, majd mutatok egy példát a csomagszűrésre mindössze abból a célból, hogy lehessen mit megkerülni. A példát IPSecpol paranccsal építem fel, egy kis többletet adva az előző cikkhez. A példa után pedig belevágunk a legizgalmasabb részbe, vagyis hogyan élhetünk *(vissza)* a kivételekkel. Eközben néhány hasznos eszközt is megismerünk. Szerencsére, az utolsó részben kiderül, miként lehet mégis értelmet adni „barkács" csomagszűrőnknek.

A kivételek

Az **[1]** URL-en található tudásbázis cikk alapján a következő hálózati csomagok hatolnak át a szűrőn:

- Broadcast: ha a csomag célcíme valamiféle broadcast cím, vagyis a csomagot több gépnek szánták.
- Multicast: ha a csomag célcíme valamiféle multicast cím *(a multicast címek tartománya a 224.0.0.0 -239.255.255.255)*.
- RSVP *(Resource Reservation Protocol)*: A 46-os számú IP protokoll, amit QoS *(Quality of Service)* szolgáltatásoknál használ a Windows 2000.
- IKE *(Internet Key Exchange)*: Az IKE-t használja az IPSec, hogy biztonságosan cseréljék ki a felek a paramétereket *(ha a szűrőfeltétel akcióparamétere megköveteli azt)*, és osztott titkosító kulcsot hozzanak létre. A lényeg, hogy a Windows 2000 mindig UDP csomagokat használ az IKE forgalomhoz *(500-as forrás és célcímmel)*.
- Kerberos: ez a Windows 2000 alapvető authentikációs protokollja, amit az IKE is használhat IPSec authentikációra. Mivel a Kerberos eleve titkosított, nincs szükség arra, hogy IPSec-kel titkosítsuk. Csomagszűrőnknél ez úgy köszön vissza, hogy az minden olyan UDP és TCP csomagot átenged, aminek a forrás- vagy célportja 88-as.

A nagyobb gyakorlattal rendelkező olvasó már biztosan látja, hogy milyen súlyos problémával állunk szemben, de még mielőtt belemennénk, lássuk a megkerülendő példát.

Példakonfiguráció

Legyen a példakonfiguráció a lehető legegyszerűbb. A célunk az, hogy csak meghatározott gépek érhessék el az Internetre kirakott webszerverünket, aminek IP-címe 192.168.2.4. Legyenek e „távoli" gépek mondjuk a következő IP-címekkel ellátva: 192.168.2.1 és 192.168.2.2. Az ezt megvalósító IPSecpol **[2]** parancssorozat a következő:

```
IPSecpol -w REG -p "PacketFilter" -r "HTTPin"
↳ -f 192.168.2.1+192.168.2.4:80:TCP
↳ -f 192.168.2.1+192.168.2.4:443:TCP
↳ -f 192.168.2.2+192.168.2.4:80:TCP
↳ -f 192.168.2.2+192.168.2.4:443:TCP -n PASS
IPSecpol -w REG -p "PacketFilter" -r "DenyAll"
↳ -f *+192.168.2.4 -n BLOCK
IPSecpol -w REG -p "PacketFilter" -x
```

Az így kreált politikát pedig a következő paranccsal törölhetjük:

```
IPSecpol -o -w REG -p "PacketFilter"
```

Lássuk a paraméterek jelentését:

- -w Típus: Tartomány. Az IPSecpol parancsnak kétféle működésmódja van: dinamikus és statikus. Dinamikus módban a megadott parancsok csak a „IPSec Policy Agent" szolgáltatás leállásig maradnak érvényben. Statikus esetben a gép újraindításával sem veszik el a beállítás. Ez utóbbit érjük el a –w paraméterrel. A Típus változónak két értéke lehet: REG, ami a regisztrációs adatbázisban való eltárolást jelent vagy DS, ami az AD-ban tárolja az általunk megadott politikát. Az utóbbi esetben a Tartomány változón keresztül megadhatjuk, melyik tartományról van szó.
- -p Név: a politika neve, amire a parancs vonatkozik. Ha még nem létezik ilyen néven, a parancs létrehozza azt.
- -r Név: a hozzáadandó szabály neve. Ha már létezik, akkor a parancsnak megfelelően módosul.
- -f A.B.C.D/mask:port=A.B.C.D/mask:port:protocol. Itt adjuk meg a szűrőfeltételt, vagyis azt, hogy a szabály milyen csomagokra vonatkozik. Mint a példában látható, egy szabályhoz több szűrőfeltétel is tartozhat. A szűrő két részből áll, amit vagy egy + vagy egy = jel választ el egymástól. Ha + jelet használunk, akkor lényegében két szűrő jön létre, vagyis a megadott feltétel érvényes lesz mindkét irányban. A két rész a forrás és cél beállításait tartalmazza. Az IP-cím helyén megadhatunk fix IP-címet vagy egy IP-címtartományt alhálózati maszkkal együtt. A * jellel bármely IP-cím előfordulását megengedjük, a 0-val saját IP-címünkre hivatkozhatunk, és megadhatunk DNS nevet is. A cél- és forrásport megadásának helye elég egyértelmű a példából, de akár el is hagyható, hasonlóan a protokollmezőhöz, ami a szűrőfeltétel legvégén szerepel. Lehetséges értékei ICMP, UDP, RAW ,TCP.
- -n PASS vagy BLOCK: itt definiáljuk a szabályhoz tartozó akciót *(esetünkbenben PASS)*, ami engedi a szűrőfeltételeknek

megfelelő csomagot. Lehetne BLOCK is, ami megakadályozná a csomagok átjutását.

- -x vagy –y: Akárcsak a GUI-t *(Graphical User Interface, az egerészős változat)* használó változatnál, itt is hozzá kell rendelni a politikát a géphez *(ha a helyi regisztrációs adatbázisba mentettük)*, csak azután lép működésbe a szűrőnk. Erre való az –x kapcsoló. A hozzárendelés megszüntetéséhez az –y kapcsolót használhatjuk.
- -o: Segítségével törölhetjük a létrehozott politikát.

Láthatjuk, hogy milyen egyszerűen hozható létre parancssorosan a csomagszűrő. A fenti példából könnyedén gyárthatunk parancsállományt, aminek segítségével rugalmasan konfigurálhatjuk gépünket. Az IPSecpol maga is támogatja ezt a –file paraméterrel, ami után megadott állományból a parancs végrehajtja az utasításokat. Miért járjuk ezt az utat? Talán azért, mert vannak, akik jobban szeretik a parancssoros, mint a GUI eszközöket. Azt se felejtsük el, hogy nagyon jó, ha rendelkezésünkre áll az IPSec politikát létrehozó parancsállományunk: bármikor és bármelyik gépen előállíthatjuk a kívánatos állapotot. Akkor is hasznos lehet, ha csak ideiglenesen akarjuk felvenni a kapcsolatot egy másik géppel IPSec-en keresztül, és ezért nem akarjuk megváltoztatni az AD-ben tárolt beállításainkat. Még egy fontos dologra szeretném felhívni a figyelmet: nem teljesen ugyanúgy viselkedik az IPSecpol parancs, mintha a GUI-t használtunk volna. A különbségekről a **[3]** címen lehet bővebben olvasni.

...hogyan lehet portszkennelő eszközzel a szűrőt megkerülni

Megkerülés

Az alábbi teszteket a VMWare **[4]** szoftver segítségével végeztem, ami azt jelenti, hogy három operációs rendszer futott egyszerre a tesztgépen. Az áldozat egy alap Windows 2000 Server volt, a támadó operációs rendszereken pedig a Windows 2000 Professional futott.

Mi a rosszszándékú támadó első lépése? A támadható rendszerek, szolgáltatások felderítése. Mi is ezzel kezdjük. Megnézzük, hogyan lehet portszkennelő eszközzel a szűrőt megkerülni *(a portok végigpróbálgatásával szolgáltatást keresve a célrendszeren)*. Erre a célra szerintem legjobb program az nmap, ami letölthető az **[5]** címről. A program eredetileg Unixos világra készült, de a forrást letöltve könnyedén lefordítható Windows platformon is *(a forrásfában megtalálható a Visual C++ projektállomány)*. Az nmap rengeteg paraméterrel rendelkezik, ezért itt csak a példához szükségeseket adjuk meg. A megkerüléshez azt a tulajdonságot használjuk ki, hogy megadható, milyen forrásportról folyjék a portok végigpróbálgatása.

```
nmap -sS -P0 192.168.2.4
nmap -sS -g 88 192.168.2.4
```

Az első parancs futása az adott környezetben 1096 másodpercig tartott, aminek örülhetünk, hiszen sikerült jelentősen lelassítani a támadót *(a kezdőknek ez gyakran elveszi a kedvét)*, IPSec nélkül mindössze 6 másodpercig tartott ugyanez. Sajnos a kapott eredmény már nem ilyen szép:

Port	State	Service
88/tcp	closed	kerberos-sec

Hogy mért nem, ahhoz először tekintsük át az nmap működését. A használt paraméterek jelentése a következő:

- -s[S, U, X...]: A szkennelés típusát adja meg, ami többek között lehet UDP, Xmass, FIN, NULL, de akár végig is pingelhetünk egy címtartományt *(erről részletesebben szintén az **[5]** címen lehet olvasni)*. Amit esetünkben használunk, az a SYN szkennelés. A TCP portokat próbálja végig, méghozzá úgy, hogy a háromlépcsős TCP kapcsolat felépítésből csak az első két lépcsőt valósítja meg. Elküldi a SYN csomagot, és a visszajövő SYN/ACK-ra ACK helyett RST-vel válaszol. *(Akinek ez kínaiul van, annak javaslom a tech.net 2001/III. számában a NetMon cikk elolvasását: a TCP csatornát elemzi.)* Mivel a TCP kapcsolat nem épül ki, alkalmazásszinten nem történik bejegyzés a naplókba a kísérletről. Ezért is szokták ezt a módot „stealth", rejtőzködő szkennelésnek nevezni. Nyilván a portot csak akkor jelenti nyitottnak, ha SYN/ACK-ot kap válaszul.
- -P0: Az nmap a szkennelés előtt megpróbálja megállapítani, hogy az adott IP-címek élnek-e. Egyszerűen megpingeli *(nem csak ICMP-n keresztül)* őket. Ha ez nem volt sikeres, akkor bele sem kezd a munkába. Ez a tulajdonság kapcsolható ki a –P0-val. Nagyon sok tűzfal úgy van beállítva, hogy az nmap kezdeti próbálgatásai sikertelenek maradnak. Ilyen a mi IPSec szűrőnk is.

Azért látom problémásnak az első eredményt, mert ha egy porton nem fut szolgáltatás, akkor a protokoll szerint a kapcsolatot kezdeményező gépnek RST/ACK választ küld vissza a célgép. Már az előző cikkben is szerepelt, hogy az IPSec egyik előnyös tulajdonsága, hogy nem küld semmiféle választ a sikertelen próbálkozásokra, de amint láthatjuk, az első port-szkennelésünk eredményéből *(a kivételként kezelt Kerberos protokoll miatt)* a 88-as portról visszakaptuk az RST/ACK-t. A port ugyan zárva van, de erről értesítést kaptunk *(RST)*. Ez baj. Ebből már gyaníthatja is a támadónk, hogy itt egy IPSec–kel védett Windows 2000-ről van szó. Ekkor próbálkozik a második nmap utasítással, ahol a –g paraméter hatására a program minden kérést 88-as portról **indít.** Úgy teszünk tehát, mintha Kerberossal kopogtatnánk – pedig dehogy! Így a kapott eredmény a következő lesz:

```
C:\WINNT\system32>nmap -sS -g 88 192.168.2.4

Starting nmap V. 2.54BETA29 (www.insecure.org/nmap
)
Interesting ports on  (192.168.2.4):
(The 1531 ports scanned but not shown below are in
state: closed)
Port        State       Service
7/tcp       open        echo
9/tcp       open        discard
13/tcp      open        daytime
17/tcp      open        qotd
19/tcp      open        chargen
21/tcp      open        ftp
25/tcp      open        smtp
42/tcp      open        nameserver
53/tcp      open        domain
80/tcp      open        http
135/tcp     open        loc-srv
139/tcp     open        netbios-ssn
443/tcp     open        https
445/tcp     open        microsoft-ds
1025/tcp    open        listen
1031/tcp    open        iad2
3389/tcp    open        msrdp
Nmap run completed -- 1 IP address (1 host up)
scanned in 5 seconds
```

Ebből látható mekkora lyukat okozott a csomagszűrőben, hogy az IPSec kivételként kezeli a Kerberos protokollt. Most lássuk, hogyan lehet ezt mélyebben kihasználni.

IIS

Olyasmire van szükségünk, amivel elérhetjük, hogy az általunk kiküldött összes csomag a 88-as portról induljon ki. Ezt legkönnyebben egy port-átirányító programmal tehetjük meg. Két programról érdemes itt beszélni. Az egyik a **[6]** címről letölthető fpipe, a másik a **[7]** címről letölthető winrelay. *(Mindkét oldalon érdemes több időt is eltölteni.)* A továbbiakban az fpipe-on keresztül mutatjuk a példákat. Lássuk az elsőt:

```
fpipe -v -i 192.168.2.3 -l 80 -r 80 -s 88
192.168.2.4
```

A –v paraméter hatására a program bőbeszédűen ecseteli, hogy éppen mit tesz. Az –i határozza meg, hogy melyik interfészen, az –l, hogy melyik porton hallgatózzon. Az –r után megadott portra továbbítja a csomagokat az –s-sel megadott forrásportról. Az utolsó paraméter pedig a cél IP-cím. Vagyis a fenti parancs hatására, ha a 192.168.2.3-as IP-címre kapcsolódunk a 80-as porton, akkor minden, amit elküldünk átirányítódik a 192.168.2.4-es IP-cím 80-as portjára, és a csomagok forrásportja 88-as lesz. Így a támadó már szabadon elérheti a rendszerre installált webszervert. Windows esetében elég nagy az esélye, hogy ez IIS lesz, de támadónk óvatos, úgyhogy leellenőrzi. Ez megtehető egy egyszerű telnettel is, de van erre sokkal alkalmasabb eszköz: a Netcat. Szintén a Unixos világból érkezett, de Windowson is remekül működik. A **[8]** címről tölthető le. Gazdag funkcionalitásából most csak ennyit használunk:

> ***Olyasmire van szükségünk, amivel elérhetjük, hogy az általunk kiküldött összes csomag a 88-as portról induljon ki.***

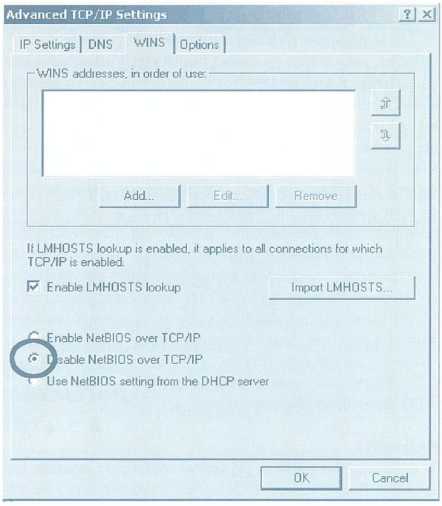
```
C:\WINNT\system32>nc -p 88 192.168.2.4 80
HEAD / HTTP/1.0


HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sun, 24 Mar 2002 09:15:08 GMT
Connection: Keep-Alive
Content-Length: 1270
Content-Type: text/html
Set-Cookie:
 ASPSESSIONIDGQGGQWZO=KECNIMMCNCIKIGAHHLIAEAMF;
  path=/
Cache-control: private
```

(A figyelmes olvasó észrevehette, hogy nem használjuk ki a fentebb indított portátirányítást. A Netcat –p paraméterével megadható, milyen forrásportról induljon a kapcsolat.) A támadó megnyugodhat, IIS fut a célrendszeren. Gondolom mindenki előtt világos, hogy ha a tesztrendszerünk gazdája, megnyugodva az IPSec nyújtotta védelemben, nem frissíti rendszerét, a támadónak nyert ügye van.

NetBios

A fenti módszerrel a Windows rendszerek achilles-sarkán, a NetBios-on keresztül is támadhatunk. Itt van szükség a harmadik gépre *(192.168.2.5)*, mert így sokkal egyszerűbb a NetBios forgalmat a célrendszer felé irányítani. A közbülső gépen a következő beállítást kell megtenni:



A 139-es port így már nem foglalt

Így a portátirányító program már képes kapcsolatkéréseket fogadni a 139-es porton:

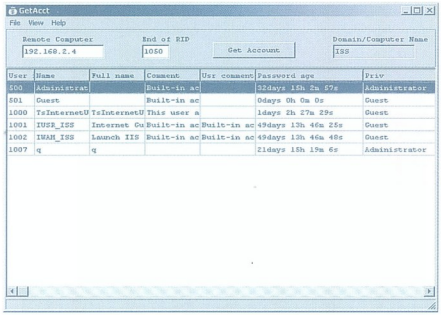
```
fpipe -v -i 192.168.2.5 -l 139 -r 139 -s 88
192.168.2.4
```

Innen már indulhat is lerágott csontként a NetBios-on keresztüli enumeráció *(a hírhedt „null session")*. Számos eszköz létezik, Dunát lehet velük rekeszteni. Itt én három enumerációs, és egy támadóeszközt említenék meg, melyeket belső, betörési tesztjeim során előszeretettel alkalmazok. Az első a Languard Network Scanner **[9]**, ami gyönyörű grafikus felületen tálalja elénk a megszerzett információkat. Véleményem szerint a legkönnyebben használható eszköz, és minden információt megszerez, amit lehetséges. Sajnos esetünkben nem használható: a tesztek során nem volt képes együttműködni a portátirányító eszközzel. *(Az okokat itt most nem részletezném, csak annyit, hogy a probléma gyökere az, hogy a portátirányítás miatt egy forrásportról továbbítódik minden kérés.)* A második egy sima parancssori eszköz *(az enum)*, és a **[10]** címről tölthető le.

```
C:\WINNT\system32>enum -U -M -S -P -G -L
192.168.2.5
server: 192.168.2.5
setting up session... success.
password policy:
  min length: none
  min age: none
  max age: none
```

```
   lockout threshold: none
   lockout duration: 715827288 mins
   lockout reset: 30 mins
opening lsa policy... success.
server role: 3 [primary (unknown)]
...
```

A paraméterek jelentését megkaphatjuk, ha nem adunk meg semmit, így egyszerű, de világos segítséget ad a használatáról. Azért kedvelem, mert lekérdezhető vele a rendszer kizárási politikája. Mint fentebb látható, a támadó örülhet, mert a rendszer nem zárja ki a felhasználót sokszori sikertelen bejelentkezés után, így nyugodtan indíthat nyers erő *(brute force)* támadást a jelszavak ellen (*ami szintén megtehető az enummal.*) A harmadik eszköz, melyet előszeretettel használok, a GetAcct **[11]**, ami rendkívül hasznos nagy felhasználói számnál, ráadásul a RestrictAnonymous=1 beállítás sem jelent számára akadályt.



A GetAcct működés közben

A GetAcct eszköz a felhasználónevek begyűjtésére való. Azért kedveltem meg belső betörési tesztjeim során, mert le lehet menteni a kimenetét csv típusú szöveges állományba. Ez egyszerűen beolvasható Excelbe, és ott különböző szempontok szerint rendezhető *(pl. a „Priv" és utána a „Password Age" oszlop alapján)*, megkönnyítve a könnyen támadható felhasználók kiválasztását. A képen látható, hogy a legsebezhetőbb felhasználónak a „q" tűnik. Valószínűleg tesztelési célokra használták, és ottfelejtették, ráadásul „Administrator" a privilégiuma. Feltételezhető, hogy könnyen kitalálható jelszava van. Itt jön az utolsó NetBios eszköz, mely a pstools csomag része **[12]**. Számomra meglepő módon nem szokták hackereszközként említeni. Talán azért, mert használatához Administrator jogokkal kell rendelkezni a megtámadott gépen *(de ez vonatkozik a többi távoli menedzsment eszközre is, amit ilyenkor szóba hoznak, pl. VNC* **[13]**, *radmin* **[14]**). A szóbanforgó eszköz, a psexec, melynek segítségével bármit lefuttathatunk egy távoli rendszeren:

```
D:\>psexec \\192.168.2.5 -u q -p q cmd.exe

PsExec v1.22 - execute processes remotely
Copyright (C) 2001 Mark Russinovich
www.sysinternals.com


Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>cd \

C:\>
```

Látható, hogy sikerült cmd.exe-t elindítani, mivel a q felhasználó jelszava megegyezett nevével. Olyan ez, mintha „betelneteltünk" volna: a „cd \" parancs már a szerveren futott. A psexec paraméterezése egyszerűen megérthető a példából, így külön nem térek ki rá, inkább továbblépek egy másik sebezhető szolgáltatásra.

SNMP

A hálózat-rendszergazdák kedvenc protokollja a Simple Network Management Protokoll. Segítségével statisztikát gyűjtenek, hálózati térképet készítenek, ellenőrzik a rendszerek állapotát. A protokoll UDP alapú, és a 161-es porton fogadja a kéréseket. *(162-es porton az snmptrap szolgáltatás fut, mely az SNMP ügynökök által generált üzeneteket, riasztásokat fogadja.)* Három ok miatt foglalkozom vele. Először, mert segítségével megmutatható, hogy a portátirányító módszer UDP protokollnál is működik *(lásd Kerberos-kivétel)*. Másodszor, példaként szolgálhat arra, hogy miért veszélyes az, hogy a broadcast csomagokat sem szűri meg az IPSec. Harmadszor, számos hasznos információ van az SNMP szolgáltatás adatbázisában – ami lényegében egy faszerkezet – melyet egy támadó felhasználhat. Az

```
snmputil walk 192.168.2.3 public
↳.iso.org.dod.internet.private.enterprises.
↳ lanmanager
```

utasítás hatására megkapjuk az SNMP-t futtató Windows rendszerekről a felhasználólistát, a futó szolgáltatásokat és a megosztásokat. Az snmputil a Resource Kit része. A fenti utasításban a 192.168.2.3-as IP-címet kérdezzük. A walkkal jelezzük, hogy az utolsó paraméterként megadott részfát szeretnénk bejárni. A public az installálás utáni úgynevezett community név. Ez az SNMP „jelszava". Szóval nemcsak a „NULL Session"-ön keresztül szerezhetünk hasznos információkat...
Az fpipe használata most már egyértelmű kell, hogy legyen:

```
fpipe -v -i 192.168.2.3 -l 161 -r 161 -u -s 88
↳192.168.2.4
```

Egyetlen új paraméter tűnt fel az –u. Arra utasítja a programot, hogy UDP módban működjön. Ezzel az utasítással lehet eredményes az előző snmputil parancs.
A broadcastproblémát a következő utasítással lehet szemléltetni, mely az snmpset paranccsal SET, azaz írási műveletet küld a 192.168.2.255 címre – vagyis a teljes subnetre *(azon belül persze a támadott gépre is)*:

```
snmpset 192.168.2.255 private
↳.iso.org.dod.internet.mgmt.mib-2
↳.sysName.0 s test13
Timeout: No Response from 192.168.2.255
```

A portátirányításra itt nincs szükség, a broadcast bejut a szűrőn. Mint látható, választ nem kaptunk, hiszen a forrásport most nem 88-as volt, így a válaszcsomagot, melynek a célportja nem 88-as, nem engedte át az IPSec. Viszont ha lekérdezzük az adott értéket a MIB-ből *(Management Information Base, így hívják az*

SNMP faszerkezetű adatbázisát), akkor a „sysName" már a mi beállításunkat fogja tartalmazni. A támadás azért volt sikeres, mert az UDP nem kapcsolatorientált protokoll, így elég volt, hogy a parancs elérte a célrendszert, a válasz már nem fontos. A használt eszköz az ucd-snmp **[15]** csomag része, ami szintén a Unixos világból származik, de mint látjuk használható Windows platformon is. Paraméterezése hasonló az snmputil parancséhoz, de itt meg kell adni a változó típusát *(az „s" azt jelenti, hogy string típusú)* és az értékét is. Azért használtuk ezt, mert az snmputil nem támogatja az SNMP set parancsát. *(Az igazat megvallva sok kárt nem okoztunk ezzel a rendszeren, de a veszélyt remélem sikerült vele érzékeltetni.)* Még hozzátenném, hogy Windows 2000-en a telepítés utáni állapotban az SNMP szolgáltatásnál külön kell engedélyezni, hogy távolról írni lehessen a MIB-et.

Nincs minden veszve

Szerencsére az SP1 telepítése után lehetőségünk van beállítani a következő DWORD típusú változót a regisztrációs adatbázisban:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\IPSec\NoDefaultExempt
```

Ha a NoDefaultExempt értéke 1, akkor az IPSec nem kezeli kivételként a Kerberos és az RSVP protokollokat, elhárítva ezzel a legnagyobb veszélyt, de így is megmarad a broadcastcsomagok kivételként kezelése.

Összefoglalás

Tanulság: jóllehet teljesen biztosak vagyunk a gép védelmében, mégse hanyagoljuk el az alapvető biztonsági intézkedéseket, *(mint például a fölösleges szolgáltatások tiltását, a már nem használt felhasználók törlését, az erős jelszavak használatának kényszerítését és a biztonsági javítások telepítését.)*
A célom nemcsak az volt e cikk írása közben, hogy felhívjam a figyelmet az IPSec szűrő problémáira, hanem szerettem volna érzékeltetni, milyen módszerek és eszközök állnak a „rosszoldal" rendelkezésére. Írásom a terjedelmi korlátok miatt nem törekedhet a teljességre, de remélem sikerült növelnem a cikket olvasó rendszergazdák paranoiaszintjét.
És végül, ha rákényszerülünk az IPSec csomagszűrő használatára, ne felejtsük el beállítani a fentebb említett kulcsot!

Tóth László
laszlo.toth@kpmg.hu
KPMG Hungária KFT.

A cikkben szereplő URL-ek:

[1] http://support.microsoft.com/default.aspx?scid=kb;EN-US;q253169
[2] http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/IPSecpol-o.asp
[3] http://support.microsoft.com/default.aspx?scid=kb;EN-US;q254728
[4] http://www.vmware.com/
[5] http://www.insecure.org/nmap/
[6] http://www.foundstone.com/knowledge/assessment.html
[7] http://www.ntsecurity.nu/toolbox/winrelay/
[8] http://www.atstake.com/research/tools/
[9] http://www.gfi.com/languard/lanscan.htm
[10] http://www.cotse.com/tools/netbios.htm
[11] http://www.securityfriday.com/ToolDownload/GetAcct/getacct_doc.html
[12] http://www.sysinternals.com/ntw2k/freeware/pstools.shtml
[13] http://www.uk.research.att.com/vnc/
[14] http://www.famatech.com/
[15] http://sourceforge.net/projects/net-snmp/

DumpSec

NTFS jogosultság listázása

Több alkalommal is felmerült a NetAcademia listáin a kérdés, vajon hogyan lehetne megállapítani, hogy egy felhasználónak milyen tényleges jogai vannak egy könyvtárban, illetve milyen eszközzel lehetne akár egy egész könyvtárstruktúra aktuális hozzáférési szabályairól kimutatást készíteni. Nos, egy beépített eszköz és egy szabadon használható szoftver megoldást jelenthet a problémával küzdőknek.

Röviden a mappa- és állományhozzáférésről

Kevesen tudják, hogy a Microsoft új funkciót épített be a Windows XP-be a fájlhozzáférések ellenőrzésére. A funkció neve „hatályos jogok ellenőrzése", és arra a kérdésre válaszol, hogy pontosan milyen jogai vannak egy felhasználónak egy adott mappában vagy állományon.

Miért? Lehet ez egyáltalán kérdés? Csak meg kell nézni a jogokat és kész. Aki így gondolkodik, az bizony nem tudja, hogyan kell a mappák hozzáférését szabályozni. Dióhéjban a három alapszabály a következő:

1. Tegyük a felhasználókat globális csoportokba!
2. Készítsünk „Domain local" csoportokat, majd a megfelelő mappában rendeljünk melléjük hozzáférési engedélyeket!
3. Helyezzük el a globális csoportokat a „Domain local" csoportokba!

Miért ilyen bonyolult ez? Nos, ha az elnevezéseket megváltoztatjuk, mindjárt kiderül, hogy a valósághoz egész jól idomul ez a modell. Használjuk a „globális" név helyett a „szervezet", „projekt" vagy „team" szót *(pl.: informatika, számvitel, kereskedelmi igazgatóság stb.)!* Amikor egy felhasználót egy ilyen csoportban elhelyezünk, nem teszünk mást, mint vállalatunk szervezeti hierarchiáját csoportokra képezzük le.

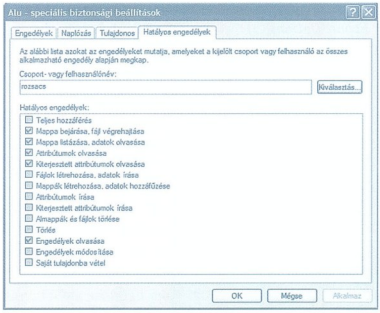
Most gondoljuk végig, hogy hányféle jogot adunk egy mappán! Háromfélét: olvasási jogot a vezetőknek, írási jogot néhány beosztottnak és teljes jogot a rendszergazdáknak.

Ha létrehozunk három lokális csoportot, akkor mindegyikbe beledobhatjuk a megfelelő szervezetet reprezentáló globális csoportot. Így elértük, hogy minimális bejegyzés legyen a jogosultságlistában, és állandó maradhasson a lista, továbbá, hogy más tartományokban létrehozott csoportoknak is ugyanúgy adhassunk jogokat, hiszen a lokális csoportok más tartománybeli globális csoportokat is tartalmazhatnak. Ha felvesszünk egy új munkatársat, elegendő betenni a fiókját a megfelelő szervezetet vagy teamet reprezentáló listába, máris engedélye van több tucat könyvtárhoz, nem kell átnyálazni minden alkalommal az összes mappa listáját.

A megoldás persze hordoz hátrányokat is, például nehéz megállapítani, hogy miért nincs egy felhasználónak valamihez joga, amikor kellene, hogy legyen. Ezt a problémát oldja meg az effektív jogokat megjelenítő panel a Windows XP-ben.

Hatályos jogok ellenőrzése Windows XP-vel

Ha egy mappa biztonsági beállításainál XP alatt a Speciális gombra kattintunk, a Windows 2000-hez képest bővebb panellel találkozunk. Történetünk szempontjából a legutolsó fül az érdekes. Ide kattintva az alábbi ábrát láthatjuk:



A pontos engedélyek egy felhasználó esetében

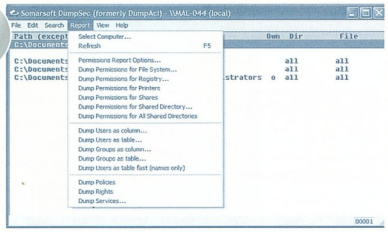
Akármilyen sok csoportunk van is, és akármilyen bonyolult is a mappárendszer, amelyben a jogosultságokat ki kell értékelni, a Windows XP pillanatok alatt elvégzi a kiértékelést. Meg kell adni a felhasználó vagy csoport nevét, és máris minden hatályos jog kiderül.

Egyetlen felhasználó esetén ez egészen jól használható funkció. A teljes jogosultságlista dokumentálására a Systemtools cég birtokába került, de még a Somarsoft által fejlesztett DumpSec nevű ingyenes eszköz alkalmas.

> ***...a Microsoft új funkciót épített be a Windows XP-be a fájlhozzáférések ellenőrzéséhez***

Egy sokoldalú apró program

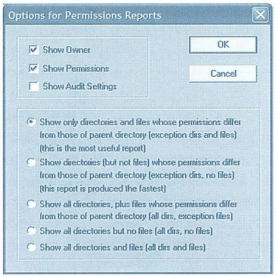
A DumpSec leánykori neve DumpAcl, és a Systemtools weblapján az áll, hogy már nem fejlesztik tovább. Kár, mert nagyon hasznos kis segédprogram. Eredetileg a fájlrendszer jogosultságlistáinak kinyerésére készítették, de aztán egyre bővült a jelentések sora.



A DumpSec puritán felülete és legfontosabb funkciói

Ma már nemcsak a fájlrendszer, hanem a regisztrációs adatbázis, a megosztások és a nyomtatók jogosultságai is listázhatók mind helyi, mind távoli gépről. Adatokat kaphatunk továbbá a szolgáltatásokról, az NT4-es értelemben vett házirendről, de még a felhasználói jogokról is.

A program használatát nagyon gyorsan el lehet sajátítani. Mindenekelőtt ki kell választani a célszámítógépet, amelyről a jelentést el akarjuk készíteni *(Report/Select Computer...)*. Ezután be kell állítanunk, hogy milyen listát szeretnénk *(Report/Permission Report Options...)*.



A leghasznosabb listát az első gombóccal készíthetjük

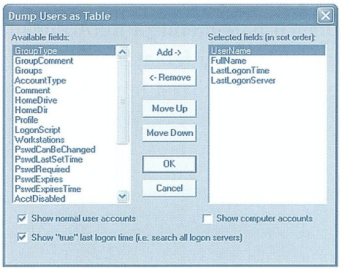
Ötféle lehetőségünk van. A legegyszerűbb, ha csak a kivételeket jelenítjük meg. Ekkor egy mappa vagy állomány tartalmának jogosultsága csak akkor jelenik meg, ha az eltér a szülőmappától. Azért ez a leghasznosabb jelentési forma, mert az almappák és állományok nagyon nagy része a szülőmappától örökli a jogosultságokat, tehát elég a szülőt ismerni, a gyerek úgyis a szülőre ütött. Ha egy mappát vagy állományt áthelyeztünk valahonnan, akkor a jogosultsága eltér a mostani szülőjétől: ez az, ami minket leginkább érdekel.

Persze az is lehet, hogy csak a kivételes mappákkal akarunk foglalkozni *(második gombóc)*, vagy minden mappával és csak a kivételes állományokkal *(harmadik gombóc)*, a többi értelemszerű.

Beállíthatjuk továbbá, hogy jelölje-e a tulajdonosokat a lista, illetve, hogy a naplózási információk megjelenjenek-e. Egy biztonsági audit esetén nagyon hasznos lehet az utóbbi lehetőség.

Miután beállítottuk a lista paramétereit, nincs más dolgunk, mint kiválasztani az elemezésre kiszemelt mappát. Ha az egy távoli szerveren található, sajnos előbb csatlakoztatnunk kell, mint hálózati meghajtót, de ezt tekintsük szépséghibának. Az elkészült listát rögtön elemezhetjük, de el is menthetjük ötféle formátumban, melyek közül az egyik a DumpSec sajátja.

A fájlok és könyvtárak biztonsági tulajdonságainak megjelenítése mellett az alkalmazás egyéb listázási tulajdonságai sem elhanyagolhatók. Különös figyelmet érdemel a felhasználók listázása. Igaz, a DumpSec még az NT4 világot ismeri, az Active Directoryt is csak úgy látja, mintha egy NT4 címtár lenne. Ez azonban nem kevés lehetőséget nyújt a rendszergazdáknak. Minden korábbi tulajdonság tetszőleges sorrendben listázható, sőt, még egy speciális jelenléti ívet is készíthetünk. A DumpSec minden egyes tartományvezérlővel képes felvenni a kapcsolatot azért, hogy megállapíthassa, ténylegesen melyik felhasználó mikor lépett be utoljára. Több telephely esetén ez jelentős hálózati forgalommal járhat, ezért inkább csak kevésbé forgalmas időszakokban javaslom ilyen lista készítését.



Katalógus: Ki nem dolgozik mostanában?

Hab a tortán

A segédprogram készítői a rendszergazdák fejével gondolkodtak, amikor nemcsak a grafikus felületet alkották meg, hanem azt is biztosították, hogy minden funkció elérhető legyen parancssorból is.

Két kötelező paramétere van a programnak: a riport típusa és a kimenetet tartalmazó állomány neve. Minden más opcionális. A szintaxis tehát ez:

```
DumpSec.exe /rpt=<report type> [/options=<value>
/options=<value>...] /outfile=<output file name>
```

A riport típusa tizenhatféle lehet, a kimenet formátumát pedig a /saveas=<format> kapcsolóval állíthatjuk olyanra, amilyen nekünk a legkedvesebb.

Tizennégy nem kötelező kapcsoló finomíthatja az elkészítendő lista formáját és tartalmát. Egy konkrét példa:

```
DumpSec.exe /computer=\\filesrv01
/rpt=share=jelentesek /showalldirs /noowner
/savesas=csv /outfile=c:\adm\logs\users.csv
```

Az alkalmazás csatlakozik a filesrv01 szerverhez, majd a „Jelentesek" megosztás minden könyvtárának jogosultságairól készít egy listát úgy, hogy az nem tartalmazza a tulajdonosokat, az eredményt pedig csv formátumban egy megadott állományba menti. Ezt a parancsot már csak időzíteni kell, máris kész a havi jelentés a hozzáférési szabályokról.

Nincs most alkalom és mód bemutatni minden egyes kapcsolót – szerencsére nem is kell. A DumSecet nagyon jó súgóval látták el, amely nemcsak a parancssori lehetőségeket taglalja, hanem részletes magyarázatot nyújt az elkészített riportok jelöléseiről, a várható teljesítményről, de még az ismert hibákról is.

Akinek felkeltettem az érdeklődését, az látogassa meg a következő címet: **http://www.systemtools.com/somarsoft/**

Sok sikert a jelentések elkészítéséhez!

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

XMLgessünk

Az XML Schema (II. rész)

Az előző részben láthattuk, hogyan kell közvetlen egymásba ágyazással, referenciákkal és típusok definiálásával egyszerűbb sémákat szerkeszteni. Részletesen megnéztük hogyan lehet egyszerű típusokat létrehozni a már meglévő típusokból. Ebben a részben a komplex típusokat tekintjük át. Megnézzük, hogy összetettebb típusok létrehozására milyen fejlettebb nyelvi elemek állnak rendelkezésre.

Összetett típusok

A korábbi példákban a complexType sémaelemet mindig egy-egy elem deklarációján belül használtuk fel, mint például:

```
<xsd:element name="konyv">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="cim" type="xsd:string"/>
...
```

Ebben az esetben összetett típusunknak nincs neve, ezért nem is használható fel másutt, csak a konyv elemen belül. Az ilyen típusdeklarációt anonymousnak hívjuk. Ennek hátránya, hogy nem lehet újra és újra felhasználni, azaz csak azon a helyen hasznos, ahol definiáltuk.
A típusdeklarációkat a schema elem alá is kiemelhetjük. Ha nevet is adunk nekik, bármely elem definiálásánál felhasználhatjuk őket:

```
<!-- konyvsema3.xsd -->

<!-- Az egyszerű típusok deklarációja -->
<xsd:simpleType name="nevTipus">
 <xsd:restriction base="xsd:string">
  <xsd:maxLength value="32" />
 </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="szuletettTipus">
 <xsd:restriction base="xsd:date"/>
</xsd:simpleType>
<xsd:simpleType name="jellemzesTipus">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<!-- Az összetett típusok definíciója -->
<xsd:complexType name="szereploTipus">
 <xsd:sequence>
  <xsd:element name="nev" type="nevTipus"/>
  <xsd:element name="baratja" type="nevTipus"/>
  <xsd:element name="szuletett"
   type="szuletettTipus"/>
  <xsd:element name="jellemzes"
   type="jellemzesTipus"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="konyvTipus">
 <xsd:sequence>
  <xsd:element name="cim" type="nevTipus"/>
  <xsd:element name="szerzo" type="nevTipus"/>
  <xsd:element name="szereplo"
   type="szereploTipus"/>
 </xsd:sequence>
 <xsd:attribute name="isbn" type="isbnTipus"
  use="required"/>
</xsd:complexType>

<xsd:element name="book" type="konyvTipus"/>
```

A példában két összetett típust *(a szereploTipust és a konyvTipust)* deklaráltuk. Látható, hogy a típusokba foglalt elemek hivatkozhatnak további típusokra, így alakul ki a tervezett elemhierarchia.
A típusok a kiemeléssel és megnevezéssel újrahasznosíthatóvá váltak. Nyilvánvaló, hogy ha egy megnevezett típust bármely elem típusaként felhasználhatunk, ezzel munkát spórolunk meg a séma megírása és karbantartása során. Hamarosan kiderül, hogy ennél jóval többet is tudnak a kiemelt és elnevezett elemek. Ehhez azonban még át kell tekintenünk néhány fogalmat.

Tartalomtípusok

Mi lehet egy elem tartalma?

- Lehet üres, azaz nincs se gyermekeleme, se közvetlen tartalma.
- Csak gyermekelemei vannak.
- Csak szöveges tartalma van.
- Gyermekelemek és szöveges tartalom vegyesen található benne.

Emellett még mind a négy esetben attribútumokat is tartalmazhat az adott elem. Hogyan lehet e különböző tartalmakat csoportosítani és formálisan leírni?
Az eddigi példákban már láttuk, hogyan kell olyan összetett típusokat létrehozni, amelyek csak gyermekelemeket és/vagy attribútumokat tartalmaztak, ilyen volt például a konyvTipus.
Tisztán tartalmat hordozott például a cim elem. Hogyan lehet olyan típust definiálni, amely közvetlen tartalmat hordoz, és vannak attribútumai is? Ehhez be kell vetnünk egy új sémaelemet, a simpleContentet:

```
<!-- konyvsema4.xsd részlet-->
<xsd:complexType name="osszetettNevTipus">
  <xsd:simpleContent>
    <xsd:extension base="nevTipus">
     <xsd:attribute name="nem" type="xsd:string">
     </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Azért egyszerű tartalom *(simpleContent)*, mert a komplex típusunknak nincsenek gyermekelemei, csak közvetlen szöveges tartalma és attribútumai.
Az xsd:extension a simpleContenten belül azt jelzi, hogy a base attribútumban megadott **egyszerű típust** *(simple type)* vagy egyszerű tartalmat hordozó **összetett típust leszármaztatás** segítségével egészítjük ki. Ez azt jelenti, hogy az így előállt egyszerű tartalmú összetett típusunk minden jellemzőjében a nevTipussal lesz azonos, csak kiegészítjük egy „nem" nevű attribútummal. Ha a nevTipusnak lennének attribútumai, akkor azokat automatikusan örökölné az osszetettNevTipus összetett típusunk.
Próbáljuk ki mire jutottunk! Ha megnézzük az előző *(konyvsema3.xsd)* forrást, akkor láthatjuk, hogy a szereplő neve egy nevTipus típussal van leírva. Ez nem enged meg semmilyen attribútumot a névben, így a következő részlet nem érvényes az eredeti séma alapján:

```
<nev nem="fiú">Snoopy</nev>
```

Ha azonban a sémában a

```
<xsd:element name="nev" type="nevTipus"/>
```

sort kicseréljük az alábbira:

```
<xsd:element name="nev"
type="osszetettNevTipus" />
```

akkor mindjárt elfogadja a „nem" attribútumot is. Ebből látszik, hogy működik ugyan az új attribútum deklarációja, de még nem látszik a leszármaztatás hatása. Ezt is könnyen ellenőrizhetjük, hisz a nevTipusunk maximum 32 karakter hosszú neveket engedett meg, így ha működik az öröklés, akkor ez igaz lesz az osszetettNevTipusra is.

```
<nev
 nem="fiú">Snoopyiiiiiiiiiiiiiiiiiiiiiiiiiiii</nev>
Validation Error: The 'nev' element has an invalid
value according to its data type.
```

Remek, megy a leszármaztatás, a 32 karakternél hosszabb szövegeket az osszetettNevTipus sem fogadja el.
Most tehát egy egyszerű típusból származtattuk le az egyszerű tartalmat hordozó összetett típusunkat. Ha kell egy olyan típus, ami mindenben azonos az osszetettNevTipussal, csak még kell hozzá egy plusz attribútum, akkor egyszerűen le kell származtatnunk, és ki kell egészítenünk az új típust a plusz attribútummal. Például lehet mindenkinek megszólítása:

```
<nev2 nem="fiú" megszolitas="Mr.">Snoopy</nev2>
Egy ilyen elemet a következő komplex típus ír le:
<!-- konyvsema5.xsd részlet -->
<xsd:complexType name="mrNevTipus">
 <xsd:simpleContent>
  <xsd:extension base="osszetettNevTipus">
   <xsd:attribute name="megszolitas"
    type="xsd:string">
   </xsd:attribute>
  </xsd:extension>
 </xsd:simpleContent>
</xsd:complexType>
```

Ebben a példában egy egyszerű tartalmat hordozó összetett típusból örököltünk.
Azaz *(egyszerű vagy összetett típusból örököltetve)* most már attribútummal, és anélkül is létre tudunk hozni egyszerű tartalommal rendelkező elemeket.
Ha egy összetett típusban vegyesen, gyermekelemeket és közvetlen tartalmat is szeretnénk használni, akkor ezt a mixed attribútum segítségével jelezhetjük. Például ahhoz, hogy a szereploTipusnak lehessen tartalma is a gyermekelemeken felül:

```
<szereplo>Itt ugyan semmi értelme
  <nev nem="fiú">Snoopy</nev>
  szövegnek, de miért ne?
...
```

Az ezt megengedő komplex típus deklarációja a mixed attribútummal jelzi szándékunkat:

```
<!-- konyvsema6.xsd részlet -->
<xsd:complexType name="szereploTipus"
 mixed="true">
```

A gyermekelemek között megbújó szöveges tartalom még abból az időből lehet ismerős, amikor az xml elődjét, az SGML-t, szövegek kiegészítésére használták *(pl. a HTML-ben is ezt tesszük)*.
A mai világban, amikor az xml-t főleg információk átvitelére használjuk, a mixed modell használata nem nagyon elterjedt. Az információt attribútumok és egyszerű tartalommal rendelkező gyermekelemek hordozzák.
Hogyan lehet üres elemet definiálni? Egyszerűen nem írunk gyermekelem-definíciókat a complexType belsejébe, és nem engedjük meg a mixed modellt sem.
Már csak egy dolog maradt hátra. Ha van simpleContent, akkor várhatóan lesz complexContent is. Ezzel hozhatunk létre leszármaztatott összetett tartalmat *(gyermekelemeket, attribútumokat, esetleg közvetlen tartalmat)* hordozó típusokat.
Például minden szereplőről le szeretnénk írni annak korát is *(illékony adat, de miért ne?)*:

```
...
  <kor>13</kor>
 </szereplo>
```

Mivel már van egy összetett típusunk a szereplők leírására, kár lenne veszni hagyni, inkább származtassunk abból egy újabb típust, és egészítsük ki egy „kor" elemmel:

```
<!-- konyvsema7.xsd részlet -->
<xsd:complexType name="bovitettSzereploTipus">
 <xsd:complexContent>
  <xsd:extension base="szereploTipus">
   <xsd:sequence>
    <xsd:element name="kor" type="xsd:int" />
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

Ha belegondolunk, ez a klasszikus, OOP-s értelemben vett öröklés. A sématípusok nagyon hasonlóak a programozott típusokhoz, például C# *(C++)* nyelven így nézne ki *(egyszerűsítve)* a szereploTipus és a bovitettSzereploTipus:

```
class szereploTipus {
  string nev;
  string nev2;
  string baratja;
  datetime szuletett;
  string jellemzes;
}
class bovitettSzereploTipus: szereploTipus {
  int kor;
}
```

A séma típusszármaztatási lehetősége jól kihasználható lenne az xml séma és a programozott osztályok közötti átjárást biztosító sémafordítókban *(schema compilers, .NET-ben a wsdl.exe)*, azonban ezek az eszközök jelen pillanatban nem nagyon élnek ezzel a lehetőséggel. Mi magunk viszont egyszerűen használhatjuk őket. Nem túl bonyolult, a névterek bevezetése után azonban eléggé kacifántos tud lenni. Egyesek úgy érvelnek, hogy ne használjuk a séma öröklési lehetőségeit. Viszont a típusok újrafelhasználásra szükségünk van, és erre van is lehetőségünk: a csoportosítás.

Csoportosítások

Gyakori, hogy több elemet vagy attribútumot több összetett típusban is fel szeretnénk használni. Összeszedhetjük őket egy-egy alaptípusba, és leszármaztatással bárhol felhasználhatjuk őket. Másfelől össze tudjuk terelni őket egy-egy csoportba, és típusainkból a csoportokra hivatkozhatunk!

```
<!-- konyvsema8.xsd részlet -->
<!-- elemcsoport létrehozása -->

<xsd:group name="konyvFoElemek">
 <xsd:sequence>
  <xsd:element name="cim" type="nevTipus"/>
  <xsd:element name="szerzo" type="nevTipus"/>
 </xsd:sequence>
</xsd:group>

<!-- attribútumcsoport létrehozása -->

<xsd:attributeGroup name="konyvAttributumok">
 <xsd:attribute name="isbn" type="isbnTipus"
  use="required"/>
 <xsd:attribute name="rendelheto"
  type="xsd:string"/>
</xsd:attributeGroup>
```

A group sémaelemen belül hasonló tartalmat láthatunk, mint amit az összetett típusok deklarálásánál elemeztünk. Az attribútumok összefogására külön sémaelem van, az attributeGroup. Mivel az attribútumok sorrendje *(definíció szerint)* érdektelen, ebben az esetben nem kell a sequence vagy bármely más sorrendet és/vagy számosságot befolyásoló compositor.
Az elkészült csoportokra *(hivatalos nevük Model Group)* hasonlóan hivatkozhatunk, mint az egyedi elemekre és attribútumokra: a ref attribútumon keresztül. Lássuk a csoportokat alkalmazó konyvTipust:

```
<xsd:complexType name="konyvTipus">
  <xsd:sequence>
    <xsd:group ref="konyvFoElemek" />
    <xsd:element name="szereplo" ... />
  </xsd:sequence>
    <xsd:attributeGroup
ref="konyvAttributumok" />
      </xsd:complexType>
```

Pont olyan, mint az előző számban látott globális elem- és attribútumhivatkozás ref-fel, csak itt csoportokra hivatkozunk.
A Compositorok közül eddig csak a sequence compositort láttuk, ami azt írja elő, hogy a benne felsorolt particle-öknek a megadott sorrendben kell szerepelni a példánydokumentumokban. A particle elemek, elemcsoportok és a bármilyen elemet reprezentáló any elemek összessége. A célnévtérbe elemeket generálni képes séma komponenseket összefoglalóan particle-öknek hívjuk *(szép magyar szó!)*.
További két compositor is van, a choice és az all. A choice, ahogyan a neve is utal rá, a benne definiált elemek vagy elemcsoportok között biztosít választási lehetőséget. Például hozzunk létre egy olyan csoportot, amely egy ember nevét ábrázoló elemeket tartalmaz. A szabály legyen az, hogy vagy meg kell adni az ember teljes nevét egyben, vagy szétbontva családi és kereszt-, valamint egy opcionális második keresztnévre:

```
<xsd:group name="nevek">
  <xsd:choice>
    <xsd:element name="nev" type="xsd:string" />
      <xsd:sequence>
      <xsd:element name="csaladinev"
       type="xsd:string" />
      <xsd:element name="keresztnev"
       type="xsd:string" />
      <xsd:element name="masodikKeresztnev"
       type="xsd:string" min0ccurs="0" />
    </xsd:sequence>
  </xsd:choice>
</xsd:group>
```

Látható, hogy a choice-on belül nemcsak egyszerűen elemeket vagy csoporthivatkozásokat, hanem minden további nélkül további compositorokat is használhatunk.
Az all compositor azt jelenti, hogy a benne felsorolt elemek *(és semmi más, még csoport sem)* bármilyen sorrendben szerepelhetnek a példánydokumentumban. Például a konyvTipus esetén nem biztos, hogy cim, szerzo és szereplo elemek sorrendje fontos. Eddig ott sequence compositor volt, cseréljük azt ki all-ra. Ebből problémáink származnak, mert szereplo elemekből több mint egy is megengedett, viszont az egyértelműség miatt az all compositor ezt nem engedi meg: minden elem maxOccurs értéke *(kardinalitása, számossága)* legfeljebb 1 lehet. Emiatt azt gondolnánk, hogy csak a cim és a szerzo elemek lesznek az all-ban, a szereplo elemdeklarációt pedig belerakjuk egy sequence-be. Azonban egy összetett típuson belül nem lehet csak egy compositor, azaz vagy all, vagy sequence. Mit lehet tenni? Sajnos ezt a feladvány nem lehet megoldani az all compositorral, azaz le kell nyelnünk, hogy meg lesz kötve az adatok sorrendje, marad a sequence.
Ennek ellenére számos helyen jól bevethető az all compositor. Például gyakori, hogy adatbázistáblák tartalmát generáltatjuk le xml-ként. A táblák sorait egy-egy elem reprezentálja:

```
<cuccok>
  <tabla1><oszlop1/><oszlop2/></tabla1>
  <tabla1><oszlop2/><oszlop1/></tabla1>
</cuccok>
```

Ilyenkor a feldolgozás szempontjából általában teljesen érdektelen a táblák oszlopait ábrázoló elemek sorrendje, ezért például a következő séma írhatja le az adatokat:

```
<xsd:element name="cuccok">
  <xsd:complexType>
    <xsd:sequence max0ccurs="unbounded">
      <xsd:element name="tabla1"
       type="tabla1Tipus" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="tabla1Tipus">
  <xsd:all>
    <xsd:element name="oszlop1" />
    <xsd:element name="oszlop2" />
  </xsd:all>
</xsd:complexType>
```

Megszorítások

Gyakran logikai kapcsolat van az xml dokumentum által szállított információk között. Ezen kapcsolatokat, és az adatokban rejlő szabályszerűségeket általában constraintek, megszorítások írják elő a különböző adattároló és szállító rendszerekben. Közismert például, hogy az adatbázistáblákban *(relációkban)* a sorok, azaz a modellezett entitáspéldányok egyediségét az elsődleges kulcs hivatott ellenőrizni, „megszorítani". Ha a táblák között logikai kapcsolat van, akkor az idegen kulcsok lehetséges értékeinek szerepelnie kell a kapcsolódó tábla elsődleges kulcshalmazában *(tartomány épségi szabály)*.

Egy xml dokumentumban gyakran többféle információtartalmat találunk. Az adatok között a sémadokumentumban definiálhatunk kötöttségeket. A sémakötöttségek gyakorlatilag megegyeznek az adatbázisokban megszokott megszorításokkal, hisz a gyakorlatban a legtöbb xml dokumentum forrása egy-egy adatbázis. Az egyik leggyakoribb feladat az egyediség biztosítása valamely elem vagy attribútum értékkészletén. Erre való a unique sémaelem:

```
<!-- konyvsema10.xsd részlet -->
<xsd:element name="konyv" type="konyvTipus">
  <xsd:unique name="EgyediSzereploNev">
    <xsd:selector xpath="szereplo" />
    <xsd:field xpath="nev" />
  </xsd:unique>
</xsd:element>
```

A selector elem xpath attribútumában megadott XPath kifejezés jelöli ki azt az elemet, amelyen értelmezni kell a field elem xpath attribútumában megadott érték egyediségét. Adatbázis-hasonlattal élve a selector választja ki a táblát, a field az egyedi értékeket tartalmazó oszlopot. A példánkban nem lehet két azonos nevű szereplő ugyanabban a könyvben. De különböző könyvekben minden további nélkül lehetnek azonos nevű szereplőink.

Hasonló módon szeretnénk biztosítani a könyvek isbn számainak egyediségét. Ehhez egy kicsit átalakítjuk a korábbi példánkat, hogy több könyvet is tudjon tárolni, és megadjuk, hogy az isbn attribútum egyedi legyen:

```
<xsd:element name="konyvek">
  <xsd:complexType>
    <xsd:sequence max0ccurs="unbounded">
      <xsd:element name="konyv" type="konyvTipus">
        <!-- Az előző szereplőnév megszorítás -->
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="EgyediISBN">
    <xsd:selector xpath="konyv" />
    <xsd:field xpath="@isbn" />
  </xsd:unique>
</xsd:element>
```

Elég valószínű, hogy az isbn attribútumot a könyvek egyedi azonosítására használjuk, erre a célra azonban nem a unique elem az igazi, mert az megengedi azt is, hogy az egyedi érték *(field)* ne szerepeljen a céldokumentumban. Adatbázis-hasonlattal élve: megengedi a null *(xml-ben nil)* értéket is.

Valódi elsődleges kulcs jellegű adatok megkötésére inkább használjuk a key sémaelemet. Ez egy olyan unique megkötés, ami nem engedi meg a null értékeket. A használata teljesen azonos a unique-éval.

Egymástól függő adatok esetén hasznos lehet idegen kulcsok definiálása. Erre a keyref sémaelem használható, mellyel hivatkozást (referenciát) hozhatunk létre egy key-jel vagy unique-kal azonosított kulcsra. Például kössük meg, hogy egy szereplő barátja csakis a könyvben definiált szereplők közül kerülhet ki.

```
<!-- konyvsema11.xsd részlet -->
<xsd:keyref refer="EgyediSzereploNev"
 name="FKSzereplő">
  <xsd:selector xpath="szereplo" />
  <xsd:field xpath="baratja" />
</xsd:keyref>
```

Példánkban a keyrefet is a konyv elem belsejébe kell helyezni, azonban legtöbbször a hierarchia különböző pontjai között szoktunk hivatkozásokat definiálni.

Láthatjuk, hogy ezeknek a funkcióknak semmi közük a korábbi részekben tárgyalt struktúradefiniáló elemekhez. Azok olyan típusok leírására valók, amelyeket sokszor programozott típusok *(class-ok)* és xml dokumentumok közötti átjárásra használunk. Nem nehéz kitalálni, hogy a fő motiválóerő a Webszolgáltatás technológia háttértámogatása volt. Ezzel szemben a megszorítások, kulcsok egyértelműen adatok értékeinek megregulázására alkalmasak, ez pedig adatok átvitele, üzleti adatok cseréje közben lehet hasznos. De a kettőt lehet ötvözni is, mert például egy .NET-es Webszolgáltatásban egy paraméterként átadott típusos DataSet táblái között lehetnek kapcsolatok, és ezt a DataSet key-keyref elemekkel modellezi le. A táblák adatai xml-ként mennek át, és az adatok épségét a kapcsolatok is elősegítik.

Zárszó

Szinte megismertük a séma összes elemét, így a következő záró részben áttekintjük a gyakorlati felhasználás részleteit COM és .NET világban is.

Soczó Zsolt
Zsolt.Soczo@netacademia.net

A cikkben szereplő URL-ek:
[1]: A cikkben szereplő példák
http://technet.netacademia.net/download/xml

ADO.NET
1+3 sávos adathíd aszfaltozási munkálata

A tech.net magazin februári és márciusi számában XMLgessünk címen Soczó Zsolt bemutatta az ADO.NET adatbáziskezelésének központi elemét, a DataSet objektumot. Egy memóriában tárolt adatbázis óriási kincs, de csak akkor, ha stabil technológiai kapoccsal összeköthető a meglévő adatbázisszerverekkel és a helyi gépeken tárolt XML állományokkal. Az alábbi cikk a két kapcsolódási felület közül az elsőt elemzi, és az SQLServer adatbázis és a DataSet objektum közötti híd „aszfaltozási munkálataiba" avatja be az Olvasót.

Mielőtt az aszfaltozógépeket ráengednénk a hídra, vizsgáljuk meg a híd vázszerkezetét és a két pillért. Az adathíd alapszerkezetét az SqlDataAdapter objektum alkotja. Az egyik pillér egy SQLServer adatbázis, a másik pedig egy DataSet objektumhoz tartozó DataTable objektum. A híd lehet egyirányú és egysávos, ha csak arra használjuk, hogy az SQLServer adatbázisból feltöltsük a DataTable objektumot. Az SqlDataAdapterrel kétirányú – és ezzel akár 1+3 sávos – híd is kialakítható, ha a DataTable-ben bekövetkezett változásokat vissza akarjuk vinni az eredeti SQLServer adatbázisba.

Adatok kiolvasása, azaz a híd első sávjának felépítése

A híd egyik irányának felépítéséhez az SqlDataAdapter objektum SelectCommand tulajdonságadatát kell feltölteni az adatok visszanyerését szolgáló SqlCommand objektummal.

```
Dim conSzamla As New SqlConnection(
    "user id=sa;password=haliho;" & _
      "database=Szamla;data source=Gep1")
Dim cmdPartner As New SqlCommand(
    "Select * From Partner Where PaTipus=1",
      conSzamla)
Dim daPartner As New SqlDataAdapter
Dim dsPartSzla As New DataSet
daPartner.SelectCommand = cmdPartner
```

Ezzel tehát a híd az SQLServer adatbázisból a DataSet irányába már járható, így átjuttathatók rajta az adatok a Fill() metódus meghívásával.

```
daPartner.Fill(dsPartSzla,"Szallito")
```

A fenti metódushívás a daPartner nevű SqlDataAdapter objektumnak a SelectCommand tulajdonságadatába betöltött SqlCommand parancsát fogja felhasználni a dsPartSzla objektum feltöltéséhez. A tényleges adatáttöltés elkezdése előtt az adott SqlCommand parancsban definiált adatbáziskapcsolatot *(conSzamla)* megnyitja a rendszer, majd az SQL parancs által visszaadott adatszerkezeti információk alapján felépít egy DataTable objektumot "Szallito" néven. A DataTable objektum neve *(TableName tulajdonságadat)* – ahogy a példában is látható – nem szükségképpen egyezik meg az adatbázisbeli táblanévvel. Az adatszerkezet kialakítása után az adatbázisból érkező adatsorokkal *(a PaTipus=1 feltételnek eleget tevő partnerek adataival, azaz mondjuk a szállítókkal)* feltölti a dsPartSzla-hoz tartozó „Szallito" nevű DataTable tartalmát. A „feltöltés" tulajdonképpen nem más, mint annyi DataRow objektum létrehozása és DataTable objektumhoz történő hozzákapcsolása, ahány eredménysort a fenti SQL parancs visszaadott. Az adatfeltöltést követően a rendszer lezárja a conSzamla objektummal azonosított adatbáziskapcsolatot. Az adatbázis és a DataSet közötti kapcsolat ily módon csak a Fill() metódus végrehajtásának idején él, előtte, és utána nem. *(Hát hiszen éppen ezért disconnected ez az egész...)*

Az SqlDataAdapter objektum Fill() metódusa a mintapéldában bemutatott, tipikusnak tekinthető beállítás mellett többféle egyéb paraméterezést is elfogad. Ha csak az első paramétert adtuk volna meg, akkor a DataTable objektum „Table" névvel születetett volna meg. Ha már létezik egy DataTable objektumunk, akkor azt is megadhattuk volna az első paraméterben a DataSet objektum helyett.

```
daPartner.Fill(dsPartSzla) '"Table" nevű DataTable
daPartner.Fill(dtSzallito) ' Ha már van ilyen dt
```

Fontos figyelembe vennünk, hogy egy SqlDataAdapter objektum a DataSetnek csak egyetlen DataTable objektumához kapcsolódik. Ha egy DataSet objektumnak „Szallito", „Szamla" és „Rendeles" adattáblákkal kell rendelkeznie, akkor ehhez három különálló SqlDataAdapter hidat kell megépítenünk.

A módosítások visszaírása, azaz a híd másik három sávjának elkészítése

Az adatbázistól függetlenné vált DataSet, DataTable és DataRow objektumokban többféle módosítás is lejátszható. Koncentráljunk most a tényleges adatokban, azaz a DataRow objektumokban végrehajtott változtatásokra!

A változtatások tipikusan „felvitel-módosítás-törlés" jellegűek. Ennek megfelelően új sorok *(új DataRow objektumok)* építhetők be a rendszerbe, meglévő sorok adott mezőit módosíthatjuk, illetve adatsorok törlését is kezdeményezhetjük. Az adatkarbantartás programozott módon is megvalósulhat, de a leggyakoribb helyzet az, amikor az adott DataTable közvetett módon *(pl. egy DataView objektumon keresztül)* valamilyen felhasználói interfész elemhez *(pl. DataGrid vagy DataList)* kötődik. Ilyenkor a fel-

használó által begépelt adatmódosítások közvetlenül átírják a DataTable-hez tartozó megfelelő DataRow objektum tartalmát.

A „közvetlen átírás" akár erős csúsztatásnak is tűnhet, ha mögénézünk a dolgoknak. A valóság ugyanis az, hogy ugyanabból a DataRow objektumból több verzió is létrejön a módosításkor. Az Original verzióban például ott csücsül az eredeti adattartalom, míg a Current verzió az aktuális tartalmat hordozza. A „közvetlen átírás" tehát valójában a Current verziót babrálja.

Akár programozott technikával, akár a felhasználó általi begépeléssel módosul egy adattábla tartalma, előbb vagy utóbb szükségessé válik a módosítások átvezetése a központi adatbázisban. Ehhez először is meg kell építeni a híd másik irányba vivő sávját. A bekövetkezett változások háromfélék lehetnek *(Insert, Update, Delete)*, ezért igazából nem is egysávos utat, hanem egy háromsávos sztrádapályát kell megalkotni. Mindhárom sávra természetesen csak akkor van szükség, ha új sor felvitelek, módosítások és sortörlések egyaránt előfordulhatnak az adattáblában. Magyarul, csak azokat a sávokat kell megépíteni, amelyeken számítani lehet adatforgalomra.

Az útsávok tényleges létrehozásához az SqlDataAdapter objektum InsertCommand, UpdateCommand és DeleteCommand tulajdonságadatait kell a megfelelő SqlCommand objektummal feltölteni.

Az útsávok tényleges létrehozásához az SqDataAdapter objektum InsertCommand, UpdateCommand és DeleteCommand tulajdonságait kell a megfelelő SqlCommand objektummal feltölteni

Az InsertCommand sáv működtetéséhez egy olyan SqlCommand objektumot kell készíteni, amelyik egy INSERT SQL utasítást vagy egy tárolt eljárást tartalmaz. A ténylegesen beillesztendő adatok SqlParameter objektumokként kapcsolódnak az SqlCommand objektumhoz.

Az UpdateCommand tulajdonságadatba olyan SqlCommand objektumot kell bevinni, amelyik egy UPDATE SQL utasítást vagy egy hasonló funkciójú tárolt eljárást tartalmaz. Az adatbázisban módosítandó mezők itt is SqlParameter objektumokként jelennek meg.

A DeleteCommand-beli SqlCommand objektum egy DELETE SQL utasítást vagy egy megfelelő tárolt eljárást tartalmazhat. A törlendő adatsor beazonosításához szükséges mező vagy mezők itt is SqlParameter objektumokként épülnek be a rendszerbe.

Ha a szükséges útsávokat kiépítettük, akkor az SqlDataAdapter objektum Update() metódusával elindítható a háromsávú adatforgalom. Az Update() metódus paraméterében kell megnevezni azon adatsorok körét, amelyekre vonatkozóan az aktualizálást el akarjuk végezni. Néhány példa arra, hogyan lehet megfogalmazni az aktualizálási feladatot:

```
daPartner.Update(dsPartSzla)
↳    ' A teljes DataSet-re vonatkozik
daPartner.Update(dtSzallito)
↳    ' Csak az adott DataTable soraira vonatkozik
daPartner.Update(dsPartSzla,"Szallito")
↳    ' Az előzővel megegyezik
daPartner.Update(drTomb())
↳    ' Az adott DataRow() tömb adatsoraira szól
```

Az Update() metódus a tényleges adatforgalom megindítása előtt megnyitja az InsertCommand, az UpdateCommand és a DeleteCommand által igényelt adatbáziskapcsolatot. Ugyanaz a kapcsolati objektum *(pl. conSzamla)* használható mindhárom parancshoz egyidejűleg. A rendszer ezután sorra veszi a paraméterben beazonosított adatsorokat *(DataRow objektumokat)*, és kihalássza közülük azokat, amelyeknél a RowState tulajdonságadat változásra utal. Azokat az adatsorokat, amelyeknél a RowState értéke DataRowState.Added, az InsertCommand sávra tereli, vagyis ezekre egyenként meghívja az ott megadott INSERT SQL utasítást vagy a megfelelő tárolt eljárást. Azok az adatsorok, amelyeknél DataRowState.Modified érték szerepel a RowState tulajdonságban, az UpdateCommand sávon indulva érik el a megfelelő UPDATE SQL utasítást, illetve az erre a célra megírt tárolt eljárást. A DataRowState.Deleted jelzéssel ellátott adatsorok pedig a DeleteCommand-hoz kapcsolt DELETE SQL parancsot hajtják végre. Az aktualizálási folyamat végén a rendszer bezárja a megnyitott adatbáziskapcsolato*(ka)*t.

Az alábbiakban nézzünk egy példát az UpdateCommand feltöltésére:

```
Dim conSzamla As New SqlConnection(
↳    "user id=sa;password=haliho;" & _
    "database=Szamla;data source=Gep1")
Dim cmdPartner As New SqlCommand(
↳    "Select * From Partner Where PaTipus=1", _
                                conSzamla)
Dim daPartner As New SqlDataAdapter
Dim dsPartSzla As New DataSet
Dim cmdModos As New SqlCommand(
↳    "UPDATE Partner SET " & _
    "PartAzon=@ID, PartNev=@Nev WHERE " & _
    "PartAzon=@EredID", conSzamla)
cmdModos.Parameters.Add(New SqlParameter(
↳    "@ID", SqlDbType.Int32, 4, _
    ParameterDirection.Input, False, 0, 0, _
    "PartAzon", DataRowVersion.Current, Nothing))
cmdModos.Parameters.Add(New SqlParameter(
↳    "@Nev", SqlDbType.Char, 30, _
    ParameterDirection.Input, False, 0, 0, _
   "PartNev", DataRowVersion.Current, Nothing))
cmdModos.Parameters.Add(New SqlParameter(
↳   "@EredID", SqlDbType.Int32, 4, _
   ParameterDirection.Input, False, 0, 0, _
   "PartAzon", DataRowVersion.Original, Nothing))
daPartner.SelectCommand = cmdPartner
daPartner.UpdateCommand = cmdModos
daPartner.Fill(dsPartSzla,"Szallito")
...' Itt történik meg a Szallito adattábla
↳ módosítása
daPartner.Update(dsPartSzla,"Szallito")
dsPartSzla.AcceptChanges()
```

A fenti kódrészletben a módosítások visszavezetésére létrehoztunk egy cmdModos nevű SqlCommand objektumot. Ez az objektum a conSzamla adatbáziskapcsolaton keresztül egy UPDATE utasítás végrehajtását célozza meg. Az UPDATE SQL utasítás három paramétert vár: a partner eredeti azonosítóját *(@EredID)*, a partner esetlegesen megváltoztatott azonosítóját *(@ID)* és a partner új nevét *(@Nev)*. *(A példa kedvéért most egy pillanatra vonatkoztassunk el attól, hogy mennyire nem egészséges dolog megengedni egy tábla egyedi azonosítójának megváltoztatását. A lényeg az, hogy akár egy ilyen módosítás is visszavezethető az eredeti adatbázisba.)* Három SqlParameter objektumot kell tehát létrehoznunk és hozzákapcsolnunk a Parameters.Add() metódus

segítségével a cmdModos objektumhoz. Az SqlParameter objektum konstruktormetódusai közül a mintapéldában látható tízparaméteres változat a legtipikusabb, ezért ezen keresztül érdemes megismerkedni az egyes tulajdonságadatok szerepével.

```
Dim Para1 As New SqlParameter(ParameterName, _
  SqlDbType, Size, Direction, IsNullable, _
  Precision, Scale, SourceColumn, SourceVersion, _
  Value)
```

Az alábbi táblázatból kiolvasható az egyes tulajdonságadatok tartalma:

Tulajdonság	Leírás
ParameterName	A paraméter neve.
SqlDbType	A paraméter adattípusa.
Size	A paraméter mérete byte-ban.
Direction	A paraméter iránya tárolt eljárásoknál: Input, Output, InputOutput, ReturnValue.
IsNullable	A paraméter fogadhat-e NULL értéket. Alapértelmezésben false az értéke, azaz nem fogadhat NULL-t.
Precision	A Value adatban szereplő számjegyek maximális száma.
Scale	A Value adatban szereplő szám tizedes jegyeinek száma.
SourceColumn	Annak az oszlopnak a neve a DataTable objektumon belül, ahonnan az adatot venni kell az Update() metódus végrehajtása során.
SourceVersion	A DataRow objektum verziója, ahonnan az adatot venni kell.
Value	A paraméter értéke. Inputparamétereknél a parancs végrehajtása előtt meg kell adni. Outputparamétereknél és a visszatérési értéknél ebben kapjuk vissza a tárolt eljárás vagy a függvény által juttatott adatot.

Mindezen információk birtokában kanyarodjunk vissza az eredeti UpdateCommand feladathoz, és nézzük meg, hogyan kell beállítani az egyes paramétereket a feladat korrekt elvégzéséhez!
Az első paraméterre az UPDATE utasításban a „@ID" néven hivatkozunk, és a konkrét paraméterértéket *(a Value tulajdonságadatot)* a „Szallito" nevű DataTable objektum „PartAzon" nevű DataColumn oszlopából kérjük kiemelni annál a sornál, amelynél valamilyen módosítást észlelt az SqlDataAdapter objektum Update() metódusa. Mivel az UPDATE utasítás SET opciójában a módosított értéket várjuk, a DataRowVersion.Current által azonosított adatot kérjük.
A második paraméterre a „@Nev" néven hivatkozunk az UPDATE utasításban. A konkrét adatot a „PartNev" nevű oszlopból kérjük kiemelni. Itt is az esetlegesen módosított értékre van szükségünk, ezért DataRowVersion.Current értéket állítunk be az SQLParameter konstruktorának kilencedik paraméterében.
A harmadik paramétert az UPDATE utasítás WHERE feltételében használjuk „@EredID" néven, és ezzel kell rátalálnunk az eredeti partnerrekordra – még akkor is, ha a partnerazonosítót gonosz módon megváltoztatták. Itt tehát a „PartAzon" oszlopnak a DataRowVersion.Original verzióját kell kiemelnie a rendszernek, és beépítenie az UPDATE utasításba.
Ha az így összeállított SqlCommand objektumot hozzákapcsoljuk az UpdateCommand tulajdonságadathoz, akkor a daPartner.Update() metódus minden egyes módosított adatsornál megfelelően kitölti a paraméterek Value adatát, majd meghívja az UPDATE parancsot. Ha új sorokat is felvittünk, és nincs megadva a megfelelő InsertCommand, akkor futás közbeni hibát kapunk.
Az Update() tényleges végrehajtását követően a DataTable-hez tartozó adatsoroknál is célszerű lehet bejelölni az aktualizálás tényét, hogy a következő Update() már ne vigye át ismételten az eredeti módosításokat. Ehhez az adott DataTable objektum AcceptChanges() metódusát kell meghívnunk. Az AcceptChanges() metódus az összes Added és Modified státuszú adatsort Unchanged státuszúra alakítja át, a Deleted jelzésű sorokat pedig kitörli. Az AcceptChanges() kiadható DataSet és DataRow szinten is. Az előbbi esetben a DataSethez tartozó összes adattáblára végrehajtja az adott műveletet, az utóbbi esetben csak egyetlen adatsorra történik meg a módosítások véglegesítése.

Nem kell persze mindenáron összekoszolni a kezünket a hídsávok fáradságos kézi megépítésével, hiszen a Data Adapter Configuration Wizard elvégezheti helyettünk a piszkos munkát. Előállítja a szükséges programkódot, sőt még tárolt eljárást is ír helyettünk, ha akarjuk. Nem árt azért tudni azt, hogy mennyi részlet pontos összeillesztése kell egy ilyen adathíd felállításához.

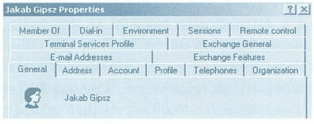
Endrődi Tamás
endrodi@okk.szamalk.hu
MCP

Exchange 2000
Felhasználók, csoportok, címlisták

Az előző szám végén ott tartottunk, hogy a felhasználók beállításai jönnek. Jöjjenek, és nemcsak a felhasználók, hanem az e-mail címmel rendelkező egyéb objektumok – tehát a kapcsolatok *(Contacts)* és az e-mail címmel rendelkező csoportok - tulajdonságait is megnézegetjük ebben a részben.

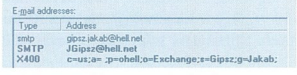
Felhasználók tulajdonságlapjai

Arról már az előző számban is szó volt, hogy attól függően, milyen nézetet választunk az Active Directory Users and Computers konzolon, mindig más tulajdonságok láthatók. Ha nincs bekapcsolva a „View Advanced Features", Gipsz Jakab tulajdonságlapja így jelenik meg:



Gipsz Jakab egyszerűbb nézetben

Mivel az Active Directory az Exchange címtára, a felhasználók beállításai közt sok olyan van, amelyeket ha kitöltünk, a levelezőprogramból nézve megjelennek a címlistában. Ilyenek az Organization, az Address vagy a Telephones tulajdonságlapon beállítható adatok. Izgalmasabb az E-mail Addresses tulajdonságlapja, ahol a felhasználóhoz rendelt e-mail címek találhatók. Házirenddel állítható, hogy milyen típusú címek jöjjenek létre a postaládával egy időben. Alapértelmezésben minden felhasználó számára egy SMTP és egy X400 cím jön létre. Egy típusból *(maradjunk az SMTP-nél)* több változatot is meg lehet adni, ilyenkor egyet mindig ki kell nevezni elsődlegesnek *(ez látszik az elküldött levelek fejlécében)*, a többi címre a felhasználó csak fogadni tud levelet.



Gipsz Jakabnak is két SMTP típusú e-mail címe van

A vastagon szedett az elsődleges cím, a gipsz.jakab kezdetű pedig alárendelt. *(A JGipsz volt az Alias, amit a postaláda létrehozásakor megadtam.)* Minden esetben az Aliast fogja a varázsló a létrehozott címben a @ elé tenni. A postaláda létrehozása után rögtön nem látszanak az e-mail címek, várni kell, míg a Recipient Update Service frissíti a címeket, de lehet őt sürgetni is.

Az SMTP és az X.400-as cím mellett lehetőség van Lotus, cc:Mail, MS Mail, Groupwise címek létrehozására is, amiket akkor használunk, ha az Exchange-et más levelező rendszerekkel, speciális csatolók segítségével kapcsoljuk össze. Az SMTP az Interneten keresztüli levelezés „de facto" szabványa, SMTP címzéssel utaznak a levelek az Interneten.

Az X400 címzést más X400 rendszerekkel való közvetlen kapcsolattartásra használjuk.

A felhasználók General lapján is van egy E-mail tulajdonság, amely az E-mail Addresses fülön található elsődleges címre mutat. Ha az egyiket átírjuk, megváltozik a másik is.

Az Exchange General tulajdonságlapon megnézhetjük, hogy az adott felhasználó postaládája melyik szerveren található. Mozgatni a Move Mailbox varázslóval lehet. A postaláda létrehozásakor megadott Aliast is meg tudjuk itt változtatni. Ennek nem kell azonosnak lennie más nevekkel, általában az e-mail cím @ előtti részével egyezik meg. Már volt arról szó, hogy a postaláda létrehozáskor megadott Alias lesz az elsődleges e-mail címben is. Ha utólag megváltoztatjuk az Aliast, az e-mail cím már nem változik vele.

Minden felhasználóra külön-külön be lehet állítani az általa elküldhető és fogadható levelek maximális méretét.

Ha POP3 segítségével húzogatjuk le a leveleket egy Exchange kiszolgálóról, előfordulhat, hogy egy felhasználó nem tud bejelentkezni. Ez abból adódhat, hogy az Alias és a felhasználó login neve - pontosan a User Logon Name *(Pre-Windows 2000)* – nem egyezik meg. Ilyenkor ezeket vagy szinkronba kell hozni *(kézzel)*, vagy a POP3-as csatlakozáskor <tartomány>\<login név>\<Alias> hármast kell megadni bejelentkezési névként.

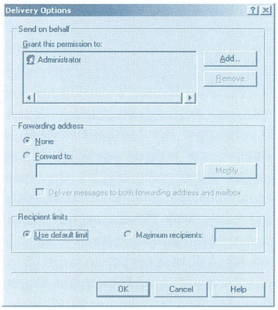
Az Exchange General **Delivery Restrictions** lapján állíthatók a levélküldéssel és fogadással kapcsolatos beállítások.

Minden felhasználóra külön-külön be lehet állítani az általa elküldhető és fogadható levelek maximális méretét. Természetesen nemcsak egyesével lehet beállítani ezt, hiszen az Exchange globális beállításai közt is megtaláljuk a levélméret-korlátozást. Itt csupán ahhoz képest tehetünk kivételeket. Szintén itt határozhatjuk meg, hogy egy felhasználó kitől kaphat levelet. Akkor tudunk megadni idegen címeket, ha azokat előzőleg Kapcsolatként *(Contact)* felvesszük az AD-ba.

A **Delivery Options** ablakban lehet beállítani, hogy az adott felhasználó nevében egy másik tudjon levelet küldeni: ez a „Send on behalf". Ez nem jelenti azt, hogy a postaládájához hozzáfér, csupán azt, hogy a felhatalmazó személy nevében írhat levelet *(a From mezőben megadhatja annak nevét)*. Ilyenkor a megkapott levél From mezőjében pédául ezt olvassuk: „Gipsz Jakabné; on behalf of; Gipsz Jakab". Ez azt jelenti, hogy Gipszné Gipsz Jakab nevében küldte az üzenetet.

Ugyanitt lehet beállítani a felhasználó leveleinek más címre továbbítását. Ha külső címre szeretnénk továbbítani, akkor előző-

leg fel kell venni egy Kapcsolatot *(Contact)* a külső e-mail címmel, majd meg kell adni továbbítási célként. Így az összes levél arra a címre utazik tovább. *(Lehetőség van arra, hogy a postaládában is megőrizzük a továbbított leveleket, ahhoz a „Deliver messages to both..." checkboxot be kell jelölni.)*



Exchange General – Delivery Options

Végül még egy lehetőség: szabályozhatjuk, hogy egy felhasználó egyszerre hány címre küldhet levelet. Alapértelmezésben a „Maximum recipients" száma 5000 – ez a Global Settings –> Message Delivery lapon található a System Managerben. Ha ez bizonyos helyzetekben nem megfelelő, akkor egyes felhasználóknak többet vagy kevesebbet állíthatunk be.

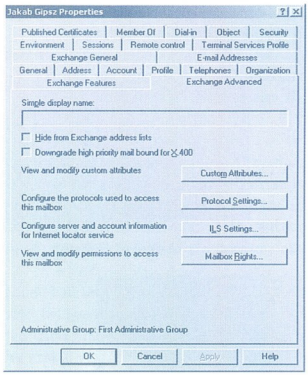
Az Exchange General – Storage Limits lapon a postaládák méretét illetően kivételeket lehet tenni az Exchange házirendhez képest. Ugyanúgy mint a házirendben, itt is meg lehet adni egyrészt egy küszöbméretet, aminek átlépése után figyelmeztető üzenetet kap a felhasználó, másrészt egy olyan határt, ami fölött nem tud levelet küldeni, harmadrészt a végső határt, aminek elérésekor mindaddig nem tud levelet küldeni és fogadni, amíg nem csökkenti a postaláda méretét a megengedett határ alá.

Ha egy olyan felhasználónak próbálunk levelet küldeni, akinek már betelt a postaládája, rögvest egy üzenetet kapunk vissza, amiben az áll, hogy vagy megtelt a címzett postaládája, vagy túl nagy levelet próbálunk küldeni, amit a beállítások nem engednek meg.

Központilag is, és felhasználónként is lehet állítani azt, hogy a postaládából letörölt leveleket – pontosabban a Deleted Items mappából törölt dolgokat – meddig tartsa meg visszaállítható formában az Exchange. Ez a „Deleted Item Retention" értéke, ami szintén a Storage Limits lapon található.

Az Active Directory Users and Computers „Advanced Features" módjában található az Exchange Advanced lap. Itt adhatunk meg egyszerű, ékezetes vagy egyéb extra karaktereket nem tartalmazó nevet a felhasználóhoz, ami akkor érdekes, ha valamilyen levelezőprogram nem tudja az extra karaktereket megjeleníteni.

Központilag is, és felhasználóként is lehet állítani azt, hogy a postaládából letörölt leveleket meddig tartsa meg visszaállítható formában az Exchange.



Gipsz Jakab „Exchange Advanced" lapja

Ha el akarjuk rejteni a felhasználót a címlistákból, akkor kell a „Hide from Exchange address lists" elé pipát tenni. A Custom Attributes gomb mögötti lapon egyéb tulajdonságokat lehet megadni, ha a meglevő attribútum-készlet esetleg nem lenne elég. Sajnos az itt megjelenő tulajdonságok nevét nem lehet megváltoztatni, csupán értéket adhatunk nekik. Ezek a tulajdonságok tulajdonképpen az Exchange 5.5 rendszerekkel való kompatibilitás fenntartása miatt léteznek. Előfordulhat hogy az 5.5-ös és az Exchange 2000 közötti replikáció megváltoztathatja a neveket. Ha azt szeretnénk, hogy a felhasználó adatai közt megjelenjenek más tulajdonságok is, az AD sémát kell bővíteni.

A **Protocol Settings** lapon felhasználónként lehet szabályozni a HTTP, IMAP és POP3-as elérést. Ezek a beállítások a System Managerben központilag állíthatók, itt kivételeket tehetünk a szerver beállításaihoz képest. Alapállapotban mindenkinek engedélyezve van mind a három protokollon keresztüli csatlakozás, amit felhasználónként letilthatunk.

A HTTP protokoll nem sok beállítással kényeztet minket: tiltani vagy engedélyezni lehet az Outlook Web Access-elérést. Az IMAP4 és POP3 beállításai ennél többet engednek. Ha szükséges, felülbírálhatjuk a szerverről örökölt beállításokat, egyénileg szabályozhatjuk a protokollhoz kapcsolódó levélformátumokat, karakter-beállításokat.

Az ILS Settings lapon beállítható Internet Locator Service kiszolgáló és név akkor használatos, amikor a felhasználó az Outlookból próbál online találkozót összehozni.

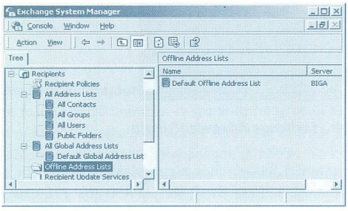
Az Exchange Advanced lapon az utolsó lehetőség a postaládához tartozó jogosultság szabályozása, a Mailbox Rights. Egy frissen készített felhasználó esetén nem látszik más, csupán a SELF „Full Mailbox Access". Amint ez a felhasználó belép, vagy levelet küldünk neki, megjelennek a valódi jogok, amelyek csaknem teljes egészében öröklődéssel kerülnek ide: a pipák szürkén látszanak. Ha egy felhasználónak „Full Mailbox Access" jogot adunk, be tud lépni a másik felhasználó postaládájába. Csupán az Outlook profilt kell beállítani, és már működik is.

Eddig már kétszer volt arról szó, hogyan tud egy felhasználó más nevében levelet küldeni. Az egyik a „Send on Behalf" a másik „Full Mailbox" jog. Van egy harmadik módja is, méghozzá a „Send As" jog. Ezt a beállítást a felhasználó objektumának Security lapján találjuk. Ha mondjuk Gipsznének „Send As" joga van Gipsz Jakab postaládáján, levélküldéskor a From mezőbe Gipszné a sajátja helyett Gipsz Jakab nevét tudja írni, és a levélből nem fog kiderülni, hogy azt valójában Gipszné küldte. Tessék óvatosan bánni ezzel a joggal!

A csoportok és kapcsolatobjektumok tulajdonságlapjai kevés beállítható pontot tartalmaznak. Ugyanúgy van Exchange General és Exchange Advanced valamint E-mail Addresses lap, rajtuk kevesebb illetve kicsit más beállítási lehetőséggel.

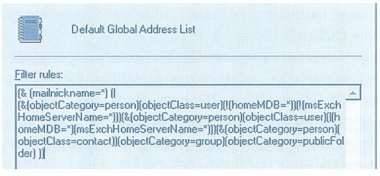
A címlisták

Amikor levelet írunk, címlistából választhatjuk ki a címzett nevét. A címlisták tartalmazzák az összes e-mail címmel rendelkező objektumot, kivéve amelyeken szándékosan megtiltjuk, hogy látszódjanak a címlistákban. Alapértelmezésben is többféle címlista található az Exchange-ben.



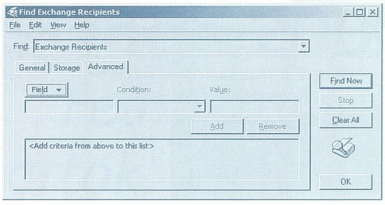
Címlisták az Exchange-ben

A címlistákat a System Managerben a Recipients alatt találhatjuk, van belőlük jó pár: úgy látszanak, mintha konténerobjektumok lennének. Ha megnézzük egy ilyen objektum tulajdonságait, *(jobb klikk – properties)* látható, hogy valójában csak egyszerű objektumok. A címlisták LDAP lekérdezéseket tartalmaznak. Az Active Directoryból szűrik ki a megfelelő objektumokat. A legismertebb talán a Default Global Address List, vagy röviden GAL, amely egymaga tartalmazza az összes e-mail címmel rendelkező objektumot. Leggyakrabban ezt használjuk. Alapértelmezésben külön címlista van a felhasználóknak, a csoportoknak, a kapcsolatoknak, illetve a címmel rendelkező nyilvános mappáknak is. Ezeken kívül van még egy, az „Offline Address Book", amely letölthető az ügyfélgépekre, így akkor is tudunk leveleket írni, ha nem vagyunk hálózatközelben. Az így írt leveleket az Outboxból a következő csatlakozáskor az Outlook automatikusan elküldi. Az alapértelmezett offline címlistának a GAL az alapja, hisz ez a legteljesebb lista. Természetesen ezt a típusú címlistát is alakíthatjuk.



A GAL LDAP lekérdezése

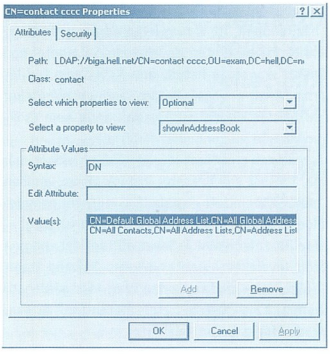
Mindezeken kívül rendkívül rugalmasan, az objektumok típusára, tulajdonságaira szűrve mi magunk is készíthetünk egyedi címlistákat, mintha csak az AD-ban keresnénk egy objektumot.



LDAP szűrés a címlistához

A General lapon ki tudjuk választani az objektumtípusokat, a Storage lapon azt, hogy melyik szerverről és adatbázisról van szó. A képen látható Advanced lapon pedig minden egyes objektumhoz tartozó összes tulajdonság alapján állíthatunk feltételeket. Ha több feltételt szabunk, azok ÉS kapcsolóval lesznek összekötve. Ha VAGY kapcsolót szeretnénk a feltételek közé, sajnos *(?)* egyedi LDAP lekérdezést kell készítenünk. A legfelső legördülő listából az Exchange Recipients helyett a Custom Search lehetőséget kell kiválasztani, és már írhatjuk is az LDAP lekérdezést, melynek RFC 2254 kompatibilisnek kell lennie, egyébként nem fog működni.

Mivel az Exchange címtára az Active Directory, a címlisták elemei is onnan kerülnek ki. A System Attendant szolgáltatás része a Recipient Update Service, röviden RUS, amely felelős a címlisták tartalmáért. Minden objektumnak – felhasználónak, csoportnak, nyilvános mappának, kapcsolatnak – van egy úgynevezett showInAddressBook tulajdonsága, amelynek értéke megadja, hogy az adott objektum mely címlistákban fog látszani.



Egy kapcsolat ShowInAddressBook tulajdonsága

Amikor a rendszergazda létrehoz egy címlistát, akkor nem tesz mást, mint szűrőket definiál a System Managerben. Ezután jön a RUS, ami a szűrő alapján minden egyes érintett objektumon módosítja a showInAddressBook tulajdonságot. Ez frissül a Global Catalog szervereken, és amikor a felhasználó az Outlookból a címlistában keres valakit, akkor tulajdonképp onnan, a Global Catalog szervertől kapja az információt.

A RUS-nak vannak egyéb feladatai is. Például a házirend alapján frissíti az e-mail címek beállításait stb. Kétféle RUS objektum van a System

Managerben. Az Enterprise RUS az AD konfigurációs konténerében levő objektumok e-mail címeit frissíti. A másik a tartományhoz kapcsolódik, az ott levő e-mailes objektumok címeinek frissítését végzi, valamint a címlistákat kezeli. Minden egyes Exchnage 2000-et tartalmazó Windows 2000 tartományhoz tartozik egy ilyen RUS.

Alapértelemzésként a RUS percenként fordul az Active Directoryhoz változásokat szimatolva, de *(puszta kézzel)* mi is frissíthetjük vagy újraépíttethetjük a címlistát.

Újraépítés esetén a teljes címlistatagság felülbírálásra kerül, míg frissítéskor a csupán a változásokkal frissül a lista.

Annyi címlistát csinálhatunk, amennyit csak szeretnénk. Létrehozhatunk több GAL-t is, egy felhasználónak azonban egy időben csak egy fog látszani. Hogy melyik, az attól függ, hogy melyikhez van jogunk, melyiknek tagja a felhasználó - és függ a címlista méretétől is. Ebben a sorrendben. Vagyis ha több olyan is van, amelyikhez egy felhasználónak joga van, az fog látszani közülük, amelyiknek a felhasználó maga is tagja. Ha többnek is tagja, a méret a döntő, a nagyobb fog győzni.

Minden címlistához meg lehet adni, hogy ki láthatja a tartalmát. Ezt a lista Security lapján tehetjük meg. Az Open Address List jog kulcsfontosságú. Az láthatja a címlista tartalmát, akinek ez a joga megvan. Ha nem akarjuk minden egyes címlistán egyedileg állítani a megfelelő jogokat, akkor a felette levő konténeren, például az All Address List Security lapján egyszerre állíthatjuk. A jogok öröklődni fognak az alatta lévőkre.

Ha az egész címlistát el szeretnénk tüntetni a kíváncsiskodók elől, nem elég megvonni az Open Adderss List jogot a listán, ugyanis ezzel csak a címlista tartalmát tüntetjük el, de magát a címlistát nem. Ehhez egy szinttel feljebb kell lépni, és ott kell rejtegetni. Létrehozunk egy üres címlistát, amit a valódi címlisták tárolására használunk. Az alábbi ábrán ez Bizalmas néven látható.

A „Külsősök" címlista elrejtése

Ebben kell létrehozni a rejtett címlistát – ez az ábrán a „Külsősök" listája. A jogosultságok öröklődése miatt az „Authenticated Users" csoportnak kezdetben joga van minden címlistához. Ha csak a vezetés számára szeretnénk láthatóvá tenni a Külsősök címlistáját, le kell vennünk az „Authenticated Users" csoportot a jogosultsággal rendelkezők közül, és helyette a vezetőknek létrehozott csoportnak kell megadni az Open Address List jogot. Mégegyszer: mindezt nem a Külsősök címlistán kell tenni, hanem egy szinttel felette, a „Bizalmas" lista Security lapján. Persze a „Bizalmast" ettől még látják mások is. Ha ezt is tiltani szeretnénk, akkor még feljebb kell menni, és az All Address Lists szintjén kell a jogokkal babrálni.

A házirendekkel folytatjuk...

Dorner Csilla
MCSE

Exchange 2000

Server és web storage system

Korábban bemutattuk, hogyan lehet az Exchange2000 Server Web Storage System-ét *(WSS)* elérni ADO-n *(ActiveX Database Objects)* keresztül. Most a .NET keretrendszer segítségével állunk neki a feladatnak. Egyelőre azonban – sajnálatos módon előre láthatatlan ideig – ADO.NET-ben nincs támogatás az Exchange2000 Server és a WSS elérésére. Ezért más, .NET alatt is használható megoldások után kell néznünk.

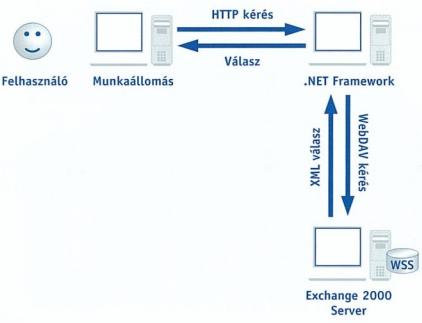
Az utóbbi időben nagy változások történtek: sokak szerint a .NET világa olyan a natív kódokhoz és a korábbi programozási módszerekhez képest, mint a Windows megjelenése a DOS-hoz képest. Ezzel a „forradalommal" együtt természetesen az ADO is megváltozott, s mint minden ilyen esetben, némi inkompatibilitás is felütötte a fejét. Ez abból adódik, hogy az ExOLEDB providert nem tudjuk felhasználni az ADO.NET menedzselt osztályaiban *(pl. Connection)*. Lássuk a többi lehetőséget!

A WebDAV *(Web-based Distributed Authoring and Versioning)*

A WebDAV a HTTP protokoll egyik kiterjesztése, amely lehetővé teszi, hogy távoli kiszolgálókon található állományokat érjünk el, szerkesszünk, olvassunk, módosítsunk stb. Fő célja - a felhasználók igényeinek megfelelően - az együttműködés elősegítése a különböző webes alkalmazások között. Egyik legfontosabb tulajdonsága, hogy a HTTP protokollt használja, így mindenhonnan könnyedén elérhető, és egyszerűen használható. Ráadásul a kommunikáció XML formátumban folyik, ami szintén ismerős és kedvelt mostanában.

Az ADO helyét kiváltó megoldások közül kettőt fogunk megvizsgálni, közös tulajdonságuk, hogy WebDAV-ra épülnek...

Az ADO helyét kiváltó megoldások közül kettőt fogunk megvizsgálni, közös tulajdonságuk, hogy WebDAV-ra épülnek, működésük hasonló elvek alapul:



A WSS adatainak elérése WebDAV-on és .NET FrameWork-ön keresztül

A .NET FrameWork-höz elérkezik a kérés *(pl. egy ASP.NET lap)*. A CLR *(Common Language Runtime)* feldolgozza azt, és egy WebDAV kérést továbbít az Exchange2000 Server felé *(amely fizikailag akár másik gépen is lehet)*. Az Exchange válasza XML formátumú, mint ahogy azt el is várjuk. Ezt a .NET FrameWork feldolgozza *(pl. egy DataSet-be tölti, táblázatba vagy file-ba írja)*, majd visszaküldi a kész adathalmazt a kliensoldali felhasználónak *(pl. egy weboldal formájában)*.

Az XMLHTTP

Az XMLHTTP első ránézésre jó megoldásnak tűnhet a WSS *(Web Storage System)* elérésére. Működik. Hátránya, hogy COM objektum. Ez azt jelenti, hogy használhatjuk ugyan .NET-es kódokból, viszont nem tartozik a FrameWork hatáskörébe. Ez azért lényeges, mert az adattípusok, metódusok, hibakezelés stb. nagyon eltérő a két világban: a natív és a menedzselt kódok mintha nem is ugyanazon a bolygón lennének. Az átjárás biztosítása a COM interop feladata, erről bővebben az MSDN-ben olvashatunk **[1]**, **[2]**.

Nyilván előnyei is vannak az XMLHTTP használatának, egyébként senki emberfiának *(-lányának)* nem születne az az elképesztő ötlete, hogy kipróbálja, mit tud. Hogy ez mégis megtörtént, annak az az oka, hogy a COM-os objektumok a régi világból ismerősek, használatukat megszoktuk, talán néhányan még meg is szerettük *(?)*. Vágjunk tehát bele!

Először is azt kell tudni, hogy az XMLHTTP30 nevű interfész az MSXML2 névtérben lakik *(msxml.dll)*, ezt kell referenciaként beimportálni a programunkba, hogy az működőképes legyen.

A WebDAV nagyon sokféle adatelérést tesz lehetővé. Ez a sokoldalúság a különböző metódusokon keresztül valósul meg, néhány ezek közül:

Név	Rövid leírás
COPY	Másolatot készít a megnevezett erőforrásról a célhelyre.
DELETE	Törli a megnevezett erőforrást.
MOVE	Átmozgatja a megnevezett erőforrást a célhelyre.
MKCOL	Új collectiont hoz létre.
PROPFIND	Az erőforrás jellemzőivel *(properties)* tér vissza.
PROPPATCH	Az erőforrás jellemzőinek módosítása.
SEARCH	Keresés a WSS-ben.

Részletesebb leírás magazinunk 2001/IX. számában, a szabványok rovatban olvasható, a metódusok teljes listája az MSDN-ben megtalálható **[3]**.
Lássuk először a keresés megvalósítását, hiszen talán ez a legalapvetőbb elvárásunk. A következő kódrészlet ezt hivatott megoldani. Első lépésként meg kell adnunk azt a WSS mappát, ahol az elérendő adatok találhatók:

```
string sURL = "http://myserver/public_folder/";
A kérés felépítése a következőképpen néz ki:
StringBuilder sQuery = new StringBuilder();
sQuery.Append(@"<?xml version='1.0'?>");
sQuery.Append("<g:searchrequest xmlns:g='DAV:'>");
sQuery.Append("<g:sql>");
sQuery.Append(" SELECT \"DAV:displayname\" ");
sQuery.Append(" FROM \"" + sURL + "\" ");
sQuery.Append("</g:sql>");
sQuery.Append("</g:searchrequest>");
```

Természetesen vesszővel elválasztva több jellemzőt is megadhatunk a SELECT után *(akár SELECT *-ot is használhatunk)*, WHERE szűrőfeltételeket szúrhatunk be, sorrendezhetjük az eredményhalmazt *(ORDER BY)*, stb., az SQL-ből megszokott szintaxisnak megfelelően. De hogyan jut el a kérés a szerverhez, és hogyan kaphatunk választ onnan? Itt jön a képbe az XMLHTTP.

```
XMLHTTP30 davKeres = new XMLHTTP30();
davKeres.open("SEARCH", sURL, false,
↳ @"myserver\user", "password");
davKeres.setRequestHeader("Content-type:",
↳ "text/xml");
davKeres.send(sQuery);
Console.Write(davKeres.ResponseText);
```

Létrehozunk egy példányt davKeres néven, majd az open meghívásával beállítjuk a megfelelő metódust *(SEARCH)*, az URL-t, ahol a keresést végezni kell *(sURL)*, illetve megadni a hozzáférést kezdeményező felhasználó azonosítóját és jelszavát. Nagyon fontos a fejlécben megadnunk *(setRequestHeader)*, hogy XML típusú adatot küldünk *(text/xml)*.
A kérést csak ezek után *(a send meghívásával)* küldhetjük el, amelynek paraméterül átadjuk a lekérdezést tartalmazó stringet. A kiszolgáló válasza ezek után a ResponseStream jellemzőn keresztül érhető el. Ennek további felhasználásáról, alkalmazásáról a későbbiekben lesz szó.
Érdekességképpen megmutatom, hogyan lehet e-mailt generálni, és állományokat csatolni hozzá a WSS-ben. Mindezt két lépésben oldjuk meg:
Először a kereséshez hasonlóan egy kérést kell felépítenünk egy stringben, amit majd elküldhetünk a szervernek. Természetesen most nem lesz SELECT-ünk, hiszen nem keresni szeretnénk a létező dokumentumok között, valamint az XML kérés formátuma és tartalma is eltér az előzőektől:

```
StringBuilder sQuery = new StringBuilder();
sURL = http://myserver/public/probamail.eml;
sQuery.Append(@"<?xml version='1.0'?>");
sQuery.Append("<d:propertyupdate xmlns:d='DAV:'
↳ xmlns:c='urn:schemas:httpmail:'>");
sQuery.Append("<d:set>");
sQuery.Append("<d:prop>");
sQuery.Append("<c:displayname>attachmail.eml
↳ </c:displayname>");
sQuery.Append("<c:subject>Level csatolt
↳ allomannyal</c:subject>");
sQuery.Append("<c:htmldescription>Ez a
↳ levél targya</c:htmldescription>");
sQuery.Append("</d:prop>");
sQuery.Append("</d:set>");
sQuery.Append("</d:propertyupdate>");
```

A különbségek tehát a következők: az XML kérés gyökéreleme propertyupdate lett, majd a prop és set tagek belsejében megadhatjuk a létrehozandó levél tulajdonságait. Ebben a példában a displayname-en kívül a levél tárgyát, illetve a törzsét adtuk meg a htmldescription jellemzőben.
Ezzel az e-mail létrehozásához szükséges kérés megvan, küldjük el a szervernek:

```
XMLHTTP30 oXML = new XMLHTTP30();
oXML.open("PROPPATCH", sURL, false,
↳ @"myserver\user", "password");
oXML.setRequestHeader("Content-type:",
↳ "text/xml");
oXML.send(sQuery);
```

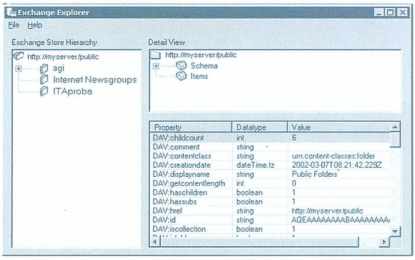
Ugye ismerős? Egyetlen eltérés van az előző küldéshez képest: a PROPPATCH metódust hívtuk meg. A levél létrejött, csatoljunk hát hozzá egy állományt!
Első lépésként olvassuk be a lokális fájlrendszerből:

```
FileStream fs = new FileStream(
↳"C:\\Agi\\valami.doc", FileMode.Open);
byte[] bytes = new Byte[fs.Length];
fs.Read(bytes, 0, (int) fs.Length);
```

Most jön az érdekesség: a WSS a levelekhez csatolt fájlokat streamként tárolja, ami azt jelenti, hogy következőképpen lehet rá hivatkozni:

```
String fURL = "http://myserver/public/
↳ probamail.eml/proba.doc";
```

Mintha a levelünk maga is egy collection lenne!
Vajon mi látszik mindebből a propertyk szintjén? Csupán annyi, hogy a urn:schemas:httpmail:hasattachment *(read-only)* property jelzi, van-e csatolt állomány a levelünkhöz.
Itt jegyzem meg, hogy az Exchange SDK-val együtt telepíthető a gépünkre egy Exchange Explorer nevű alkalmazás, amellyel jól nyomon követhetők a WSS-ben történő változások.



Az Exchange Explorer felhasználói felülete

1. Az ablak bal oldali része *(Exchange Store Hierarchy)* a WSS hierarchiánkat mutatja, abból a gyökérből kiindulva, amit az indításkor megadunk neki.
2. A jobb oldali ablakrész a részleteket szemlélteti. Felül a hierarchiában kijelölt mappához tartozó sémadefiníciókat, és a mappában található elemeket tekinthetjük meg.
3. Alul egy-egy elem kijelölésekor megjelennek annak propertyjei, névtereik szerint rendezve.

Az Exchange Explorer használata tehát nagyon egyszerű, logikus, struktúrája világos, áttekinthető.
Térjünk vissza azonban eredeti problémánkhoz, a csatolt fájl beszúrásához. Ott tartottunk, hogy megvan a beszúrandó állomány, megvan, hogy hová kell beilleszteni, jöjjön tehát, aminek jönnie kell:

```
XMLHTTP30 fXML = new XMLHTTP30();
fXML.open("PUT", fURL, false,
  @"myserver\user", "password");
fXML.setRequestHeader
  ("Content-type:","text/msword");
fXML.send(bytes);
fs.Close();
```

Két dolgon akadhat meg a szemünk: a PUT metódus jelöli, hogy állományt akarunk feltölteni; a senddel pedig nem kérést küldünk el, hanem a fájlunk tartalmát reprezentáló bytes tömböt. Máris láthatjuk például az Exchange Explorerben, vagy az Outlookban próbálkozásunk eredményét!

Ne felejtsük azonban: az XMLHTTP még COM-os objektum, előttünk áll még a feladat, hogy szebb, „modernebb" megoldás után nézzünk.

WebRequest és WebResponse

Ezennel elérkeztünk a .NET világába. A WebRequest és a WebResponse már .NET-es osztályok, a System.Net névtérben találhatók. Nem kell külső DLL-ekre, COM objektumokra hivatkozni, a Framework kielégíti minden igényünket. Lássuk, hogyan!
A kérés, ami a WSS-hez érkezik, hasonló lesz az előzőekhez, azonban a kiszolgálóhoz történő küldésének módja kissé más. Most a WebRequest példányunk jellemzőit állítjuk be a megfelelő értékekre:

```
WebRequest request = WebRequest.Create(sURL);
request.Method = "SEARCH";
request.Credentials = new NetworkCredential
  ("user", "password", "domain");
request.ContentType = "text/xml";
request.ContentLength = sQuery.Length;
```

Beállítjuk tehát a metódust, a felhasználót és a tartalom típusát. Végül meg kell adnunk *(nagyon fontos!)* a kérés hosszát is, mert e nélkül a kiszolgáló nem tudja értelmezni azt.
A kérés továbbításához létrehozunk egy StreamWritert *(System.IO névtér)*, amely a Request Streamünkbe ír, majd beleírjuk a lekérdezést, s lezárjuk:

```
StreamWriter writer = new
  StreamWriter(request.GetRequestStream());
writer.Write(sQuery);
writer.Close();
```

A választ egy WebResponse objektumban kapjuk vissza:

```
WebResponse response = request.GetResponse();
```

Ennek GetResponseStream() metódusával érhető el a választ reprezentáló Stream, amit aztán ismét igényeinknek megfelelően dolgozhatunk fel.

A válasz feldolgozása

Mindkét típusú keresésnél hivatkoztam arra, hogy válaszként streamet kapunk. Ennek feldolgozására mutatok most egy egyszerű példát. Az utóbbi, .NET-es megoldásra alapozva az alkalmazás elve az, hogy egy XmlReaderbe kiolvassuk a választ *(System.Xml névtér)*, majd ezen megyünk végig.

A WebRequest és a WebResponse már .NET-es osztályok, a System.NET névtérben megtalálhatóak.

```
WebResponse response = request.
  GetResponse();
Stream stream =
  response.GetResponseStream();
XmlTextReader xr = new
  XmlTextReader(stream);
xr.MoveToContent();
```

Az xr.Read() metódus a következő XML elemre ugrik a dokumentumban, és egy bool típusú értéket ad vissza: mindaddig igaz, amíg el nem éri az XML dokumentum végét. Azt, hogy az XML tag nyitó vagy záró éppen, az IsStartElement() metódus adja meg: értéke igaz nyitótagekre, hamis a tag záró párjára, illetve a tagek belsejében szereplő értékekre. Például:

```
<pelda>Ez a belseje</pelda>
```

esetén:

xr kurzor (xr.Read())	IsStartElement()
<pelda>	True
Ez a belseje	False
</pelda>	False

Lássuk tehát, hogyan írathatjuk ki a képernyőre a kiszolgáló XML válaszát:

```
while (xr.Read()) {
  stringBuilder s = new stringBuilder();
  if (xr.IsStartElement()) {
    s.Append(xr.Name);
    s.Append(" = ");
    xr.Read();
    if (!xr.IsStartElement()) {
      s.Append(xr.Value);
    } // if
    Console.WriteLine(s);
  } // if
} // while
```

Ha kényelmesebben, DOM-on keresztül akarjuk feldolgozni a kapott xml választ, akkor az XmlDocument osztályt inicializálhatjuk az XmlTextReader példányunkból.
A lehetőségek végeláthatatlanok: a képernyőre íratással nem merül ki a tudomány. Az adatokból készíthetünk DataSetet, építhetünk táblázatot, szűrhetjük azokat kliensoldalon *(például különböző nézetek létrehozására)*, stb.

Mire jó mindez?

Jogos a kérdés, hiszen manapság annyi alkalmazás van már a piacon, ami mind az Exchange Server elérését támogatja, hogy már csupán ezek nyomon követése is szép harci feladat: hogy ne menjünk messzire, ott az Outlook. Az Interneten is találhatunk jónéhányat, akár az MSDN-en is **[4]**.

A helyzet azonban az, hogy ezek az alkalmazások már készen kerülnek elénk, és vagy egyáltalán nem, vagy csak kis mértékben alakíthatók a mi igényeinkhez. *(Természetesen nem arra gondolok, hogy az Office Assistant kiskutya vagy mosolygó piros labda formájában jelenjen-e meg...)* Szükségünk lehet például arra, hogy saját WSS mappánkhoz olyan nézeteket generáljunk, amelyek a saját jellemzőinket is megmutatják, a saját igényeink szerint. Például ha az elmúlt évi nyaraláson készült képeinket a WSS-ben szeretnénk tárolni, akkor azokhoz nyugodtan hozzárendelhetünk egy sajatnevter:mivoltez jellemzőt, amelyben leírhatjuk, milyen eseményhez, a nyaralás melyik pillanatához kapcsolódik a kép. Vagy létrehozhatunk egy sajatnevter:kikvannakrajta jellemzőt. És még sorolhatnám. Ha saját alkalmazást írunk, mindez lehetővé válik, sőt, még a dokumentumok *(képek)* is úgy jelennek meg, ahogyan mi szeretnénk. Például a képek kis méretben, mellette a helyszín, az időpont, az esemény, stb. Ezek alapján aztán már szűrhetünk is, csoportosíthatunk, a lehetőségeknek csak a képzeletünk szab határt.

Jó móka, nem?

Molnár Ágnes
agnes.molnar@dataware.debis.hu

A cikkben szereplő URL-ek:

Common Language Runtime az MSDN-en:
[1] ms-help://MS.NETFrameworkSDK/cpguidenf/html/cpconcommonlanguageruntimeov
[2] http://msdn.microsoft.com/library/en-us/cpguide/html/cpconthecommonlanguageruntime.asp?frame=true

A WebDAV metódusok az MSDN-en:
[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wss/wss/_webdav_methods.asp

Exchange2000 Serverhez kapcsolódó alkalmazások:
[4]http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/msdn-files/027/001/833/msdncompositedoc.xml

[5] Exchange SDK Development Tools March 2002
http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/msdn-files/027/001/833/msdncompositedoc.xml

Dupla KV

Routolás RRAS nélkül

K: RRAS nélkül szeretnék routerként használni Windows 2000-et. A Windows NT 4.0 TCP/IP beállításai közt, az egyik ablakban volt egy olyan lehetőség, hogy „Enable IP Forwarding". Hova lett ez a Windows 2000-ben?

V: Sem a Windows 2000-ben, sem az XP-ben nincs felület a routing bekapcsolására. A registry módosításával viszont be lehet kapcsolni. Registry editorral a következő kulcs alá, REG_DWORD típussal kell felvenni az IPEnableRouter értéket.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\Tcpip\Parameters
IPEnableRouter=1
```

Ezzel az összes hálózati csatolóhoz bekapcsoljuk a routolást. Ha 0-ra állítjuk az értéket, vagy letöröljük a létrehozott értéket, megszüntethetjük a routolást.

(Forrás: MS Technet Q230082)

Terminálkiszolgáló leállítása

K: Ha a Windows 2000 terminál szervert szabályosan indítom újra, a felhasználók nem értesülnek róla, hogy alattuk a szerver újraindul, s egyszer csak eltűnik. Van arra egyszerű módszer, hogy a felhasználók automatikusan értesüljenek arról, hogy a szerver újraindul?

V: Ha a **tsshutdn** parancsot használjuk a terminál szerver leállításához, az elintézi a felhasználók értesítését is, sőt ki is lépteti a felhasználókat. Ez a parancs az operációs rendszer része. Vigyázat! Az egész gép újraindul, nem csak a Terminal Services! A parancs szintaktikája a következő:

```
F:\Documents and Settings\Administrator>tsshutdn /?
Shut down a server in a controlled manner.

TSSHUTDN [wait_time] [/SERVER:servername] [/REBOOT] [/POWERDOWN]
         [/DELAY:logoffdelay] [/V]

  wait_time             Seconds to wait after user notification before
                        terminating all user sessions (default is 60).
  /SERVER:servername    The server to shut down (default is current).
  /REBOOT               Reboot the server after user sessions are terminated.
  /POWERDOWN            The server will prepare for powering off.
  /DELAY:logoffdelay    Seconds to wait after logging off all connected
                        sessions (default is 30).
  /V                    Display information about actions being performed.
```

- wait_time: megadhatjuk, hogy a parancs kiadása után menynyi ideig várjon arra a leállítás, hogy kilépjenek a felhasználók. Alapértelmezésben ez 1 perc. Amikor letelt az idő, elkezdi kiléptetni a bennmaradt felhasználókat. Természetesen rögtön a parancs kiadása után a felhasználók értesülnek arról, hogy a szerver x idő múlva leáll.
- /server <server name> : ha távoli szervert szeretnénk leállítani, ezt a kapcsolót használjuk. Ha nem adunk meg távoli gépet, akkor értelemszerűen a helyi szerver lesz az áldozat.
- /reboot – szerver leállítás után rögtön újraindítás következik – ennek a kapcsolónak nem sok értelme, hisz az újraindítás az alapértelmezett művelet, mégha nem is használjuk ezt a kapcsolót.
- /powerdown – ezzel a szervert leállítás után kikapcsolhatjuk.
- /delay <time> - miután kiléptette az összes felhasználót, menynyi időt várjon a folyamat a leállítás előtt. Ez alapból fél perc.
- /V – bőbeszédűbb infót kapunk vissza arról, mi is történik.

Ha a parancsot egyszerűen, minden kapcsoló nélkül kiadjuk, akkor a következő történik: a folyamat a felhasználók értesítése után 1 perccel kilépteti a még bennlevőket, majd vár még egy fél percet, mielőtt újraindítja a gépet.

DFS replikáció finomhangolása

K: Hol tudom állítani, hogy a DFS replikáció mikor fusson le, illetve hogyan lehetne időzíteni?

V: A DFS a File Replication Service (FRS) segítségével replikál. Az FRS működését csak a registry módosításával lehet szabályozni. A registryben itt találhatók a DFS beállításai:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\
Services\NtFrs\Parameters
```

A beállításokról bővebben lehet olvasni a Microsoft Technet Q221111 KB cikkben.

(Forrás: Windows 2000 levelezési lista)

Exchange 2000 Mailbox jogok

K: Hogyan lehet megcsinálni, hogy a rendszergazda lássa az Exchange 2000 M: meghajtóján az összes felhasználó mappáját?

V: Legalább három úton lehet eljutni a megoldásig.

1. A legelegánsabb, ha a rendszergazda „Receive As" jogot ad magának az Exchange Organization szintjén. Ezt úgy tehetjük meg, hogy a következő registry túrással megjelenítjük a Exchange System Manager Security fülét:

```
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Exchange\
ExAdmin\
ShowSecurityPage=1
```

   Majd a System Managerből a legmagasabb szinten ellátjuk magunkat „Receive As" joggal.
   Ugyanezt registrymódosítás nélkül, ADSIEdittel is elvégezhetjük. A „Receive As" jogot a következő objektum tulajdonságainak Security fülén állíthatjuk be:

```
Configuration Conatiner
CN=Configuration
CN=Services
CN=Microsoft Exchange
CN=<Organization Name>
```

   Vigyázat!! Ha a felhasználó tagja a Domain Admins, vagy az Enterprise Admins csoportnak, ez a jog gyárilag „deny", azaz tiltott. Ilyenkor elegendő a tiltás megszüntetése, és máris az összes levelesládába belelátunk. Ezért két évig terjedő szigorított fegyház jár *(lásd 18. oldal)*!
2. Nem elegáns, de működő módszer, ha betesszük magunkat az Exchange Domain Servers csoportba. Kicsit veszélyes is, ezért öt év jár!

3. A harmadik módszer, hogy a levelesládák szintjén játszunk a „Receive As" joggal, illetve „Full Mailbox" jogot adunk magunknak. Felhasználói szinten a „Receive As" jogot az Active Directory Users and Computersben, a felhasználó tulajdonságainak Security lapján találjuk. A „Full Mailbox" jogot pedig az Exchange Advanced lapon, a Mailbox Rights alatt. Ez csak szabálysértés, és pénzbüntetést von maga után.

(Forrás: Exchange 2000 levelezési lista)

Group Policy megnyitása egy adott DC-ről

K: *A Domain Administrators csoport tagjaként a Group Policyt egy olyan tartományvezérlőn próbálom szerkeszteni, amelyikre hisecdc template lett rá húzva. Nem ez a tartományvezérlő a PDC emulátor. A következő hibaüzenetet kapom a házirend megnyitásakor:*

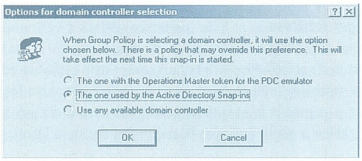
„Failed to open the Group Policy Object. You may not have the approriate rights. The specified domain either does not existor could not be contacted."

Az igaz, hogy a PDC emulátor épp nem megy. Hogyan tudnám mégis megnyitni a GPO-t?

V: Itt a „hiba" oka! **„...the specified domain either does not exist or could not be contacted..."**

A GPO Editor alapértelmezésben először a PDC Emulátorhoz kapcsolódik. Ha ez nem fut, akkor feldobja a „Select another DC" ablakot. Tegyél így, select another!

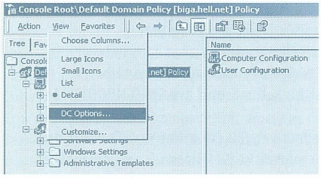
A GPO Editornak meg is lehet mondani, hogy hova csatlakozzon legközelebb. Három lehetőség van:



DC Options

1. A PDC emulátorhoz csatlakozik – ez az alapbeállítás.
2. Ahhoz a tartományvezérlőhöz kapcsolódik, amelyikhez az Active Directory Users and Computers MMC snap-in is.
3. Bármelyik elérhető tartományvezérlőhöz csatlakozik.

Mindezt úgy lehet átállítani, hogy megnyitunk egy Policyt, majd a Policy objektumon állva a View menüben megkeressük a DC Options menüpontot.



DC választás a GPO Editorban

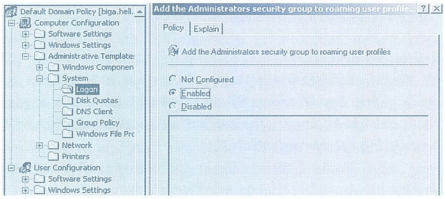
(Forrás: Security levelezési lista)

A Windows 2000 roaming profile könyvtárak jogai

K: *Windows 2000 környezetben ha egy új felhasználónak Roamain Profilet állítok be, később még Admin jogokkal sem tudom a Profile könyvtárat és tartalmát törölni. Rendszergazdaként hozzá kellene férnem a felhasználók Roaming Profiljaihoz. Van erre megoldás?*

V: Háromféle megoldás is létezik:

1. Az egyik, ha a felhasználó profilkönyvtárán a „Take Ownership" joggal élünk, és – az Administrators csoport tagjaként – magunkévá tesszük a felhasználó teljes Profile könyvtárát és tartalmát, majd megfelelő jogokat adunk magunknak.
2. Jobb módszer, ha a felhasználó létrehozásakor *(még mielőtt belépne)* létrehozzuk a Roaming Profile-hoz azt a könyvtárat, amit beállítottunk. Így nemcsak a felhasználónak, hanem a rendszergazdá*(k)*nak is jogot adhatunk a könyvtárhoz.
3. Legjobb módszer, ha a Default Domain Policyben beállítjuk, hogy a profile könyvtárak jogaihoz adja hozzá az Administrators csoportot is.



Profile könyvtárak jogainak állítása Group Policy segítségével

Ez a beállítás csak számítógépekre értelmezett. Tehát nem elég a profilokat tartalmazó szerveren beállítani, hanem ki kell terjeszteni az ügyfélszámítógépekre is, mivel az első belépéskor a jogok beállítását az ügyféloldal szabályozza.

Könyvtár/fájlméretek meghatározása egy könyvtárstruktúrában

K: *Van tudomása bárkinek olyan programról, amivel egy könyvtárstruktúrában rekurzívan ki lehet gyűjtetni a legnagyobb méretű fájlokat vagy könyvtárakat? Nálunk napról napra csak fogy a hely, és rejtélyes könyvtárakat, bennük hatalmas telepítőkészleteket találok. Ezeket szeretném valahogy automatikusan megtalálni.*

V: Használjuk a Windows 2000 Resource Kit diruse segédprogiját, az pont erre való.

Vigyázzunk, hogy a vizsgált területen mindenhova legyen jogunk, mert csak azokat a könyvtárakat tudjuk elemezni, amelyek tartalmát el tudjuk olvasni. A profilkönyvtárak tartalmát például nem. Kibúvó: legjobb, ha Backup Operators csoport tagjaként futtatod a programot.

(Forrás: Windows 2000 levelezési lista)

Jelszóváltoztatás webes felületről

K: *Webes felületen hogyan tudom a felhasználónak biztosítani, hogy megváltoztassa a jelszavát?*

V: Ha NT4 domainben, vagy Active Directoryban lévő felhasználóról van szó, akkor ADSI-val lekérhetjük a felhasználót egy IADSUser objektumba, majd annak meghívhatjuk a SetPassword metódusát. Példa:

```
Dim usr As IADsUser
Set usr = GetObject("WinNT://Microsoft/JSmith,user")
usr.SetPassword "topsecret98"
```

(Forrás: fejlesztői levelezési lista)

RAM vizsgálat programból

K: Csereprocesszor nélkül meg lehet-e valahogy vizsgálni a processzort valamilyen diagnosztikai programmal? (Mert amennyire én tudom, a RAM-ot is csak cserélgetéssel lehet.) A RAM hibát ki lehet zárni a három RAM modul cserélgetésével. Úgy tűnik, nem ezzel van a gond...

V: Már miért ne lehetne a RAM-ot programmal tesztelni? Igaz, kisebb hatékonysággal és alapossággal, mint egy speciális teszterrel, de vannak nagyon jó programok is erre. Ami nálunk nagyon bevált, a ramexam nevű program, az **[1]** címről beszerezhető.
A processzor vizsgálatára is vannak programok, mondjuk valamelyik Checkit. Azért ez nem RAM, ahol minták beírásával, illetve annak korrekt visszaolvasásával lehet a hibát meghatározni.
(Forrás: Windows 2000 levelezési lista)

Kredencek ütközése

K: Most installáltam egy Active Directoryt a hálózaton az egyik gépre. Egy Windows NT-ről be akarok lépni erre a gépre, és azt írja ki, hogy: „The credentials supplied conflict with an existing set of credentials."

V: Az a baj, hogy hozzá vagy már kapcsolódva.
Parancssorból add ki a net use * /del parancsot, és meglátod, menni fog! Vagy kapcsolódj FQDN néven a másik szerverhez. Az FQDN *(vagy IP-cím)* és a NetBIOS névvel történő kapcsolódás két külön dolog, ezért egymással párhuzamosan, más felhasználói névvel is lehet így csatlakozni.
Tehát kétféle módon is kapcsolódhatsz egy időben:

```
net use \\server1 /user:domain\Jozsi
és
net use \\server1.domain.net /user:domain\Pisti
vagy
net use \\192.168.0.1 /user:domain\Pisti
```

(Forrás: Windows 2000 levelezési lista)

Kiterjesztés nélküli fájlok megnyitása

K: Az egyik programunk kiterjesztés nélküli fájlokat generál. Már nagyon unom, hogy mindig rákérdez ezeknél a fájloknál az Explorer, hogy mivel is akarom megnézni, és nem engedi, hogy rendeljek hozzá programot. Van valami megoldás erre?

V: Ilyen fájlok esetén a registry módosításával lehet megadni az alapértelmezett programot. Egy .reg kiterjesztésű fájlba írjuk be a következőket:

```
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT\.]
@="SPEC"
[HKEY_CLASSES_ROOT\SPEC]
[HKEY_CLASSES_ROOT\SPEC\shell]
[HKEY_CLASSES_ROOT\SPEC\shell\open]
[HKEY_CLASSES_ROOT\SPEC\shell\open\command]
@="c:\\winnt\\notepad.exe \"%1\""
```

Az első és az utolsó sor az operációs rendszertől függően más lehet. Természetesen a notepad.exe csak egy példa. Az utolsó sorba annak a programnak az elérési útját és nevét kell rakni, amelyikkel a kiterjesztés nélküli fájlokat meg akarjuk majd nyitni.
A .reg fájlt könnyedén hozzá tudjuk adni a registryhez *(jobb klatty – Merge, vagy egyszerűen dupla klikk a fájlon)*.
Ha nem csinálunk .reg fájlt, egyszerűen registry editorral is létrehozhatjuk ezeket a kulcsokat és értékeket.

XP-be épített ZIP tömörítő kiiktatása

K: Nincs szükségem a Windows XP-be beépített tömörítőre, szeretnék megszabadulni tőle. Lehetséges ez?

V: Igen. A beépített tömörítést a system32-ben található zipfldr.dll végzi. Ki lehet iktatni, a következő paranccsal:

```
regsvr32 /u %systemroot%\system32\zipfldr.dll
```

A gép újraindítása után már nem fog működni a beépített tömörítő. Ha később mégis használni kellene, akkor a /u nélkül kiadva ugyanezt a parancsot, újra használhatóvá válik a funkció.

XP Windows Messenger kikapcsolása

K: Szeretnék megszabadulni a Windows Messengertől. Nem használják a felhasználók és nem is szeretném ha használni tudnák. Ne is jelenjen meg induláskor, sem később.

V: Group Policy segítségével le lehet tiltani a Messengert. A Local Policy-ban ezt két helyen is meg lehet találni, egyszer a felhasználók, egyszer pedig a gép beállításai közt. Mind a két helyen a az Administrative Templates\Windows Components\Windows Messenger útvonalon van két beállítás.

A Messenger letiltása

Ha mind a kettőt „Enabled" állapotra hozzuk, akkor nem tudják majd használni a felhasználók a Messengert és a gép indulásakor sem fog betöltődni. Ha a gép és a felhasználók beállításai közt is állítjuk a konfigurációt, akkor a gépre érvényes házirend fog érvényre jutni.

XP és a NetBEUI

K: Jól látom, hogy nem látom a NetBEUI-t a Windows XP Professional-ben? Ha lehet ilyet használni XP-ben, akkor hogyan kell beállítani?

V: A Microsoft KB Q301041 cikke adja meg a választ. Lehet, de „manuálisan" kell feltenni, mert a NetBEUI ezentúl nem támogatott protokoll.
A cikk tartalma nagyjából a következő: Összesen két fájl kell a NetBEUI működéséhez: a netnbf.inf és a nbf.sys. Ezek a fájlok az XP telepítő CD-n a Valueadd\MSFT\Net\NetBEUI könyvtárban találhatóak. A telepítés menete a következő:

1. Az nbf.sys-t be kell másolni a %SYSTEMROOT%\System32\Drivers könyvtárba.
2. Az Netnbf.sys-t be kell másolni a %SYSTEMROOT%\Inf könyvtárba.
3. A Control Panelben vegyük elő a Network Connectionst.
4. A hálózati kapcsolatok közül válasszuk azt, amelyikhez szeretnénk NetBEUI-t rendelni, majd jobb kattintás után válasszuk a Tulajdonságokat.
5. General fülön kattintsunk az Install gombra, majd Protocol – Add, ezután válasszuk a NetBEUI-t, végül az OK gombot.
6. Utolsó lépésként indítsuk újra a gépet.

A cikkben szereplő URL-ek:

[1] http://www.qualitas.com/product/ramexam/whatisre.htm

Farsite

Úgy tíz évvel ezelőtt egy Deltában láttam, amint japán kutatók bemutatták, hogyan tudnak szemétből gyémántot előállítani, mondhatnám varázsolni. Önkéntelenül ez jutott eszembe, amikor először hallottam a Farsite-ról, mert ezzel temérdek ócskavasból *(azaz idejétmúlt, de még nem teljesen használhatatlan számítógépből)* gigászi tárhelyet, amolyan internetes bőségszarut építhetünk.

Mi is tehát a Farsite? Egy **szimbiotikus, kiszolgáló nélküli, elosztott** állományrendszer. Ez tuti jól hangzik, de mit jelentenek e felemelő, marketingízű jelzők egy állományrendszer esetében? Vegyük ezeket sorra!

A biológiából kölcsönzött szimbiózis szó kölcsönösen hasznos együttélést jelent. Az ebből alkotott jelző rendkívül találóan írja le a Farsite közösség számítógépeinek *(felhasználóinak)* helyzetét. Mindenki hasznot húz a tagságából, hiszen élvezi a Farsite „növelt értékű" szolgáltatásait. A kölcsönösség abban nyilvánul meg, hogy „cserébe" mindenki ad tárhelyet a többiekhek *(nem kötelező, de elég valószínűtlen, hogy ne ezt tenné az, akinek akad némi fölösleges bájt a merevlemezén)*.

A kiszolgálóval működő elosztott állományrendszer nem ismeretlen fogalom sem a Windows, sem a Unix világában. Sajnos azoknál egyetlen, központi gép tartja számon, hogy a globális névtér egyes részei hol találhatók a valóságban. Ha ez a gép elesik, oda minden. Nem úgy a Farsite esetében, ahol maga az állományokat nyilvántartó címtár is elosztott. Ennek eredményeképpen nem számít, ha néhány gép tönkremegy, netán kikapcsolják azokat, mert a redundáns felépítés következtében továbbra is összeállítható lesz a kérdéses állomány a még működő gépekről. Magától értetődően előállhat olyan olyan állapot, amikor egyes fájlok nem elérhetők, de ez már egy adott rendszer tervezésének kérdése. A központi kiszolgáló mellőzésének van még egy nagy előnye, amire a vezető beosztású informatikusoknak biztosan felcsillan a szeme: nem kell hozzá rendszergazda, és ezzel tovább faraghatók a bérköltségek. A közösség tagjai maguk szabják meg, hogy erőforrásaikat ki használhatja. A házirendet jelenleg úgy képzelik, hogy a közösbe bedobottal arányos nagyságú tárhely használható. Tulajdonképpen ez a legfontosabb tulajdonság, és nemcsak azért, mert a költségcsökkentés a döntéshozók vesszőparipája, hanem mert ezt mindez idáig csak a Farsite tudta megvalósítani.

A címtár mellett magukat az állományokat is elosztott módon tárolják. A recept szerint mindent apró „kockákra" vágnak, majd ugyanazt a darabot több helyen is elmentik. Mindezt leginkább egy hálózati RAID rendszerhez lehetne hasonlítani. Nem vész el némi tárhely a redundancia miatt? Talált, süllyed, de csak félig, mert ezt az a technika hivatott kompenzálni, amely az egyazon gépen lévő, azonos fájlokat összeolvasztja – részben a Windows 2000-ben már megtalálható Single Instance Storage módszer segítségével.

Az I/O teljesítmény drasztikus esése miatt sem kell aggódni, mert az adott időn belül használt fájlok egy változó méretű, lokális gyorsítótárban is megtalálhatók.

Nem is tudom miért, de úgy érzem, hogy itt egy termékgyanús rendszerrel állunk szemben. Ebben az esetben viszont piac is kell. Kinek kell majd egy ilyen állományrendszer, és miért?

A Farsite projekt célközönségét olyan intézmények alkotják, ahol mintegy 10^5 gépen 10^{10} számú állományban 10^{16} bájt adatot tárolnak. Százezer bizony nagy szám, és itt sokan talán azt gondolják, hogy ezt már nem is érdemes tovább olvasni. De igen, érdemes. A Farsite rendszerek kialakítását ugyanis nem feltétlenül egy vállalatnak, egyetemnek stb. kell kialakítani. Ezt a hálátlan, de valószínűleg jól jövedelmező feladatot direkt erre specializálódott szervezetek, esetleg a jelenlegi Internetszolgáltatók fogják felvállalni. Ők lesznek a tárközvetítők. Vállalatok, egyetemek és más intézmények, sőt, akár magánszemélyek is ezeken a közvetítőkön keresztül férnek hozzá némi díjazás ellenében a bőségszaruhoz, és ugyanígy ajánlhatják fel saját táruk egy részét a Farsite közösség javára, szintén némi díjazás ellenében. Így vagy úgy, előbb-utóbb mindenki számára elérhető lesz a szolgáltatás.

Akár egyéni felhasználót, akár egy intézményt tekintünk, felmerül a biztonság kérdése. Beszállok ebbe a buliba, és vadidegen gépeken tároljam a szerelmes leveleimet? Ez így nem kóser. Valóban. De egy gépen legfeljebb néhány morzsa, apró „kocka" található ugyanabból az állományból, így több felhasználó összefogása kellene ahhoz, hogy a teljes puzzle-t kirakják. És mi van, ha összefognak, hogy legbelsőbb titkaim felfedjék? Csak semmi pánik! Ha még nem mondtam volna, a morzsákat titkosítják, a hozzáférést pedig digitális aláírások révén ellenőrzik, így aztán összefogás ide vagy oda, a lecke fel van adva a cracker-társadalomnak.

Legyünk paranoiásak, és tegyük fel, hogy a gonosz crackerek átvereksznek magukat a biztonsági vonalakon, és néhány gépen sikerül megváltoztatniuk a szerelmeslevelet felmondólevéllé. Ekkor az a furcsa helyzet áll elő, hogy a levelem néhány gépen szerelmes lesz, néhányon viszont felmondó. Mi az igazság? Ez az ún. bizánci probléma: a csatára készülő seregeket több tábornok és a főparancsnok vezeti, akik futárok segítségével üzenhetnek egymásnak. Vagy mindannyian egyszerre támadnak, vagy egyikük sem. A döntést a főparancsnok hozza meg. A bibi mindössze annyi, hogy a tábornokok, de maga a főparancsnok is küldhet megtévesztő üzeneteket, vagyis míg egyes tábornokokat támadásra szólíthatnak fel, másokat épp az ellenkezőjére. A probléma megoldására többféle algoritmus is létezik, és természetesen a Farsite fejlesztői is alkalmaznak egy ilyet a vázolt eset feloldására.

Fontosnak tartom még azt is megemlíteni, hogy ebből a nagy masszából mit látok én, a felhasználó. Látok egy globális névteret, amiben az állományokat azok tényleges, fizikai helyétől függetlenül érhetem el. Ennyit, és nem többet. Olyan ez, mintha egyetlen kiszolgáló állna velem szemben. A jelenlegi változatban ezt a hatást még az is erősíti, hogy a legfelső szintet egy betű jelöli: egy virtuális állománykiszolgáló virtuális meghajtójáé.

Zacco@fw.hu

A cikkben szereplő URL-ek:

[1] http://research.microsoft.com/sn/farsite

Következő számunk tartalmából

Windows: SAK? MATT!
Nemrégiben egy kicsiny cég ügyvezetőjével beszélgettem, aki a maga szemszögéből a következőképpen látta a Linux kontra Windows harcot: van Windowsuk is, Linuxuk is, de a Linux – szerinte – nagyságrendekkel kevesebb törődést igényel *(igaz, szinte semmit sem tud, mert mindent kivettek belőle, akik telepítették)*. A Server Appliance Kit megjelenéséig nehéz volt erre okosat mondani, ma már van válasza a Microsoftnak erre a problémára.

Research:
Elérkeztünk a világgép sorozat harmadik részét képező második komponenshez, az elosztott, online alkalmazásokhoz. A „táptalaj" a Pastry, amely a résztvevő gépek *(peer-to-peer)* társalkalmazásai számára elosztott, méretezhető, önszervező és hibatűrő réteget biztosít, továbbá hatékonyan valósítja meg a Pastry-gépek közötti útválasztást, a terheléselosztást és az objektumok helyének meghatározását.

Exchange:
Hasonlóan a Windows 2000 Csoportos házirendekhez, házirendek segítségével az Exchange egyes beállításait is központilag lehet elvégezni. Mint minden házirend, ez is azért hasznos, mert így egy helyről lehet beállítani olyan dolgokat, amelyek tágabb körben érvényesek.

Biztonság: IPSec a gyakorlatban
E havi számunkban bemutattuk, mihez vezethet, ha az IPSec-ben, mint csomagszűrőben megbízunk. De vajon nyugodtak lehetünk-e akkor, ha az IPSec-et arra használjuk, amire eredetileg kitalálták? Hogyan működik az IPSec? Minderre megkapjuk a választ a következő számban.

W2000: DNS mégegyszer
Nem is olyan régen bemutattuk a Windows 2000 dinamikus DNS szolgáltatását. Most kicsit visszanyúlunk az alapokhoz: hogyan viselkedik a hagyományos DNS a Windowsban?

Exchange Web Storage System
Az előzőekben láthattuk, hogyan tudunk információt kinyerni a WSS-ből különféle technológiákkal, illetve hogyan tudunk fájlokat csatolni a WSS-beli elemekhez. Mindezek mellett csak említést tettünk arról, hogy van lehetőségünk saját jellemzők felvételére, kezelésére. Most ezt a kérdéskört járjuk körbe.

XMLGessünk:
Az előző két részben áttekintettük az XML Schema alapvető elemeit, így a következő részben már megfelelő elméleti alapokkal felvértezve nekiláthatunk a séma gyakorlati felhasználásának COM és .NET alapokon is.

.NET Akadémia
A string alaptípus lett a .NET-ben, és elég sok mindent tud. Mit érdemes róla tudni, hogyan használjuk? Milyen tömböket és kollekciókat definiálhatunk a .NET-ben? Hogyan működik az átjárás a stringek és bájttömbök között? Ezekre a kérdésekre keressük a választ a következő részben.

Szimpla KV

Hogyan kell felkészíteni az Active Directoryt az XP-specifikus házirendbeállítások befogadására?

a) A Windows 2000-es tartományvezérlőről Terminal Servicessel egy XP-re csatlakozunk, és ettől a háttében megtörténik a csoda: a csoportos házirend kiegészül az XP újdonságaival.

b) A Windows XP-ről beterminálunk a tartományvezérlőre, és így megnyitjuk a csoportos házirendet. Ettől megtörténik a csoda...

c) A Windows XP-n, helyben indítjuk el a házirendszerkesztőt, majd ennek segítségével csatlakozunk az Active Directoryra, azon belül is a megfelelő szervezeti egységre, végül a házirendre, és ettől a háttérben megtörténik a csoda...

A helyes válasz: c

Előfizetési szelvény

Címzett: NetAcademia Kft.
Faxszám: (1) 261-7145

fax

Tisztelt Olvasónk!
Lapunk hírlapárusi forgalomba nem kerül, ezért ha kíváncsi megkezdett sorozataink folytatására, kérjük töltse ki és juttassa el hozzánk az alábbi megrendelőlapot. Előfizetőink kedvezményesen vehetnek részt konferenciáinkon, tanfolyamainkon illetve egyéb rendezvényeinken.
Kérjük töltse ki ezt az előfizetési szelvényt és faxolja el az (1) 261-7145-ös faxszámra.

Új előfizetési AKCIÓ!

A 2002. április 1-től indult új előfizetési akciónkban minden új előfizetőnk aki 2002. július 30-ig egy évre előfizet a tech.net magazinra, ajándékul megkapja tőlünk a 2002. januári, februári és márciusi számokat. Fizessen elő egy évre most, hogy küldhessük Önnek az ajándék magazinokat!

Az akció határideje: 2002. július 30.

http://technet.netacademia.net/subs • e-mail: terjesztes@netacademia.net • fax: (1) 261-7145

Előfizetem a **tech.net** magazint: ...példányban egy évre (14.784 Ft)
...példányban fél évre (7.392 Ft)
...pld.-ban.NET akcióval ***(12+3 szám)*** (14.784 Ft)

Az előfizetés kezdete:...../...../.....

Előfizető neve:. . . . . .

Cég neve: . . . . . .

Cím: ☐☐☐☐ . . . . . .

E-mail cím: . . . . . .

Telefon: . . . . . .

Fax:. . . . . .

Fizetés módja: ☐ csekken *(postán küldjük)* ☐ átutalással

Kelt:...../...../..... Aláírás:.....

Amennyiben a számlázási cím nem egyezik meg a szállítási címmel, kérjük az alábbi részt is töltse ki!

Számlázási cím: **Szállítási cím:**

....

....

tech.net working with windows