tech.net
working with windows

next

Whistler

Az Exchange Server telepítése 37. oldal
Dinamikus DNS 15. oldal
Egységben az erő 44. oldal

III./04.szám
1344 Ft

ISSN 15865185
04
9 771586 518005

Tanévzáró

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 átütő 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

Mint minden rendes iskolában, a NetAcademia-ban is véget ér júniusban a tanév. Egyfelől azért, hogy lélegzetvételhez jussunk mi magunk: bepótolhassuk elmaradásainkat, kimehessünk végre ügyfeleinkhez, akik január óta hiába várnak ránk... Másfelől azért, hogy az informatikus-társadalom is fellélegezhessen egy kicsit. Nincs elegük az állandó rohanásból? Ha nincs, ne hagyják olvasatlanul nyári számainkat, hisz most fogunk igazán belehúzni az újságírásba. A lelkesebbek idén is, mint tavaly nyáron, vigyenek magukkal mindenhova egy tech.net magazint: tengerpartra, hegyek közé, völgyek közé.

A tanévzáróra június 7-én kerül sor, ahova szeretettel várunk mindenkit! A rendezvényt elsősorban azok tiszteletére rendezzük, akik az elmúlt félévben tanfolyamon *(NATE, Ugrósuli vagy hagyományos MOC)* voltak nálunk, de természetesen azokra is számítunk, akik a komoly előadások, vagy éppen a komolytalan végkifejlet miatt érdeklődnek. Ezen az utolsó tanítási napon valami tanulnivalóval is szeretnénk fárasztani a jelenlévőket, de nem lenne az évzáró NetAcademiás, ha a vakáció örömére nem vennénk komolytalan fordulatot a nap végére. De lássuk az elejétől.

Két előadás is lesz: az egyik elsősorban rendszergazdáknak szól, és az IPSec hálózati adatfolyam-titkosítás lesz a témája. A Windows 2000-be beépített adatfolyam-titkosítás teljesen zökkenőmentesen, a háttérben dolgozva védi meg a vállalati adatokat az avatatlan szemek bepillantásától. Hiába „ingyenes", nagyon kevés helyen használják, mert túl bonyolultnak tűnik. Első ránézésre az is.

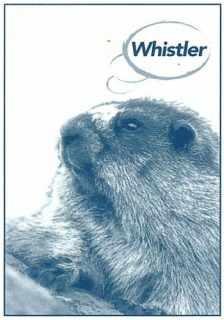
A második inkább a fejlesztőkre kacsint: a régóta várt Design Patterns elmélettel ismerkedhetünk meg. Ez ahhoz kell, hogy megértsük a .net Framework filozófiáját *(is)*. Tulajdonképpen az objektumorientált programozás kulcskérdése a tervezési minták ismerete: hogyan építsünk az egyedi objektumokból *(téglák)* házat? A témakörtől nem kell megijedni: mindenről lehet úgy beszélni, hogy érthető legyen! E két előadás után majd jöhet a ballagás: akit érdekel, beballaghat velünk a CEU konferenciaközpont úszómedencéjébe. *(A víz hideg, de nem mély. Fázósaknak jelentem: szauna is van. A medencében koktélt szolgálunk fel. A ballagás nem kötelező, kihagyható.)* Ezután következik az össznépi kolbászsütés, majd érzelgős búcsú egymástól, a tavasztól és a tanévtől. Részletesen:

14:00-15:00	Biztonságos adatátvitel IPSEc-kel	Fülöp Miklós
15:15-16:15	Design Patterns	Soczó Zsolt
16:15-16:30	Tanévbúcsúztató beszéd	Fóti Marcell
Szünet		
16:30-17:30	Ballagás. Fürdőruha viselése ajánlott!	
18:00-19:30	Interaktív kolbászsütés a kertben	

A részvétel az extra szolgáltatások miatt sajnos nem ingyenes, hanem 10.000 Ft+áfa, *volt hallgatóinknak 5.000 Ft+áfa*. Érdeklődni, jelentkezni az info@netacademia.net címen lehet – volt hallgatóinkat pedig személyesen is megkeressük.

Fóti Marcell
marcellf@netacademia.net

2002 Április



Windows.NET Server Beta 3

Beta 3 állapotba érkezett a Windows kiszolgáló operációs rendszerek következő verziója. A korábbi tech.net számokban még csak Whistler kódnéven emlegetett terméknek már végleges neve is van: Windows.NET Server. Mi mégis a Whistler kódnév eredete után nyomozunk...

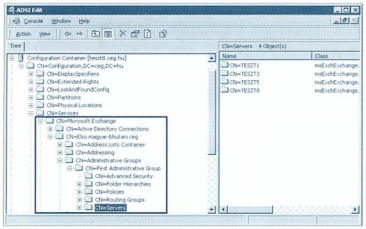
10. oldal

Farkasokkal táncoló

Cluster a gyakorlatban (VI. rész)

Elkanyarodunk a különböző típusú erőforrások taglalásától – az a néhány, ami még hátra van, megvár minket, amikor a nagyobb alkalmazások fürtösítését vizsgáljuk. Most azt vesszük górcső alá, miként viselkedik a fürt, ha a feladata a hálózati infrastruktúra ellátása, címtárszolgáltatás, globális katalógus és egyéb, a Windows 2000 számára kritikus szolgáltatás biztosítása.

5. oldal



Ki mivel ♥?

Exchange telepítés felsőfokon

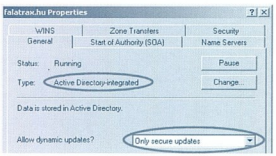
Hagyományos problémabemutató rovatunkban ma egy ritkán előforduló, ám tanulságos Exchange balesetet elemzünk, mely rögtön telepítéskor agyonvágja a világ legátgondoltabb rendszertervét is – ha megfelelően gyorsak vagyunk :). Megtudhatjuk azt is, hogyan távolítható el a félkész Exchange 2000 abban az esetben, ha a SETUP sem képes azt eltávolítani.

8. oldal

Dinamikus DNS *(II. rész)*

Az előző részben bemutattuk a dinamikus DNS regisztráció működését. A regisztrációt abban a formájában tulajdonképpen bárki elvégezheti, aki hálózati kapcsolatán keresztül képes felvenni a kapcsolatot a DNS kiszolgálónkkal. Szerencsére – Windows 2000 tartományon belül – a zónákon az adatok módosítását előzetes bejelentkezéshez kötheti: ez a biztonságos zónafrissítés, azaz az „Only secure updates".

15. oldal



Etheral

Ingyenes hálózatanalizátor

Cikkeinkben régóta hivatkozunk a Microsoft Network Monitorra, mint kötelező eszközre. Egy valamirevaló rendszergazda nem nagyon megy semmire egy rendes hálózatanalizátor program nélkül – mondjuk.

19. oldal

Microsoft Metadirectory Services

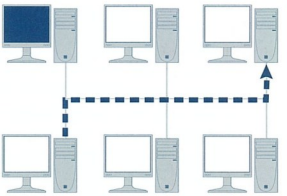
(I. rész)

Magyarországon is egyre több vállalat jut arra a felismerésre, hogy a heterogén infrastruktúra hosszú távú örökség is lehet – az egységesítés fontos, ám költséges és technikai kihívásokkal teli folyamat. Metacímtár kialakításával rövid távon is égető problémákat oldhatunk meg, miközben megtesszük a kezdőlépést hosszú távú egységes címtárstratégiánk megvalósítása felé.

22. oldal

IP Multicast

Unicast és Broadcast



Az IP Multicast az a hálózati forgalomtípus, mely a Unicast és Broadcast között helyezkedik el félúton, s amely szép új jövőnk alapja. E nélkül ugyanis nehezen lehetne elképzelni Pirx kapitány kalandjait, hisz ha nincs Multicast, nincs sokrésztvevős videokonferencia sem itt a Földön, sem a jövőben, az űrben.

26. oldal

.NET Akadémia

Delegate-ek és események (IV. rész)

Az előző számban megismerkedtünk a delegate-ekkel, és láttunk egy egyszerű példát a használatukra. Valójában sokkal többet tudnak, mint amit akkor felvázoltunk, képesek sok metódusreferencia meghívására is. Miután kiteljesítjük a korábban megkezdett atomreaktoros példánkat, rájövünk, hogy az ott végzett munkánk jelentős részét megspórolhatjuk multicast delegate-ek felhasználásával.

29. oldal

XMLgessünk

Az XML Schema (XII. rész)

Számtalanszor hivatkoztunk már XML sorozatunkban az XML sémára, így hát itt az ideje, hogy közelebbről megismerjük. Ebben a részben áttekintjük az alapismereteket, és megnézzük az egyszerű típusok képzésének fortélyait. A következő részben áttekintjük az összetett típusok kialakításának részleteit, a harmadik, záró részben pedig tervezési mintákat nézünk át újrafelhasználható és jól olvasható sémák írásához, valamint megnézzük milyen eszközökkel lehet ténylegesen kihasználni a sémák adta előnyöket.

33. oldal

Az Exchange Server felügyelete

Ismerkedjünk meg az Exchange 2000 felügyeleti eszközeivel, valamint nézzük meg az Active Directory Users and Computers snap-inbe integrált Exchange varázslókat!

37. oldal

A Web Storage System

Programozott levélkezelés

Exchange sorozatunkhoz kapcsolódóan néhány cikk erejéig betekintünk a Web Storage System programozott elérésébe. Legelőször az ADO-n keresztüli hozzáférést vizsgáljuk meg.

41. oldal

Patchwork

Adminisztráljunk Windows 2000-et Windows XP professional kliensről

43. oldal

Egységben az erő

44. oldal

Farkasokkal táncoló

(VI. rész) – Cluster a gyakorlatban

Elkanyarodunk a különböző típusú erőforrások taglalásától – az a néhány, ami még hátra van, megvár minket, amikor a nagyobb alkalmazások fürtösítését vizsgáljuk. Most azt vesszük górcső alá, miként viselkedik a fürt, ha a feladata a hálózati infrastruktúra ellátása, címtárszolgáltatás, globális katalógus és egyéb, a Windows 2000 számára kritikus szolgáltatás biztosítása.

A farkasfalkára egy nagyvállalatot is rá lehet bízni – és ilyen, majdnem tisztán fürtökből álló rendszer már van Magyarországon.

A feladat

Nem sokkal a Windows 2000 megjelenése előtt azt a feladatot kaptam, hogy készítsek egy tervet szerverparkunk megújítására. Több telephelyes vállalat a miénk, öt városban, hat helyszínen vagyunk jelen. Három üzemünkben egyenként 120-160 felhasználónk dolgozik, a kisebb irodaépületekben 10-40 munkatárs végzi napi tevékenységét – az egyik kis telephely ezek közül az informatikai központ.

Olyan kiszolgálókat kellett lecserélni, amelyek már majdnem négy éve üzemeltek, nem voltak bővíthetők, s mivel a felhasználók számának növekedését nem tudták követni, itt is, ott is egy-egy újabb, PC-ből avanzsált társuk kerekedett. A „szerves" fejlődés eredményeképp az egyik tartományvezérlő volt, a másik nem, az egyik nyomtató-kiszolgálóként és Exchange levelezőként egyaránt funkcionált, a másikon csak egy SQL Server árválkodott – elég vegyes volt a kép, nem kell ezt ragozni.

A feladat minket érdeklő része két egyszerű, egymásnak teljesen ellentmondó cél elérése volt:

- legyen kevesebb kiszolgáló, és
- legyen nagyobb a rendelkezésre állás.

Miért ellentmondó célok ezek? Ha az állománymegosztások három kiszolgálón, szétszórva működnek, még ha kiesik is az egyik, az állományelérés, mint funkció mégsem szűnik meg teljesen – nem mondhatjuk, hogy nincs semmi, legfeljebb, hogy nem működik minden. Ha valamennyi közös mappát egyetlen vasra koncentráljuk, annak kiesése a teljes funkció elvesztését jelenti – ez botrány lenne. A kiszolgálók számának csökkentése viszont nemcsak azt jelenti, hogy kevesebb lesz belőlük, hanem azt is, hogy tervezetten ugyan, de eleve több funkciót látnak majd el. Ez még roszszabb helyzetet teremt: a megosztásokkal együtt az összes hálózati nyomtató is eltűnik! A megoldás a fürtszolgáltatás után kiált.

Elkezdtem tehát a cluster után kutatni – ám leginkább olyan esettanulmányokat találtam, amelyek a sorozat első részében emlegetett tiszta konfigurációt tartalmaztak, vagyis amikor a fürtöt egyetlen feladatra, például egy ERP rendszer üzemeltetésére használták. Ez nem jó hír, ugyanis telephelyenként legalább két Windows 2000 tartományvezérlő, két DNS- és két globális katalógus kiszolgáló dukál, nem is beszélve a DHCP és a WINS infrastruktúráról. Úgy tűnt, hogy a fürtszolgáltatás nem visz előre, mert ugyan nagyobb megbízhatóságot nyújt, de több szervert követel. Vagy mégsem?

tech.net – kimerítő mélységekig

Elhatároztam, hogy mindent összegyűjtök a fürtökről, és ha lehet, akkor erre a szolgáltatásra építve készítem el a tervet. Először néhány alapvető problémát kellett tisztázni.

Világossá vált, hogy a fürtszolgáltatás csak egy szolgáltatás a szerveren, az tehát nem lenne újdonság, hogy emellett más szolgáltatások is megjelennének. A kérdés csak az, hogy hogyan viszonyulnak a fürthöz. Irodalmazással kiderült, hogy sokféle lehet ez a viszony:

- A fürtözésre felkészített *(angolul: cluster-aware)* szolgáltatások szeretik és várják a rendelkezésre állás növelését. Számos ilyen létezik: az állománymegosztások, a nyomtatósor, a WINS, a DHCP, az IIS, illetve a külön megvásárolható termékek, mint az Exchange, az SQL, a Biztalk stb. Az ilyen funkciók elkészítésekor a programtervezők számoltak azzal, hogy az alkalmazás fürtre kerül, ezért beépítették a fürt támogatását.
- Vannak fürtözésre fel nem készített, de fürtözhető alkalmazások. Ezek általában nem Microsoft-termékek, de szerkezetükben olyanok, hogy fürtözésük lehetséges. A nálunk működő J.D. Edwards vállalatirányítási rendszer is ilyen, de gyakorlatilag a legtöbb, szabványos NT szolgáltatást létrehozó alkalmazás fürtözhető.
- Vannak nem fürtözhető alkalmazások, amelyek azonban működhetnek a fürtszolgáltatás mellett. A Windows 2000 önmaga is tartalmaz ilyeneket, például a DNS-t és a címtárszolgáltatást. Önálló termékként megemlíthetjük a SharePoint Portal Servert.
- Végezetül vannak olyan alkalmazások, amelyek ütköznek a fürtszolgáltatással, és nem telepíthetők Windows 2000-re, amennyiben a cluster működik. Ilyen a Systems Management Server minden verziója vagy harmadik féltől származó programok közül a Sotfware Metrics „Print Accounting Server" nevű terméke, amely a nyomtatási költségeket képes nagyon precízen költséghelyekre lebontani.

A fürtözésre felkészített (angolul: cluster-aware) szolgáltatások szeretik és várják a rendelkezésre állás növelését.

Nos, történetünk szempontjából az a fontos, hogy minden, az operációs rendszerrel kapott Windows 2000 szolgáltatás képes működni a fürt mellett, még ha a fürtbe nem is lehet bevonni. Ezek után nincs más teendő, mint megtervezni a teljes hálózatot.

Fürtökre alapozott infrastruktúra

Vállalatunk az alábbi alapvető informatikai szolgáltatásokat nyújtja minden telephelyen a felhasználóknak:

- Egységes címtár *(beléptetés, hitelesítés)*
- Állományok központi elérése
- Központi *(kiszolgálón definiált)* nyomtatók
- Levelezés, kiszolgálón tárolt postaládákkal
- Internet, központi kijárattal
- Antivírus-rendszer, automatikus vírusadatbázis-frissítéssel
- Intranetoldalak *(nem túl sok)*

Mindezek mellett vannak a felhasználók által nem ismert, de létező szolgáltatások, mint a névfeloldás és a tallózás *(DNS, WINS)*, az IP cím adminisztráció *(DHCP)* és persze a mentés. Természetesen nálunk is van vállalatirányítási rendszer, de a mi szemszögünkből az nem infrastrukturális szolgáltatás, hanem alkalmazás – tehát most nem tárgyaljuk. *(Egyébként persze az is fürtön fut.)* A cluster nem olcsó, tehát csak bizonyos számú felhasználó esetén érdemes alkalmazni. Ökölszabályként azt mondhatjuk, hogy ha egy szolgáltatást 100 felhasználó igényel, akkor azt érdemes fürtösíteni. Nálunk négy ilyen telephely van, vagyis legalább négy egyforma fürt kellene, a következő feladatokkal:

1-es állomás, **nem fürtözött** szolgáltatások:

- Tartományvezérlő
- DNS, GC
- Antivírus szoftver

1-es állomás, **fürtözött** szolgáltatások:

- Állománymegosztások
- Webszolgáltatások
- Nyomtatómegosztások
- DHCP, WINS
- Mentés

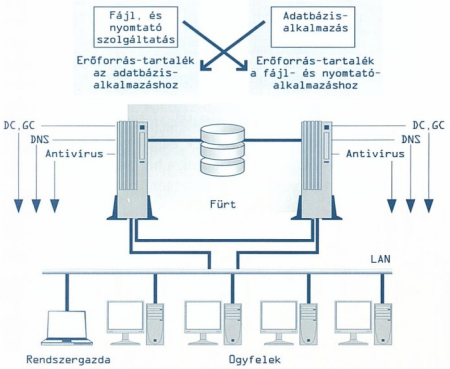
2-es állomás, **nem fürtözött** szolgáltatások:

- Tartományvezérlő
- DNS, GC
- Antivírus szoftver

2-es állomás, **fürtözött** szolgáltatás:

- Levelezés *(Exchange 2000)*

Egyszerűbb ezt egy ábrán megmutatni:



Egy infrastruktúrafürt felépítése és funkciói

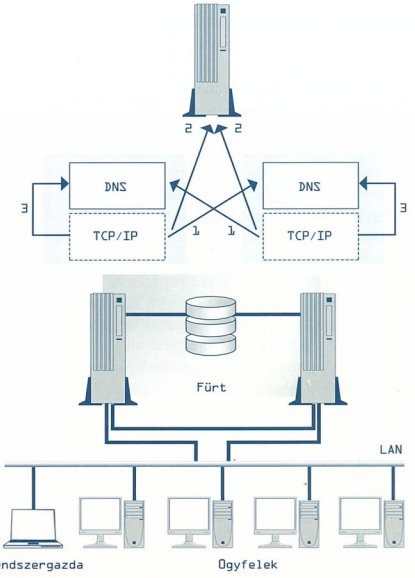
Az adatbázisalkalmazás jelen esetben az Exchange 2000. *(Tényleg az! Ha valaki nem hiszi, járjon utána!)* Minek nevezzük ezt a konfigurációt? Aktív-passzívnak, vagy aktív-aktívnak? Tulajdonképpen mindkét állomás aktív, olyan feladatokat lát el normál esetben, amilyeneket a másik nem. A szolgáltatások felől nézve azonban ez aktív-passzív szerkezet, mert egy szolgáltatás *(a fürtözhetők persze)* egy időpillanatban csak az egyik kiszolgálón érhető el. Nevezzük el tehát a fenti alkotást hibrid konfigurációnak, ezzel mindenki elégedett lehet. Persze néhány apróságot még meg kell oldani. Ezek a problémák nem a fürt meglétéből fakadnak, hanem abból, hogy a funkciókat rázsúfoljuk egy-két vasra. Ekkor is működik minden – csak egy picit oda kell figyelni a tervezéskor.

DNS és globális katalógus

Tiszta Windows 2000 környezetet tervezve nem kétséges, hogy a címtárral integrált DNS zónáknak több előnye is van, legyen tehát a névfeloldás ilyen. A GC funkció viszont bizonyos esetekben ütközik az így konfigurált DNS szolgáltatással. A Q252695 számú cikk részletesen leírja a jelenséget, melynek lényege a következő: ha AD-integrált DNS zónákkal dolgozunk, és a tartományvezérlőnk TCP/IP beállításai a vezérlő DNS-ére mutatnak, akkor újraindításkor a globális katalógus szolgáltatás nem képes rekordokat regisztrálni a DNS Serverbe, mivel az csak később indul el. A leírás már adja is a megoldást: a tartományvezérlőnk abban az esetben nyújthat GC és címtárral egyesített DNS szolgáltatást, ha a TCP/IP beállításai nem a saját DNS-ére mutatnak. Azt ajánlom, hogy a fürtkiszolgálók DNS beállításait a következőképp adjuk meg:

1. DNS kiszolgáló – a fürt másik tagja
2. DNS kiszolgáló – a központban egy DNS szerver
3. DNS kiszolgáló – önmaga

Azért célszerű egy nem fürtbeli DNS kiszolgálót is megadni, mert előfordulhat, hogy a fürt másik tagja nem érhető el. Végső esetben jobb, ha önmagához fordul a vezérlő, minthogy feladja a próbálkozást.



A fürt TCP/IP beállításaiban a helyes DNS kiszolgáló sorrend

Az FSMO szerepek

Kezdünk felépíteni egy szép, szimmetrikus világot, csakhogy a valóság ennél egy kicsit bonyolultabb. A Windows 2000 Serverek címtárszolgáltatásánál létezik öt olyan funkció, amelyet egy erdőben csak egyetlen kiszolgáló nyújthat. Ezeket angolul „Flexible Single Master Operation"-nek, nevezzük amit magyarul „rugalmasan állítható egyedi szerepkör"-nek fordíthatnánk. Ismertnek tekintve a fenti funkciókat, a következőket érdemes tudni, ha fürtözött környezetben dolgozunk:

Az FSMO szerepköröket tetszőlegesen tehetjük bármelyik fürtállomásra, amely egyben tartományvezérlő is – egyetlen kivételszabály van csupán: az a kiszolgáló, amely globális katalógus szerver, nem lehet egyben az „Infrastructure Master" szerepkör tulajdonosa. A szabálynak nincs köze a fürtökhöz, egyébként is érvényes, de lassan formálódó szimmetriánkat tönkreteszi. Ha van olyan fürtállomásunk, amelyik ezt a szerepet viseli, akkor csak a társa lehet GC hordozó, önmaga nem.

Amit még feltétlenül ajánlok az az, hogy nagybetűkkel vessük fel, hova tettük az egyedi FSMO szerepeket. Az FSMO funkcionál nem a működésük a gond, hanem a hiányuk és a helyreállításuk. Fürtök tervezésekor mindenképp ki kell dolgozni egy tervet arra az estre, ha egyik vagy másik állomás véglegesen felmondja a szolgálatot, és ki kell cserélni. A katasztrófa-elhárítási tervben kiemelt helyet kell kapnia az FSMO szerepeknek.

Előnyök és hátrányok

Bevallom, hogy a cikkben felvázolt infrastruktúráról még sehol sem hallottam. Egyes elemei megtalálhatók a tudásbázis cikkeiben, egészében azonban nincs publikált esettanulmány. Mindeddig. Mostantól azonban van. A leírt koncepció működik, méghozzá egy magyar nagyvállalatnál – lényegében háromnegyed éve problémamentesen. Joggal teszi fel a a kérdést a Kedves Olvasó : milyen előnyei és hátrányai vannak egy ilyen szerkezetű hálózatnak?

Nos, előnyként jelentkezik az egyszer megvásárolt, de többszörösen élvezhető redundancia. Kettő, azaz kettő kiszolgálóval a fenti szolgáltatások magas rendelkezésre állásúvá váltak. A legalapvetőbb szolgáltatások is betonbiztosak – csak a beállításokra kell egy kicsit ügyelni. Fontos, hogy a tartományvezérlők mindig kéznél vannak – így nem fordulhat elő, hogy a fürtszolgáltatás nem indul el, vagy a felhasználók hitelesítése nem történik meg. Ne felejtsük: a fürt rendelkezésre állása kisebb vagy egyenlő a tartományvezérlő szolgáltatás rendelkezésre állásánál. Esetünkben ez nem probléma, míg ha a vezérlő külön kiszolgálón üzemel, és elérhetetlenné válik, akkor a felhasználók nem lesznek képesek hozzáférni a szolgáltatásokhoz, mert nincs hitelesítés.

Hátrányként emlegetik tudásbázis cikkek, hogy a tartományvezérlők többleterőforrást igényelnek. Erre az a válaszom, hogy Magyarországon semmiképp. Egy ilyen fürt amúgy is rengeteg tartalékkal rendelkezik – nem tapasztaltam problémát vagy lassulást emiatt. Ha valaki nagyon aggódó alkat, akkor ajánlom neki a „tartományocskák" szakasz elolvasását.

Sokkal inkább érdekes az alkalmazások *(az Exchange, az SQL)* és a tartományvezérlők egy állomáson való elhelyezése, de a Q298570-es cikkben leírtakon kívül nem tudok gondokról.

A cikkek nem sorolnak viszont néhány olyan hátrányt, amelyek mégiscsak léteznek. A kisebbik probléma, hogy olyan szolgáltatások, mint a betárcsázás, nem fürtözhetők, biztonsági okokból pedig nem ajánlott tartományvezérlőkre telepíteni őket – tehát kilógnak a sorból. A nagyobbik probléma, hogy több memóriaéhes alkalmazás azonos kiszolgálóra telepítése *(pl. SQL és Exchange)* méretezési és válaszidő problémákhoz vezethet. Absztrakt szabályt úgy fogalmazok a fentiekből, hogy az infrastruktúra kialakítása a fenti módon megoldja a rendelkezésre állás és a felfelé történő méretezhetőség problémáját, szűkíti azonban a funkcióbővítés lehetőségét. Az előre megtervezett szolgáltatásokon felül nehéz újakkal kiegészíteni a fürtöt. Mindezen korlátok ellenére úgy gondolom, hogy az egyik legstabilabb rendszert sikerült létrehozni.

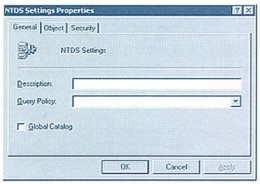
Azt azonban nem szeretném állítani, hogy ez a tökéletes megoldás. Ha valakinek kiegészítenivalója, esetleg kritikai megjegyzése van, szívesen válaszolok a Netacademia tech.net levelezési listáján.

Függelék: „Tartományocskák"

Nem árt eléggé hangsúlyozni: ha egy nagy rendelkezésre állású szolgáltatás egy külső erőforrástól függ, akkor annak is nagy rendelkezésre állásúnak kell lennie, különben a teljes rendszerben Achilles sarkot hagyunk. Ez a helyzet a tartományvezérlőkkel is. Ha valaki aggódik amiatt, hogy egy tartományvezérlő funkció további terheket jelent a fürt számára, de szeretné biztosnak tudni a fürt indulását, később pedig a hitelesítést, annak a „tartományocska" *(angolul: domainlet)* megoldást ajánlom. Ennek lényege a következő: létrehozunk az AD erdőben egy új tartományt, és minden fürtállomást ebbe a tartományba telepítünk tartományvezérlőként. Ha a tartományban nem hozunk létre felhasználókat és egyéb AD objektumokat, akkor jelentősen csökken a címtárszolgáltatás erőforrásigénye. Csak a globális katalógusok okoznak problémát – hiszen azok továbbra is méretesek maradhatnak. Ezen azonban segíthetünk: helyezzük el az alábbi kulcsot minden vezérlő regisztrációs adatbázisában. Nem kell értéket adni – a Windows 2000 csak a kulcs meglétét vagy hiányát ellenőrzi.

```
HKLM\SYSTEM\CurrentControlSet\Control\Lsa\
↳IgnoreGCFailures
```

Ha a kulcsok már léteznek, akkor kikapcsolhatjuk a globális katalógus funkciót az „AD Sites and Services" mmc modul segítségével az alábbi párbeszédpanelen.



Ha nincs pipa, nincs globális katalógus sem

Így minimális lesz a tartományvezérlők erőforrásigénye, és mindig elérhetők lesznek. Több fürtnek elég egyetlen tartományt létrehozni. A megoldásnak ugyanazok a hátrányai, mint egy közönséges újabb tartománynak: többletadminisztráció.

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

Q237366	SMS: Microsoft SMS and Clustered Server Environments
Q174837	Microsoft BackOffice Applications Supported by MSCS
Q266650	BackOffice Program Support on Windows 2000 Datacenter Server
Q252695	DNS Server Generates Event 4011
Q223346	FSMO Placement and Optimization on Windows 2000 Domains
Q281662	Windows 2000 Cluster Nodes as Domain Controllers
Q298570	BUG: Virtual SQL Server 2000 Installations May Fail if Installed to Windows 2000 Domain Controllers

Ki mivel ♥?

Exchange telepítés felsőfokon

Hagyományos problémabemutató rovatunkban ma egy ritkán előforduló, ám tanulságos Exchange balesetet elemzünk, mely rögtön telepítéskor agyonvágja a világ legátgondoltabb rendszertervét is – ha megfelelően gyorsak vagyunk :). Megtudhatjuk azt is, hogyan távolítható el a félkész Exchange 2000 abban az esetben, ha a SETUP sem képes azt eltávolítani.

Az itt elemzett telepítési probléma egyelőre nem szerepel a Microsoft tudásbázisában *(Knowledge Base)*. A tech.net magazin mindig is élen járt a különböző exkluzív marhaságok publikálásában. Szerzőink közvetlenül a tűzvonalból küldik jelentéseiket! Hihetetlen kalandja volt? Írja meg nekünk: cikktipp@netacademia.hu!

Az Exchange 2000 és az Active Directory

Nemrégiben elindult Exchange 2000 sorozatunkban olvasható, hogy az Exchange 2000 – elődeitől eltérően – nem rendelkezik önálló címtárral, hanem az Active Directoryban tárolja beállításait. Ennek megfelelően a kiszolgáló beállításainak változtatása is mintegy 138,7%-ban az AD módosításán keresztül történik. Innen veszik az Exchange szolgáltatásai futási paramétereik új értékeit; ebben a System Attendant megfelelő végrehajtási szálai *(Recipient Update Service, DSAccess, DSProxy, AD2MU)* játszanak szerepet. Összefoglalva:

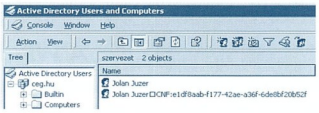
- Ha az Exchange kiszolgáló beállításait módosítom a System Managerrel, valójában az AD egyik névterében, a Configurationban matatok.
- Ha felhasználóknak készítek e-mail címet az Active Directory Users and Computerssel, akkor pedig a DC névtérben dolgozom.

Replikációs ütközések kezelése az Active Directoryban

A Windows 2000 címtára, az AD rengeteg újdonsággal szolgált elődjéhez, az NT4-hez képest *(hierarchikus, DNS alapú stb.)*. Ezek közül problémánk szempontjából a többforrású *(multimaster)* replikáció érdekes: az AD tartományvezérlők bármelyikén módosítható az adatbázistartalom. Ha – netán – két gépen egymásnak ellentmondó beállításokat eszközlünk, a replikációs folyamat kisimítja az apró ráncokat. Ha például egyszerre, egy időben két helyen módosítjuk Józer Jolán leánykori nevének attribútumát, akkor a két változtatás közül az egyik *(a korábbi)* automatikusan elvész. Az automatikus konfliktusfeloldás gondoskodik arról, hogy az AD tartalma konzisztenssé váljon a tartományvezérlőkön.

Ha az ellentmondás nem oldható fel automatikusan – akkor is feloldódik!

Hozzunk létre egyszerre két helyen teljesen egyforma felhasználókat!



Két hajszálpontosan egyforma Józer Jolán nem lehet a címtárban. Még ha egypetéjű ikrek is, valamiben biztosan eltérnek. Ha nem a DNS-ben, hát a GUID-ben ;-)

A konfliktus úgy oldódik fel, hogy az egyiknek a nevét megváltoztatja az AD: mögéír egy GUID-t. Ettől ugyan roppant ronda lesz a megjelenése, de egyedivé is válik! A fenti ábrán két Józer Jolán veszett össze, s az egyikük GUID-vel a hátában végezte.

Szomorú történet, de könnyű elkerülni: ne csináljunk ostobaságot, ne hozzunk létre egyidőben teljesen azonos felhasználókat.

FSMO *(Floating Single Master Operations)* szerepek

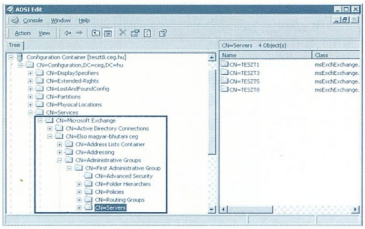
Hogyan lehet garantáltan elkerülni az egymást ütő módosításokat? Mi sem egyszerűbb: le kell tiltani az elosztott módosítás lehetőségét. Az Active Directoryban öt olyan funkció van, mely nem végezhető el tetszőleges tartományvezérlőn, ezeket csak kijelölt masinák hajthatják végre: a FSMO szerepek hordozói *(PDC Emulator, RID Master, Schema Master, Domain Naming Master, Infrastructure Master)*. A Domain Naming Master őrzi például a gyermektartományok nevének egyediségét: valahányszor új gyermektartományt hozunk létre, a DNM ad engedélyt adott nevű gyermek létrehozására. Ha egy kézben *(gépen)* van a döntés, nem fenyeget az elosztott módosításoknál véletlenszerűen bekövetkező ellentmondások felbukkanása. Egyszerű, mint az egyszeregy. Kár, hogy csak ez az öt megkülönböztetett szerep létezik!

Az Active Directoryban öt olyan funkció van, mely nem végezhető el tetszőleges tartományvezérlőn

Automatizált Exchange telepítés

Ha a rendszerterv szerint 5-10 darab Exchange Server telepítésére van szükség, az ember kísértésbe esik, hogy automatikus telepítést használjon. Csak elkészítjük a megfelelő .INI fájlt, és hajrá! Párhuzamosan elindul nyolc telepítés a nyolc gépen.

Mit is csinál az Exchange telepítő? Felépíti az Active Directory Configuration névterében az Exchange Organizationt; sok-sok objektumot hoz létre, többek között – ha nem talál ilyet – az Administrative Groups nevű tárolót, melybe majd a szerverünk bejegyzése kerül. Ezt alakítgatja:



Az Exchange 2000 fészket rak magának az AD Configuration névtérben. Ez a „természetfotó" ADSIEdittel készült.

Ha jó a rendszerterv, akkor:

1. biztosan egynél több tartományvezérlőnk van *(mondjuk három)*
2. az Exchange Serverek nem feltétlenül tartományvezérlőn futnak *(a várható terhelés elosztása miatt)*

A nyolc telepítő mindegyike AD-t keres magának, s véletlenszerűen rácsatlakozik a három tartományvezérlő egyikére, majd nekiáll a fészekrakási ceremóniának. Van már Administrative Groups tároló? Nincs? Csinál magának egyet. Igen ám, de három tartományvezérlő esetén bizony három ilyen konténer születik! Ajjaj! Mi lesz ebből! Hát replikációs konfliktus! Feloldása? A GUID hozzácsapásával! A végeredmény?

- egy rendes, plusz két GUID-vel megspékelt nevű Admin Group
- néhány félresiklott Exchange 2000 telepítés, mert elvették *(átnevezték)* a játékát...

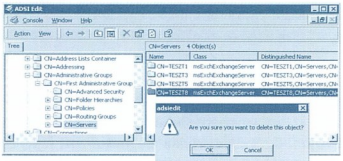
Az azonos Józer Jolánokkal könnyedén elbánunk: csak DEL és vége. Ugyanez a sors vár nyilván a GUID végződésű Admin Groupokra. De mi legyen a *(se)*besült Exchange telepítésekkel? Lássuk sorjában:

1. A GUID végződésű Admin Groupok lekaszálása

A Windows 2000 Support Tools eszközkészlet becses darabja az ADSIEdit MMC modul. Ennek segítségével ugyanolyan könnyedén törölhetünk a Configuration névtérben is, mint ahogy azt a DC Naming Contextben Jolánnal tennénk. Egy nyisszantás, és vége.

2. Az Exchange Server objektum törlése

A megmaradó, hibátlan AdminGroupból az ábra szerint töröljük ki a Servers alól a hibás telepítésű gép bejegyzését.



A hibás, félrement Exchange Server nyomának eltüntetése a Configurationből

3. A beszorult Exchange eltávolítása

Azok a telepítések, amely alól kirántódott az Administrative Groups, még akkor is cigányútra mennek, ha közben a replikáció odahozza nekik az egyetlen, valódi konténert, mivel a telepítés első felét megírják, majd *(átneveződés után)* egy másik, ugyanolyan nevűben folytatódik a művelet. Két lovat kellene megülni, ráadásul vágtatás közben!

Reinstall? Nem megy!
Deinstall? Nem megy!
Ráinstall? Nem megy!

Most akkor dobjuk ki a hardvert? Nem. Dobjuk ki az Exchange Servert! Mintha sohasem lett volna ott!

3/a. A telepítőinformációk eltávolítása

Registry editorral *(regedt32.exe)* töröljük a HKLM/SOFTWARE/MICROSOFT/EXCHANGE kulcsot. Ezzel eltávolítjuk a telepítés leíróinformációit. A SETUP.EXE a következő alkalommal úgy fog futni, mintha még sohasem lett volna Exchange Server ezen a gépen.

De ez még nem elég: a félkész telepítés valószínűleg sikeresen felpakolt néhány szolgáltatást. Ezeket is ki kell venni a rendszer lelkéből.

A Windows 2000 Support Tools eszközkészlet becses darabja az ADSIEdit MMC modul

3/b. A szolgáltatások kitörlése

Állítsuk le az összes MS Exchange szolgáltatást az Administrative Tools->Services eszközzel, majd Registry editorral töröljük a HKLM/SYSTEM/CurrentControlSet/Services alól az összes MSExchange kezdetű szolgáltatás bejegyzését!

Végül törölnünk kell a telepítési könyvtárat, de abba erőteljesen csimpaszkodik az Internet Information Server, ezért mindenekelőtt állítsuk le az IISAdmin szolgáltatást, ami magával húzza a mélybe az SMTP és az NNTP szolgáltatásokat.

3/c. A fájlok törlése

Most már törölhetjük a Program Files\Exchsrvr könyvtárat. *(Már amennyit sikerül belőle. Nem baj, ha egy-két fájl marad, csak az MDBDATA könyvtár tűnjön el!)*

4. Telepítés

Újraindítás után jöhet a SETUP.EXE!

Az eset tanulságai

- A rendszerek összetettsége miatt sajnos a legjobban végiggondolt cselekvéssorozat is nem várt eredményre vezethet. Még az is előfordulhat, hogy mi vagyunk a világon a legelsők, akik belefutunk egy jó kis problémába.
- Az Active Directory címtár nem szent tehén. Ha a szükség úgy hozza, nyugodtan*(?)* mártsuk bele a késünket. *(De előtte számoljunk háromig.)*
- Az AD alapú alkalmazások viszontagságainak túlélése – bármilyen meglepő – az AD alapos ismeretén áll vagy bukik.
- Az AD hibáknak mégsem tejles köre vezethető vissza DNS hibára. Csak az esetek 123,82%-ában igazolódik a DNS hiba.
- Legyen pilotrendszered.

Fóti Marcell
marcellf@netacademia.net

Windows.NET Server Beta 3

Beta 3 állapotba érkezett a Windows kiszolgáló operációs rendszerek következő verziója. A korábbi tech.net számokban még csak Whistler kódnéven emlegetett terméknek már végleges neve is van: Windows.NET Server. Mi mégis a Whistler kódnév eredete után nyomozunk...

A név kötelez

Az új kiszolgáló esetében a dotnet kifejezés immáron nem csupán marketingfogásként jelenik meg a névben. *(Mint korábban az ún. NET Enterprise Serverek némelyike esetében... Nem, nem a Host Integration Server 2000-re gondolok, ugyan... :-)*

Jó egy éve a Microsoft már belül is kemény szabályokhoz köti, hogy mi kaphatja meg a .NET jelzőt, és mi nem – a piciny kis utótagot csak az a szoftver illesztheti a neve végére, amely az Interneten hosztolható, provizionálható *(automatizálható adminisztrációval rendelkezik)*, és egyéb hasonló követelményeknek is eleget tesz. A korábbi „buzzword", az XML támogatás, mint feltétel tehát önmagában már nem elégséges. Ennek fényében nem lett „dotnet" a Windows XP, hiszen még nem tartalmazza a .NET Framework fejlesztői keretrendszert, ami a .NET Server-nek viszont már része lesz.

A Windows.NET Server, mint megszokhattuk, egy kiszolgálócsalád, ám a Windows 2000-hez képest eggyel több tagot tartalmaz. Van természetesen Datacenter változat, mely a teljes kiépítést, a legbővebb szolgáltatást adja. Az ennél egy fokkal kevesebb funkcionalitással bíró változat neve .NET Enterprise Server lesz a korábbi Advanced Server helyett *(visszatérve az NT4-es gyökerekhez)*. Az eddigi legalsó kategóriájú „sima" Standard Server mellett az új tag a Windows.NET Web Server változat. Ne feledkezzünk meg az Embedded, vagyis a „beágyazott", modulonként összerakható változatról sem, amivel úgynevezett .NET Server Appliance-ek készíthetők, pl. fájlkiszolgáló, vagy VPN célszámítógép, fekete doboz távoli felügyelettel.

Whistler és Whistler anyja

Mielőtt elmerülnénk a Beta 3-ban megjelent új funkciók taglalásában, előbb válaszolnunk kell két olyan fontos kérdésre, melyek a tech.net magazin szerkesztőségébe érkezett visszajelzések szerint meglehetősen sokunkat foglalkoztatnak: mi is az a Whistler *(1) (a bennfentesek számára síparadicsom, ahogy egy korábbi címlapborító hirdette)*, és vajon miért ez lett az új Windows kódneve *(2)*?

Szerencsére mindkét kérdésre elsőkézből, abszolút tiszta forrásból kaptunk választ, melyet exkluzívan csak a tech.net magazin Kedves Olvasóival osztunk meg a következőkben.

Whistler egy falu neve. A falu tőszomszédságában található egy hegy is, aminek szintén Whistler a neve. A falu határában mellesleg több hegy is található, mivel a falu egy völgyben épült. Mind a falu határának, mind a további hegyek egyikének jelentős szerepe lesz a továbbiakban.

Whistler Village télen valóban egy síparadicsom, nyáron pedig golfparadicsom, az itt élők foglalkozási összetételére könnyen következtethetünk ezen tényekből.

I Got You Babe...

Whistler angolul sípolót jelent. A helyiek szerint a név onnan ered, hogy nyáron *(tehát a golfszezonban)* a környező hegyeket ellepik a mormoták, akik különleges jellegzetességet kölcsönöznek a tájnak – valami furcsa módon ugyanis sípoló hangokat hallatnak az odúikból, amitől naphosszat visszhangoznak az erdők. *(Arról nem tudni, hogy vajon Whistler lakossága ünnepli-e a Mormota-napot, amikor is a hagyomány szerint az odújából kibúvó, avagy nem kibúvó mormota viselkedése alapján vonnak le messzemenő következtetéseket a közeljövő időjárását illetően. Valószínűsítem, hogy igen...)*

Beta 3 állapotba érkezett a Windows kiszolgáló operációs rendszerek következő verziója. A korábbi tech.net számokban... - no jó, csak vicceltem. *(A szerző itt valószínűleg az „Idétlen időkig" című filmre utal Bill Murray főszereplésével, eredeti címe „Groudhog day" azaz Mormota-nap – a szerk.)*

Whistler falu Kanadában található, Vancouver-től cirka 2 órányi útra kocsival. Történetünk szempontjából talán érdemesebb úgy fogalmazni, hogy a falu Seattle-től északra autóval jó 4 óra alatt megközelíthető. Ez nyomban sejteti a választ második kérdésünkre: a Seattle melletti Redmondban dolgozó Microsoft fejlesztők számára remek hétvégi program kínálkoznak – télen síelni, nyáron golfozni lehet.

Sífelvonó a Whistler hegyre

Brian Valentine

A névadás hiteles és pontos körülményeit maga Brian Valentine mesélte el – ő jelenleg a Windows fejlesztési csoport vezetője. A Kedves Olvasó már hallhatott róla, nemrég például az internetes The Register című informatikai bulvárlapban jelentek meg a Microsoftból kiszivárogtatott, a Linuxról írt belső céges körlevelei...

Történt tehát, hogy az egyik Windows csoport fejlesztési Program Managerének kedve támadt ingyen síbérletet szerezni a Whistler-beli sípályákra. Tüstént kedett, forgolódott, míg munkája eredményeként az új Windows belső kódneve Whistler lett. Ezzel a hírrel nagy büszkén odaállt Whistler falu főpolgármestere elé, és az idegenforgalmat fellendítő hírverésért szerény *(ámde nem sértő)* jutalmul pusztán egy éves felvonóbérletet kért.

A polgármester persze elhajtotta. Végül is kit is érdekel, mi a kódneve egy PC-s szoftvernek? Hol hírverés ez? Hol itt a faluimázs? Ugyan kérem...

Habár a Program Manager pórul járt, a fejlesztők ettől függetlenül megszerették a Whistler nevet, hiszen kedvenc kirándulóhelyüket juttatta eszükbe. A Whistler utáni, következő Windows változatot *(hiszen ennek fejlesztése is már régebben, a Whistler-rel párhuzamosan megkezdődött)* éppen emiatt a falu melletti másik, a Whistler-rel szomszédos, talán még egy kicsit meredekebb sípályáknak otthont adó hegyről nevezték el. Ez a Blackcomb. Érdekesség, hogy a két hegy és a falu együtt egy közös WhistlerBlackcomb marketing branddel, logóval rendelkezik, és saját weboldala is van: **http://www.whistlerblackcomb.com**

Sífelvonó a Blackcomb hegyre

Blackcomb

A Blackcomb újabb mérföldkő lesz a Windowsok történetében. A Windows 2000 és a Whistler közötti váltás tulajdonképpen nem olyan nagy – mondjuk az NT 3.51 és a 4.0 közötti különbséghez hasonlítható – míg ha párhuzamot akarunk vonni, a Whistler-Blackcomb lépés az NT 4.0 és Windows 2000 közötti ugrással mérhető.

A Blackcomb tervezése során ugyanis szép lassan, de módszeresen alapvető architekturális újítások tömege került bele a termékbe. Ennek megjelenése *(leghamarabb)* 2004-2005 környékére várható. Hogy mégse legyen annyira sokkoló a sok újdonság hirtelen megjelenése *(ahogyan az a Windows 2000 esetében volt; a hazai NT4-es társadalom talán épphogy csak mostanában kezd belejönni)*, időközben módosítottak a terveken.

Megközelítőleg egy éve került nyilvánosságra, hogy a Windows .NET/XP és a Blackcomb között mégis lesz egy köztes Windows-változat, amely a Blackcomb terveiből egy kisebb részhalmazt valósít meg. Ennek a fejlesztésnek a kódneve: Longhorn.

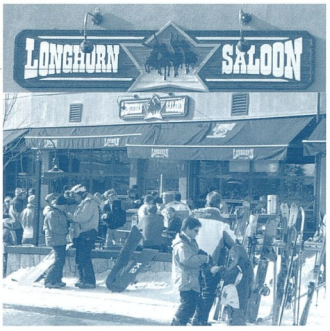
Longhorn szintén egy hegy Kanadában, bár Whistlertől egy kicsit távolabbra esik – állítólag nagyon szép természeti tünemény. A kódnév azonban mégsem innen ered. Korántsem.

Longhorn

Akkor hát honnan? Ismét Brian Valentine-t hívjuk segítségül, remek Whistler-béli helyismeretével:

Whistler Village határához elsétálva *(mely tehát itt nyeri el végső jelentőségét történetünkben)* egy nagy térre jutunk, aminek jobb oldaláról a Whistler hegy felé indul egy sífelvonó, bal oldalán a Blackcomb-gondola kabinjai repítik a sportolókat a magasba. A sífelvonók között, a tér köztes oldalán két lesiklás között megpihenhetünk egy kellemes kis kávézóban, a nyitott teraszon télen is zsibongó forgatagban ücsörögnetünk egy picit – ennek a vendégmarasztaló helynek a neve: Longhorn Saloon. Egyszóval Longhorn tulajdonképpen egy kocsma... Brian Valentine-t szó szerint idézve: „Longhorn is the best place to stop between Whistler and Blackcomb."

A redmondi fejlesztők egyébként fenemód jól érezhetik magukat, ha a fentiekhez még azt is hozzávesszük, hogy a Windows CE operációs rendszer következő, .NET Framework alapú változatának kódneve „Talisker". A Talisker egy kizárólag malátából készített *(ún. single malt)* skót whisky neve, a Skye szigetek vidékéről – az édesebb, mézes, vaníliás ízű, nem pedig a markáns, füstös fajtából...



A Longhorn bár terasza egy napos téli délután

Végezetül: hogy jön mindehhez Whistler anyja? Nos, Whistler anyja egy híres festmény neve – de ez már egy egészen más történet... Most már térjünk rá végre tényleg az új .NET Server funkciókra.

Active Directory – még mindig

Az Active Directory újdonságai eddig a tech.net magazin minden egyes Whistler beszámolójában terítékre kerültek, a Beta 3-ban azonban számos olyan új funkció is megjelent, amiről eddig még nem esett szó.

Alkalmazáspartíció

Egy korábbi számban már pedzegettük például az úgynevezett alkalmazáspartíció *(Application Partition)* fogalmát, mely egy új Naming Context *(NC)* az Active Directoryban. Már Magyarországon is akad jónéhány példa olyan vállalatra, amely saját fejlesztésű alkalmazásának adatait az AD-ban tárolja. Ehhez a Windows 2000-ben mindenképp sémabővítés szükséges, aminek hátránya, hogy a változás az egész AD erdőben érvényes lesz. Számos helyen felmerült azonban az igény arra, hogy az egyedi alkalmazás adatai csak egy korlátozott területen, nem pedig az egész erdőben legyenek elérhetőek *(erre jó példát nyújtanak a multinacionális cégek hazai leányai, ahol ugyan egy AD erdőben van a cég a bécsi, müncheni, párizsi, londoni központtal, ám a kis helyi fejlesztésű alkalmazás adatainak nem kellene az országhatáron túl is látszania)*. Ha ilyen, sémát bővítő vállalati alkalmazást fejlesztünk, akkor az éles és a fejlesztői környezet szétválasztásának igénye is egy újabb ok arra, hogy két erdőből álló Windows 2000 AD struktúra kialakítása mellett döntsünk. Két erdő felügyelete pedig mindig nehezebb lesz, mint egyé.

Ha többtartományos környezetben vagyunk *(hazánkban szerencsére ez meglehetősen ritka, nagyon kevés helyen merülhetnek fel komoly technikai indokok az egytartományos modelltől való eltérésre)*, tovább csökken a rugalmasságunk. Ebben az esetben az előbbi igényeknek éppen a fordítottjával szembesülünk, azaz nemcsak saját tartományunkban, hanem az erdő tetszőleges tartományában is elérhetővé kell tenni az alkalmazásspecifikus adatokat. Ilyenkor az egyetlen megoldás a globális katalógusba *(GC)* való publikálás. Ekkor sem tudunk azonban telephely- *(site)* vagy kiszolgálószintű szabályozást biztosítani az alkalmazásadatoknak: a GC kétállapotú – vagy van egy gépen példány belőle, vagy nincs. A replikát már nem tudjuk finomabban szabályozni, szűrni – nem tudjuk azt mondani, hogy ezen a GC-n csak X alkalmazás adatai érhetők el, míg egy másik GC-n csak az Y-é. Ha egy mezőt beveszünk (propagálunk) a GC-be, akkor az minden GC-ben megjelenik.

Filtered replika

Ha a fenti szűrési feladat ismerős lenne, akkor mondjuk ki, hogy pontosan ezt valósítja meg az ún. „filtered replika", amit például a Novell NDS címtára használ a 8.5 változattól kezdődően. Az ilyen „filtered", azaz szűrt replika használatakor minden egyes objektumról *(vagy akár az objektum minden egyes mezőjéről)* megmondhatom, hogy a sok replika közül melyikben legyen jelen, és melyekre ne replikálódjon.

Nekem személy szerint tetszik az NDS filtered replika megközelítése. Egyetlen probléma van vele – rendkívül nehéz kezelni. Ember legyen a talpán, aki átlátja, hogy a címtárhierarchia sokezer objektuma közül éppen melyiket hova engedte bemásolódni, és hova nem. A replikációs topológia kialakítása sem egyszerű feladat, hiszen lehet, hogy egy objektum számára megfelelő az A-B-C replikációs útvonal, azonban egy másik számára ez már nem jó, mert előírtuk, hogy B gépen őt nem tároljuk *(ott „kiszűrtük")*, így az ő „forgalma" nem haladhat B-n keresztül. Neki egész más útvonal kell, és ez így megy tovább, minden egyes egyedi szűrési beállításra.

Technikailag rugalmas tehát ez a megoldás, azonban az életben gyakran nincs szükségünk ekkora szabadságra, mert nem tudunk mit kezdeni vele: nem tudjuk kézben tartani, átlátni.

Az AD megoldás

A versenytárs termékek felé tett kitérő után térjünk vissza a Windows.Net Server-hez. Hogyan oldja meg az új AD a fenti problémát?

Nos, az alkalmazáspartíciót tetszőleges helyre replikálhatom a teljes erdőn belül – kiszolgálószinten mondhatom meg, hogy az adott gép tárol-e egy példányt az adott partícióból, vagy sem. Az alkalmazáspartíció tehát nemcsak az adott tartományban, hanem az egész erdőben él *(csakúgy, mint a GC)*. Tulajdonképpen egy plusz GC-t kapunk, azaz még egy erdőszinten független replikációs topológiát alakíthatunk ki.

Hogy jó példával járjon elöl, a Microsoft néhány alkalmazás *(szolgáltatás)* adatát máris ebben az új partícióban tárolja – ilyen pl. az AD integrált DNS a Windows.NET Server-ben, illetve akár a RAS, RADIUS *(IAS)*, DHCP adatok is az alkalmazáspartícióban kaphatnak helyet *(majd a végleges kódban kiderül...)*. A hírek szerint az ISA Server következő változata is az alkalmazáspartíciót használja majd – az Exchange oldaláról egyelőre nincs ilyen információ, én legutóbb februárban hallottam azt, hogy az AD csapat leült erről a témáról beszélgetni az Exchange fejlesztőkkel. No, de a Titanium *(az Exchange következő változatának kódneve)* még messze van... *(elég sivár kódnév, különösen az előbbi történet után...)*

Független alkalmazás-címtárszolgáltatás

Mellesleg, ha továbbgondoljuk a folyamatot, elképzelhető, hogy ez az alkalmazáspartíció egy, az AD-tól független önálló alkalmazás-címtárrá növi ki magát, míg az AD megmarad hálózati *(NOS)* címtárnak, együtt alkotva vállalati címtár *(enterprise directory)* megoldást. Erre utal néhány jel: a .NET Serverben pl. az alkalmazáspartícióban bármilyen objektumtípust tárolhatunk *(kivéve biztonsági objektumokat: security principals – felhasználók, csoportok, számítógépek)*. Ha a felhasználó azonosítását az AD végi, az alkalmazások konfigurációinak tárolása ettől függetlenül, egy másik infrastruktúrán történhet – a különválasztott alkalmazáscímtárral így tulajdonképpen elosztott regisztry*(szerű)* szolgáltatást kapnánk. Ez szerintem egész jól jönne pl. a .NET Framework-re írt alkalmazásoknak: egy dotnet alkalmazás ugyanis a saját konfigurációját a saját könyvtárában, szövegfájlokban tárolja – ez hasznos a mobileszközök és egyéb más platformok közötti hordozhatóságot tekintve, azonban nehézkessé teszi a konfigurációk központi felügyeletét.

A függetlenedés talán még ott is látszik, hogy a tervek szerint a Domain Controller funkció *(azaz maga az Active Directory)* bármelyik kiszolgálón, dcpromo használata, vagyis telepítés nélkül szolgáltatásként, szervízként lenne futtatható *(és leállítható!)*. Ezzel majd talán csak a „kocsmában" *(a Longhorn változatban)* találkozhatunk, a .NET Serverben egyelőre még nem.

Mindez persze csak találgatás. A jelenlegi fejlesztések kísérleti továbbgondolása – a .NET Server funkcionalitása – egyelőre még nem végleges a Beta 3 változatban sem.

Az biztosan látszik, hogy a jelenlegi vállalati rendszerekben a különböző alkalmazások címtárainak egységesítése egyre inkább égető feladat – egy ilyen címtárstratégia része lehet a Metadirectory szolgáltatás is. *(Erről bővebben a MMS cikk első részében olvashatunk jelen számunkban)*

Hogy konkrétan mi lesz a .NET után, a Longhorn és a Blackcomb változatban, arról még a redmondi fejlesztőknek is csak vázlatos terveik vannak, 2005-ig még akár gyökeresen is megváltozhatnak. Ezekbe a tervekbe éppen ezért általában csak szigorú titoktartási nyilatkozatok *(NDA – Non-Disclosure Agreement)* aláírása után engednek külső szemlélődőt betekinteni – a tech.net Ojság Kedves Olvasóinak így csak a folyosói pletyka és a találgatások maradnak...

Telephelyek közötti replikáció

Érdekes új funkció, hogy ha akarjuk, kikapcsolhatjuk az AD telephelyek közötti replikáció tömörítését. Amerikában, úgy tűnik, bőven van sávszélesség, így ha valaki inkább a vonalat akarja terhelni, a CPU-kon egy kicsit könnyíthet azzal, hogy nem zaklatja őket a be- és kitömörítéssel. Azt hiszem, mi még nem járunk itt...

Mellesleg érdekes tudni, hogy a Windows 2000 sem tömöríti minden esetben a telephelyek közötti *(intersite)* replikációs forgalmat. Ha 10 Kbyte alatt van a replikálandó adat mennyisége, akkor nem történik tömörítés. Hogyan? A Kedves Olvasó 32k-ra emlékszik? Andreas Luther replikációs tanulmánya pedig 50k-t ír? A Windows 2000 Akadémián ráadásul 10 változást említettek küszöbértékként. No, és mit mond a Resource Kit?

Nos, senki nem találná ki: valójában nincs ilyen bedrótozott korlát, azaz mégis van: 256 byte *(nem kilobyte!)*. E felett a Windows 2000-ben az AD replikációs motor mindig *(!)* betömöríti az intersite replikálandó adatokat, majd összehasonlítja a tömörítetlen és a tömörített adatot. Ha az utóbbi kisebb, akkor tömörítve küldi, ha nem, akkor tömörítetlenül. Az átbillenési határ néhány 10 KByte-nál van *(nyílván változó, hogy éppen hol)*. A CPU-t tehát minden esetben terheli a tömörítés.

„Ez tényleg így működik?" – hallom is már az Olvasó hitetlenkedő kérdését. Azon a belső Microsoft fejlesztői levelezési listán is ezt kérdezték, ahol mindezt olvastam. „Ez tényleg így működik" – jött a válasz az egyik tagtól. „Itt van előttem a kód, onnan olvasom..."

A tömörítés 32 Kbyte-os blokkokkal dolgozik, ez adhatta az egyik népszerű határ alapját – természetesen a blokkméretnek nincs köze ahhoz, hogy mikor lesz kisebb a tömörített adat, hiszen a blokkot nem kell teljesen kitölteni a hívásnál.

Andreas Luther és csapata pusztán empirikus úton, tesztekkel, mérésekkel elemezték a replikációs forgalmat, így jutottak a másik, széles körben elterjedt 50 KByte-os határ létezéséhez. *(Intő jel ez kérem – mások például a SID-ek használatát vizsgálják hasonló empirikus módon, pusztán mérésekkel. Érdekes lenne tudni a kód ismeretében, hogyan is használja az NT/2000 pontosan a SID-eket... De ez már egy másik téma)*. Ettől függetlenül Andreas Luther replikációs tanulmánya persze messze az egyik legjobb írás az AD témakörében **[1]** – ez a tévedés végül is teljes mértékben akadémikus, bár kétségkívül érdekes.

A .NET Server telephelyeken belüli replikációs tömörítési algoritmusa egyébként ugyanolyan hatékonyság mellett akár 2-7-szer gyorsabban működik, mint a Windows 2000-es a korai tesztek szerint *(empirikusan, ugyebár... :)*

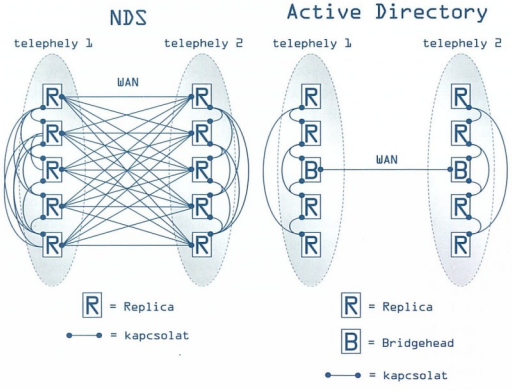
A telephelyek közötti replikációs topológia kialakításáért felelős Inter-Site Topology Generator *(ISTG)* algoritmusában is újdonságokat hoz a .NET Server, azonban ez is azon *(egyébként nem túl számos)* funkciók egyike, amik csak a .NET Server natív üzemmódjában érhetők majd el.

Globális Katalógus

Ha már szóba került a globális katalógus – Windows 2000 esetén, ha egy újabb mezőt, objektumot propagálunk a GC-be *(azaz módosítjuk a sémát)*, az teljes GC replikációt indít el, tehát nemcsak a változások, hanem az egész GC tartalma ismét elkezd vándorolni a hálózaton. A .NET Server esetén a GC attribútumok bővítésekor is megőrzi replikációs állapotát a katalógus, tehát ekkor is csak a változások mennek át.

A telephelyek közötti replikációs forgalom csökkentése – érthető okokból – úgy tűnik, egy kulcsterület. A Kedves Olvasóban persze ilyenkor felütheti fejét a kétkedés ördöge: nem gyanús, hogy amint az újabb termék megjelenése közeledik, úgy lesz a korábban optimálisnak hirdetett Windows 2000 AD replikációnak egyre több apró-cseprő hibája, amit természetesen a .NET Server majd megold?

Nos, a helyzet egyáltalán nem ilyen vészes – ehhez ismét nézzünk körül a versenytársaknál. A Novell NDS-ben például nem létezik a telephely fogalma, azaz nem lehet a replikációt LAN-WAN sávszélesség szerint bontani. Bridgehead szerverek sincsenek, amik proxy-ként, gyűjtőként szolgálnának a WAN vonal jó kihasználásához. Mindenki mindenkivel replikál, az alábbi ábra szerint:



Az Active Directory replikációs topológiája könnyedén igazítható a vállalati vonalak sávszélességéhez

Az NDS-ben éppen emiatt az egyik fő tervezési szempont az, hogy partíció lehetőleg ne nyúljon át telephelyhatáron. *(Vessünk még egyszer egy pillantást a fenti ábrára a következmények megértéséhez! Remélem ez egy amolyan „aha" pillanat lesz (ahogy azt az amerikai mondja, mikor a fejére csap) azon Kedves Olvasók számára, akik cégénél az NDS replikációs forgalma csak úgy zabálja a vidéki vonalakat...)*

Az Active Directory fejlesztésekor ezzel szemben a fő cél az volt, hogy a logikai struktúra *(a partíciók, azaz a tartományok)* tervezése teljesen független legyen a fizikai topológiától *(attól, hogy milyen vonalak és transzportok állnak rendelkezésre a logikai struktúra alatt)*. Ugye emlékszünk még az alábbi ábrára a Windows 2000 TechNet előadásokról? *(A mű egyébként Bátorfi Zsolt kollégám alkotása, szerintem nagyon erős szemléltető hatású rajz...)*

Egyetlenegy olyan eset van, amire nem igaz ez a teljes függetlenség: ha SMTP, azaz e-mail alapú *(S-MIME segítségével titkosított)* transzportot akarunk használni RPC helyett az AD replikációra, akkor ezt csak tartományok között tehetjük meg, tartományon belül *(intra-domain)* nem.

Ennek pediglen történeti okai vannak. Az USA védelmi hivatala *(DOD – Department Of Defense)* ugyanis már régóta Microsoft Exchange alapú levelezést használ *(kb. 4-500 ezer felhasználó)*, pontosabban az Exchange egy speciálisan a DOD részére készített, módosított változatát az ún. DMS-t *(Departmental Messaging System)*. A DOD-ben alapvetően csak az SMTP portok voltak nyitva szerte a belső tűzfalakon – ennek fényében még 1998-ban, a Windows 2000 *(akkor még NT5)* tesztelésében segítő ügyfeleknek tartott JDP *(Joint Development Program)* konferencián megrökönyödve hallották, hogy a tervek szerint RPC lesz a transzport az AD replikációhoz. Egy DOD méretű *(és biztonsági szintű)* szervezetben egy újabb port kinyitásának évekig tartó tesztelési és engedélyeztetési fázisokon kell átmennie! Ennek reális esélyeit látva a Microsoft inkább belefejlesztette a termékbe az SMTP transzportot *(400 ezer ügyfél, az mégiscsak 400 ezer ügyfél...)*.

Mivel a meglévő Exchange struktúra Organization-ök szerint volt NT4 tartományokra bontva, ez adta a migrációs utat is. Emiatt a DOD-ben csak a tartományok közötti replikációra kellett SMTP, mert ez ment át belső osztályok közötti tűzfalakon.

Persze kérdezhetjük, hogy ettől függetlenül miért nem került bele a tartományon belüli *(intra-domain)* replikáció SMTP támogatása a végleges Windows 2000-be, hiszen ha a DOD-nak nem is, de másnak esetleg lehetett volna szüksége rá?

Nos, a választ talán több MCSE2000 is tudja: tartományon belül az AD replikációt nem csak az AD végzi, hiszen a csoportházirend objektumok *(GPO, Group Policy Object)* egyrészt az AD-ban, másrészt a SYSVOL megosztáson, a fájlrendszerben tárolódnak. A SYSVOL replikálását pedig a File Replication Service *(FRS)* végzi, amely csak RPC transzporton megy. Ahhoz, hogy tartományon belül is lehessen SMTP-vel replikálni az AD-t, az FRS-t kellett volna megírni SMTP használatára – ennek fejlesztésére, tesztelésére sem idő, sem igény nem volt már.

Jó kérdés, hogy vajon a Windows.NET Serverben az FRS tud majd SMTP-vel replikálni? Kétlem. Talán már a DOD-nak sincs szüksége az

SMTP-re, most valószínűleg IPSec-et használnak. Komoly technikai indok nem igazán merülhet fel az SMTP mellett, így okafogyottá válik ez a probléma. A történetet azért érdemes megjegyezni.

Újabb kitérő a Utah állambeli Provo városába

Visszakanyarodva az eredeti témához: az AD-ban tehát alapvetően elkülönül a logikai és a fizikai struktúra tervezése. Az NDS 8.5 -től kezdve a fenti nx*(n-1)* kapcsolat replikációs forgalmát úgy lehet csökkenteni, hogy nem egyszerre replikál mindenki, hanem szép sorban egymás után *(az átvitt adatmennyiséget ez nem csökkenti, de a lökésszerű terhelést időben szétkeni)*. Bridgehead nélkül persze nem lehet jó megoldást találni – mert ahhoz telephelyeket kellene tudni létrehozni – de ezt sem lehet. A Novell nem ebbe az irányba indult el, hanem bevezette a már korábban említett filtered replikát, amivel ténylegesen szűrhető, csökkenthető az átvitt adatmennyiség.

A globális katalógus Windows 2000-es replikációs forgalmából indultunk el – érdemes tudni, hogy az NDS-ben egyáltalán nincs a GC-hez hasonló partíció. Van egy különálló katalógusszolgáltatás, aminek számos hátránya van: ez nem egy partíció, azaz nem replikálódik, és főleg nem mezőszinten *(ahogy egyébként a többi NDS replika működik)*. A katalógust az NDS Catalog Service Manager ún. Dredger komponense pull módszerrel gyűjti össze. Mivel a változásokról ő nem kap értesítést, ezt a gyűjtést periodikusan meg kell ismételni, azaz folyamatosan nulláról fel kell építeni a katalógust *(emlékszünk: a replikációs forgalom csökkentéséből indultunk ki)*. Egy másik nagy hiányosság, hogy a katalógusba összegyűjtött objektumok *(kivonatok)* elvesztik eredeti jogosultsági listáikat – a hozzáférési jog csak a teljes katalógus szintjén szabályozható, a katalógusban lévő objektumok szintjén nem. Ennek jelentős biztonsági vonzatai vannak.

A Novell itt is más irányba indult el – a katalógusszolgáltatás továbbfejlesztése helyett ezt a feladatot is a filtered replika oldja meg. Hiszen ha minden egyes objektumról és mezőről egyénileg szabályozhatom, hogy merre replikálódik, hol van jelen, és hol nincs, akkor nem kell katalógus; ez kiváltja a funkcióját *(a kezeléséről mondottak azonban továbbra is igazak...)*.

A filtered replika koncepciója tehát alapvetően jó dolog – ha egy jó felügyeleti modellt lehetne kialakítani hozzá, bizonyos funkciókra még az AD-ban is el tudnék képzelni ilyen működést. Majd meglátjuk.

Korábban hallhattunk arról, hogy a .NET Server-ben esetleg egy Domain Controller több különböző tartomány vezérlője is lehet *(több partíciót, domain adatbázist is tárolhat)*, ám a Beta 3-ban úgy néz ki, még mindig nincs meg ez a funkció, így majd talán a Longhorn vagy a Blackcomb tudását gazdagítja. Az NDS-ben egy fizikai kiszolgáló több NDS replikát is tárolhat. Csak a tényszerűség kedvéért.

Az AD-ban elkülönül a logikai és a fizikai struktúra tervezése

Sémabővítés

Fóti Marcell tollából már korábban olvashattuk, hogy a natív Windows.NET AD erdő, az úgynevezett 1-es funkcionalitási szint használatához sémát kell bővítenünk. Ezt az Exchange2000-hez hasonló módon egy adprep.exe nevű kis eszközzel végezhetjük el. A /forestprep kapcsolóval a Schema Master FSMO szerepkörű tartományvezérlőn, a /domainprep kapcsolóval pedig az Infrastructure Master FSMO szerepkörű DC-n kell lefuttatni *(utóbbi a csoportobjektum egyetlen mezőben, listában tárolt csoporttagságainak replikációját javító újítás miatt szükséges) (erről még a legelső Whistler előzetesben számoltunk be...)*

Május 15 – Windows.NET Server TechNet előadás

Beszámolónkat a következő számban folytatjuk, ahol is megtudhatjuk, mit jelent a Release Candidate kifejezés, további újdonságokat olvashatunk az AD *(pl. küzdelem a zombi objektumok ellen)*, a biztonság, az NLBS és a Windows Update területeiről. A kíváncsiaknak addig is ajánlom a május 15-i, Lurdy házas, ingyenes TechNet szemináriumot a témában – jelentkezés a szokásos helyen **[2].**

Horváth Tamás
MCSE2000
Microsoft Magyarország

A fent közöltek a szerző saját véleményét tükrözik, és semmilyen módon nem tekinthetők a Microsoft Corporation hivatalos álláspontjának.

A cikkben szereplő URL-ek:

[1] http://www.microsoft.com/TechNet/prodtechnol/windows2000serv/deploy/ntopt11.asp
[2] http://www.microsoft.com/hun/events/

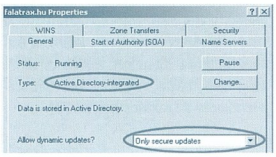
A Windows 2000 és a dinamikus DNS

(II. rész)

Az előző részben bemutattuk a dinamikus DNS regisztráció működését. A regisztrációt abban a formájában tulajdonképpen bárki elvégezheti, aki hálózati kapcsolatán keresztül képes felvenni a kapcsolatot a DNS kiszolgálónkkal. Szerencsére – Windows 2000 tartományon belül – a zónákon az adatok módosítását előzetes bejelentkezéshez köthetjük: ez a biztonságos zónafrissítés, azaz az „Only secure updates".

A biztonságos zónafrissítés

Ha a DNS kiszolgálónk egy Windows 2000 tartományvezérlőn fut, és a zónánk adatait a címtárban tároljuk *(azaz a zóna „Active Directory-integrált")*, a zóna önmaga, és minden egyes bejegyzés is rendelkezik egy biztonsági leíró táblázattal, amiben meghatározhatjuk, hogy a zónával, vagy bejegyzésekkel ki és mit jogosult művelni.



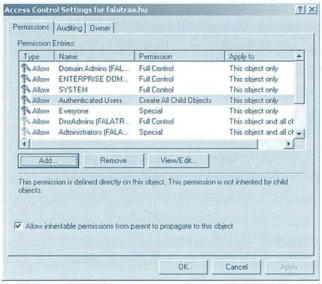
A biztonságos zónafrissítést csak a címtárba integrált zónákon engedélyezhetjük

Alapértelmezett zónajogosultságok

Korábban már bemutattuk, hogy a címtárintegrált zónák bejegyzései valójában az Active Directory dnsZone és dnsNode objektumai. A zónák és rekordok így kapnak biztonsági hozzáférési listát *(Access Control List – ACL)*: amikor a DNS konzolban ezeket szerkesztjük, valójában a címtárobjektumok hozzáférési jellemzőit módosítjuk. *(A jogosultságlistát egyébként nyugodtan módosíthatnánk közvetlenül a címtárobjektumokon is, de a DNS konzolból azért talán kicsit kényelmesebb).*

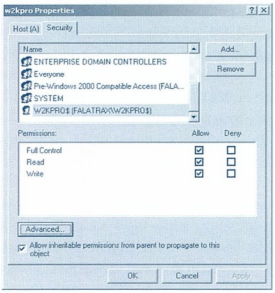
A címtárban a rekordok *(a dnsNode objektumok)* a zónaobjektum *(dnsZone)* „gyermekei". Így a rekordok létrehozását a szülőobjektumon *(a zónán)* definiált jogosultságok korlátozzák; a létrejött bejegyzések pedig ugyaninnen öröklik a jogosultságlistájukat. A zónaobjektumon alapértelmezésben beállított érdekesebb jogosultságok a következők:

- SYSTEM, Enterprise Domain Controllers, Domain Admins, Enterprise Admins, DNSAdmins: Full Control – azaz teljes hozzáférés,
- Authenticated Users: Create All Child Objects – gyermekobjektumok (rekordok) létrehozása,
- Everyone: List Contents, Read All Properties, Read Permissions – tartalom listázása, gyermekobjektumok jellemzőinek és jogosultságlistáinak olvasása.



A zónában bárki létrehozhat új bejegyzést

A tartományvezérlők, a rendszer, illetve a rendszergazdák teljes hozzáférése, illetve a mindenki számára biztosított olvasási jog nem igényel különösebb magyarázatot. Az „Authenticated Users" csoport magába foglal mindenkit, aki képes volt bejelentkezni a tartományba *(ideértve a felhasználókat és a számítógépeket is: minden számítógép rendelkezik egy „felhasználói" fiókkal a tartományban, melynek neve: <gépnév>$)*. A csoportnak definiált (gyermekobjektumok létrehozása) jog tehát azt jelenti, hogy bármely felhasználó, illetve számítógép létrehozhat bejegyzést a zónában, amennyiben sikerült bejelentkeznie. A regisztrációk zömét persze mindig a számítógépek végzik *(emlékezzünk, a Windows 2000 már induláskor megpróbálkozik a regisztrációval)*.



A számítógép az általa bejegyzett rekordon teljes jogú hozzáférést kap

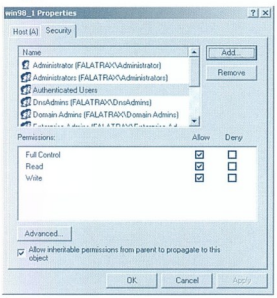
Bár a regisztráció során létrejött rekordobjektum tulajdonosa a SYSTEM lesz, a regisztráló felhasználó *(számítógép)* Full Control jogot kap a saját bejegyzésére. Így azt a későbbiekben bármikor módosíthatja, vagy akár törölheti is, mások pedig legfeljebb az örökölt olvasási jogokat gyakorolhatják a rekordon.

DNSUpdateProxy

De mi van azokkal a rekordokkal, amelyeket a kliens helyett a DHCP kiszolgáló jegyez be? *(Láttuk, van ilyen: a Windows 2000 DHCP Service szívesen bejegyzi a DNS-be azokat a címeket, amelyeket az önálló címregisztrációra nem képes ügyfeleknek – Win95/98/Me/NT – osztott ki)*. Az előbb kiderült, hogy a bejegyzett rekordhoz teljes hozzáférést – a tartományvezérlőkön és rendszergazdákon kívül – csak a bejegyzés létrehozója kap. Mi történik, ha a DHCP kiszolgáló néhány rekord bejegyzése után meghibásodik, és a tartalék DHCP kiszolgálónk veszi át a helyét? Ha a tartalék szerver nem tartományvezérlőn fut, akkor nem lesz joga módosítani az előd által létrehozott rekordokat!

Erre találták ki a DNSUpdateProxy csoportot, amelynek leírásában – kissé félrevezetően – az szerepel, hogy: „DNS clients who are permitted to perform dynamic updates on behalf of some other clients *(such as DHCP servers)*." *(Azaz: „DNS ügyfelek, amelyek más nevében is regisztrálhatnak (például DHCP kiszolgálók)")*.

Valójában a mások által beregisztrált rekordokat a DNSUpdateProxy csoport tagjai sem módosíthatják, hiszen – láttuk – a rekordok jogosultságlistáiban ennek a csoportnak nyoma sincs. A csoport működésének valódi elve: a DNSUpdateProxy csoport tagjai által beregisztrált rekordokat később bárki módosíthatja! Ha tehát a DHCP kiszolgálónkat felvesszük ebbe a csoportba, később az általa regisztrált rekordokat bárki frissítheti, módosíthatja, törölheti.



Egy DNSUpdateProxy-tag által bejegyzett rekord: aki kapja, marja!

A DNSUpdateProxy csoport tagjai által regisztrált rekordok tehát egy plusz jogosultságot kapnak, ez pedig az Authenticated Users – Full Control. Elgondolkodhatnánk azon, hogy miért van szükség ilyen szintű engedékenységre: nem lenne-e elég, ha csak a DNSUpdateProxy csoport tagjai kapnák meg ezt a jogot? Nos, ez megoldaná a DHCP kiszolgáló kiesésének esetét, amennyiben a tartalék DHCP kiszolgáló is tagja a csoportnak. Nem oldaná meg viszont a másik fontos okot, ami miatt a DNSUpdateProxy csoportot kitalálták: ha ugyanis egy Windows NT 4 munkaállomást *(ami nem képes önálló regisztrációra, így helyette a DHCP kiszolgáló regisztrál)* frissítünk Windows 2000-re, a frissítés után az ügyfél már önmaga próbálkozna a bejegyzéssel *(ő azonban nem tagja a DNSUpdateProxy csoportnak)*. Ezért van szükség az Authenticated Users jogosultságaira.

A dolog közben egy fontos veszélyt is magában rejt: soha ne vegyünk fel olyan kiszolgálót a DNSUpdateProxy csoportba, ami egyben tartományvezérlő is! Ekkor ugyanis a tartományvezérlő által regisztrált összes tartományi erőforrás-rekord védtelenné válik (hacsak kézzel nem módosítjuk azok jogosultságait). A tartományvezérlőn futó DHCP kiszolgáló egyébként mindenképpen hozzáfér az erőforrás-rekordokhoz, hiszen az Enterprise Domain Controllers csoport jogosultsága megtalálható mindegyik bejegyzésen. Ez persze nem segít a fenti *(NT4->W2K)* eseten; ilyenkor az egyik megoldás az, hogy kézzel töröljük a DNS-ből a DHCP által regisztrált „régi" rekordot. A másik megoldás kézenfekvő: a DHCP kiszolgálót ne telepítsük tartományvezérlőre.

> ***...a DNSUpdateProxy csoport tagjai által beregisztrált rekordokat később bárki módosíthatja***

A biztonságos DNS frissítés szabványai

A Windows 2000 dinamikus DNS szolgáltatása nem az RFC 2137 *(Secure Domain Name System Dynamic Update)*, és ugyancsak nem az RFC 2535 *(Domain Name System Security Extensions)* alapján működik. Ehelyett egy olyan módszert használ, ami képes összekötni a Windows 2000 tartományban amúgy is rendelkezésre álló Kerberos felhasználóazonosítást és a dinamikus DNS frissítést: ez pedig az úgynevezett GSS-API *(Generic Security Service Application Program Interface, RFC 2078* **[1]***)*. A GSS-API tulajdonképpen egy olyan keretrendszer, ami lehetővé teszi tetszőleges felhasználóazonosítási megoldás *(esetünkben a Kerberos)* tetszőleges hálózati szolgáltatással való összekapcsolását *(esetünkben a DNS-sel)*. Ehhez mindössze két új DNS regisztrációs erőforrásrekord-típust kellett definiálni, ezek a TKEY, illetve a TSIG erőforrásrekordok *(RR's – Resource Records)*.

A felhasználóazonosítást mind az ügyfél,- mind a kliensoldalon a GSS-API, azon belül is a Windows 2000 Kerberos biztonsági alrendszere végzi. A sikeres felhasználóazonosítás után az ügyfél a kapott Kerberos jegy segítségével előállít egy speciális *(TKEY)* rekordot tartalmazó „bejelentkező" DNS csomagot. A TKEY lekérdezésre a kiszolgálótól TKEY válaszrekord érkezik; ebben a pillanatban az ügyfél azonosította magát a kiszolgáló felé és viszont; a TKEY rekordban kapott adatok segítségével pedig a további kommunikáció *(a valós címregisztráció)* minden egyes csomagja digitálisan aláírható. Ez a digitális aláírás utazik a TSIG *(Secret Key Transaction Signature)* rekordokban.

A Windows 2000 biztonságos DNS frissítése

Akárcsak az előző alkalommal, a Network Monitor illetve a jelen számunkban bemutatott ingyenes Ethereal program segítségével megtekinthető elkapott hálózati forgalmak letölthetők a **[2]** címről. A probootdyn_e_f.cap fájl egy tartományi tag *(Windows 2000 Professional)* indulásakor generált hálózati forgalom egy részét mutatja. *(A # jellel jelölt számok a csomagok sorszámát jelölik)*:

#1: a kliens lekérdezi a szülőtartomány SOA rekordját, hogy abból kiolvashassa a tartomány elsődleges zónáját kezelő DNS kiszolgáló címét *(a dinamikus frissítéseket ugyanis – mint tudjuk – ennek kell küldeni)*

#2: megérkezik a kiszolgáló válasza, benne a DNS kiszolgáló IP címével

#3 és #4: a címregisztrációhoz szükséges prerekvizítumok lekérdezése, és a kapott válasz ***(részletesebb magyarázatukat lásd az előző számban)***

#7: a kliens elküldi a prerekvizítumok alapján előállított regisztrációs kérelmet a kiszolgálónak, benne a saját címével.

... eddig a módszer megegyezik a hagyományos frissítéssel. Ekkor jön viszont – derült égből – az „Operation Refused" *(#8)*:

```
-DNS: 0x4:Dyn Upd Resp. PRE/UPD records to w2kpro.falatrax.hu. of type Canonical name
  DNS: Query Identifier = 4 (0x4)
+DNS: DNS Flags = Response, OpCode - Dyn Upd, RCode - Operation refused
  DNS: Zone Count = 1 (0x1)
  DNS: Prerequisite Section Entry Count = 2 (0x2)
  DNS: Update Section Entry Count = 1 (0x1)
  DNS: Additional Records Count = 0 (0x0)
+DNS: Update Zone: falatrax.hu. of type SOA on class INET addr.
+DNS: Prerequisite: w2kpro.falatrax.hu. of type Canonical name on class Unknown Clas
+DNS: Update: w2kpro.falatrax.hu. of type Host Addr on class INET addr.
```

A kiszolgáló elutasítja a névtelen regisztrációs kérelmeket

A Windows 2000 ügyfél ekkor megpróbál bejelentkezni. A #9-es csomag egy Kerberos jegy-kérés *(KRB_TGS_REQ)* a server.falatrax.hu kiszolgáló DNS szolgáltatásához:

```
+IP: ID = 0x94; Proto = UDP; Len: 1310
+UDP: Src Port: Unknown, (1060); Dst Port: Unknown (88); Length = 1290 (0x50A)
-Kerberos: KRB_TGS_REQ
 +Kerberos: Message Type = 108 (0x6C)
 +Kerberos: Kerberos Protocol Version = 5 (0x5)
 +Kerberos: Kerberos Message Type = KRB_TGS_REQ
 +Kerberos: Pre-Authentication Data
 -Kerberos: KDC Request Body
  +Kerberos: KERB_KDC_REQ Type = KDC-Req-Body (req-body[4])
  +Kerberos: Ticket Flags
  +Kerberos: Realm Name = FALATRAX.HU
  +Kerberos: Server Name = DNS/server.falatrax.hu
  +Kerberos: Expiration Date = 09/13/2037 02:48:05 UTC Time Zone
  +Kerberos: Random Number = 2100107035 (0x7D2D171B)
  +Kerberos: Supported Encryption Types
```

Kerberos jegykérés (bejelentkezés)

A Kedves Olvasó sajnos valószínűleg ebben a formában nem fogja látni a Network Monitor által elfogott csomagot, ugyanis a Kerberos kommunikáció értelmezéséhez szükséges bővítmény nyilvánosan nem hozzáférhető. Arról azért meg lehet ismerni a Kerberos kommunikációt, hogy a *(Kerberos kulcselosztási szolgáltatást – KDC – futtató tartományvezérlő)* kiszolgáló UDP 88-as portjára csatlakozik, illetve a válasz is onnan érkezik. *(Az Ethereal képes a Kerberos csomagok visszafejtésére is)*. A kérésben azt láthatjuk, hogy a munkaállomásunk jegyet *(hozzáférést)* kér a server.falatrax.hu kiszolgáló DNS szolgáltatásához. A #9-es válaszban már érkezik is a Kerberos jegy. Az ügyfél ezt a jegyet eltárolja, és a segítésével előkészíti a TKEY rekordot. Mivel a TKEY rekorddal kibővített DDNS kérés mérete már meghaladja az egyetlen UDP csomagban átvihető adatmennyiséget, ezért a kliens TCP csatornát *(!)* épít a kiszolgáló 53-as portjára *(#11-#13)*. A felépült csatornán pedig elküldi a GSS-API bejelentkezéshez szükséges speciális DNS csomagot, benne a TKEY rekorddal *(#14-#15)*:

```
+DNS: 0x70AE:Std Qry for 910533066770-3. of type Scrt Key Estbl on class INET addr.
  DNS: TCP Length = 2547 (0x9F3)
  DNS: Query Identifier = 28846 (0x70AE)
+DNS: DNS Flags = Query, OpCode - Std Qry, RCode - No error
  DNS: Question Entry Count = 1 (0x1)
  DNS: Answer Entry Count = 1 (0x1)
  DNS: Name Server Count = 0 (0x0)
  DNS: Additional Records Count = 0 (0x0)
-DNS: Question Section: 910533066770-3. of type Scrt Key Estbl on class INET addr.
  DNS: Question Name: 910533066770-3.
  DNS: Question Type = Secret Key Establishment
  DNS: Question Class = Internet address class
-DNS: Answer section: 910533066770-3. of type Scrt Key Estbl on class Req. for any
  DNS: Resource Name: 910533066770-3.
  DNS: Resource Type = Secret Key Establishment
  DNS: Resource Class = Request for any class
  DNS: Time To Live = 0 (0x0)
  DNS: Resource Data Length = 2489 (0x9B9)
  DNS: Additional Resource Data = 03 67 73 73 73 09 6D 69 63 72 6F 73 6F 66 74 03 63 6F 6D
```

A DNS bejelentkezés, benne a TKEY rekorddal

A kiszolgáló válasza a #17-es csomagban érkezik: ha nem történt hiba, a válasz már digitális aláírással *(azaz egy plussz TSIG rekorddal)* kibővítve érkezik. Innentől kezdve az ügyfél és a kiszolgáló azonosította egymást, és a bejelentkezést követően *(hála a digitális aláírásnak)* más már nem avatkozhat bele a kommunikációba. A bejelentkezés után a TCP csatornára már nincs szükség, azt be is zárjuk *(#18-#21)*. Más már nincs is hátra, mint beregisztrálni a címrekordot: a TSIG rekorddal felturbózott regisztrációs kérést a #22-es csomagban látjuk:

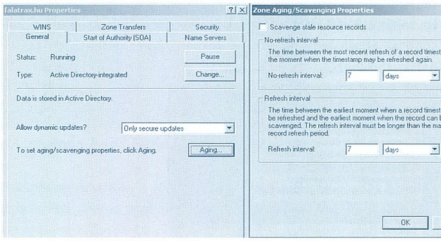
```
+DNS: 0x77FD:Dyn Upd PRE/UPD records to w2kpro.falatrax.hu. of type Canonical name
  DNS: Query Identifier = 30717 (0x77FD)
+DNS: DNS Flags = Query, OpCode - Dyn Upd, RCode - No error
  DNS: Zone Count = 1 (0x1)
  DNS: Prerequisite Section Entry Count = 2 (0x2)
  DNS: Update Section Entry Count = 1 (0x1)
  DNS: Additional Records Count = 1 (0x1)
+DNS: Update Zone: falatrax.hu. of type SOA on class INET addr.
+DNS: Prerequisite: w2kpro.falatrax.hu. of type Canonical name on class Unknown C
-DNS: Update: w2kpro.falatrax.hu. of type Host Addr on class INET addr.
  DNS: Resource Name: w2kpro.falatrax.hu.
  DNS: Resource Type = Host Address
  DNS: Resource Class = Internet address class
  DNS: Time To Live = 1200 (0x4B0)
  DNS: Resource Data Length = 4 (0x4)
  DNS: IP address = 192.168.77.111
-DNS: Additional Records Section: 910533066770-3. of type Scrt Key Trans Sig on c
  DNS: Resource Name: 910533066770-3.
  DNS: Resource Type = Secret Key Transaction Signature
  DNS: Resource Class = Request for any class
  DNS: Time To Live = 0 (0x0)
  DNS: Resource Data Length = 72 (0x48)
  DNS: Additional Resource Data = 03 67 73 73 09 6D 69 63 72 6F 73 6F 66 74 03 6
```

Secure Dynamic DNS Update: digitálisan aláírt DDNS regisztrációs kérés

A kiszolgáló ellenőrzi a digitális aláírást, és a #23-as csomagban – ugyancsak aláírva – visszaküldi a választ. Ezzel a biztonságos dinamikus DNS frissítést véghezvittük.

Rekordéllettartam, automatikus szemétgyűjtés

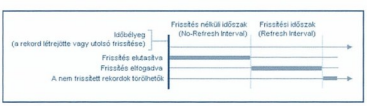
A DNS zónákban regisztrált rekordokat azok gazdája nem minden esetben törli onnan. A Windows 2000 DNS kiszolgálója beállítható úgy, hogy automatikusan törölje a régi, régóta nem frissített rekordokat. Az automatikus törlés *(scavenging)* alapértelmezésben nincsen bekapcsolva *(általában nincs is rá szükség)*; ha használni szeretnénk, kiszolgálónként, illetve azon belül zónánként is engedélyezhetjük:



Az automatikus szemétgyűjtés beállításai

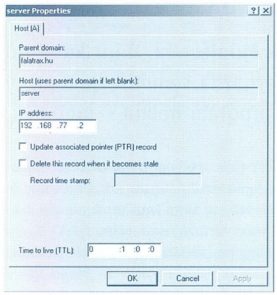
Az engedélyezés után a kiszolgáló minden dinamikusan létrehozott rekordot időbélyeggel lát el. Ez az időbélyeg frissül minden módosítás esetén, illetve bizonyos esetekben akkor is, ha a rekordhoz tartozó IP cím nem módosul, de a kliens frissíti azt. Az engedélyezéskor két időszakot kell meghatároznunk:

- No-refresh Interval: ezidő alatt a rekordok időbélyege nem frissül, még akkor sem, ha az ügyfél regisztrálni próbálja a meglévő bejegyzést *(természetesen, ha az IP cím megváltozik, akkor igen)*. Ezzel csökkenthetjük a felesleges címtárreplikációs forgalmat.
- Refresh Interval: ezalatt az időszak alatt a rekordok időbélyege frissülni fog. A Refresh Interval a No-refresh Interval-t követi, és az automatikus szemétgyűjtés csak a Refresh Interval lejárta után következhet be:



Az automatikus szemétgyűjtés időzítése

Alapértelmezésben mindkét időszak 7 nap, tehát az árva rekordok 14 nap után törlődnek majd a DNS kiszolgálóról.
A rekordok időbélyegét megtekinthetjük a rekordok tulajdonságlapján, de csak akkor, ha a DNS konzol View menüjében előzőleg engedélyeztük az „Advanced" nézetet:



A rekordok időbélyegei

A lényeg a „Delete this record when it becomes stale" jelölőnégyzet: ha ez engedélyezve van, akkor a beállított idő lejárta után a kiszolgáló a rekordot törölheti az adatbázisból. Ugyanitt megnézhetjük a rekord időbélyegének aktuális értékét is. Az ábrán látszik, hogy a „server" rekord automatikus törlése nem engedélyezett. A DNS kiszolgáló ugyanis nem fog törölni olyan rekordokat, amelyek még az automatikus szemétgyűjtés bekapcsolása előtt kerültek bele az adatbázisba. Persze, ha akarjuk, a rekordokon ezeket a jellemzőket akár kézzel is beállíthatjuk.

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:
[1] http://www.ietf.org/rfc/rfc2078.txt
[2] http://technet.netacademia.net/download/ddns/

Ethereal

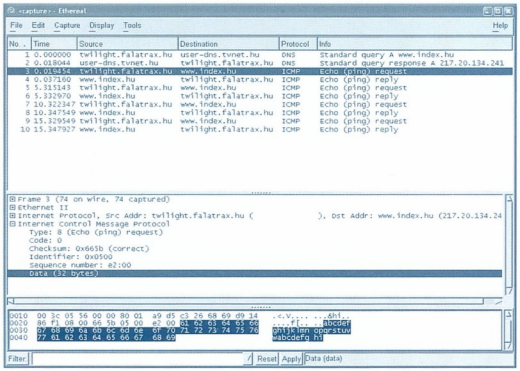
Ingyenes hálózatanalizátor

Cikkeinkben régóta hivatkozunk a Microsoft Network Monitor-ra, mint kötelező eszközre. Egy valamirevaló rendszergazda semmire sem megy rendes hálózatanalizátor program nélkül – mondjuk.

Egy pici baj van csak: a Microsoft Network Monitor csak a Windows NT/2000 Server-től „felfelé" érhető el, és abban is csak a butított, „lite" verzió. Ha a teljes változatot szeretnénk – legálisan –, akkor bizony meg kell vennünk az SMS Server csomagot, mert a Network Monitor abban bújt el. Most bemutatunk egy olyan ingyenes eszközt, ami nagyon sok tekintetben felveszi a versenyt a Network Monitor-ral, sőt, esetenként le is körözi azt. A legújabb változat *(forráskódostul)* az [1] címről tölthető le, és a Windows-os *(mármint Win95, 98, Me, NT, 2000, XP és .NET)* verziókon kívül találunk itt szinte minden operációs rendszerhez: nem kivétel a Linux, a Solaris, a MacOS, a FreeBSD, az AIX és még sok más sem.

A WinPCap driver

A programnak működéséhez szüksége van egy hálózati eszközmeghajtóra, ami a hálózat közvetlen elérését biztosítja számára. Ehhez a [2] címről le kell töltenünk – az ugyancsak ingyenes – WinPCap telepítőkészletét. A Windows XP és .Net Server legalább a 2.3-as verziót igényli – a cikk írásának pillanatában épp ez a letölthető változat. A meghajtó letöltése és telepítése egy percig sem tart. Ha ezzel megvagyunk, már telepíthetjük is az Ethereal-t. Az Ethereal elindítása után a következő kép fogad minket:



Az Ethereal főképernyője

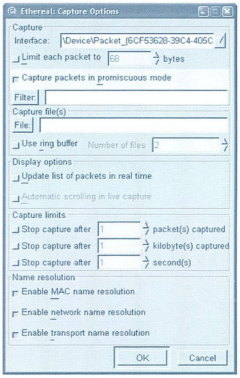
Az Ethereal főképernyője *(hasonlóan a Network Monitor-hoz)* három fő részből áll:

- A felső harmadban az elfogott csomagok listáját látjuk. Minden sor egy csomagnak felel meg. Ha egy sorra kattintunk, a képernyő többi részén megjelenik a csomag kifejtett változata.
- A középső harmad a kiválasztott csomag *(helyesebben keret, „frame")* részletes, hierarchikus, böngészhető tartalma. A protokollanalizátorok ide fejtik nekünk vissza azt, amit az adatfolyamból kinyertek. Ha itt egy sorra kattintunk, az alsó harmadban kijelölődik a hozzá tartozó adatfolyam.
- Az alsó harmad a csomag hexadecimális tartalmát mutatja.

A képernyő alján található Filter: gomb és mező a szűrés beállítására szolgál; később azt is bemutatjuk. Előbb azonban fogjunk el valamilyen hálózati forgalmat: ehhez a Capture... menü Start... parancsát használhatjuk.

A hálózati forgalom elfogása

A parancs hatására megjelenik a csomagok elfogásának paramétereit firtató dialógusablak:

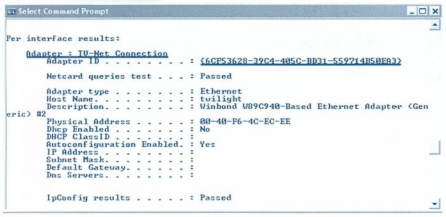


A csomaggyűjtés paraméterei

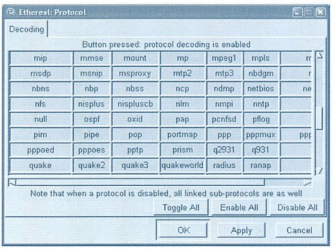
Az „Interface" sorban válasszuk ki, melyik hálózati csatolóról szeretnénk adatot gyűjteni. Ha ez a lista üres, valószínűleg nem volt sikeres a WinPCap eszközmeghajtó telepítése. Ha a lista tartalmaz sorokat, akkor sincs könnyű dolgunk, ugyanis itt csak a hálózati csatolók belső azonosítóit látjuk. Ha telepítettük a Windows 2000/XP Support Tools programcsomagot *(a Windows telepítő CD-ken rátalálunk a /Support/Tools alkönyvtárban)*, akkor adjuk ki a

```
netdiag /debug /test:ipconfig
```

parancsot. A megjelenő válaszban keressük ki az alábbi *(„Per Interface Results")* részt:



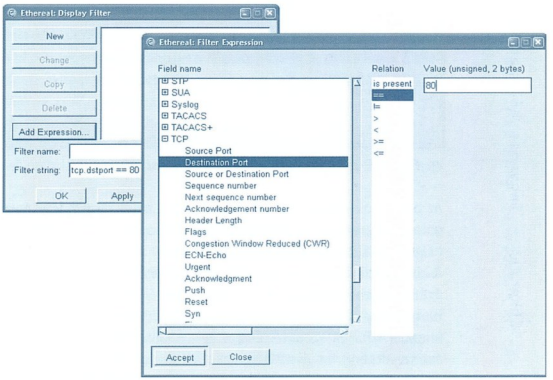
A netdiag parancs elárulja nekünk a csatolóink belső azonosítóit



Néhány a rendelkezésre álló protokollok közül

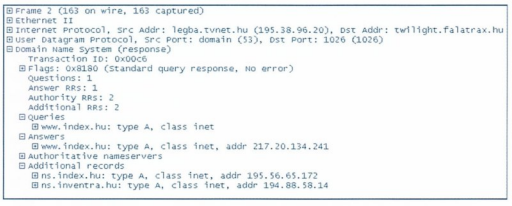
Szűrés

Ha a megjelenített csomagok tömegét szűrni szeretnénk, kattintsunk az ablak alján található Filter gombra! *(A szűrőparancsot – ami egyébként nem más, mint egy szöveges érték – kézzel is begépelhetnénk, de az Ethereal segít nekünk azt felépíteni).*



A szűrőparancs összekattintgatható egérrel is

Ha az elfogott csomagok listájába kattintunk, a középső és az alsó harmadban megjelenik a csomag visszafejtett változata. A képernyő középső harmadában a beépített protokollanalizátorok munkájának eredményét látjuk *(esetünkben éppen egy DNS kiszolgálótól visszaérkező választ; a kérdésben a www.index.hu címét firtattuk)*:



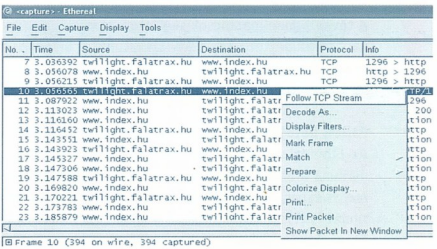
Egy DNS válasz belső lelkivilága

A hierarchia tükrözi a hálózati csomag belső felépítését: a hierarchia egyes pontjain böngészhetjük az Ethernet keret, az IP csomag, az UPD fejléc vagy – mint ábránkon – a DNS válasz tartalmát és értelmezését. Az Ethereal több mint 250 beépített protokollértelmezőt tartalmaz, közöttük sok olyat, amit a Microsoft Network Monitor nem: ilyen pl. a Kerberos, vagy mondjuk a Quake :-); és persze nem esik kétségbe akkor sem, ha mondjuk dinamikus DNS kéréseket, vagy IPSec csomagokat kell visszafejtenie! A protokollanalizátorokat egyenként ki- és bekapcsolhatjuk ha a Edit menü Protocolls... parancsára kattintunk:

Ha a szűrőnk túl szigorú lett, és a listából minden eltűnik, a Reset gomb segítségével visszaállíthatjuk az alapértelmezést. A szűrők szintaxisának teljes, részletes leírását megtaláljuk a program dokumentációjában.

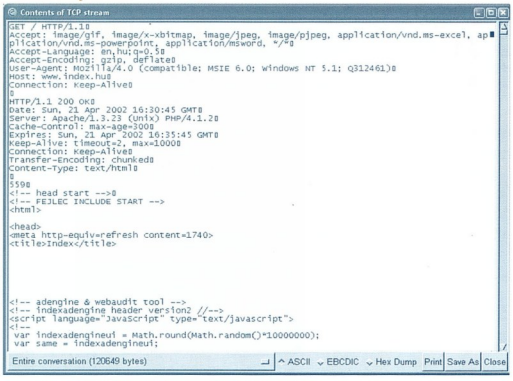
Jobbklikk egy csomagon...

... és elénk tárul az Ethereal igazi világa. Kattintsunk jobb gombbal egy csomagra:



A csomagokon végrehajtható parancsok

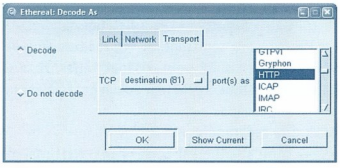
A „Follow TCP Stream" egy nagyon ügyes eszköz: összeválogatja és megjeleníti az adott csomag által kijelölt kommunikációs csatorna teljes tartalmát. Példánkban a **http://www.index.hu** webkiszolgálónak elküldött kérést tartalmazó csomagra kattintottunk, és eredményként visszakaptunk a weboldal teljes tartalmának forráskódját, hibamentesen összerakva, a különböző irányú kommunikációt különböző színekkel jelölve *(bár ez nem biztos hogy az újságban is látszani fog)*:



Egy TCP csatorna „összerakott" forgalma

Az ábrán jól látható és szétválasztható a különböző irányú forgalom, sőt, akár az oldal teljes forráskódja is; nem kell többé a hexadecimális panelen vakoskodnunk!

A következő parancs a „Decode As".... A beépített protokollanalizátorok ethernet kerettípus *(ethertype)*, IP protokollazonosító, illetve TCP/UDP portcím alapján „harapnak" rá a hálózati forgalomra. *(A TCP 80-as port például így „lesz" HTTP)*. Ha viszont mi azt szeretnénk, hogy az Ethereal a 81-es port forgalmát is HTTP forgalomként kezelje, válasszuk ki a csomagot, és a Decode As... csomag paneljében állítsuk be ezt:

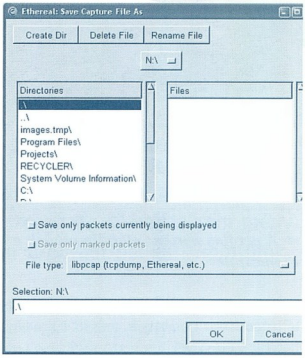


Mostantól a TCP 81-es port is HTTP

A további menüparancsok segítségével kijelölhetjük a csomagokat, figyelmeztető színekkel láthatunk el bizonyos típusú csomagokat, vagy éppen kinyomtathatjuk azokat *(bár a nyomtatás még nem igazán kiforrott)*.

Fájlok betöltése és kimentése

Az Ethereal számos fájlformátumot ismer. Alapértelmezett formátuma a libpcap/tcpdump formátum, de kezeli *(olvasni és menteni is képes)* többek között a Microsoft Network Monitor formátumát is *(!)*. A formátumot egyébként csak mentéskor kell beállítani, beolvasáskor az Ethereal automatikusan felismeri a használt formátumot. A kimentő/betöltő panellel bánjunk óvatosan, mert úgy viselkedik, mintha egy kisdiák írta volna tíz éve Pascalban, nyári gyakorlaton...

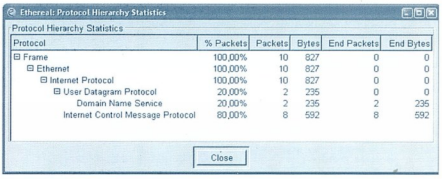


A mentőpanel.. ha kitapasztaljuk, hogy működik, már nincs gond

Ha viszont megszoktuk egymást, minden valószínűség szerint használni is sikerül majd. Ha csak a szűrőnkön fennmaradt csomagokat szeretnénk menteni, akkor válasszuk a „Save only packets currently being displayed" opciót. Ugyanez igaz a kijelölt csomagokra is *(ha vannak)*: ilyenkor a „Show only marked packets" az érdekes.

Analízis és összefoglaló információk

A Tools menüben további szolgáltatásokat találunk: a Plugins... parancs hatására láthatjuk a telepített kiegészítő DLL-eket, a TCP Stream Analysis néhány grafikonnal kényeztet minket, a Protocol Hierarchy Statistics pedig az elfogott hálózati forgalom protokolleloszlását mutatja:



Egy ping parancs protokolleloszlása: két DNS (egy kérdés, egy válasz), és nyolc ICMP (négy kérés, négy válasz) csomag

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:
[1] http://www.ethereal.com
[2] http://winpcap.polito.it/install/default.htm

Microsoft Metadirectory Services *(I. rész)*

Magyarországon is egyre több vállalat jut arra a felismerésre, hogy a heterogén infrastruktúra hosszú távú örökség is lehet – az egységesítés fontos, ám költséges és technikai kihívásokkal teli folyamat. Metacímtár kialakításával rövid távon is égető problémákat oldhatunk meg, miközben megtesszük a kezdőlépést hosszú távú egységes címtárstratégiánk megvalósítása felé.

A központi címtár problémája

Azt hiszem mindenki számára ismerős a magyar *(de valószínűleg nemzetközileg is gyakori)* helyzetkép: a vállalat rengeteg címtárral vagy csak egyszerű felhasználói adatbázissal rendelkezik. Az alapok általában mindenhol megvannak, létezik egy széles körben használt hálózati címtár *(Network Operation System, NOS Directory – felhasználók, kiszolgálók, munkaállomások adatainak tárolására – ezt a funkciót tipikusan az NT4 tartományi modell, Windows 2000 Active Directory vagy Novell NDS tölti be)*. A hálózati címtár jó esetben nagyvállalati címtárként is használható *(Enterprise Directory)*, ilyen pl. az Active Directory, amely alkalmazások *(Exchange2000, ISA2000, stb...)* adatait is tárolhatja. Elvben és technikailag arra kellene törekednünk, hogy minden vállalati alkalmazás ezt az egyetlen, központi vállalati címtárat használja *(jó esetben az Active Directoryt)* – ám szembe kell néznünk a tényekkel és a realitásokkal: korábbi beszerzésekből, saját fejlesztésekből vagy éppen a külső gyártó stratégiájából fakadóan a vállalat egyedi címtárszigetekből épül fel. Hosszútávon kijelölhető ugyan egyetlen címtár központinak *(pl. az Active Directory)*, ám a régi, egyedi címtárak átírása, lecserélése a stratégiának megfelelően meglehetősen költséges – olykor nem is lehetséges – előfordulhat, hogy a külső gyártó nem támogatja az általunk preferált címtárat, és nincs más megoldás a piacon. Ez gyakori eset a speciális, kis részterületre összpontosító üzleti alkalmazások esetén *(pl. speciális banki szoftverek, értékpapír-kezelés, devizakezelés, stb.)*.

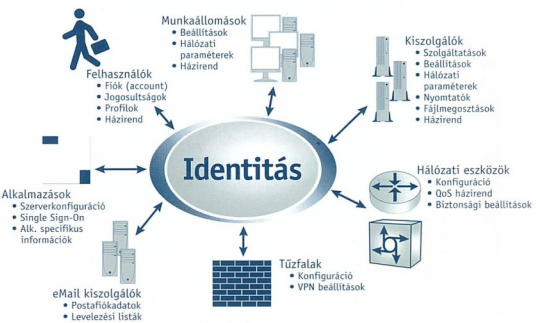
Az egységes központi címtár sokunk számára elsősorban a konzisztens adatokat jelenti. Egyedi címtárszigetek között ezt a konzisztenciát az adatok szinkronizációjával is megvalósíthatjuk. Két vagy három címtár szinkronizációja még megoldható ún. konnektorok, szinkronizációs felületek segítségével, e felett azonban már bonyolult, nehezen kézben tartható topológiát kapnánk a vezetékek bekötögetésével. Logikus tehát egy csillagtopológiában gondolkoznunk, azaz mégiscsak beültetni valamit középre, egy szinkronizációs útvonalválasztót, router-t, ami intelligensen vezényli a szinkronizációs folyamatokat. Ha a vezénylés szó ismerős lenne a BizTalk szerver terminológiából, valljuk be: valójában itt egy köztes rétegről, üzleti logikát tartalmazó speciális middleware-ről van szó.

A szinkronizáció irányításával ugyanis valójában leegyszerűsítettük a feladatot – egy ilyen útvonalválasztó még nem nevezhető metacímtárnak. A metacímtár valóban tartalmazza a fent említett üzleti logikát, és – a szinkronizáción keresztül – számos más gondunkra is orvosságot jelenthet: egypontos bejelentkezés *(single-sign-on)*, központi jogosultságkezelés, automatizált folyamatok workflow-alap vezénylése több címtárat érintő változtatásoknál stb.

A kép így egyre árnyaltabb lesz. Mielőtt konkrétan tárgyalnánk a Microsoft Metadirectory Services metacímtár megoldásának működését *(melyre csak a következő számban kerül sor)*, most elvben tekintjük át, hogy mik is pontosan az elvárásaink egy általános metacímtár rendszerrel szemben, mik a gondok amiket egy metacímtárnak meg kell oldania – azaz az ún. identitáskezelési problémakört vizsgáljuk meg.

Identitáskezelés

Identitás alatt a személyekről, alkalmazásokról és erőforrásokról meglévő, címtárakban és adatbázisokban elszórtan tárolt információinkat értjük. Egy személy identitásadata lehet például a neve, e-mail címe, fizetése vagy beosztása. Az alkalmazások identitásadatai például az alkalmazást kiszolgáló szerverek hálózati címei vagy az egyes alkalmazások által publikált szolgáltatások köre. A hálózati erőforrásoknak – pl. nyomtatóknak – szintén vannak identitásadatai – például a helyük, vagy az általuk támogatott nyomtatási módok.



Az identitáskezelési drámakör

Az identitáskezelési problémakör

Az identitásadatok változatossága és tárolási helyük sokfélesége több problémát vet fel:

- Nem minden identitásinformáció érhető el címtárakon *(pl. LDAP-on, Lightweight Directory Access Protocolon)* keresztül – sok rendszer saját objektumainak identitásadatai kizárólag programozói felületeken *(API-kon)* keresztül érhetők el.

- Az identitásadatok gyakran több helyen, több példányban fordulnak elő, a különböző változatok gyakran eltérnek egymástól, ami az idő előrehaladtával csak rosszabbodik.
- Általában nincs egyetlen hely, ahol az adminisztrátorok vagy az alkalmazások a konszolidált adatokat ***(az ún. join-t)*** elérhetik.
- Minden egyes új alkalmazás bevezetésével az identitásadatok újabb tára jelenik meg a rendszerben.

Rendszerezve ezeket a jelenségeket egy identitáskezelő rendszerrel kapcsolatban az alábbi elvárások jelennek meg általában:

Közös „telefonkönyv"/felhasználói adatbázis. Több *(leány)*vállalat, szervezeti egység telefonkönyvének, postafiókadatainak szinkronizációja, mely lehetővé teszi, hogy a különálló *(leány)*vállalatok dolgozóinak névjegy-információi a munkatársak rendelkezésére álljanak.
Felvétel/távozás támogatása. A frissen felvett dolgozó adatainak eljuttatása az érintett rendszerekbe segít abban, hogy az illető munkafeltételei gyorsan teljesüljenek. Ugyanennek gyors, fordított irányú végrehajtása szükséges akkor, amikor az illető a vállalattól távozik.
E-commerce alkalmazások. Vállalati szintű indentitásadatok *(pl. digitális aláírások)* szinkronizációja beszállítókkal és extranet felhasználókkal vagy más, a tűzfalon kívül található címtárakkal.
Egyszeri belépés kezdeményezések *(Single Sign-On)*. Felhasználói nevek, jelszavak és jogosultságok megosztása platformok és alkalmazások között, melyek esetleg eltérő hozzáféréskezelési és titkosítási technikákat alkalmaznak.

Követelmények

Az egyetlen, központi címtár bevezetésére irányuló erőfeszítések rendszerint kudarcot vallanak az alábbi egyszerű problémák valamelyike következtében:

- Sok alkalmazás nem módosítható úgy, hogy a közös címtárat használja.
- Sok esetben az alkalmazásnak jó *(pl. adatbiztonsági)* oka van arra, hogy ezen adatokat saját formátumában tárolja.
- Az egyébként lehetséges technológiai megoldások bevezetését gyakran politikai megfontolások akadályozzák. Az egyes önálló szervezeti egységek, osztályok nem szívesen engedik át a felügyeletet saját címtáradataik felett egy központi szervezetnek.

A fentiek alapján valószínű, hogy az identitásadatokat továbbra is sok helyen, különböző formákban fogjuk tárolni, fel kell tehát készülnünk arra, hogy ezen szolgáltatások együttműködését biztosítsuk. A fenti feltételezést elfogadva a megoldásnak gondoskodnia kell a következőkről:

- Kapcsolódás lehetőségének megteremtése nagyszámú különböző rendszerhez.
- Az identitásadatok különböző rendszerek közti áramlásának kezelése.
- Az adatok konzisztenciájának biztosítása a teljes identitáskezelő rendszerre vonatkozóan.

A következőkben a fent említett szempontokat tárgyaljuk.

Kapcsolódási lehetőségek *(Connectivity)*

A kapcsolódási lehetőségekkel szemben támasztott követelmény egyszerű: minél több címtárhoz, adatbázishoz és alkalmazáshoz képes kapcsolódni a rendszer, annál nagyobb értéket képvisel alkalmazója számára. Ha valamely információ nem áll rendelkezésre az egyik címtárban, esetleg elérhető egy másikban.
Az identitáskezelő megoldás akkor képes kapcsolódni valamely adattárhoz, ha képes követni az ott lezajlott változásokat, új elemeket felvenni, meglévőket törölni, tulajdonságait megváltoztatni.

Az átfogó megoldás érdekében a megoldásnak képesnek kell lennie az alábbi adatforrások kezelésére:

- Szabványokon alapuló, LDAP V3-on keresztül elérhető címtárak *(Active Directory, Novell NDS, Netscape Directory Services, stb.)*,
- Elterjedt levelezőrendszerek és nem-LDAP címtárszolgáltatások *(Exchange 5.5, Lotus Notes, egyedi fejlesztésű céges telefonkönyv, stb.)*,
- Integrált vállalatirányítási rendszerek *(ERP, Enterprise Resource Planning – SAP, JDEdwards, Oracle Financials, Peoplesoft, Scala, stb.)*,
- Adatbázisok, valamely elterjedt lekérdezőfelületen keresztül *(pl. SQL, ODBC, OLE-DB)*.
- Olyan alkalmazások, melyek kizárólag programozói felületeken *(Application Programming Interfaces, API)* keresztül érhetők el.

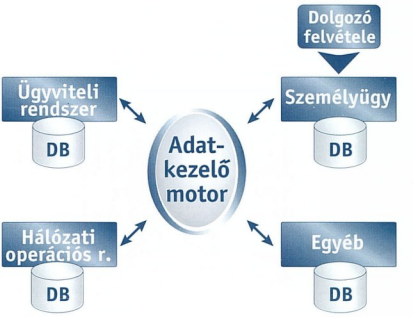
A kapcsolattal szemben támasztott követelmények

Adatáramlás *(Information Management Flow)*

A rendszer által megvalósított adatáramlásnak támogatnia kell az identitásadatok különböző adattárak közti áramlását, képesnek kell tehát lennie az identitásadatok változásának detektálására és a módosulások továbbítására más rendszerek felé, a különböző adattárak adatainak egyetlen, a vállalat teljes képét tartalmazó metacímtárba való összegyűjtésére, egymással összefüggő adatok nyomon követésére, miközben azok helye változhat az egyes adattárakon belül.

Változtatások követése *(Change Event Processing)*

A változás eseménye akkor következik be, amikor egy adminisztrátor, felhasználó vagy alkalmazás valamely adattárban identitásadatot hoz létre, töröl vagy módosít. Az identitáskezelő megoldásoknak képeseknek kell lenniük ezen változások észlelésére, a szükséges adatformátum-transzformációkra, majd valamennyi adattár megfelelő módosítására annak érdekében, hogy azok konzisztens állapotban maradjanak.



Változtatások követése

Ha például a személyügyi alkalmazásban új dolgozót rögzítenek, a dolgozó által használt alkalmazások adattárait szintén módosítani kell – ezt szemlélteti az előző ábra.

Aggregációs képességek

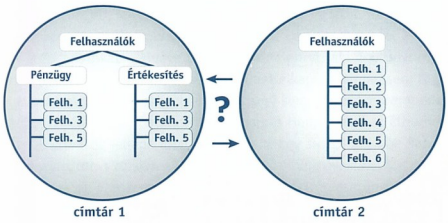
Amíg az identitásinformáció különböző rendszerekben elszórtan található, az ezt az információt egyesítő címtárak komoly értéket képviselnek. Magát a metacímtár-koncepciót és kifejezést a Burton Group használta először. Szintén ők alkalmazták a join kifejezést a vállalat aggregált identitásadatainak reprezentációjára. Egy metacímtár használatával az alkalmazások az információ széles körét érhetik el egy helyen, egyetlen hozzáférési módszerrel. A metacímtárak az adatok ideiglenes tárolásával és indexelésével *(illetve particionálásával és replikálásával)* a teljesítmény növelésére is képesek lehetnek: a lekérdezőnek nem kell közvetlenül az adatforráshoz fordulnia, mely esetleg egy WAN vonal túloldalán található. Az aggregációs mechanizmusnak képesnek kell lenne az információ összegyűjtésére számos különböző forrásból, pl. címtárakból, adatbázisokból és alkalmazásokból.

Aggregáció a metacímtárban

Az összefüggő információt csoportosítani kell tudnia, függetlenül annak származási helyétől és tárolási módjától *(lásd. az illusztrációt az előző ábrán, ahol egyetlen felhasználó a különböző rendszerekben különböző azonosítókkal rendelkezik)*. Az információt vissza kell tudnia áramoltatni a forrásrendszerekbe a metacímtárban elvégzett módosítások *(az üzleti logika lefuttatása)* után.

Kapcsolódó objektumok követése *(Related Object Tracking)*

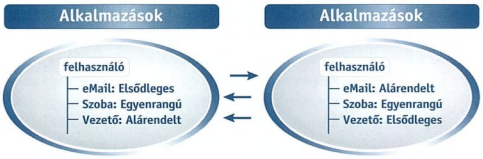
Az identitáskezelő rendszer bevezetésekor a rendszernek támogatnia kell az összefüggő objektumok azonosítását *(pl. hogy Kovács Eugén, ekovacs és kovacse egy és ugyanaz a személy)*. Ezután *(ahogy az a következő ábrán látható)* a rendszernek követnie kell ezeket az összefüggéseket az identitásadatok esetleges átszervezése ellenére is. A megoldás nem veszítheti el egy dolgozó nyomát akkor sem, ha az illető a pénzügyről a kereskedelembe, és ezáltal valamely címtárban egyik fából a másikba kerül.



Kapcsolódó objektumok követése

Integritáskezelés *(Integrity Management)*

Az integritáskezelés feladata annak biztosítása, hogy az identitásinformáció az esetleges módosítások ellenére se váljon hibássá vagy inkonzisztenssé a különböző rendszerekben. Az integritáskezelésnek kezelnie kell az adatok tulajdonosi *(ownership)* viszonyait, megfelelő szabályok alapján fel kell oldania a több rendszerbeli együttes módosítás közben esetlegesen előforduló ütközéseket *(hibakezelés)*, meg kell őriznie az identitásadatok közti hivatkozások helyességét. A következőkben ezt a három feladatot részletesebben is bemutatjuk.



Birtoklási relációk kezelése

Birtoklás *(Ownership)*

Az identitáskezelő-rendszerek egyik fontos jellemzője az egyes rendszerek és a bennük tárolt adatok birtoklási kapcsolatainak felismerése.

Példaképpen: a legtöbb vállalatnál az alkalmazott postafiókjának címét a levelezőalkalmazás birtokolja. Ezzel egyidejűleg a személyügyi rendszer szintén tárolhat ilyen információt. Identitáskezelő-rendszer alkalmazása nélkül mindkét birtoklási szabály érvényre juthat, hiszen nincs harmadik alkalmazás, mely a két, eltérő rendszer azonos jellegű adatát szinkronban tartaná.

A birtoklás jellege attribútumszintű kell legyen: a fenti esetben a postafiókcím a levelezőrendszer, a munkahelyi vezető attribútum viszont a személyügyi rendszer birtokában kell legyen. Nem állhat mindkettő a személyügyi rendszer fennhatósága alatt, hiszen a postafiók címének megváltoztatása a levelezőrendszer tudta nélkül komoly gondokat okozna. Ugyanakkor lehetnek olyan attribútumok *(pl. szobaszám)*, melyek mindkét rendszerből módosíthatóak.

A követelmény tehát, hogy a rendszerben attribútumszinten megadhatók legyenek a birtoklási viszonyok. Amennyiben egy módosítás összhangban van e birtoklási szabályokkal, azt a többi rendszer felé továbbengedhetjük, ha nincs, a továbbítást blokkoljuk, vagy éppen visszafordítjuk. Ha például a fenti esetben a beosztotti viszonyt a személyügyi rendszer tudta nélkül a levelezőrendszerben átállítjuk, akkor a rendszernek képesnek kell lennie ezt visszaállítani a személyügyi rendszerben szereplő értékre.

Hibakezelés *(Failure Management)*

A változások több adattárba való továbbításának lehetősége az identitáskezelő-technológiákkal szemben támasztott alapvető követelmény. Az egyidejűleg elvégzendő több módosítás miatt azonban fennáll annak a lehetősége, hogy közülük valamelyik nem sikeres, és így sérül az adatok konzisztenciája, ahogy az az alábbi ábrán látható.

Ha például egy személy beosztása, fizetése és költségkerete egyszerre kell, hogy módosuljon, de a metacímtár nem képes valamennyi alkalmazásban módosítani a beosztást, az identitásinformációk inkonzisztens állapotba kerülnek, melynek elhárítása általában személyes vizsgálatot és beavatkozást igényel.

Hibakezelés és a hivatkozási integritás fenntartása

Az adatbázisrendszerek ezt a problémát általában tranzakciók segítségével kezelik, melyek biztosítják, hogy a módosításnak vagy valamennyi művelete sikeresen végrehajtódik, vagy egyetlen egy sem. Sajnos azonban a legtöbb címtár- és alkalmazásprogramozói felület nem támogatja a tranzakciók kezelését, az identitáskezelő rendszereknek tehát más megoldást kell választaniuk: általában naplózáson alapuló, „kívánatos állapot" mechanizmusokat, melyek addig próbálják a változást érvényre juttatni, amíg az sikeres nem lesz.

Hivatkozási integritás *(Referential Integrity)*

A másik, adatbázisrendszerekkel közös probléma a több adattár között fennálló hivatkozási integritás biztosítása. Hivatkozási integritásnak nevezzük azon szabályok összességét, melyeket bizonyos, különböző helyen tárolt adatelemek közti összefüggéseknek ki kell elégíteniük. Az identitáskezelő-rendszereknek például biztosítaniuk kell azt, hogy egy alkalmazottnak a pénzügyi rendszerben tárolt költségkerete megfeleljen a személyügyi rendszerben tárolt beosztásának. Az adatbáziskezelő rendszerek e probléma megoldására ún. triggereket és tárolt eljárásokat biztosítanak, melyekkel a programozó vagy az adatbázis adminisztrátora bizonyos adatelemek megváltozása esetén valamely üzleti szabályt érvényesíthet. A mai címtárak nem tartalmaznak ilyen szolgáltatásokat, az identitáskezelő rendszereknek tehát biztosítaniuk kell olyan szolgáltatásokat, melyek a hivatkozási integritást sértő változtatásokat visszautasítják. Mindezeknek az elvárásoknak összességében tehát csak egy már fent is említett metacímtár rendszer tud megfelelni.

A metacímtár megoldás

A metacímtár lehetővé teszi az elszórtan, szigetekben létező, változatos adatformátumokat alkalmazó címtárak integrációját, és így az identitásadatok elsődleges forrásává válik a vállalaton belül. A metacímtár az eltérő forrásokból származó attribútumokkal rendelkező objektumok egyesített képét valósítja meg. Ez az integráció lehetővé teszi az adminisztrációs költségek csökkentését, a duplikációk megszüntetését, az eltérések csökkentését, és az identitásadatok széles körben való elérhetőségét. A metacímtár elég rugalmas, hogy bármely szervezet felépítéséhez, politikai viszonyaihoz, üzemeltetési eljárásaihoz alkalmazkodjon, és elég dinamikus, hogy a szervezet változásával együtt változzon.

Források

A metacímtár adatait a szervezet más, a metacímtárhoz kapcsolt adat- és címtáraiból szerzi. Szinte minden levelező-, adatbázis- és címtáralkalmazás képes tartalmát valamilyen formában exportálni. A metacímtár ezen adatokat fájlcsere, e-mail üzenet, vagy valamilyen online kapcsolat révén képes fogadni. Természetesen manuális úton is bevihető információ a rendszerbe.

Tartalom

Általánosan elterjedt nézet, hogy a címtárak csak a személyek identitásadatait *(pl. e-mail címét)* kezelik; a valóságban a lehetőségek ennél sokkal tágabbak: a metacímtár sok más, valós világbéli objektumról képes információt tárolni, ilyenek pl.

- fizikai objektumok *(pl. személyek vagy számítógépek)*,
- fogalmak *(pl. szervezetek vagy osztályok)*,
- földrajzi objektumok *(pl. országok vagy városok)*,
- digitális formában létező objektumok *(pl. online megtekintésre szánt dokumentumok)*.

A metacímtár egyetlen követelménye ezen adatokkal szemben, hogy azok valamiféle hierarchiába szervezhetők legyenek. A példa kedvéért egy személy tagja egy osztálynak, amely része egy szervezetnek, amely valamilyen Internet tartományban vagy országban található. Egy multinacionális szervezetben egy alkalmazott tagja lehet valamely üzletágnak, amely valamely országban a szervezeti fába illeszkedik.

A „személy" nem feltétlenül a hierarchia legalsó szintje. Elképzelhető például, hogy az egyes alkalmazottak birtokában lévő dokumentumok vagy eszközök a személy alá rendelt szinten kapnak helyet.

Rendszerfelügyelet

A metacímtár tartalmának és biztonsági kérdéseinek kezelése lehet központosított, elosztott vagy a kettőnek valamilyen kombinációja. Elképzelhető olyan megoldás, melyben a változások bizonyos bejegyzések esetében kizárólag a forrásrendszerekben, más bejegyzések esetében viszont csak a metacímtárban kezdeményezhetők. A metacímtár bizonyos tartományait más és más adminisztrátorok kezelhetik, ez pedig vonatkozik az egyes elemekre éppúgy, mint bizonyos attribútumaik halmazára. Elképzelhető például, hogy az egyes felhasználók saját telefonszámaikat, címüket maguk szerkeszthetik. A metacímtár tehát nem kényszerít a szervezetre semmilyen rendszerfelügyeleti modellt, hanem lehetővé teszi olyan címtár létrehozását, mely alkalmazkodik a szervezet valós felépítéséhez, biztonsági és hozzáférési igényeihez.

Microsoft Metadirectory Services

A fentieket olvasva számos fogalom, működési mód már ismerős lehetett az Active Directory ismereteinkből. Az Active Directory nem metacímtár, csak nagyvállalati címtár, annak ellenére, hogy a fenti elvárások jó részét teljesíti *(pl. tartalmaz konnektorokat bizonyos adatforrásokhoz)*. A Microsoft metacímtár megoldása az MMS *(Metadirectory Services)*, de a két termék egyre inkább közelebb kerül egymáshoz, így Active Directory tudásunkat tovább építve érdemes megismerkednünk az MMS lehetőségeivel is. Erre részletesen a tech.net magazin következő számában kerítünk majd sort.

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

IP Multicast

Unicast és Broadcast

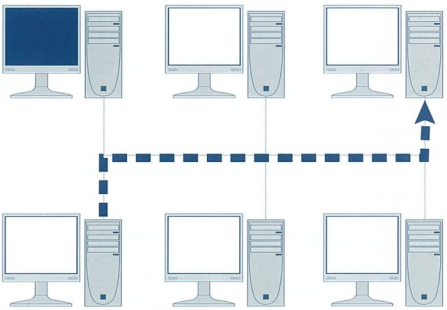
Az IP Multicast az a hálózati forgalomtípus, mely a Unicast és Broadcast között helyezkedik el félúton, s amely szép új jövőnk alapja. E nélkül ugyanis nehezen lehetne elképzelni Pirx kapitány kalandjait, hisz ha nincs Multicast, nincs sokrésztvevős videokonferencia sem itt a Földön, sem a jövőben, az űrben.

Unicast és Broadcast

Elsőként vizsgáljuk meg, hogyan is zajlik az a két hálózati forgalomtípus, amelyeket mindannyian ismerünk: a Unicast és a Broadcast. A Broadcast az egyszerűbb a kettő közül: egy adott hálózati szegmensen egy gép mindenkihez szól. Például amikor egy Windows 95/98/2000/XP indul, úgynevezett Browse Announcement Broadcast üzenettel adja tudtára az adott szegmens összes gépének, hogy ő nem más, mint \\KARALABE, és nála fut mind a Workstation, mind a Server szolgáltatás. Az operációs rendszerek indításának hálózati forgalmáról szóló cikkem a tech.net magazin májusi számában bitszinten elemzi ezt a forgalomtípust. Ez a címzésmód Ethernet-nyelvre lefordítva annyit jelent, hogy míg a csomag feladója egyetlen gép, addig címzettje az összes létező masina, beleértve az Ethernet kártyával felszerelt nyomtatókat, hűtőgépeket és kólaautomatákat is. Bitszinten a dolog úgy néz ki, hogy míg az Ethernet keret feladó mezője valóban a feladó címével *(MAC Address, például 00-A0-CC-C0-71-18)* van kitöltve, addig a címzett címének minden bitje csupa egyes *(FF-FF-FF-FF-FF-FF)*, melyre minden gép figyel. Ebből is látszik, hogy a sok-sok broadcast miért nem kívánatos a hálózatokon: minden egyes gépnek foglalkoznia kell vele még akkor is, ha valójában nem tud mit kezdeni vele. Hogy még egy szemléletes példával éljek a Broadcast címzésmód ellen, vajon mit szól a kólaautomata a hálózat sok-sok gépének DHCP címkéréséhez? Ugyanis a címkérés kezdeti csomagja *(DHCP Discover)* Broadcast címzésű *(hisz a kérelmező gép nem is használhat mást: fogalma sincs, ki a DHCP Server)*, így az minden gépen feldolgozásra kerül, zörög a winchester, pereg az órajel, s a végén az üzenet minden normál gépen kikukázódik – egyedül a valódi DHCP Server nem dogozott feleslegesen.

A Broadcast címzés használata sokszor erőforráspazarlással jár, mert válogatás nélkül minden gép megkapja a csomagot.

A Unicast címzésről a következő állapítható meg: mind a feladó, mind a címzett MAC Addresse helyesen van kitöltve az Ethernet keretben, így csak és kizárólag a címzett gép dolgozza fel a benne rejlő adatokat – kivéve talán Hacker Henryt, aki egy úgynevezett Snifferrel minden hálózati forgalmat elkap a szegmensen, függetlenül attól, hogy az neki szólt-e vagy sem *(kötekedő barátaim: hubot feltételezve)*. Erre azért lenne képes Henry, mert az Ethernet üzenetszórásos jellege miatt *(amin csak egy switch üzembe helyezése változtat, s az sem elvileg, hanem kizárólag gyakorlatilag szüntetni meg az üzenetszórást)* még Unicast címzés esetén is eljut a csomag mindenhova – viszont ebben az esetben már a hálózati kártyák elhajítják az érdektelen csomagokat, nem kell megvárni, míg az operációs rendszer kimondja a halálos ítéletet.



A szaggatott vonal mutatja a feldolgozott Unicast forgalmat. A többi gép hálókártyája eldobja a csomagot – bár oda is megérkezik!

A Unicast címzés hatékonysága is megkérdőjelezhető, ha egyszerre nem egy gép lenne a címzett – de nem is mind! Szép példája a Unicast sávszélességpazarló felhasználásának a NET SEND parancs, amelynek ha tartománynevet adunk meg, akkor a tartomány összes bejelentkezett felhasználójához eljut az üzenetünk, de úgy ám, hogy a NET SEND szépen egyesével, Unicast címzéssel kiértesíteni mind a 3634 bejelentkezett felhasználót.

A Broadcast címzés használata sokszor erőforráspazarlással jár, mert válogatás nélkül minden gép megkapja a csomagot.

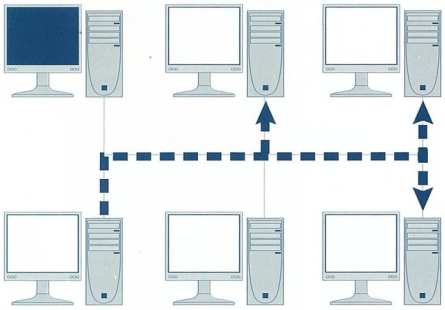
A Unicast címzés használata sokszor erőforráspazarlással jár, mert sok gép megcímzése nem lehetséges egyetlen csomaggal.

Miután ilyen szépen leszidtuk mindkét címzésmódot, lássuk, mit nyújt a Multicast!

Csoportos címzés – a Multicast

A Multicast létjogosultságát a fenti egy-két példa is ékesen bizonyította. Nem vitás, hogy például több résztvevős videokonferenciázás esetén Unicast címzéssel pillanatok alatt eldugulna a hálózat, hisz minden beszélő fél videó és audióadatát egyenként el kellene küldeni a hálózaton, ha hat résztvevő van, hát hatszor-hatszor *(=36)*. A Broadcast is ugyanilyen rossz lenne, mert bár lehetővé válna, hogy a konferenciaanyagot egy példányban küldjük

el, de bizony ilyenkor a videokonferencia becsorogna a hűtőszekrény Ethernet kártyáján is, ahol természetesen némi feldolgozás után a szemétre kerülne. Az alábbi ábra az ideális megoldást mutatja, ahol egy feladó *(a bal alsó sarokban)* egyszerre nulla, vagy több felhasználónak küld adatot, egyetlen példányban!



A csoportos címzés elvi felépítése. De mi lesz a gyakorlattal?

A gyakorlati megvalósításnál azonban számtalan elvi probléma vetődik fel:

1. Mi legyen a protokollszintű, végponttól-végpontig kísérő címzéssel, az IP-vel?
2. Hogy lehetne megmagyarázni hálózati kártyák egy csoportjának, hogy ők innentől egy brancsba tartoznak?

IP címzés Multicast adatforgalomnál

Kézenfekvő elgondolásnak tűnik a csoport tagjainak felruházása egy közös IP címmel, ám ez több kérdést is felvet.

- Ha több gépnek egyszerre azonos IP címet adunk, azt nem IP cím ütközésnek hívják? Mit szól ehhez az oprendszer?
- Ha egy csomó gépnek azonos az IP címe, de külön alhálózaton vannak, akkor vajon hogyan kommunikálnak egymással? Az útválasztón keresztül? Miért? Hogyan?
- Ki fogja beállítani a közös IP címet? A júzer?

Gordiuszi csomó, de ha odasuhintunk a kardunkkal, tüstént kibomlik. A csoport minden gépére beállítandó IP cím az IANA által lefoglalt Multicast IP tartományból származik *(D osztályú, 224.0.0.0-239.255.255.255)*, s mint ilyen, az operációs rendszerek részéről különleges bánásmódban részesítendő, IP cím ütközést sikítani illetlenség. A címtartományból látszik, hogy valószínűleg nem lesz elegendő Multicast címünk az Internet összes, kamerával felszerelt gépének ellátásához, mely csoportot akar alkotni társaival. Ez igaz. Azonban egyelőre nincs olyan nagy tülekedés, hogy fogyóban lenne e címtartomány. S hogy ki fogja beállítani az adó IP címét? Vagy kézzel a rendszergazda (ha egy odabetonozott, központi médiakiszolgálóról van szó), vagy dinamikusan (ha egy ad hoc konferencia van kibontakozóban). Ez volt a könnyebbik eset, a közös címre küldött csomagot majd szépen az IP leszállítja a megfelelő gépekre. De mit szól ehhez az Ethernet?

Ethernet címzés Multicast adatforgalomnál

Az a bökkenő, hogy a hálózati kártyák alapesetben összesen két MAC Addressre „harapnak", a sajátjukra, és a Broadcastra *(emlékeztetőül: FF-FF-FF-FF-FF-FF)*. Csoportos, vagyis több gépen közös MAC Addressről ki hallott? Olyan nincs! Vagy mégis? A 1112-es RFC elolvasása után *(Host Extensions for IP Multicasting)* kiderül, hogy ilyen állat márpedig van! Mégpedig az Ethernet kártyák azon tulajdonságának kihasználásával, hogy legtöbbjükön a gyári eredeti MAC Addressen kívül további MAC Addressek állíthatók be, melyekre szintén „harap". Már csak azt kell biztosítani, hogy az azonos csoportba tartozó gépek azonos dinamikus MAC Addresst vegyenek fel. De mi legyen az?

Kézenfekvő megoldásnak tűnik a dinamikus MAC Address képzésénél felhasználni a csoport tagjain amúgy is közös IP címet. Ám az IP cím négy, míg a MAC Address 6 bájtos. Sebaj! Kiált fel a szabvány, és a következőket mondja:

1. A hat bájtos MAC Address első három bájtja legyen 01-00-5E. Ez a lefoglalt Ethernet címtartomány Multicast címzésre van fenntartva.
2. A hátsó három bájton még 24 bitnyi infó férne el az IP címből – de az sajnos 32 bites!
3. A D osztályú IP címekben a hasznos bitek száma amúgy is csak 28, mivel az első bájt első négy bitje mindig 1110 *(decimálisan 224-239)*. Ha még „lemondunk" néhány bitről, és megelégszünk az utolsó 3 bájt felhasználásával, akkor már meg is kapjuk a csoport tagjain közösen használandó MAC Addresst.

Bepillantás a Multicast MAC Address kotyvasztás rejtelmeibe

Ezzel a módszerrel jól láthatóan sajnos olyan MAC Addresseket kapunk, ami több IP címből is előállhat, de hát ilyen az élet, a kompatibilitás, meg a későn ébredés. Tetszettek volna már 1969-ben előállni a nyavalyás videokonferencia ötletével! Ez biza így le van írva a szabványban, még ha kissé szabadosan fordítottam is:

„Because there are 28 significant bits in an IP host group address, more than one host group address may map to the same Ethernet multicast address."

Ez van. Ezt a kis konfúziót majd az oprendszer lesz szíves lekezelni-lenyelni.

Címek

Eddig még nem ejtettem szót a Multicast csoporthoz történő csatlakozás mibenlétéről, s arról, vajon hogyan jövünk rá, hogy egy olyan csoport alakult, aminek hej de jó lenne tagjává válni.

- Először is vannak olyan előre megadott csoportok, melyeknek számítógépünk by default része. Ilyen például a 224.0.0.1 cím *(All Systems on this Subnet, RFC1112)* és a 224.0.0.2 *(All Routers on this Subnet)*. Csak érdekességképpen említem *(az 1700-as RFC böngészése közben akadtam rá)*, hogy vannak egészen speciális lefoglalt Multicast címek is, például a 224.0.1.24, mely microsoft-ds szerepre van lefoglalva, s a cím felelőse **arnoldm@microsoft.com.** Ez a válaszom a NetAcademia listán feltett kérdésre, hogy vajon mi ez a cím.
- Másodszor, szert lehet tenni a kívánatos Multicast címre például a vállalati Intranetről, MADCAP kiszolgálótól ***(egyfajta DHCP Server, de multicast címeket oszt)***. A belső, dinamikus Multicast címeket a szabvány szerint a 239.192.0.0

tartományból kell osztani ***(hasonlóan a belső Unicast IP címek fenttartott tartományaihoz: a 172.16.0.0-ról mindenki tudja, hogy belső IP cím)***

- Harmadszor, internetszolgáltatótól leosztott Multicast címünk is lehet, amely megtalálhatóvá teszi publikus Multicast kiszolgálónkat ***(ugyanolyan egyedi forráscím, mint webszerverünk fix IP címe)***. A publikus Multicast címek 233-mal kezdődnek, utána két bájton a szolgáltató Autonomous System száma szerepel, majd az utolsó bájtról ad a szolgáltató nekünk egyedi Multicast címet. ***(Magyarországon persze ez az egész nem működik, mert a szolgáltatók útválasztói nem engedik át a Multicastot...)***

A megfelelő IP címek birtokában már mehet is a Multicast. A Windows 2000 Resource Kit tartalmaz egy mini-Multicast ügyfélkiszolgáló párost, hogy két végpont között PING szerű kapcsolatesztet végezhessünk: ez a MCAST.EXE.
Multicast PING elindítása:

```
mcast /send /grps:224.2.2.2 /srcs:172.16.0.3
```

Multicast PING fogadása:

```
mcast /recv /grps:224.2.2.2
```

Ideális esetben a válasz:

```
Started.... Waiting to receive packets...
Received [1]: [GOOD] SRC- 172.16.0.3 GRP-
224.2.2.2 TTL- 5 Len- 256
```

Alkalmazások

Végül tekintsük át, mi mindenre használja a Multicastot már ma is operációs rendszerünk:

- A Windows 2000 Serverben található Media Services segítségével központi adásban lehet „sugározni" a főnök beszédét, PowerPoint bemutatókat, tanfolyamokat.
- Az összes Windows *(98-tól felfelé)* képes megtalálni a hálózatán a Default Gatewayt az All IP Routers címre küldött Router Solicitation üzenettel
- Az Exchange 2000 Conferencing Server a MADCAP *(Multicast DHCP)* felhasználásával sokszereplős Multicast konferenciázást tesz lehetővé. Kipróbálva!
- Az NLBS *(Network Load Balancing System)* fürtök Multicast címzéssel oldják meg, hogy a munkaállomások egyetlen IP címet használhassanak a fürt szolgáltatásainak kihasználására. Erről bővebben a tech.net magazin tavaly júniusi számában értekeztem, Network Monitor trace fájlokkal súlyosbítva a mondanivalómat.

Más gyártók is ráharaptak az egyszeres adatátvitel miatt a Multicastra, például a klónozószoftverek *(pl. GHOST)* már elég régóta lehetővé teszik, hogy ha egyszerre másolok X darab gépre valamit, akkor az csak egy példányban menjen át a hálózaton.

Multicast buktatók

Minden előnye és fantasztikussága dacára a Multicast nem univerzális eszköz. Nem lehet vele valódi TCP csatornát létrehozni, mivel a TCP csatornának sajnos minimum és maximum két vége van. Korlátot szabhat bevezetésében az a szomorú tény is, hogy egyes régebbi hálókártyák nem képesek dinamikusan új MAC Addresst felvenni, enélkül pedig a Multicast nem működik. Útválasztós környezetben sajnos mindegyik érintett routernek ismernie kell a Multicastot, s ez az olcsóbb eszközök esetén nem teljesül. Ilyenkor dobjuk ki az olcsó eszközt, és routoljunk Routing and RAS-sal! Igaz, a Multicastalapú alkalmazások száma egyelőre nem végtelen, de egyre szaporodnak, így nagy routercsereberék előtt állunk!

Mire nem tértem ki?

Terjedelmi korlátok, valamint az elbonyolódó lehetőségek miatt nem emlékeztem meg többek között az IGMP protokollról, mely a csoportba be- és kilépés vezérprotokollja, nem néztük meg a Multicast és az útválasztók harcát, valamint a multicast feletti MTP „csatorna" protokollt sem, de az erősebb idegzetűek kedvéért, íme a forrásmunkák, ahol tovább lehet olvasni:

RFC 1112:	Host Extensions for IP Multicasting
RFC 1301:	Multicast Transport Protocol
RFC 2588:	IP Multicast and Firewalls
RFC 2730:	MADCAP
RFC 2908:	The Internet Multicast Address Allocation Architecture
RFC 2236:	IGMP

Fóti Marcell
marcellf@netacademia.net

.NET Akadémia

Delegate-ek és események (IV. rész)

Az előző számban megismerkedtünk a delegate-ekkel, és láttunk egy egyszerű példát a használatukra. Valójában sokkal többet tudnak, mint amit akkor felvázoltunk, képesek sok metódusreferencia meghívására is. Miután kiteljesítjük a korábban megkezdett atomreaktoros példánkat, rájövünk, hogy az ott végzett munkánk jelentős részét megspórolhatjuk multicast delegate-ek felhasználásával.

Multicast delegate-ek

Delegate-ek felhasználásával egyszerűen meg tudtuk oldani a megfigyelők *(szivattyúk)* értesítését, azonban nekünk kellett egy listában nyilvántartani a hívandó metódusokra hivatkozó delegate-eket. Pedig ezt is tudják, csak nem a közönségesek, hanem a multicast típusúak. A multicast delegate egy olyan konstrukció, amelynek több mint egy célpontja is lehet, azaz több mint egy metódust is képes meghívni. A metódushívások sorrendje nem determinisztikus, és nem garantált, hogy mindegyik meghívódik, ha valamelyik hívott „felemel" egy kivételt *(Exception)* és nem kezeli le helyben, azaz a kivétel eléri a delegate-et.

Ismétlésképpen az előző számban felvázolt példában a lényegi rész a következő volt:

```
//A szivattyúosztályokba hívó delegate
public delegate void SzivattyuBekapcsolas();
//A szivattyúk hívandó metódusait tároló
//delegate-ek listája
private ArrayList szivattyuk = new ArrayList();
//A szivattyúk bekapcsoló metódusainak hívása a
delegate-eken keresztül
foreach(SzivattyuBekapcsolas szbekapcsolo
        in szivattyuk) {
  szbekapcsolo();
}
```

Azaz egy ArrayList-ben *(~ nyújtható tömb)* tároltuk a szivattyúk bekapcsolását végző metódusokra hivatkozó delegate-eket, és egy ciklusban egyenként végeztük el a metódushívásokat a delegate-eken keresztül.

Hasonlóan működik a multicast delegate is. Van egy belső listája, aminek Invocation List *(hívási lista)* a neve. Ez valójában nagyon egyszerűen van megvalósítva, minden multicast delegate-nek van egy belső *(private)* _prev nevű multicast delegate referenciája, ami a következő hívandó multicast delegate-re mutat. Amikor egy delegate-en keresztüli metódushívást hajtunk végre, akkor a delegate-híváslánc végigfut a _prev-ek mentén, és minden delegate meghívja a Method és a Target jellemzői mögött tárolt osztály vagy objektumpéldány megfelelő metódusát. A Target egy referencia arra az osztálypéldányra, amiben a célmetódust meg kell hívni. Nyilván ennek értéke null, ha statikus *(osztályszintű)* metódust hívunk meg.

Hogyan kell definiálni egy multicast delegate-et? Igazából pont úgy, mint egy „közönségeset", ahogy az előbb a SzivattyuBekapcsolas példában is láthattuk. Ez azt jelenti, hogy eddig is multicast delegate-et hoztunk létre, csak nem használtuk ki a képességeit? Igen! A hivatalos leírás alapján akkor lesz egy delegate deklarációból multicast delegate, ha a delegate által reprezentált metódusnak void a visszatérési értéke. Azért, mert egy multicast delegate hívásakor úgyis csak az utoljára meghívott metódus visszatérési értékét kaphatjuk vissza, a többi elveszik, nem tárolja a multicast delegate.

Na, de mi van azokkal a metódusokkal, amelyeknek ugyan van visszatérési értéke, de nem mindig akarjuk felhasználni? Azaz mindenképpen multicast delegate-et akarunk használni az egyszerű csoportos hívás miatt, és nem érdekel minket, ha elveszik a metódus visszatérési értéke. Ha a voidos szabály igaz lenne, akkor ilyet nem tehetnénk. De szerencsére tehetünk. A lefordult kódokat IL Disassemblerrel megtekintve az összes delegate-ből multicast delegate lesz *(legalábbis Windows platformon és C# fordítóval)*. Miért?

Amikor a Microsoftnál a delegate-eket tervezték, különválasztották a két típust, és az egyiket a System.Delegate, a másikat a leszármazott System.MulticastDelegate osztályban valósították meg. A megkülönböztetés zavarta a fejlesztőket, a fordító *(compiler)* írókat, a CLR csapatot és még sokan másokat, ezért össze akarták őket vonni egy osztályba, amely tulajdonképpen multicast delegate, csak bizonyos esetekben 1 hosszúságú a hívási listája. Azonban ez az ötlet túl későn jött, és túl sok kihatása lett volna ez egész .NET framework-re, ezért már nem merték meglépni a változtatást. Majd a jövőben...

Azaz most már tudjuk, hogy a SzivattyuBekapcsolas valójában egy multicast delegate. Hogyan kell hozzáadni a hívandó a metódusokat? A Delegate osztály Combine statikus metódusát erre találták ki. Ő vagy két delegate-et vár paraméterül, vagy delegate-ek tömbjét. Az előbbit használjuk leggyakrabban, amikor egy multicast delegate-hez hozzá kell adni egy újabb hívandó metódust. Eddig így nézett ki a hívandó metódusokat (bekapcsolandó szivattyúkat) regisztráló metódusunk:

```
public void SzivattyuHozzaad(object szivattyu) {
  szivattyuk.Add(szivattyu);
}
```

Mivel egyszerűen definiálhatunk egy multicast delegate-et, már nem is kell a delegate-eket tároló ArrayList:

```
/// <summary>
/// Ebben a multicast delegate-ben tároljuk
/// a szivattyúk listáját
/// </summary>
private SzivattyuBekapcsolas szbekapcs = null;
public void SzivattyuHozzaad(
            SzivattyuBekapcsolas szivattyu)
{
  szbekapcs = (SzivattyuBekapcsolas)
    Delegate.Combine(
    szbekapcs,
    szivattyu);
}
```

A Combine statikus metódusként van megvalósítva, így nem kell foglalkozunk azzal, hogy az szbekapcs delegate-ünk már ténylegesen hivatkozik egy delegate-re, vagy még nem, így null az értéke. Az első híváskor az szbekapcs-ban levő null és a szivattyúban kapott delegate kombinációjaként az szbekapcs multicast delegate-ünk egy metódushivatkozást fog tartalmazni.
A SzivattyuHozzaad második hívásakor az előbbi delegate-ünk meghízik, hozzáláncolódik az újabb hívandó metódusra hivatkozó delegate.
A metódusok meghívása éppoly egyszerű, mint a sima delegate-eknél, csak le kell ellenőriznünk, hogy egyáltalán inicializálva van-e a delegate referenciánk, vagy null az értéke, mert esetleg még nem is adtunk hozzá hívandó metódusokat:

```
//Szivattyúkat elinditani
if (szbekapcs != null) {
  szbekapcs();  //delegate-en keresztüli hívás
}
```

Ez az ellenőrzés fontos a nem multicast delegate-eknél is.
Ha hozzá lehet adni egy delegate-et a hívási listához, akkor valószínűleg ki is lehet venni onnan. A Delegate.Remove-ot erre találták ki. Első paramétere a hívási lista feje *(a multicast delegate-ünk)* a második az eltávolítandó delegate:

```
szbekapcs = (SzivattyuBekapcsolas)
            Delegate.Remove(
            szbekapcs,
            szivattyu);
```

Teszteljük az új, multicast delegate-ekre épülő osztályunkat *(a kód teljes környezetében a* **[1]** *címről letölthető példában tekinthető meg)*:

```
SzivattyuBosch b = new SzivattyuBosch();
SzivattyuAKG a = new SzivattyuAKG();
HoerzekeloMulticastDelegatte hm = new
  HoerzekeloMulticastDelegatte();
SzivattyuBekapcsolas sz1 = new
  SzivattyuBekapcsolas(a.Indit);

hm.SzivattyuHozzaad(sz1);
hm.SzivattyuHozzaad(
  new SzivattyuBekapcsolas(b.Bekapcs));
hm.Meres();
Console.WriteLine("Most pedig kivonjuk az egyik
↳szivattyút.");
hm.SzivattyuEltavolit(sz1);
hm.Meres();
```

Ha jobban megnézzük a kódot, akkor látható, hogy trükköztem, mert megjegyeztem az a.Indit-ra hivatkozó delegate-et, mert tudtam, hogy később el akarom távolítani a listából.
Azonban a gyakorlatban általában ez nem így történik meg, így kérdés, hogy miután többször létrehozunk delegate referenciákat ugyanarra a metódusra, a Delegate.Remove észreveszi-e, hogy tulajdonképpen ugyanarról a delegate-ről van szó?

```
SzivattyuBekapcsolas sz1 =
new SzivattyuBekapcsolas(a.Indit);
SzivattyuBekapcsolas sz11 =
new SzivattyuBekapcsolas(a.Indit);
```

Sz1 ugyanaz mint sz11? Nos, referenciájukat tekintve nem:

```
Console.WriteLine(
object.ReferenceEquals(sz1, sz11));
false
```

Azaz van két System.Delegate *(leszármazott)* objektumunk, amelyek a saját értelmezésünk szerint ugyanarra a hívandó metódusra mutatnak, azaz számunkra egyformák. És valóban, az == operátor már ennek megfelelően működik:

```
Console.WriteLine(sz1 == sz11);
true
```

Ez egy jó példa arra, hogy érdemes felüldefiniálni egy operátor jelentését, ha az intuitív működése nem ugyanaz, mint az alapértelmezett működés. Hasonló a helyzett a string típusnál is, ahol az == nem referencia, hanem érték szerinti összehasonlítást jelent. Az eddigiek miatt az alábbi kódrészlet éppúgy (jól) működik, mint a hárommal ezelőtti blokkban látható:

```
hm.SzivattyuHozzaad(
   new SzivattyuBekapcsolas(a.Indit));
hm.SzivattyuEltavolit(
   new SzivattyuBekapcsolas(a.Indit));
```

És még nincs vége az operátor felüldefiniálásoknak! Az előbbi Combine és Remove metódushívásokat végrehajthatjuk a + és a - operátorok segítségével is:

```
szbekapcs = szbekapcs + szivattyu;
szbekapcs = szbekapcs - szivattyu;
```

A + hozzáad egy delegate-et a listához, a - pedig kiveszi. Leggyakrabban az += és a -= formában, az értékadás operátorral együtt szoktuk használni:

```
szbekapcs += szivattyu;
szbekapcs -= szivattyu;
```

Ennél tömörebben nem lehet megfogalmazni a kívánságunkat. Valójában a két operátor Delegate.Combine és Delegate.Remove-re fordul le, amint az alábbi disassemblált IL részletből is látszik:

```
.method public hidebysig instance void  SzivattyuHozzaad1(class SzivattyuBekapcsolas szivattyu) cil managed
{
  // Code size       24 (0x18)
  .maxstack  3
  .language '{3F5162F8-07C6-11D3-9053-00C04FA302A1}', '{994DA5CA-E4E9-11D2-903F-00C04FA302A1}', '{5A869D0B-6611-11D3-BD2A-0
// Source File 'c:\documents and settings\soci.dotnet2000\my documents\articles\net\net4\delegatesandevents\szivattyuzas.cs'
//000001:             szbekapcs += szivattyu;
  IL_0000:  ldarg.0
  IL_0001:  dup
  IL_0002:  ldfld      class SzivattyuBekapcsolas HoerzekeloMulticastDelegattel::szbekapcs
  IL_0007:  ldarg.1
  IL_0008:  call       class [mscorlib]System.Delegate [mscorlib]System.Delegate::Combine(class [mscorlib]System.Delegate,
                                                                                           class [mscorlib]System.Delegate)
  IL_000d:  castclass  SzivattyuBekapcsolas
  IL_0012:  stfld      class SzivattyuBekapcsolas HoerzekeloMulticastDelegattel::szbekapcs
//000002:         }
  IL_0017:  ret
```

Megfigyelhető, hogy a Combine kimenetét a visszakonvertáló utasítást *(castclass)* automatikusan megírja helyettünk a fordító.

Vegyük kézbe a vezérlést!

A multicast delegate könnyedén végighívja a listában található összes metódust. Azonban, mi van, ha valamelyik hívott metódus:

- „felemel" egy kivételt *(raise an exception)*
- nagyon sokáig elmolyol magában, vagy egyáltalán vissza sem tér a hívásból

Az első esetben vagy elszáll a teljes program, vagy ha lekezeljük a kivételt a delegate hívásakor, akkor megáll a hívási láncolat. A második esetben a többi metódust csak későn vagy soha nem fogja meghívni a delegate. Lássunk az elsőre egy példát:

```
public class RendetlenSzivattyu {
  public void Puff() {
    Console.WriteLine("Most dobok egy kivételt.");
    throw new
    Exception("Én vagyok a rendetlen szivattyú!");
  }
}
```

Rakjuk be a renitens szivattyút a listába, majd hívjuk meg a mérést, amely elindítja a multicast delegate automatikus hívási láncát:

```
RendetlenSzivattyu rsz = new RendetlenSzivattyu();
hm.SzivattyuHozzaad1(
  new SzivattyuBekapcsolas(rsz.Puff));
hm.Meres();
```

A hívás kódja a Meres() mögött:

```
szbekapcs();
```

Az eredmény, hogy elhasal a programunk:

```
{C:\Documents and Settings\soci.DOTNET2000\My Documents\Articles\NET\
Bosch elindult.                                                  12:28
* és - operátorok
AKG elindult.
A rendetlen szivattyú is a listában van
AKG elindult.
Most dobok egy kivételt.

Unhandled Exception: System.Exception: Én vagyok a rendetlen szivattyú!
   at RendetlenSzivattyu.Puff() in c:\documents and settings\soci.dotnet2000\my
documents\articles\net\net4\delegatesandevents\szivattyuzas.cs:line 21
   at SzivattyuBekapcsolas.Invoke()
   at HoerzekeloMulticastDelegattel.GazVan() in c:\documents and settings\soci.d
otnet2000\my documents\articles\net\net4\delegatesandevents\szivattyuzas.cs:line
196
   at HoerzekeloMulticastDelegattel.Meres() in c:\documents and settings\soci.do
tnet2000\my documents\articles\net\net4\delegatesandevents\szivattyuzas.cs:line
109
   at AlaptipusokIII.Reaktorteszt() in c:\documents and settings\soci.dotnet2000
\my documents\articles\net\net4\delegatesandevents\szivattyuzas.cs:line 149
   at AlaptipusokIII.Main(String[] args) in c:\documents and settings\soci.dotne
t2000\my documents\articles\net\net4\delegatesandevents\szivattyuzas.cs:line 163

C:\...et4\DelegatesAndEvents\bin\Debug>_
1Left  2Right  3View  4Edit  5Print  6MkLink  7Find  8History  9Video  10Tree
```

Egy atomreaktor esetén ez nem túl szerencsés, habár estek már le műholdakat földkörüli pályára állító rakéták hasonló okokból. *(Mielőtt valaki félreértené a C# és a .NET nem műhold vagy atomreaktor vezérlésre van kitalálva.)*

Kapjuk el a kivételeket a hívó oldalon:

```
try {
  szbekapcs();
}
catch(Exception e) {
  Console.WriteLine(e.ToString());
}
```

Most már nem repül el az egész program, de a kivétel után nem folytatódik a maradék delegate meghívása. Ez baj, nagy baj. Mit lehet tenni?

A kezünkbe kell venni a vezérlést. A multicast delegate van olyan kedves, és a kezünkbe adja az általa összeláncolt összes delegate-et egy tömbben. Ezek után már csak végig kell mennünk a tömbön, és egymás után meghívni a delegate-eket. Mivel hívásonként le tudjuk kezelni a hibákat, garantáltan végig tudjuk hívni az összes résztvevőt. A bűvös metódus a GetInvocationList(), és az alábbi módon használható:

```
Delegate[] d = szbekapcs.GetInvocationList();
foreach(SzivattyuBekapcsolas sz in d) {
  try {
    sz();
  }
  catch(Exception e) {
  Console.WriteLine(e.ToString());
  }
}
```

Valójában még azt is tudjuk melyik szivattyú hibázott, mert a delegate Target jellemzőjéből kiolvasható a hivatkozott objektum, így annak minden nyilvános jellemzőjét kiolvashatjuk. Most ilyenünk nincs, de a hibázó osztály nevét könnyedén kinyerhetjük:

```
Console.WriteLine("A hibás osztály: {0}",
        sz.Target.GetType().ToString());
```

Ezek után melyik módszerrel éljünk? Az automatikus hívást, vagy a kézi végiggyalogolós módszert? Nos, ha nem bízunk meg a hívottakban, akkor marad a kézi módszer. Ha tudhatjuk, hogy azok nem „emelnek" kivételeket, és nem kell minden egyes hívás visszatérési értéke *(amit csak az egyenkénti hívással nyerhetünk ki)*, akkor kényelmesebb az automatikus módszer.

Események

Láthattuk, hogy a delegate-ek segítségével könnyedén ki lehet építeni olyan futásidőben felépíthető objektumok közötti kommunikációt, amelyben egy objektum értesíthet bármely más objektumot belső állapotváltozásáról. Általában ezt nevezzük klasszikus értelemben vett eseménykezelésnek. Az eseményforrás *(publisher)* értesíti a feliratkozókat *(subscriber)* valamely számukra érdekes belső állapotváltozásáról. Mivel mindezt könnyedén meg lehet oldani delegate-ekkel, akkor minek az események *(events)*?

Valójában munkát spórolnak meg nekünk, és lehetetlenné teszik azt, hogy egy óvatlan feliratkozó leiratkoztasson valaki mást egy eseményforrásról.

Nézzük az előbbi példánkat. Az eseményforrásként tetszelgő Hoerzekelo osztályunkban volt egy private, a külvilág számára elérhetetlen multicast delegate *(szbekapcs)*, és ehhez lehetett hozzáadni, illetve elvenni feliratkozókat. Ehhez írnunk kellett két nyilvános elérőmetódust, a SzivattyuHozzaad és a SzivattyuEltavolit-ot. Mi van, ha százféle eseményről akar értesítést küldeni egy osztály? Akkor bizony száz delegate-et és kétszáz elérő metódust kell írnunk. Ez nem túl ambiciózus feladat. Ezért a tizedik táján *(lustaságból)* úgy döntünk, hogy elfelejtjük az OOP-s egységbezárás és absztrakció elvét, és nyilvánossá tesszük a további delegate-eket. Valahogy így:

```
public class HoerzekeloEventRavezetes {
  public SzivattyuBekapcsolas szbekapcs; ...
```

Ezek után a feliratkozók nagyon egyszerűen rácsatlakozhatnak a delegate által szolgáltatott értesítési láncra, a már megismert += operátorral:

```
HoerzekeloEventRavezetes he =
  new HoerzekeloEventRavezetes();
he.szbekapcs += new SzivattyuBekapcsolas(a.Indit);
```

Ugye milyen egyszerű? Semmi cicó, használjuk a nyilvános delegate tagot. Ám ekkor csap be a ménkő! A következő szivattyú elfeledkezik róla, hogy már valaki várja az értesítéseket, és arcátlanul berakja magát a másik (előzőek) helyére:

```
he.szbekapcs =
new SzivattyuBekapcsolas(b.Bekapcs);
```

Figyelem, nem +=-t, hanem =-t használt! Megteheti? Meg! Tudunk ellene védekezni? Nyilvános delegate-tel nem, csak a korábban látott elérő metódusokkal. Ott tartunk, ahonnan elindultunk. Ki akartuk kerülni az elérő metódusok megírását, s most tessék, nélkülük nem lehet biztonságos eseménykezelést megvalósítani. Eljött az ideje, hogy belépjen a felmentő sereg, az esemény *(event)*! Az esemény nem más, mint egy delegate-re épülő réteg, amellyel pont az előbb vázolt problémát oldhatjuk meg anélkül, hogy elérő metódusokat kellene írnunk. Írja meg azt a fordító, nem?
Egy event-höz mindig definiálnunk kell egy delegate-et, ez írja le az esemény által értesíteni kívánt metódusok formátumát. Tulajdonképpen a metódushívásokat teljes egészében a delegate valósítja meg. Az előbbi példánk így néz ki event-öt felhasználva:

```
public event SzivattyuBekapcsolas szbekapcs;
```

Azaz csak annyi a változás, hogy az osztálybeli delegate referencia létrehozásakor az event kulcsszót írjuk a deklaráció elé (vö. hárommal korábbi kódrészlet).
Kívülről, a feliratkozók oldaláról semmi különbséget nem veszünk észre, egy eseményre ugyanolyan módon lehet feliratkozni, mint ahogyan a multicast delegate-nél láttuk, a += operátorral:

```
HoerzekeloEventtel hev = new HoerzekeloEventtel();
hev.szbekapcs += new
  SzivattyuBekapcsolas(a.Indit);
```

Azonban az ármánykodókra rossz világ vár:

```
hev.szbekapcs =
  new SzivattyuBekapcsolas(b.Bekapcs);
Hiba:
The event 'HoerzekeloEventtel.szbekapcs' can only
appear on the left hand side of += or -=.
```

Hát nem ezt akartuk? Az eseményre bárki rácsatlakozhat *(és lecsatlakozhat róla, -=)*, de nem üthet ki másokat a hívási listából. Igazából ezért hozták létre az eseményeket.
Mit csinál a fordító a háttérben? Valójában a publisher osztályban létrehoz egy private delegate-et az event-ben megjelölt típussal *(private SzivattyuBekapcsolas szbekapcs)*. Ezen felül ír két elérő metódust, az egyik neve add_eseménynév, a másiké remove_eseménynév. Esetünkben add_szbekapcs és remove_szbekapcs. Ezek a nyilvános elérőmetódusok, és a +=, -= eseménykezelő hozzáadások esetén ezeket hívja meg a fordító által generált kód. Mivel valódi hívásokat végző delegate nem érhető el nyilvánosan, érhető miért nem megy az = operátoros értékadás.
Emellett még egy szbekapcs nevű nyilvános esemény is bekerül a lefordított kódba, ezen keresztül találják meg a valódi add és remove metódusokat a nyelvi fordítók.
A delegate-eket olyan formátumúra írjuk, ahogyan nekünk tetszik. Ez igaz az eseményekre is, azonban a framework dokumentáció számos útmutatást ad arra nézve, hogyan írjuk meg az eseménykezelésünket. Ha alkalmazkodunk ehhez, akkor a feliratkozók egységes módon tudják lekezelni az eseményeket.
Az események nevét lehetőleg igékből képezzük, mindenféle előtag nélkül: Closed, Closing. Általában az ige folyamatos alakja *(Closing)* azt jelezi, hogy még beleavatkozhatunk az eseménybe, a múlt idejű *(Closed)* azt, hogy már csak értesítést kaptunk egy olyan eseményről, amibe már nincs beleszólásunk.
Az események mögötti delegate neve legyen az esemény neve plusz az EventHandler *(eseménykezelő)* szócska: ClosingEventHandler, ClosedEventHandler. Mivel az események forrása érdekes lehet a hívottaknak, ezért az esemény első paramétere mindig legyen sender nevű és a típusa legyen object. A publisher ezt töltse fel saját magára mutató referenciával *(this)*. Ezen felül az eseményre vonatkozó további paraméterek szállítására használjunk egy második paramétert is, amely az EventArgs osztály egy leszármazottja legyen, és a neve legyen XxxEventArgs.
Ezek a legfontosabb iránymutatók, azonban a framework dokumentáció még további javaslatokat is tartalmaz az Event Naming Guidelines részben. Érdemes egyszer átolvasni.

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

A cikkben szereplő URL-ek:
[1]: Letölthető példakódok
http://technet.netacademia.net/download/dotnet

XMLgessünk

Az XML Schema (I. rész)

Számtalanszor hivatkoztunk már XML sorozatunkban az XML sémára, így hát itt az ideje, hogy közelebbről megismerjük. Ebben a részben áttekintjük az alapismereteket, és megnézzük az egyszerű típusok képzésének fortélyait. A következő részben áttekintjük az összetett típusok kialakításának részleteit, a harmadik, záró részben pedig tervezési mintákat nézünk át újrafelhasználható és jól olvasható sémák írásához, valamint megnézzük milyen eszközökkel lehet ténylegesen kihasználni a sémák adta előnyöket.

Kezdjük az alapokkal. Mi az a séma?

Az xml sémaleíró nyelvek célja, hogy formális leírást adjanak xml dokumentumosztályok definiálására. Másképpen megfogalmazva nyelvtani leírások, amelyek segítségével xml struktúrák által ábrázolni kívánt üzleti szabályokat definiálhatunk.

Egy-egy xml dokumentumminta nem elég az egyértelmű leíráshoz. Nézzük például az alábbi xml töredéket:

```
<helyzet>
   <szelesseg>47.97426262</szelesseg>
   <hosszusag>24.290234</hosszusag>
   <bizonytalansag mertekegyseg="meter">10
   </bizonytalansag>
</helyzet>
```

Ránézésre sejthetjük, hogy egy hely földrajzi koordinátáit írja le ez a töredék. De milyen pontosságú a két koordinátát ábrázoló szám? Milyen határok között mozoghatnak az értékek? Kötelező-e megadni a bizonytalanság értékét? Milyen mértékegységeket használhatunk? Ezek mind nagyon fontosak az adatok értelmezéséhez, és ezeket kellene valahogyan formálisan leírni. Erre való a séma.

Mit ír le egy séma?

- Definiálja, milyen elemek és attribútumok lehetnek egy dokumentumban.
- Definiálja az elemek egymásba ágyazását.
- Definiálja a gyermekelemek sorrendjét.
- Definiálja a gyermekelemek számát.
- Definiálja, hogy egy elem üres, szöveget, vagy további gyermekelemeket tartalmaz.
- Definiálja az elemek és attribútumok adattípusát.
- Alapértelmezett értéket definiál elemekhez és attribútumokhoz

Egyszerűbben megfogalmazva a dokumentum struktúráját, valamint a benne található adatok típusát és értelmezési tartományait adja meg.

Mit jelent az a szó, hogy validálás?

A sémák egyik fő célja, hogy miután formális leírást adtak egy dokumentumosztályról, programozottan meg lehessen mondani, hogy egy konkrét xml dokumentumpéldány megfelel-e a sémának? Ezt a folyamatot nevezzük validálásnak, és az olyan xml doksit, ami egy sémának megfelelő, validnak.

Mire jó a séma?

Számos helyen használjuk a sémákat, Don Box szerint „a séma a villanyáram és a melegvíz az xml technológiákban", azaz e nélkül nem lehet kulturáltan, komolyan xml-ben fejleszteni.

Az első felhasználás a struktúradefiniálás. Hogyan magyarázom el az üzleti partnereimnek, hogy milyen formátumú xml dokumentumokat várok például a megrendelések leírására? Ahelyett, hogy körbemagyaráznám a formátumot, egyszerűen adok neki egy sémát, és azt mondom, hogy így néz ki a megrendelőlap. Ebből a precíz leírásból ő már képes a sémának megfelelő xml dokumentumokat generálni. Egyes eszközök *(pl. SQL Server 2000)* a séma minimális kiegészítésével azonnal képesek xml kimenetet generálni a táblák adataiból.

Miután bejött hozzám egy megrendeléstétel, le kell ellenőriznem, hogy az tényleg a sémának megfelelő formátumú, azaz validálásra használom a sémát. Validálás nélkül letárolni a kapott információkat *(minimum erkölcsi)* öngyilkosság.

A SOAP által javasolt kommunikációs modellben a kommunikáló programok között xml formátumban utaznak az információk, például a távoli eljáráshívások paraméterei. Mivel mindkét oldal objektumokban gondolkodik, szükség van egy olyan formális leírásra, amelynek segítségével a programozott típusokat *(osztályok, interfészek, primitív típusok)* automatikusan át tudjuk fordítani xml struktúrákká, és vica versa. A programozók objektumokban és tagváltozókban gondolkodnak, az xml pedig elemekben és attribútumokban. A két világ áthidalására a SOAP szabvány is az XML sémára bízza az összes olyan típus leírását, amelyet eleve ismer a séma *(amit nem, azt kiegészítették a SOAP specifikációban)*.

Milyen sémákat ismerünk?

A legrégebbi sémaleíró nyelv a Document Type Definition, a DTD. Ő az SGML világból jött át, ám az XML világban nem igazán hatékony. A fő problémák vele:

- Nem xml formátumú
- Nem ismeri a névtereket
- Nem bővíthető
- Csak egy teljes dokumentumra alkalmazható
- Fő probléma: nem ismeri a programozott világban megszokott adattípusokat

Világos volt, hogy le kell cserélni egy jobban „xmlesített" sémaleírásra. Ez azonban nem volt egyszerű meccs. Többféle javaslat is napvilágot látott, név szerint a legfontosabbak: RELAX, TREX, Schematron, SOX, DSD, XDR és XML Schema.

A sémák egy része szabályalapú, más részük nyelvtani elemekből építkezik. A szabályalapúak általában fejlettebbek az xml dokumentumpéldányok szabályellenőrzésében *(validálás)*, a nyelvtanalapúak pedig jobbak struktúra definiálásra. Az előbbire az egyik legfejlettebb nyelv a Schematron, az utóbbira az XML Schema. A szabályalapúakkal le lehet írni olyan környezetfüggő kötöttségeket is, amelyeket nem lehet nyelvtannal megfogalmazni. Például ha egy elemnek van x nevű attribútuma, akkor kell lenni y-nak is. Vagy ha egy elem szülője z, akkor kell legyen j nevű attribútuma. Ezzel szemben a nyelvtanalapú sémákból remek programozott objektumokat lehet generálni: osztályokat, interfészeket. Ez a Webszolgáltatások egyre erősödő piacán nagyon fontos feladat, és talán ez is oka annak, hogy a World Wide Web konzorciumnál az XML Schema Definition *(XSD)* győzött, mint xml séma ajánlás. Ez nem jelenti azt, hogy a többi kihal, valószínűleg hosszabb távon a szabályalapúak közül is ki fog fejlődni egy jól használható leírás.

Az XML Schema

2001. május óta az XML Schema *(XSD)* az aktuális, a w3c által javasolt sémaleírás. A formátum gyökerei a Microsoft XML Schema Reduced *(XDR)* sémához kötődnek, melyet *(micsoda meglepetés)* a Microsoft fejlesztett ki. Azért kellett az MS-nek az XDR-el bajlódni, mert a DTD kevés volt, végleges séma még nem volt a látóhatáron, így a korai xml támogatottságú termékekhez kellett egy fejlettebb sémaleíró nyelv. Az SQL Server 2000, az Internet Explorer 5.x, a Biztalk 2000, az Exchange 2000 mind-mind XDR-t használnak.
Tavaly május óta van szabványos sémánk, így az SQLXML2-től már használhatjuk az XSD-t az SQL Server programozásához, az MSXML4 már XSD kompatibilis, valamint a .NET framework osztályait már eleve úgy tervezték, hogy ismerjék az XSD-t is. Azaz mai fejlesztésekhez mindenképpen ez a megfelelő sémanyelv.
Nézzünk most meg egy xml dokumentum példányt, majd írjunk hozzá sémát!

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<konyv isbn="0836217462">
 <cim>Kutyavilág</cim>
 <szerzo>Charles M. Schulz</szerzo>
 <szereplo>
  <nev>Snoopy</nev>
  <baratja>Peppermint Patty</baratja>
  <szuletett>1950-10-04</szuletett>
  <jellemzes>extrovertált beagle</jellemzes>
 </szereplo>
 <szereplo>
  <nev>Peppermint Patty</nev>
  <szuletett>1966-08-22</szuletett>
  <jellemzes>kövér, vakmerő leányzó</jellemzes>
 </szereplo>
</konyv>
```

A séma megírásához egyszerűen végigmegyünk az összes elemen, és definiáljuk őket. Előtte azonban az xsd:schema elemmel kell indítanunk:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

A schema elem a séma dokumentum gyökéreleme, amelyben az xsd prefixhez kötjük a séma aktuális névterét. Ennek az elemnek lesz még számos attribútuma, amely a teljes séma értelmezését befolyásolja, de erről egy kicsit később.
A konyv elemünkhöz definiálnunk kell egy elemet, amelynek neve konyv. Ennek az elemnek vannak gyermekelemei és attribútumai, ezért őt az xml sémában komplex típusnak tekintjük (complex type). A gyermekelemeket a sequence elemen belül definiálhatjuk:

```
<xsd:element name="konyv">
 <xsd:complexType>
  <xsd:sequence>
```

A sequence elem egy compositor, és azt írja elő, hogy a benne felsorolt gyermekelemeknek pontosan a megadott sorrendben kell szerepelni az xml dokumentumpéldányokban. További két compositor is van, a choice és az all. Ezekre még visszatérünk. Jöhet a cím és a szerző, ezeknek nincs gyermekeleme vagy attribútuma, így ők egyszerű típusok, simple type-ok.

```
<xsd:element name="cim" type="xsd:string"/>
<xsd:element name="szerzo" type="xsd:string"/>
```

A type attribútumban írjuk elő az egyszerű típusok adattípusát. Az xsd prefix azt jelzi, hogy a séma szabványban definiált string típust szeretnénk használni.
A szereplo elem egy kicsit huncutabb, mert abból minimum egy, maximum akárhány darab is lehet. Mivel vannak gyermekelemei, egyértelmű, hogy komplex típus lesz, a számosságot pedig a elemdefinícióban tudjuk jelezni:

```
<xsd:element name="szereplo"
 minOccurs="1" maxOccurs="unbounded">
 <xsd:complexType>
  <xsd:sequence>
```

A minOccurs írja elő legalább hány példány kell az elemből, a maxOccurs pedig hogy legfeljebb mennyi. Az unbounded a végtelent jelöli. Mindkettő alapértelmezett értéke 1, azaz ha nem írjuk ki őket pontosan egy elemet kell a jelzett helyen találnunk. Ezek után jöhetnek a gyermekelemek:

```
<xsd:element name="nev" type="xsd:string"/>
<xsd:element name="baratja" type="xsd:string"
 minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="szuletett" type="xsd:date"/>
<xsd:element name="jellemzes"
 type="xsd:string"/>
```

Látható, hogy egy szereplőnek több *(vagy akár semennyi)* barátja is lehet, amit nem lehetett kiolvasni az xml mintánkból, ezt csak az tudhatja, aki ismeri az adatok logikáját, és ezért kell a séma egy mintapélda helyett.
A szuletett elem típusa xsd:date, azaz ebben az elemben csak érvényes dátumokat reprezentáló sztringeket lehet megadni.
A *szereplo* leírás kész, zárjuk le a megkezdett elemeket

```
  </xsd:sequence>
 </xsd:complexType>
</xsd:element> <!-- szereplo -->
```

A *konyv*ben található gyermekelemek leírása is kész, zárhatjuk a sort:

```
        </xsd:sequence>
```

Most jön az isbn attribútum. Az attribútumdeklarációknak kötelezően a gyermekelemek után kell szerepelni.

```
<xsd:attribute name="isbn" type="xsd:string"/>
```

Majd zárjuk a konyv elemet leíró komplex típust, magát az elemdeklarációt és a teljes sémát is:

```
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Készen is vagyunk! Ez volt az „orosz Matrjoska baba" tervezés, amelyben a séma írása közben pontosan követtük a leírandó dokumentum szerkezetét.

Az áttekintés kedvéért íme a teljes séma:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<-- konyvsema1.xsd -->
<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="konyv">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="cim" type="xsd:string"/>
    <xsd:element name="szerzo"
     type="xsd:string"/>
    <xsd:element name="szereplo"
     min0ccurs="1" max0ccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="nev"
        type="xsd:string"/>
       <xsd:element name="baratja"
        type="xsd:string" min0ccurs="0"
        max0ccurs="unbounded"/>
       <xsd:element name="szuletett"
        type="xsd:date"/>
       <xsd:element name="jellemzes"
        type="xsd:string"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element> <!-- szereplo -->
   </xsd:sequence>
   <xsd:attribute name="isbn"
    type="xsd:string"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Második nekifutás - modularizálunk

Az előző „nekirugaszkodunk és egyszuszra megírjuk" módszer bonyolultabb struktúrák esetén gyorsan áttekinthetetlenné válhat, gyorsan túlnyúlik a jobb oldal a képernyőn. Ennél laposabb sémát hozhatunk létre, ha kihasználjuk, hogy az előre definiált elemekre hivatkozhatunk a struktúra kívánatos pontján.

Ebben a tervezési módszerben globálisan, a schema elem alatt felsorolunk minden elemet, attribútumot, és az összetett típusokban csak hivatkozunk *(reference)* a már deklarált tagokra:

```
<-- konyvsema2.xsd -->
<xsd:schema ...>
 <!-- egyszerű típusok definíciója -->
 <xsd:element name="cim" type="xsd:string"/>
 <xsd:element name="szerzo" type="xsd:string"/>
 <xsd:element name="nev" type="xsd:string"/>
 <xsd:element name="baratja" type="xsd:string"/>
 <xsd:element name="szuletett" type="xsd:date"/>
 <xsd:element name="jellemzes"
  type="xsd:string"/>
 <xsd:attribute name="isbn" type="xsd:string"/>
 <!-- komplex típusok -->
 <xsd:element name="szereplo">
  <xsd:complexType>
   <xsd:sequence>
   <!-- az egyszerű típusokra a ref attribútumon
        keresztül hivatkozunk  -->
    <xsd:element ref="nev"/>
    <!-- a számosságot itt jelöljük ki, nem az
         elem definíciójánál -->
    <xsd:element ref="baratja"
     min0ccurs="0" max0ccurs="unbounded"/>
    <xsd:element ref="szuletett"/>
    <xsd:element ref="jellemzes"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="konyv">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="cim"/>
    <xsd:element ref="szerzo"/>
    <xsd:element ref="szereplo"
     min0ccurs="1" max0ccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute ref="isbn"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Ez az objektumorientált programozási elvekre emlékeztet, amelyben kis építőkockákból *(alaptípusok – egyszerű típusok)* építünk fel nagyobb blokkokat *(osztályok, struktúrák - összetett típusok)*. Felfoghatjuk úgy is, hogy felül létrehozzuk az elemek, attribútumok absztrakt leírását, a komplex típusokban pedig példányosítjuk őket, konkrét elfordulásokat hozunk létre belőlük.

Típusos megközelítés

A harmadik sémaépítési módszerünkben összetett és egyszerű adattípusokat definiálunk, és ezek felhasználásával építjük fel az elemeket és attribútumokat.

A módszer hasonló lesz az előzőhöz, csak most nem kész elemeket és attribútumokat hozunk létre, hanem a típusdefiníciókat megnevezzük, és ezekre hivatkozunk az elemekben és attribútumokban.

Eddig csak komplex típusokat használtunk, azonban az XSD lehetőséget biztosít egyszerű típusok definiálására is. Az egyszerű típusok őseit a sémában definiált alaptípusok adják. Összesen 44-en vannak, az ábrán az anySimpleType leszármazottjai *(ld. következő kép)*.

A beépített típusokból származtatjuk le saját egyszerű típusainkat listagenerálással, megszorítással vagy unióképzéssel.
A listagenerálással képzett típus az alaptípuspéldányok white-space-ekkel elválasztott listája.

```
<xsd:simpleType name="nevnapok">
  <xsd:list itemType="xs:date">
</xsd:simpleType>
```

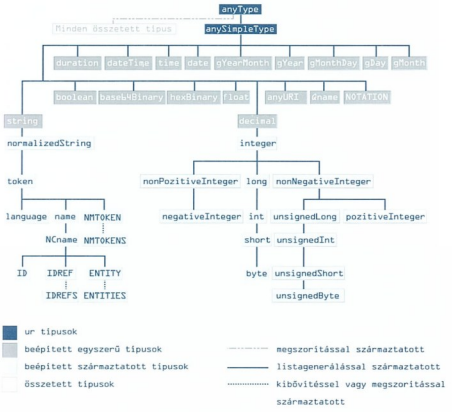
Egyféle ennek megfelelő tartalom *(a szóköz és az újsor is white-space)*:

```
2002-03-22 2002-04-10
2001-01-05
```

Jóval gyakoribb a megszorítással képzett típus, amelyben az alaptípus érvényes értékeit korlátozzuk le.

```
<xsd:simpleType name="nevTipus">
 <xsd:restriction base="xsd:string">
  <xsd:maxLength value="32"/>
 </xsd:restriction>
</xsd:simpleType>
```

Ebben a példában a netTípus névvel ellátott egyszerű típus a string séma alaptípusból származik megszorítással *(restriction)*, ahol a szöveg hosszát legfeljebb 32 karakterre korlátoztuk le.



Az XML Schema típusdefiníció hierarchiája

A megszorításokat előre definiált facetekkel írhatjuk le, ilyen volt a maxLength is. Az alábbi táblázatban összefoglaltam a többi facetet is.

Facet	Jelentés
length	Az adott adattípuson értelmezhető alapegységek száma. Például egy string esetén a karakterek száma.
minLength	Minimális hossz.
maxLength	Maximális hossz.
pattern	Reguláris kifejezéssel megfogalmazott kötöttség, amellyel az adott típus szöveges reprezentációját tudjuk megfogni.
enumeration	Az alaptípus értékeit korlátozza le egy véges halmazra.
maxInclusive	Felső határ, beleértve a megadott értéket is.
maxExclusive	Felső határ, a megadott értéket már kizárva.
minInclusive	Alsó határ, beleértve a megadott értéket is.
minExclusive	Alsó határ, a megadott értéket már kizárva.
totalDigits	Egy decimális típusból leszármazott típus számjegyeinek maximális száma *(egész + törtrész)*.
fractionDigits	Egy decimális típusból leszármazott típus törtrészében a számjegyek maximális száma.

A könyvek isbn számai pontosan tíz decimális számjegyből állnak, ezt az alábbi módon lehet megfogalmazni a pattern facet és benne reguláris kifejezések használatával:

```
<xsd:simpleType name="isbnTipus">
 <xsd:restriction base="xsd:string">
  <xsd:pattern value="[0-9]{10}"/>
 </xsd:restriction>
</xsd:simpleType>
```

Az enumeration típusra egy példa:

```
<xsd:simpleType name="szin">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="barna"/>
  <xsd:enumeration value="fekete"/>
  <xsd:enumeration value="szoke"/>
 </xsd:restriction>
</xsd:simpleType>
```

Az unióképzéssel létrehozott típusok alaptípusok és/vagy általunk definiált egyszerű típusok összessége:

```
<xs:simpleType name="variant">
   <xs:union   memberTypes="xsd:float   xsd:integer
xsd:string xsd:dateTime xsd:boolean xsd:long "/>
</xs:simpleType>
```

Az így létrehozott típusunk erősen emlékeztet a Visual Basic variant típusára, ami sokféle adattípust képes leírni.

folytatjuk...

Soczó Zsolt
Zsolt.Soczo@netacademia.net

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

Az Exchange Server felügyelete

Ismerkedjünk meg az Exchange 2000 felügyeleti eszközeivel, valamint nézzük meg az Active Directory Users and Computers snap-inbe integrált Exchange varázslókat!

Mire érdemes még odafigyelni a telepítésnél?

Ha Kedves Olvasóink elmélyedtek a ForestPrep és DomainPrep Exchange telepítési lépések rejtelmeiben, a kiszolgáló telepítése már gyerekjátéknak tűnik. Mire figyeljünk mégis oda? Hogy legyen elég helyünk a telepítéshez, illetve a kiszolgáló legalább a minimális követelményeknek megfeleljen.

- Legyen legalább 256MB RAM a gépben, és a page file legyen legalább kétszer akkora, mint a RAM tényleges mennyisége.
- NTFS partícióra telepítsük az Exchange minden részét
- Ami a helyfoglalást illeti, az Exchange – bárhová is telepítjük a kiszolgálón – a rendszerpartíción 200MB körüli helyet felemészt.
- Ideális, ha a Exchsrvr könyvtár nem a rendszerpartícióra kerül, hanem attól külön, lehetőleg hibatűrő lemezekre. A minimális hely, amivel érdemes elindulni kb. 500 MB.

Van még egy dolog, amit a telepítés előtt érdemes megfontolni, ez pedig az Exchange számára a rendszerfelügyeleti környezet ***(Administrative Groups)*** kialakítása. Az Administrative Groups arra való, hogy az Exchange kiszolgálókat üzemeltetési, felügyeleti, és bizonyos beállítások szempontból csoportosítsuk. Az első Exchange 2000 telepítésénél nem tudjuk megadni, hová is kerüljön az Exchange kiszolgáló, automatikusan a ***First Administrative Group*** nevű csoportba kerül. Ezt megelőzhetjük úgy, ha az első kiszolgáló telepítése előtt – a forestprep után – az Exchange CD-ről felrakjuk az Exchange 2000 System Manager-t, és ott létrehozzuk a megfelelő nevű Administrative Group-ot, majd csak ezután telepítjük az első Exchange 2000-et. Egy AD erdőben több Administrative Group-ot lehet létrehozni, de a kiszolgálókat utólag nem tudjuk az egyik csoportból a másikba átrakni. Sajnos.

Az Exchange System Manager nem az Exchange kiszolgálóhoz, hanem az AD-hoz csatlakozik!

Az Exchange rendszergazda eszközei

Mielőtt nekiesnénk felhasználókat, nyilvános mappákat létrehozni, áttekintjük az Exchange beállításához és a mindennapi Exchange adminisztrációs feladatokhoz használatos programokat. Megnézzük, melyik eszköz mire is való tulajdonképpen, mit és hol találhatunk bennük. Lehet, hogy sokan ezt feleslegesnek érzik, mert úgy gondolják, a gyakorlat teszi a mestert, azonban nem árt tudni mit, hol és miért ott találjuk.

Az Exchange előző verzióinál nagyjából egyetlen grafikus interfésszel bíró eszköz, és pár parancssori segédprogram volt, amivel az Exchange összes nyűgjét lehetett orvosolni. Ez a helyzet megváltozott, ami főként az Exchange és az Active Directory igen bensőséges kapcsolatából adódik. Egy dologban a legtöbb Exchange 2000 eszköz megegyezik: mindegyik egy-egy MMC modul – természetesen a parancssori programok kivételével.

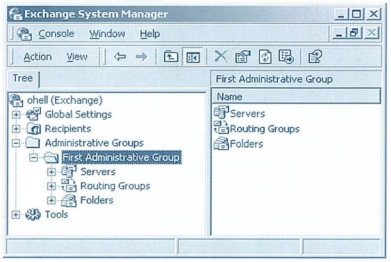
A System Manager

A Start menüben ugyan ezen a néven találkozhatunk vele, de az MMC modul neve Exchange System. Nagyon fontos tudnunk, hogy ez az eszköz az Active Directoryban matat, annak is a Configuration partíciójában.

Ha elindítjuk a Start Menüből, akkor az első, útjába akadó tartományvezérlőhöz próbál csatlakozni, azaz az ugyanazon az alhálózaton *(subneten)* levő tartományvezérlőt keres; ha ilyet nem talál, akkor a saját telephelyen *(Site)* levő tartományvezérlőhöz próbál csatlakozni. Az Exchange System Manager nem az Exchange kiszolgálóhoz, hanem az AD-hoz csatlakozik! Értelemszerűen, ha az Exchange kiszolgáló egyben tartományvezérlő is, akkor ahhoz kapcsolódik, de ez nem jelent semmit, hisz az AD-ból fogja lekérdezni az Exchange beállításokat. Ha valaki meg szeretné határozni pontosan, hogy a System Manager melyik tartományvezérlőhöz csatlakozzon, akkor indítsa el az MMC-t a Futtatás *(Run)* panelből, majd adja hozzá az Exchange System modult az MMC konzolhoz. A folyamat során az egyik ablakban meg lehet adni a tartományvezérlő nevét, amihez azután majd csatlakozik a modul.

A System Manager alapértelmezett nézete

Eltérő lehet, hogy mit is látunk itt sorban, attól függően, hogy az Exchange adminisztrációs csoportokat láthatóvá tesszük vagy sem. Ha a legfelsőbb objektum tulajdonságait előhúzzuk, ott be tudjuk állítani a nézetet, azaz hogy látszódjanak-e az adminisztrációs és útválasztó csoportok, vagy sem.

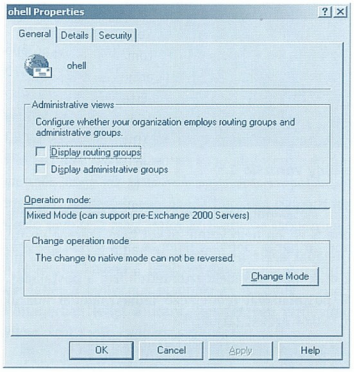


Administrative Group-os nézet

A második nézetben annyit változik a helyzet, hogy a beállítások más csoportosításban jelennek meg, vagyis az egy adminisztrációs egységhez tartozó beállításokat egy helyre gyűjti össze. Ha a képen lenne több ilyen csoport, akkor azalatt is ezeket a pontokat látnánk. Ennek a nézetnek igazán csak akkor van értelme, ha több ilyen adminisztrációs csoport létezik egy szervezeten belül.

A System Manager menürendszere

A Szervezet ikonja – a legmagasabb szintű objektum a System Managerben. Ha ennek a tulajdonságablakát előhúzzuk *(Jobb klikk – Properties)*, ezt látjuk:



Tehát itt állíthajuk be a System Manager nézetét, erről már volt szó, valamint az Exchange üzemmódját, abból a szempontból, hogy képes-e együttélni korábbi verziójú Exchange kiszolgálókkal, vagy sem. Ez más mint a Windows 2000 Active Directory üzemmódja, abban azonban hasonlítnak, hogy a változtatás itt is csak egyirányú lehet: ha egyszer feladjuk az együttélés lehetőségét és átváltunk úgynevezett natív módra, nincs visszaút.
A képen látszik a Security fül is, ami alapértelmezésben nincs ott. Általában nem is szükséges, de vannak helyzetek amikor hasznos lehet, azonban ezzel óvatosan kell bánni, mert amit itt beállítunk, öröklődni fog az exchange összes objektumára. Ahhoz hogy megjelenjen a Security fül tegyük be a ShowSecurityPage=1 értéket DWORD típusban a következő registry kulcs alá:

```
HKEY_CURRENT_USER\Software\Microsoft\
↳ Exchange\EXAdmin
```

Egyébként az ADSI Editet használhatjuk a szervezeti szintű jogosultságok beállítására. Az Exchange jogosultságai izgalmas téma, külön cikket szentelünk neki, ezért most nem szólnék többet erről, nézzük inkább tovább a System Managert.

Global settings – a nevében is benne van, itt a teljes szervezetet érintő beállításokat találjuk. Itt *(is)* meg lehet határozni a be és kimenő levelek méretét. Jó tudni azonban, ha a bejövő levelek méretéhez beállítunk korlátozást, az még nem garantálja, hogy a túlméretezett levelek nem fognak bejutni. Ugyanis ha a küldő fél nem veszi tudomásul ezt a beállítást, akkor bizony be fognak esni a nagyméretű levelek is. A Global Settings alatt található még a kimenő levélformátumok beállításai is, valamint itt tudjunk szabályozni, hogy mely e-mail címekről ne jöhessenek be levelek.

Recipients – itt találhatóak a postafiókokhoz kapcsolódva a címlisták beállításai, a postafiókokhoz rendelhető szabályok, például az e-mail címek. Ezekről egy későbbi cikkben lesz szó részletesen.

Servers – itt találhatóak az Exchange kiszolgálókhoz közvetlenül rendelhető beállítási lehetőségek, kommunikációs protokollok, valamint az Exchange adatbázisok beállításai.

Routing Groups – egy routing csoportba tartoznak azok az Exchange kiszolgálók, amelyek megbízható kapcsolattal rendelkeznek, pl. egy helyi hálózaton vannak. Egy szervezetben több routing csoport is lehet, de egy kiszolgáló egyszerre csak egyben szerepelhet. Itt lehet beállítani a különböző csatolókat *(Connectors)* más üzenetkezelő rendszerek felé.

Folders – itt találjuk a különböző ún. tárolócsoportokhoz tartozó adatbázisokat. Ugyanaz lelhető itt is fel mint a kiszolgálói objektumok alatt, csak más csoportosításban.

Tools – ide tartozik az Exchange 5.5-ről való áttérésnél használatos replikációs szolgáltatás, valamint innen meghívható egy MMC modul, ahol az üzenetek nyomonkövetését lehet beállítani.

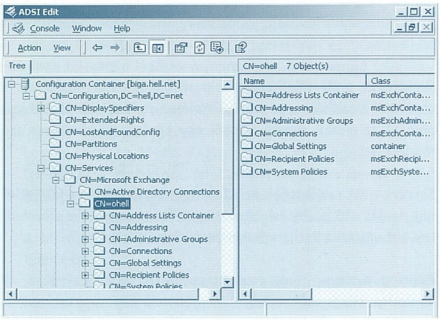
Végignéztük az összes menüpontot, és mégsem leltük a felhasználói postafiókok beállításait. Persze hogy nem, mert azok az Active Directory Users and Computersben találhatóak. Újabb bizonyítéka annak, hogy nincs Exchange 2000 Active Directory nélkül. Mielőtt azonban rátérnék a felhasználók dolgaira, kitérőt teszünk, és megnézzük az Exchange beállításait egy másfajta nézetből.

ADSI Edit – AD Service Interface Editor

Fontosság szempontjából a rendszergazdai eszközök sorában talán a harmadik az ADSI Edit, amellyel hogy úgy mondjam „hátulról" láthatjuk ez Exchange összes beállítását. Sőt, nemcsak az Exchange beállításokat láthatjuk, hanem az Active Directoryban fellelhető összes objektumot. Leginkább az előző verziós Exchange Administrator RAW módjához lehetne hasonlítani az ADSIEdit használatát az Exchange 2000 vonatkozásában.
Az ADSIEdit a Windows 2000 Support Tools része. Ahhoz hogy használni tudjuk, elegendő a Windows 2000 Server CD-ről **Support\Tools** könyvtárból a **support.cab** fájlból az **adsiedit.dll**-t kimásolni mondjuk a %systemroot%\system32 könyvtárba és regsvr32 adsiedit.dll parancsot kiadva regisztrálni a dll-t. Ezután MMC modulként hozzá tudjuk adni a konzolhoz. Az ADSI Edit szin-

tén NEM az Exchange kiszolgálóhoz csatlakozik közvetlenül, hanem a legközelebbi, vagy egy meghatározott tartományvezérlőhöz.

Az Exchange konfigurációs beállításait az Active Directory konfigurációs konténerében *(Configuration Container)* a CN=Configuration – CN=Services – CN=Microsoft Exchange alatt találhatjuk. Ezen belül van – ahogy az alábbi képen is – az Exchange szervezet *(Organization)*, és természetesen a beállítások, amiket normális esetben az Exchange System Managerből állítgatunk. Biztonsági beállításokat is lehet ilyen módon állítani, alapértelmezésben a Organization és az Administrative Groups szintjén csak itt lehet állítani a jogosultságokat, de azt már láttuk a cikk elején is, hogy ez kikerülhető.



Exchange beállítások az ADSI Editben

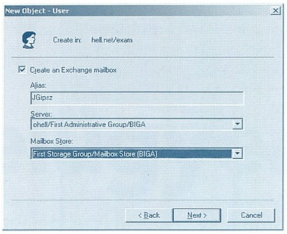
Számos esetet lehetne itt most felsorolni, hogy mikor használatos az ADSI Edit a beállításokhoz, általában akkor húzzuk elő, ha nincs mód, hogy pl. a System Managerből vagy az Active Directory Users and Computerből állítsunk be valamit. Például mindenki képes ún. top level nyilvános mappát létrehozni, amit úgy is meg tudunk akadályozni, illetve szabályozni, ha a CN=Microsoft Exchange – CN=<szervezeti név> tulajdonságait előhúzva, a Security fülön az Everyone csoporttól megvonjuk a 'Create top level public folder' jogot, illetve a megfelelő csoporthoz hozzárendeljük.

Active Directory Users and Computers

Egy másik igen fontos eszköz az AD Users and Computers MMC modul, hisz ez használatos az Exchange felhasználók, a disztribúciós listák és a kapcsolatok *(Contacts)* karbantartására is.

Kétféle felhasználó lehet az Exchange-ben. Az egyik, akinek az Active Directoryban felhasználói neve, postaládája és e-mail címe is van, ezt hívjuk mailbox-enabled felhasználónak. A másik, akinek szintén van az AD-ban felhasználói neve, de nincs postaládája a helyi Exchange rendszerben, hanem egy külső e-mail címmel rendelkezik *(így szerepel a címlistákban is)*: ő a mail-enabled felhasználó.

Kétféleképpen is létrehozhatunk Mailbox-enabled felhasználót: ha nincs a felhasználónak azonosítója a tartományban, akkor egy menetben létrehozhatjuk a felhasználói accountot és a postaládát is, ilyenkor a szokásos felhasználó létrehozásához szükséges kérdések után jön ez az ablak:

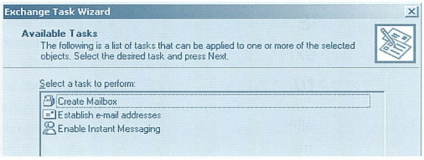


Gipsz Jakab mailbox-enabled felhasználó létrehozása

Meg lehet adni, hogy mely adminisztrációs egységbe tartozó, azon belül is melyik Exchange kiszolgálón és melyik adatbázis-csoportba szeretnénk rakni a postaládáját.

Ha már az AD-ban létező felhasználónak szeretnénk létrehozni postaládát, akkor az Exchange Task Wizardot, vagyis Exchange Feladatvarázslóját lehet segítségül hívni, a felhasználói objektumon jobb klikk – Exchange Tasks...-ra kattintva.

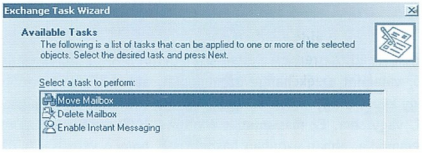
A varázsló mindig csak az adott objektumhoz *(felhasználó/csoport/kapcsolat)* illetve a környezetben végrehajtható változtatásokhoz ajánl fel lehetőségeket. Például egy olyan felhasználónál. akinek nincs postaládája még az Exchange-ben, a következőket ajánlja fel:



Helyzetérzékeny varázsló 1.

Az Enable Instant Messaging azért szerepel, mert fel van telepítve az a komponens is; ha nem volna, akkor nem szerepelne itt. Move mailbox feladat azért nem szerepel, mert még nincs mit mozgatni, hisz csak most szándékozzuk létrehozni a postaládát.

Ha egy létező, postaládával rendelkező felhasználónál húzzuk elő a varázslót, akkor ezeket látjuk:



Helyzetérzékeny varázsló 2.

Ha csak mail-enabled felhasználót vagy kapcsolatot szeretnénk létrehozni, akkor használatos az Establish e-mail address feladat. Mail-enabled felhasználót nem tudunk egy menetben létrehozni, mert a felhasználó létrehozásánál a postaládára kérdez rá a varázsló, ezért először létre kell hozni egy „natúr" felhasználót az AD-ban, és utána lehet e-mail címmel ellátni, így lesz mail-enabled.

A mail-enabled felhasználó nem összekeverendő a Kapcsolat *(Contact)* objektummal, ugyanis az utóbbinak nincs az AD-ban felhasználó azonosítója, így Exchange postaládája sincs, csupán egy külső e-mail címmel rendelkező objektum az AD-ban, amelynek címe látható a címlistában.

Természetesen ha egy új Kapcsolatot hozunk létre *(az adott OU-*

ban jobb klikk – New – Contact) akkor a varázsló rákérdez, hogy szeretnénk-e e-mail címet hozzárendelni, majd kell választani egy e-mail cím típust *(például SMTP, Notes, GroupWise, cc:Mail, vagy X.400)*. Meg kell adni a típusnak megfelelő e-mail címet, sőt be lehet állítani a feljövő ablak Advanced fülén, hogy milyen formában menjen a levél az adott címre. (MIME vagy UUEncode/Plain Text formátumban.)

Nemcsak felhasználóknak, hanem csoportoknak is lehet e-mail címe. Az AD-ban kétféle csoport létezik a biztonság szempontjából. A biztonsági v. Security csoportokhoz jogosultságok is rendelhetők, míg a disztribúciós csoportok csak üzenetek szétosztására valók. Az Exchange szempontjából mindegy, hogy milyen csoportról van szó, legyen az biztonsági vagy disztribúciós, ha beállítunk nekik e-mail címet, azaz mail-enabled csoportokká tesszük őket, akkor megjelennek a címlistákban, a csoport címére küldött leveleket a csoport minden olyan tagja megkapja, akinek van e-mail címe.

Meg kell adni a típusnak megfelelő e-mail címet, sőt be lehet állítani a feljövő ablak Advanced fülén, hogy milyen formában menjen a levél az adott címre. (MIME vagy UUEncode/Plain Text formátumban)

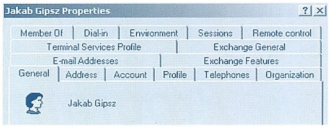
Csoport esetén egyszerűbb az Establish e-mail address varázsló, ott nem kérdez e-mail címet, csak egy Aliast kér, ami a @ előtt fog szerepelni az e-mail címben, a többit átveszi a központi beállításoktól.

Csoportok esetén értelmezett a Hide és Unhide Group membership feladat, ami értelemszerűen vagy eltünteni a kíváncsiskodók szeme elől, hogy kik tartoznak egy adott csoportba, vagy láthatóvá teszi azt. Alapértelmezésben láthatóak a csoporttagságok. Ez a varázsló nem tesz mást, mint az Everyone csoportnak explicite tiltja vagy visszaadja a Read jogát az adott csoportra vonatkozóan.

A teljesség kedvéért jegyzem meg, hogy e-mail címe ezeken kívül lehet a nyilvános mappáknak is, amit az Exchange System Managerben lehet állítani. Részletesen erről majd a nyilvános mappákról szóló cikkben szólunk.

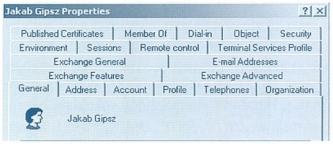
Felhasználói tulajdonságok

Egy postaládával rendelkező felhasználó tulajdonságablakai így néznek ki első megközelítésben:



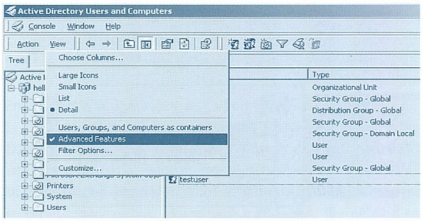
Egyszerűbb nézet

Ugyanennek a felhasználónak a tulajdonságablakai így is kinézhetnek:



Mintha több lehetőség lenne itt

Nem árulok el nagy titkot, ha azt mondom, ez azért van, mert az Active Directory Users and Computers konzol nézetét átállítottam.



Az Advanced Features nézet ki és bekapcsolása

Nem csak a felhasználóknál sokasodik meg az elérhető beállítások száma ilyenkor, hanem minden objektumon, valamint megjelennek extra konténerek is, amelyek addig rejtve voltak.

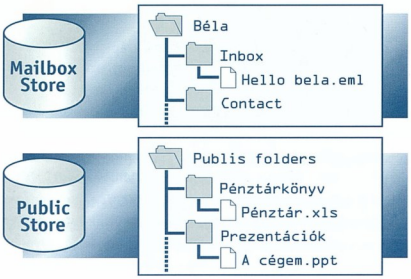
Folytatás következik a felhasználók beállításaival...

Dorner Csilla
MCSE

A Web Storage System
Programozott levélkezelés

Exchange sorozatunkhoz kapcsolódóan néhány cikk erejéig betekintünk a Web Storage System programozott elérésébe. Legelőször az ADO-n keresztüli hozzáférést vizsgáljuk meg.

A Web Storage System *(WSS)* az Exchange2000 Server új adattárolási mechanizmusa. Hatalmas előnye, hogy sokoldalú hozzáférést biztosít az adatokhoz *(HTTP, SMTP, IMAP4, POP3)*. Egyetlen szerveren egymástól függetlenül, méreteikre vonatkozó korlátozás nélkül több store is elhelyezkedhet *(nyilván a rendelkezésre álló tárterület határain belül...)*. A WSS felépítése hierarchikus: a különböző fájlok, dokumentumok mappákba szervezhetők, s ezek a mappák *(folderek)* tetszőleges mélységig egymásba ágyazhatók. E-maileket, szövegfájlokat, Office dokumentumokat, multimédiás fájlokat stb. egyaránt tárolhatunk a WSS-ben ugyanúgy, mint egy egyszerű fájlrendszerben.



A Web Storage System felépítése

Alapvetően két store típust különböztetünk meg: a Mailbox store-t és a Public store-t.
A Mailbox store olyan adatbázis, amelynek mappáihoz és dokumentumaihoz egyetlen felhasználó, a tulajdonos fér hozzá. E-maileket, hozzájuk csatolt állományokat, mappákat, dokumentumokat stb. tárolhatunk itt. Természetesen minden felhasználóhoz külön Mailbox store rendelhető.
A Public store legfőképpen abban különbözik a Mailbox store-tól, hogy nem egyetlen felhasználó kizárólagos tulajdona, megosztottan többen is hozzáférhetnek - persze a megfelelő jogosultságok alapján.

Keresés a WSS-ben

A WSS a keresést a leggyakrabban használt sémajellemzők *(schema property)* indexelésével optimalizálja. Ezek a mezők: Subject, Body, From és To. A WSS-t egyszerű SQL szintaxissal is elérhetjük, erről később, az ADO részben írok. Ezenkívül a tartalom szerinti indexelés *(content indexing)* és a full text keresés is támogatott, jelentős sebességnövekedéssel örvendeztetve meg a felhasználót. Mindez beépített lehetőség, nem igényel külön szoftvert, kódolást, álmatlanul töltött programozói éjszakákat. Mindezeket figyelembe véve elmondhatjuk, hogy a korábbi verziók keresési lehetőségei jelentősen javultak.
Nagyon hasznosak az úgynevezett keresőmappák *(search folder)*. Ezek olyan mappái a WSS-nek, amelyek korábbi keresések eredményeit tárolják, s később bármikor hozzáférhetőek.

A továbbiakban arra koncentrálunk, hogy saját kódból, programozott módon milyen lehetőségeink vannak a Web Storage System elérésére.

A WSS elérése ADO-val

Az alábbi ábra azt mutatja, hogy a Web Storage System hogyan érhető el ADO-n *(illetve CDO-n)* keresztül:

Az ExOLEDB provider kiszolgálóoldali komponens, melyen keresztül – egy COM vagy COM+ objektumba ágyazva, ASP-vel vagy más webalkalmazással – elérhetjük a kiszolgálón lévő adatokat. Ily módon akár a helyi, akár távoli tárolók is elérhetőek.
A következő forrásrészlet azt mutatja, hogy hogyan férhetünk hozzá a WSS állományaihoz a RecordSet objektum segítségével:

A WSS elérése ADO-val vagy CDO-val

```
Set conn = CreateObject("ADODB.Connection")
conn.Provider = "ExOLEDB.DataSource"
conn.Open "http://myserver/folder/"
Set rs = CreateObject("ADODB.Recordset")
strSQL = "SELECT ""DAV:displayname"",
↳""DAV:contentclass"" " & _
"FROM ""http://myserver/folder/"" " & _
"WHERE ""DAV:ishidden"" = FALSE " & _
"AND ""DAV:contentclass"" =
↳'urn:content-classes:message'"
rs.Open strSQL, conn
```

Létrehozzuk a connection objektumot, melyhez kijelöljük az ExOLEDB providert, és megnyitjuk a megfelelő kiszolgálóra. Ezzel létrejött a kapcsolat az Exchange Server megfelelő mappájával.
A CreateObject parancs segítségével létrehozunk egy RecordSet-et, majd egy egyszerű stringváltozóba betöltjük a lekérdezésnek megfelelő SQL mondatot. Azért éppen SQL, mert az ExOLEDB Provider ezt a szintaxist használja a WSS ADO-n keresztüli elérésére *(bővebben lásd az Exchange SDK-ban)*. A WSS jellemzőit

különböző névterekbe sorolva találhatjuk, ezek közül egy az általunk használt DAV:, amely a leggyakrabban használt jellemzőket tartalmazza. Néhány további névtér, a teljesség igénye nélkül:

Névtér	Leírás
urn:schemas:calendar	Naptárhoz rendelt jellemzők, például találkozó, évforduló stb.
urn:schemas:contacts	Egyénekhez rendelt jellemzők, például csoport, vagy szervezet neve
urn:schemas:httpmail	Az üzenetek törzsének leírására szolgáló jellemzők.
urn:schemas:mailheader	Az üzenetek fejrészének leírására szolgáló jellemzők
http://schemas.microsoft.com/exchange	Exchangespecifikus jellemzők.

Természetesen lehetőség van saját névterek definiálására is, például létrehozhatunk http://schemas.cégnév.hu/belso névteret, hogy az általunk újonnan felvett jellemzőket ott tároljuk. Erre olyankor lehet szükség, ha olyan leírókat kívánunk használni, amelyek eredetileg nincsenek a WSS-ben, például az alkalmazottaink fényképét, vagy gyermekeinek adatait, stb. is el kívánjuk tárolni a WSS-ben.

A fenti lekérdezés alapján az rs.Open parancs kiadásával visszakapjuk azokat a WSS-ben tárolt dokumentumokat, amelyek megfelelnek az általunk kiadott kritériumoknak. Példánkban a DAV: névtér displayname és contentclass jellemzőit *(property)* szeretnénk elérni azon e-mailek *(DAV:content-class='urn:content-classes: message')* esetében, amelyek nem rejtettek *(DAV:ishidden =FALSE)*. A message értéken kívül a content-class felveheti még appointment, person, item, calendarmessage stb. értékeket is, az urn:content-classes névtérből.
Jól látható, hogy a WSS-ben minden a jellemzőkön keresztül érhető el, ezek alapján szűrhetünk, kereshetünk, rendezhetünk, stb. Nagy előnye ennek a tárolásmódnak, hogy teljesen dinamikus, tehát, ha egy propertyhez nem rendelünk értéket, akkor az nem is jön létre. Viszont amint feltöltjük valamivel, az Exchange2000 elvégzi a helyfoglalást, létrehozza, stb. Természetesen ha kitöröljük az értékét, azonnal megszűnik a property létezése is.
Ezek után lássuk, hogyan listázhatjuk ki egy adott dokumentum jellemzőit:

```
rs.Move sorszam
Dim fld
for each fld in rs.Fields
Response.Write (fld.name & " = " &
fld.value & VbCrLf)
next
```

Az első sorban a RecordSet megfelelő elemére visszük a kurzort *(feltételezzük, hogy a sorszam nevű változó egy olyan egész típusú értéket tárol, amely ebben a környezetben értelmezhető)*, majd egy for each ciklusban megyünk végig az elem minden egyes jellemzőjén, és kííratjuk annak nevét és értékét.
Ugyanezt az eredményt érhetjük el akkor is, ha már a végrehajtás legelején tudjuk, mely dokumentumra lesz szükségünk, és RecordSet helyett egyetlen Record-ot hozunk létre. Figyeljük meg, hogy ebben az esetben közvetlenül rámutatunk az Exchange szokásos WebDAV URL-ének segítségével egy adott levélre, és ez kerül be az ADODB.Record objektumba:

```
Set conn = CreateObject("ADODB.Connection")
conn.Provider = "Ex0LEDB.DataSource"
conn.Open "http://myserver/folder/"
Set rec = CreateObject("ADODB.Record")
rec.Open "http://myserver/folder/level.eml", conn
```

Ennyi volt a móka mára...

És hogy mire jó mindez, azon kívül, hogy szép és jó, valamint számos átizzadt éjszakával képes megajándékozni azt, aki közelebbről is próbálja megismerni? A levelezésen, a levelek saját kódból történő kezelésén felül gyakorlatilag minden tárolható, kereshető, indexelhető, kezelhető a WSS-ben, így például céges nyilvántartásokat, adatbázisokat készíthetünk a dokumentációkból, bemutatókból, levelezésekből, mindenből, melyhez saját, az átlagos felhasználók számára is könnyen elsajátítható, használható, barátságos és a sokszínű igényeknek megfelelő kezelői felületet írhatunk.

A következőkben a Web Storage System WebDAV-on, illetve a .NET-es WebRequest-en és WebResponse-on keresztül történő elérését ismertetem majd, hogy az új technológiák hívei is megismerhessék a Web Storage System rejtelmeit.

Molnár Ágnes
agnes.molnar@dataware.debis.hu

A cikkben szereplő URL-ek:
[1] Exchange SDK (2002. március):
http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/834/msdncompositedoc.xml

Patchwork

Adminisztráljunk Windows 2000-et Windows XP Professional kliensről

Hódít a Windows XP. Ön is lecserélte már saját munkaállomásán régi, Windows 2000 Professional operációs rendszerét ugye? Hogyan lehet XP-ről távfelügyelni a Windows 2000 kiszolgálókat? Terminal ablakból. És még? Az új ADMINPAK.MSI segítségével!

A Windows XP térhódításával egyre gyakoribb, hogy a klienseken XP, míg a szervergépen Windows 2000 van telepítve. Ekkor rögtön felvetődik az az igény, hogy az eddig megszokott módon távolról adminisztrálhassuk a szerver beállításait.

Ezt megtehetjük Terminal Services segítségével, melyet Remote Administration módban telepítünk egy Windows 2000 Server operációs rendszert futtatató gépre, ez azonban csak 2 kapcsolódást tesz lehetővé. Megoldás jelent a Windows .NET Server Administration Tool Pack *(Adminpak.msi)* béta verziójának telepítése is.

Az Adminpak.msi önkicsomagoló fájl tartalmazza a leggyakrabban használt eszközöket, melyeket Windows 2000 és Windows .NET Server operációs rendszereket futtató számítógépek helyi és távoli adminisztrációjára használhatunk hozzájuk kapcsolódó kliensekről.

Ha adott egy Windows 2000 operációs rendszerrel rendelkező gép *(melyre korábban Windows 2000 Adminpak.msi-t telepítettünk)*, s ezek után upgradeljük az oprendszert Windows XP-re vagy Windows .NET Serverre, a telepítés folyamán a rendszer figyelmeztet minket, hogy a Windows 2000 Administration Tools nem kompatibilis az új operációs rendszerrel: távolítsuk el, és használjunk Terminal Services-t vagy telepítsük a Windows .NET Servers Adminpak béta verzióját. Ha mégsem távolítjuk el a fent említett eszközöket és megpróbáljuk őket használni, azt tapasztaljuk, hogy el sem indulnak, vagy nem működnek megfelelően.

A Windows 2000 Adminpak.msi utólag nem telepíthető Windows XP kliensre: megmakacsolja magát. Manapság a Windows XP Professionalt futtató gépekre csak az Adminpak.msi Windows .NET Server Beta 3 verziója telepíthető. A Windows XP Professional sajnos nem tartalmazza az előbb említett fájlt, mivel még nem volt elérhető az XP piacra kerülésekor.

A Windows .NET Server adminisztációs eszközeinek többsége hasonlóan működik, mint amit a Windows 2000-ben megszoktunk. Néhány esetben azonban ezek az eszközök nagyobb funkcionalitással rendelkeznek, viszont bizonyos esetekben a kompatibilitás és támogatottság hiányáról beszélhetünk mind a .NET Server eszközökkel történő Windows 2000 rendszer adminisztrálásakor, mind fordított esetben.

A Windows .NET Server Administration Tools funkcióinak többsége lehet, hogy nem engedélyezett vagy támogatott, amikor Windows 2000 operációs rendszerrel rendelkező számítógépet adminisztrálunk vele. Például a „saved query for last logon time" funkció nem támogatott a Windows 2000 alapú számítógépeken, mivel ezt a parancsot a Windows 2000 Active Directory nem ismeri. A legtöbb esetben ezek a fejlettebb eszközök nem csak nem engedélyezettek, de nem is láthatók, amikor az adminisztrációs eszközöket Windows 2000 rendszerekre irányítjuk.

Manapság a Windows XP Professionalt futtató gépekre csak az Adminpak.msi Windows .NET Server Beta 3 verziója telepíthető

A Windows .NET Server Adminpak nem telepíthető Windows 2000-re, és nem is fut ezen az operációs rendszeren. Parancsori eszközeit sem érdemes felmásolni, mivel Windows 2000 alatt azok sem működnek.

A Windows .NET Server Adminpakot kimondottan Windows XP-re és Windows .NET Serverre tervezték.

A Windows XP-k közül csak a Professionalra telepíthetjük a Windows .NET Server Adminpak.msi béta 3-as verzióját. Mivel a Windows XP Home Edition nem tud Windows NT 4.0-ra, Windows 2000-re vagy Windows .NET Serverre kapcsolódni, ezért ezen az operációs rendszeren nem is támogatott az adminpak telepítése.

Ahhoz, hogy Windows XP Professionalunkra telepíthessük a fent említett adminpakot, elég ha az **[1]** címről letöltjük az adminpak-.exe fájl, amely egy önkitömörítő állomány, tartalmazza az Adminpak-readme.txt és a B3-web-version-adminpak.msi fájlt. Telepítéshez pedig elegendő, ha duplán kattintunk az .msi fájlra.

Ne higyjük, hogy tökéletesen fognak működni a Windows .NET Server Administration eszközök, hisz még csak béta minőségűek. 2002 folyamán várható a végleges verzió. Ekkor a béta 3-at szedjük le gépünkről és telepítsük a végleges verziót.

Borsi Katalin
bobo@netacademia.net

A cikkben szereplő URL-ek:
[1] http://msdownload.netacademia.net

Egységben az erő

Egy átlagos kaliberű, komplett asztali számítógép ma megvásárolható mintegy 150 ezer Ft-ért. A legtöbb esetben – amikor egyáltalán be van kapcsolva – szövegszerkesztésre, játékra, internetezésre stb. használjuk, miközben a merevlemeznek a legjobb esetben is csak a töredékét tudjuk értelmes adatokkal feltölteni. Bob Metcalfe *(az Ethernet feltalálója)* 90%-ra saccolja a parlagon hagyott hardvererőforrások arányát. Nyugodtan higgyük el, és nyugodjunk bele, hogy 135 ezer Ft-ot kidobtunk az ablakon. Bár arra senkit sem biztatok, hogy várjon rá, a világháló horizontján már felbukkant a megoldást hozó világgép: a World Wide Computer.

Mivel ezt az írást a Scientific American **[1]** cikke inspirálta, hadd induljak el egy, éppen ennek a magazinnak több évvel ezelőtt megjelent, magyar kiadásában fellelhető érdekes cikke nyomán: kutatók számítógépes szimulációval kimutatták, hogy minél több demokratikus ország lesz a világban, annál kisebb egy háború kitörésének valószínűsége. A demokratikus országok ugyanis két- és többoldalú szerződésekkel jótékony béklyóba kötik magukat...

Mi a világgép? A válaszhoz vegyünk egy végtelenül leegyszerűsített, négyszereplős világot: ÉN, a felhasználó, Ők, a többi felhasználó, a koordinációs szervezet *(K)* és a Profit Rt *(P)*. Jelentkezem K-nál, hogy bérbeadnék tárhelyet *(mert már csak 20 gigás winyót lehet venni, míg nekem 5-ös is elég lenne)*, és programok futtatását is engedélyezném *(mivel legtöbbször úgyis csak a wördöt és a bullbrózert használom)*. Regisztrálok, miegymás, telepítek egy programot, amolyan ügynököt, amely leveszi a vállamról a gondot, és a gépemet érintő üzleteket nyélbe üti. Ők ugyanezt tették, teszik és fogják tenni. P látja, hogy ez jó, mert nem kell neki megvenni a böszme nagy kiszolgálókat, elég ha bérli az erőforrásokat K-tól. Kész a világgép, jöhet az eksön!

P egy alkalmazottja egy új rekordot akar beilleszteni a vállalati adatbázisba. A gépén lévő ügynök belekukkant a világgép elosztott adatbázisába, és kiszúr magának néhány online gépet, amelyeken van hely. ÉN is köztük vagyok, pénz áll a házhoz :-) P ügynöke a rekordból valamilyen algoritmus szerint több apró darabkát készít, és mindegyiket gondosan titkosítja, nehogy Ők bele tudjanak nézni *(ÉN persze sosem tennék ilyet)*. Aztán felveszi a kapcsolatot az ÉN és az Ők ügynökeivel, és mindegyik darabkát eltárolatja velük *(a biztonság kedvéért egy darabkát több gépen is)*. A virtuális számlámon jóváír egy adott összeget, és ekkor már ÉN is látom, hogy ez jó. Most, hogy így meggazdagodtam, megnézek egy jó filmet. A világgép sok-sok ügynöke lendül munkába a kedvemért, megkeresik a film soronkövetkező darabkáit az Ők számítógépein és rendületlenül küldik az én hűséges ügynökömnek. A közvetítési díjakból az ügynök, azaz K is jól megél.

Feltűnik a színen az 5. elem, az Internetmumus, a kódtörő Szabó néni, aki mellesleg P alkalmazottja, és álmatlan éjszakákat tölt főnöke nem ismert fizetése miatt. Otthoni gépével beáll a sorba, és reményei végül valóra válnak, nemsokára tényleg ott landol a gépén egy, a főnök fizetését leíró darabka. De micsoda bosszúság, csak az egyik, és az is titkosítva. Hol van a többi? Ekkor ráébred, az Ők gépein. Hogy melyikeken, az kemény dió, mert ez dinamikusan változhat annak függvényében, hogy éppen melyik van bekapcsolva, és az ügynökök sem ma jöttek le a falvédőről, tanulják egy ideje a matematikát. Egy kis biztatással talán segíthetünk: hajrá Szabó néni!

Ezen a bolton – Szabó nénit kivéve, aki sokat költhet majd pszichológusra – mindenki nyer! A végeredmény egy minden eddiginél megbízhatóbb, gyakorlatilag kimeríthetetlen tároló- és feldolgozórendszer. Ha egészében tekintünk erre a rendszerre, akkor egyetlen nagy számítógépet látunk, egy új paradigmát bevezető, világméretű operációs rendszerrel: ISOS – Internet-Scale Operating System. Egy ISOS munkába állíthatná az Internethez kapcsolódó 150 millió számítógép összes erőforrását *(ez a szám már évek óta hatványozottan növekszik)*, aminek eredményeként egy hihetetlenül nagy kapacitású, virtuális „mainframe" állna a közösség tagjai részére. A világgép mindemellett még önjavító is, hiszen ha elromlik a gépem, ÉN elviszem megjavíttatni.

Az ISOS kétféle alkalmazástípusnak kedvez: az elosztott adatfeldolgozásnak és az elosztott online szolgáltatásnak.

Az utóbbi kategóriába tartozik például a közismert SETI@home projekt, amely a fölösleges processzoridőt a földönkívüli, értelmes élet keresésének szolgálatába állítja, vagy a distributed.net RSA törögető alkalmazása stb. Azt azonban érdemes megemlíteni, hogy létrejöttüket szinte mindig a feldolgozásra váró, petabájtos nagyságrendű adatmennyiségnek köszönhetik.

A Farsite egy kiszolgáló nélküli, elosztott fájlrendszer

A Microsoft is megtette már az első lépéseket mindkét fentebb említett területen:

- A Farsite **[2]** egy kiszolgáló nélküli, elosztott fájlrendszer. A résztvevő kliensek között nincs kölcsönös (trust) kapcsolat, de mindegyikük hozzáfér a csak logikai síkon létező, központi állománykiszolgálóhoz. A Farsite globális, hierarchikus névterében könnyű a tájékozódás, nem tudunk, nem tudhatunk, és nem is akarunk tudni arról, valójában melyik kliens tárolja az adatainkat.
- Az elosztott online alkalmazások frontján a Pastry **[3]** áll csatasorba, amely a résztvevő gépekkel egy elosztott, méretezhető, önszervező és hibatűrő réteget biztosít a ***(peer-to-peer)*** társalkalmazások számára. Hatékonyan valósítja meg a Pastry gépek közötti útválasztást, a terheléselosztást és az objektumok helyének meghatározását.

A világgép e két lehetséges komponensét a soron következő két számban mutatom be részletesebben.

Zacco@fw.hu

A cikkben szereplő URL-ek:
[1] http://www.sciam.com/2002/0302issue/0302anderson.html
[2] http://research.microsoft.com/sn/farsite
[3] http://research.microsoft.com/~antr/pastry

Következő számunk tartalmából

Exchange:
Tovább mélyedünk az Exchange 2000 rendszerfelügyeleti feladataiba, elérkeztünk a felhasználók tulajdonságaihoz, megnézzük mit hol lehet beállítani. A felhasználók után a címlistákat fogjuk körüljárni, és ha futja erőmből, akkor a házirendekkel is foglalkozunk.

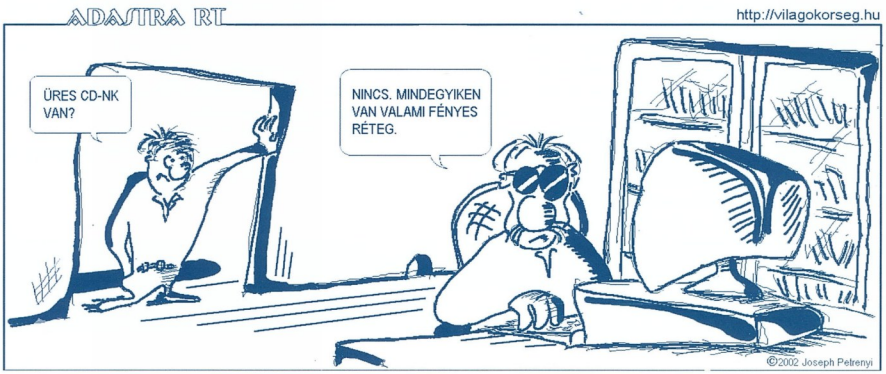
Jog:
Április 1-től az eddiginél jóval nehezebb helyzetbe kerülhetnek azok, akik az Internetet hátsó szándékkal használják. Ha valamilyen, a Büntető Törvénykönyvben meghatározott tényállást valósítanak meg, akkor adott esetben sokéves börtönbüntetéssel is számolhatnak. A korábban ismertek mellett új elkövetési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

ADO.NET
A tech.net magazin februári és márciusi számában XMLgessünk címen bemutattuk az ADO.NET adatbázis-kezelésének központi elemét, a DataSet objektumot. Egy memóriában tárolt adatbázis óriási kincs, de csak akkor, ha stabil technológiai kapoccsal összeköthető a meglévő adatbáziskiszolgálókkal és a helyi gépeken tárolt XML állományokkal. A 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

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

Farkasokkal táncoló
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. Két Microsoft terméket és egy harmadik gyártótól származó szoftvert vizsgálunk meg.

Biztonság
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...



Microsoft Project 2000
– hogy a tervekből valóság legyen!

A projektirányítás számos szervezet sikerében és bukásában játszik fontos szerepet. Egy hatékony tervezési eszköz birtokában csökkenthetők a költségek és javítható a termelékenység, ami nyereségnövekedést eredményez.

A Microsoft Project 2000 olyan egységes tervezőeszköz, mellyel a szervezet minden tagja egyszerűen megtervezheti, módosíthatja, nyomon követheti az adott projektet és annak valamennyi fázisában mérheti annak eredményességét.

Ismerje meg közelebbről tanfolyamainkon!

Microsoft Project 2000 és Project Central kurzusok

A Microsoft Project témakörben oktatóközpontunkban **kétnapos, intenzív** tanfolyam keretében ismerkedhet meg a program 2000-es verziójának használatával, haladók részére pedig szintén kétnapos Project Central kurzusnkat ajánljuk.

Rugalmas időbeosztás

Megpróbáltuk figyelembe venni, hogy a cégek dolgozóikat – különös tekintetettel a **vezető/felelős beosztású** személyekre – nem tudják nélkülözni hosszabb időre, ezért a tanfolyamokat **intenzív, egész napos formában** hirdettük meg, délelőtt 9.00 és 16.00 óra között. A tanfolyamok díja az oktatás és a tananyagok mellett az étkezést is tartalmazza a kurzus napjaira.

Testreszabott oktatások

Oktatóközpontunkon kívül, a megrendelő telephelyén, **kihelyezett formában** (vidéken is) is vállaljuk az **adott cégprojektekhez kapcsolódva** tanfolyamok vagy konzultációk megtartását, lebonyolítását. Ehhez igazodva készítjük el a tematikát és az oktatási segédletet, így az oktatás kedvezőbb időbeosztással és anyagi feltételekkel valósítható meg.

A képzésekkel, szakmai kérdésekkel kapcsolatosan kérjük keresse értékesítési vezetőnket, **Lovász Attilát** a 203-0304/3040 mellékű telefonszámon.

Menjen biztosra!

Tervezze üzleti folyamatait Microsoft Project 2000-rel!

Minőség, egyéneknek kedvező konstrukciók, cégeknek partneri kedvezmények!

Microsoft CERTIFIED Technical Education Center

SZÁMALK TOVÁBBKÉPZÉS

SZÁMALK

✆ 203-0304/3050 www.szamalk.hu/tisza 1115 Budapest, Etele út 68.