tech.net

working with windows

Az utolsó lap

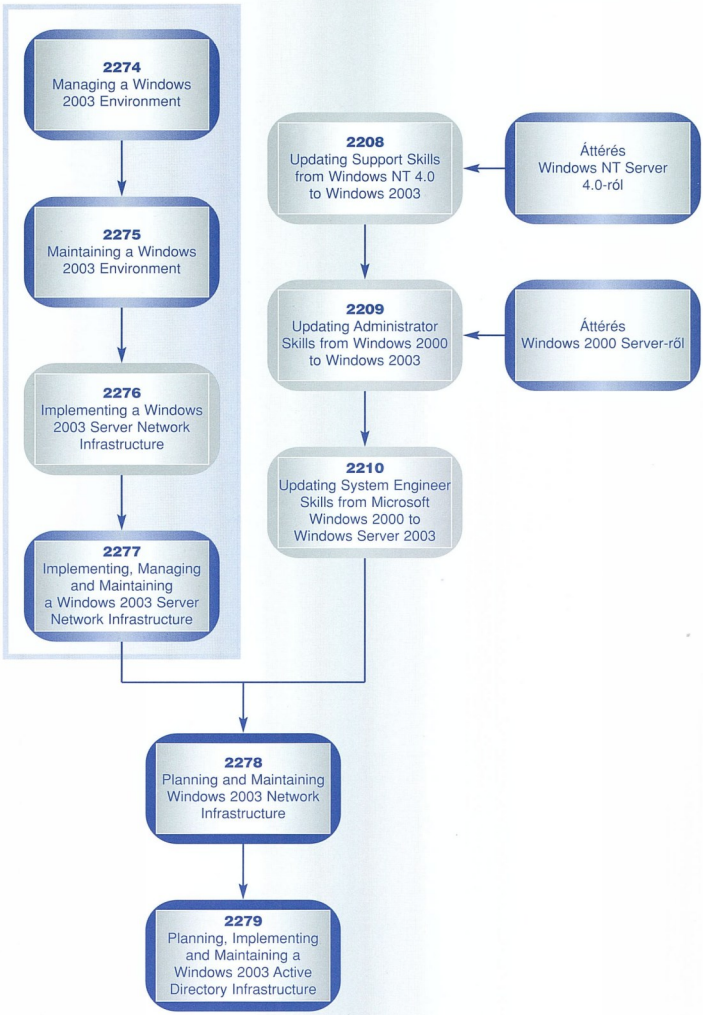
IV. / 12. szám
2003. december
1344 Ft

ISSN 15865185 12
9 771586 518005

NetACADEMIA

A LEGJOBBAKAT TANÍTJUK

Windows 2003 Server rendszergazdai tanfolyamok térképe



Szerkesztőség:
Főszerkesztő: **Fóti Marcell**
marcellf@netacademia.net

A szerkesztőség címe:
1062 Budapest, Andrássy út 62.
Telefon: 472-1214
technet@netacademia.net
Nyilvános levelezési lista:
tech.net@technetklub.hu

Kiadja és terjeszti a
NetAcademia Kft.
Terjesztési, előfizetési információ:
Telefon: 472-1214
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft

Előfizethető megrendelőlevélben a szerkesztőségnél:
1062 Budapest, Andrássy út 62.
Fax: 472-1215
http://technet.netacademia.net/subs

Hirdetésfelvétel: **Szívós Éva**
Telefon: 472-1214
Fax: 472-1215
info@netacademia.net

Nyomdai előkészítés:
Ars Luna Bt.
Vezető: Dobák Ildikó

Nyomda:
AduPrint Kiadó és Nyomda Kft.
1033 Budapest,
Csikós utca 8.
Felelős vezető: **Tóth Béláné**

ISSN 1586-5185

Az utolsó lap...

Búcsú a papírtól

A tech.net magazin utolsó számát tartja kezében a meglepett olvasó. Az alábbiakban elmesélem, miért is ér véget ez a jó hírnévnek örvendő, magasszintű szakcikkeket tartalmazó, igényes megjelenésű magazin.

Az első kérdésük nyilván ez lenne: pénzügyi okai vannak a megszűnésnek? Nos, egyáltalán nem. A második kérdésre is tudom a választ: hogyhogy ilyen hirtelen? A válasz az, hogy egyáltalán nem hirtelen. Én már több mint egy éve keresem a megoldást arra a problémára, amit úgy neveznek: túlterhelés.

Illúziók...

Annak idején 2000-ben, a tech.net magazin születésekor arról álmodoztam, hogy ez egy olyan lap lesz, amibe szinte mindenki cikket szeretne írni, hisz mindannyiunkkal történnek tanulságos, érdekes dolgok napi munkánk során. A kifejtendő témakörök száma már Microsoft technológiából is igen széles, és a végtelenhez tart – már csak le kell írni.
Bennem például igen erőteljes publikálási kényszer feszült, úgy éreztem, hogy legalább 1000 embernek kell elmondanom mindazt a tapasztalatot, amit addigi oktatási és szaktanácsadási ténykedésem során felhalmoztam.

... és a valóság

Sajnos a publikáló szerzők száma sohasem haladta meg azt a minimális létszámot, ami mellett biztosítva lett volna, hogy hónapról hónapra elegendő tartalommal rendelkezzünk.
Tökéletesen alátámasztja ezt az a tény, hogy ez a lapszám is 8 oldallal karcsúbb, mint „szokott", bár a „szokott" már nagyon gyakran 4-4 oldal lelopását jelentette.
Az én belső kényszerem így fokozatosan külső kényszerré alakult, a tech.net magazin pedig beköltözött a mindennapjaimba, a szabadságom idejébe, az estéimbe – mindenhova. Ha éppen nem korrektúráztam, akkor cikket kellett írnom, ha ezzel is megvoltam, jöhetett a szerzőkeresés, a következő lapszám tervezése, a nyelvi lektorálás és az egyeztetés a grafikusokkal. Mindezt mellékállásban, az oktatás utáni szabadidőmben végeztem.
Ezt a problémát úgy próbáltam orvosolni, hogy egy jó fél éve elballagtam a Microsoft Magyarországhoz: keressenek új főszerkesztőt, mert én szeretnék csöndben visszavonulni. Fél éves felmondási időben állapodtunk meg, és ez az idő most lejárt. Utódom sajnos nincs. Tehát a tech.net magazin a jelenlegi formájában megszűnik létezni.

Információs túlterhelés

Sokat gondolkodtam, vajon helyesen döntöttem-e, és az utóbbi hetek tapasztalata azt mondatja velem, hogy sajnos igen. Az újdonság varázsának elmúltával a tech.net magazin is beállt a soha el nem olvasott kiadványok sorába. Több előfizetővel beszéltem, akik már nem győzik elolvasni, amiket írunk. Ez pompásan találkozik azzal, hogy mi pedig nem győzzük megírni. Ha a matematika szabályai szerint mindkét oldalt elosztjuk tech.nettel, az egyenlőség nem borul fel.

Mi lesz ezután?

Az év eleje óta üzemel a NetAcademia Tudástár, ahová egyrészt folyamatosan felpakoljuk a korábbi tech.net cikkeket *(az örökévalóságnak)*. Ezek az anyagok hamarosan a www.microsoft.com/hun/technet/ oldalról is elérhetővé válnak. Emellett a NetAcademia Tudástár lesz az a fórum, ahol a belső kényszerből fakadó írásaink megjelennek. Nem havi rendszerességgel, nem 40 oldal terjedelemben – de szakcikkek.

A tech.net tervek

A Microsoft Magyarországtól származó információk alapján a keresés továbbra is folyik, tehát a papírmagazin – amennyiben sikerül megfelelő, írói vénával és komoly szakmai háttérrel egyaránt rendelkező főszerkesztőt találni – reményeik szerint 2004 első felétől kezdődően ismét megjelenik.
A magunk részéről a szakmai publikációkat eztán a Tudástárban közöljük, ahol fel lehet iratkozni az egyes témakörökre, és ha ott új cikk jelenik meg, emailben értesítjük az olvasókat.
Persze tudom, hogy a papír az papír: csak ezt lehet olvasni buszon, fürdőkádban, valamint vezetés és bandzsi dzsamping közben. Éppen ezért azon gondolkodunk, hogy *(ha kipihentük eddigi fáradalmainkat)* megjelenünk egy negyedéves szaklappal. De ez még csak annyira biztos, mint hogy a Matrix filmsorozat nem ért véget a harmadik résszel.

A formától és névtől búcsúzunk tehát, de a tartalomtól nem. Viszontlátásra 2004-ben is!

Találkozunk a weben és a tanfolyamokon!

Fóti Marcell
Főszerkesztő
MCSE, MCT, MCDBA, MZ/X

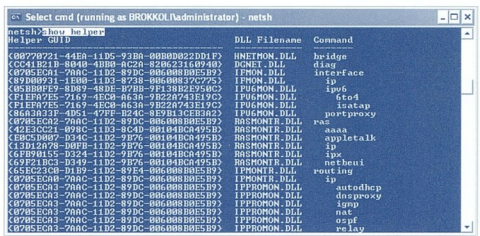
2003. 12. szám

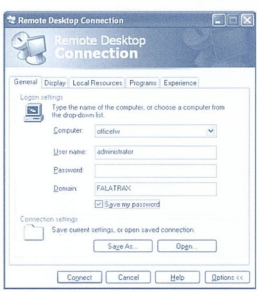
Parancssori hálózatkezelés

Netsh

A múltkori „Ne légy rendszergazda!" című cikkemben már felbukkant a Netsh parancs, amivel akkor IP-címet állítottunk egy hálózati kártyán. Most összefoglalom a Netsh-val elvégezhető feladatokat, köztük néhány olyat, amit másképp el sem lehet végezni. Hogyan mentenék le másképpen egy adott gép hálózati beállításait?

4. oldal





Terminálszolgáltatás a Windows Server 2003-ban

Alkalmazások a távolban

A terminálok segítségével elérhető központi kiszolgáló koncepciója napjainkban újra terjedőben van, mivel a központi felügyelet olcsóbbá teheti a nagy gépparkok üzemeltetését. A Windows Server 2003 Terminálszolgáltatás ennek megvalósításához kínál eszközöket.

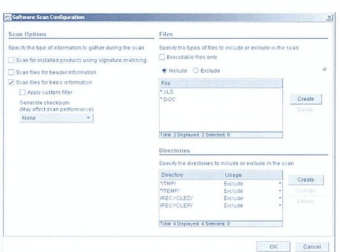
6. oldal

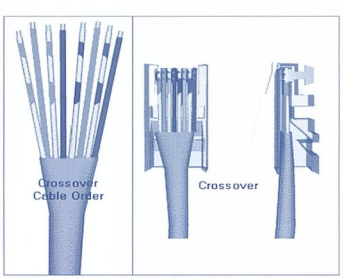
Hagyományos rendszermenedzsment

...ami már nem is annyira hagyományos

Napjaink IT infrastruktúráinak hatékony üzemeltetése csak egy rendszermenedzsment eszköz által támogatva képzelhető el. De be kell-e érnünk egy HW-SW leltárral, egy szoftverterítési eljárással és a többi jellemző funkcióval? Mi lesz például az októberi számban bemutatott 800 gépes feladattal, ha nincs a közelben az akkori szerző? Kerüljünk a közelébe egy Tivoli rendszernek!

10. oldal



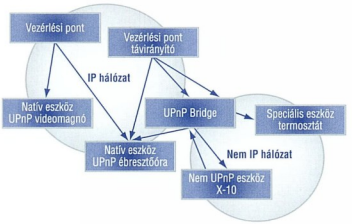


Hálózati Kábelezés

UTP, CAT5, RJ45

Ha szükségünk van egy speciális méretű hálózati kábelre... Szerelőt hívunk, mert saját magunk félünk elkészíteni? Inkább megvesszük az üzletben? Esetleg egy „szaki" ismerősünk készíti el? Ha ezekre a kérdésekre igen a válasz, a cikk végére talán majd nem az lesz...

13. oldal



Univerzális Plug and Play

Az önműködő hálózat

A UPnP hálózat segítségével megvalósítható a hálózati eszközök automatikus csatlakoztatása és eltávolítása, a műveletek semmilyen emberi beavatkozást nem igényelnek; az összes szükséges információt maguk az eszközök hordozzák és hirdetik meg a hálózaton.

15. oldal

A DHCP rejtett szépségei – VII.

A cikksorozat befejező részében három fontos témakört elemzünk. Megvizsgáljuk, milyen lehetőségeink vannak a DHCP-szolgáltatás rendelkezésre állásának növelésére, áttekintjük a beépített monitorozó eszközöket, végül néhány hasznos fogást ismertetünk, amelyek a DHCP-adatbázis karbantartását segíthetik.

19. oldal

Felhasználói bizalom, mint a fejlődés kulcsa

Az információs társadalom fejlesztésén munkálkodó Magyarországnak szembesülni kell azzal a problémával, hogy a fejlődés gátja ma már nem a rendelkezésre álló hardverek szűkössége, és nem is alapvetően a világhálóhoz való csatlakozás drágasága, hanem egyrészt az Internettel elérhető szolgáltatások kínálatának szűkössége, másrészt az a felhasználói bizalmatlanság, amely legfőbb akadálya a szolgáltatások bővülésének.

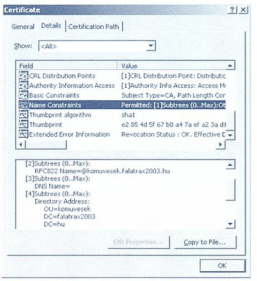
25. oldal

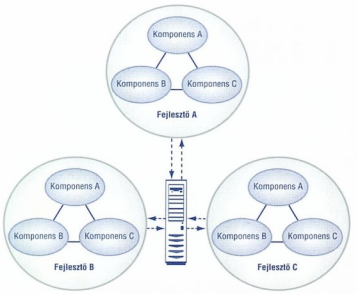
Tanúsítványkiadók a Windowsban

Qualified Subordination

Sorozatunk utolsó része az altanúsítványkiadókról szól.

27. oldal





Objektumorientált alkalmazásfejlesztés

Záró gondolatok

33. oldal

Parancssori hálózatkezelés

Netsh

A múltkori „Ne légy rendszergazda!" című cikkemben már felbukkant a Netsh parancs, amivel akkor IP-címet állítottunk egy hálózati kártyán. Most összefoglalom a Netsh-val elvégezhető feladatokat, köztük néhány olyat, amit másképp el sem lehet végezni. Hogyan mentenék le másképpen egy adott gép hálózati beállításait?

Netsh alapok

A Netsh parancs egy keretprogram, ami olyan funkciókat lát el, amilyen helper DLL-ek „alá vannak rakva". Kétféle üzemmódban futtathatjuk: parancssori és interaktív módban, hasonlóan például az NSLookup parancshoz.
Parancssori üzemmódban közvetlenül beírjuk a parancs mögé az összes utasítást és paramétert, ami a végrehajtáshoz szükséges, például:

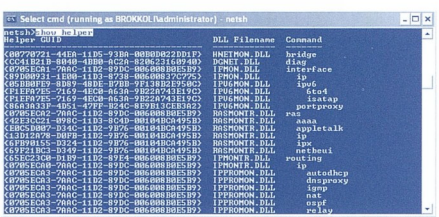
```
netsh interface ip set dns local dhcp
```

Interaktív üzemmódban pedig egy promptot kapunk, ahova akár órákon át írogathatjuk a Netsh parancsokat:

```
netsh>
```

Parancskészlet

Kezdjük azzal, hogy megnézzük, milyen parancsokat támogat mondjuk egy Windows XP-n. Az alábbi ábrán a parancslistázó parancsot *(show helper)*, valamint a lista elejét láthatjuk, a jobb oldali faszerkezet pedig az egyes DLL-ek által kiszolgált parancsokat mutatja.



Netsh helper DLL-ek és funkcifunkcióik

Netsh hierarchia

Apropó faszerkezet: a parancsok *(a DLL-ek mentén)* logikus hierarchiába vannak szervezve, és úgy váltogathatunk közöttük, mintha egy könyvtárszerkezetben mászkálnák fel és alá. Ezt a Netsh kontextusváltásnak hívja. Van néhány parancs, ami a hierarchia mindegyik pontján kiadható, ezek az alábbiak:

Parancs	Magyarázat
SET	valamilyen értéket beállít
SHOW	a beállításokat egyesével jeleníti meg
DUMP	a beállításokat scriptszerűen listázza ki
ADD	többértékes változók második és sokadik értéké nek beállítására *(pl. másodlik DNS Server-cím)*
DELETE	többértékes változók második és sokadik értékének törlése
HELP	az adott kontextusban érvényes parancsokat listázza ki
PUSHD	egy kontextus elmentése
POPD	visszatérés egy elmentett kontextushoz

Kontextusváltások

A legegyszerűbb kontextusváltás, ha egy paranccsal „beljebb" megyünk, például a gyökérből az INTERFACE kontextusba:

```
interface
```

Vagy rövidítve:

```
i
```

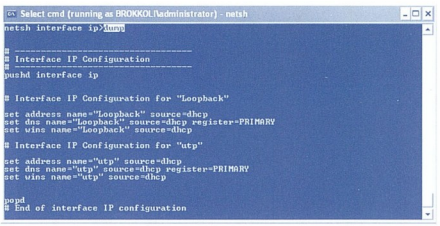
A parancsokat ugyanis addig célszerű rövidíteni, amíg egyértelműen beazonosíthatók. Egy alap Windows XP-n csak az INTERFACE kezdődik i-vel, de már például o vagy p betűs parancs kettő is van. Ha azokat rövidítjük, lehet, hogy mást csinál a parancs, mint amire gondolnánk *(lásd később)*! Ez a példa egyenesen az INTERFACE alatti IP kontextusig hatol le:

```
i i
```

Kontextusból „kijönni" a .. *(pont-pont)* jelekkel, míg a Netsh-ból tetszőleges mélységből egyből kiszállni a BYE *(rövidítve: B)* paranccsal lehet.
Két további kontextuskezelő parancs van, a PUSHD és a POPD, amelyeket hosszabb scriptekben érdemes használni. Lássunk is mindjárt egy olyan scriptet, ami kis kalandunk után lehetővé teszi, hogy egyetlen mozdulattal visszaállítsuk majd a hálózati kártyáink IP-beállításait. Tehát elmentjük a jelenlegi beállításokat.

DUMP parancs

A hierarchia tetszőleges pontján kiadva a DUMP parancsot egy olyan outputot kapunk, ami annak a kontextusnak és gyermekeinek beállítását tartalmazza – mégpedig Netsh script nyelven!
Ha a gyökérben futtatjuk, az IGMP-től az IPV6-ig minden beállítást lescriptel. Ha csak az IP-vel kapcsolatos beállításokat szeretnénk scriptelni, menjünk be az INTERFACE IP kontextusba *(i i)*, és ott dumpoljunk egyet!



Az IP-beállítások mentése scriptbe (dump)

A fenti képernyőképen nemcsak a SET parancs helyes használatára láthatunk példát, hanem arra is, hogyan alkalmazhatjuk a PUSHD, POPD parancsokat egy olyan scriptben, mely össze-vissza nyargalászik a kontextusok között. Ugyanis a legelső sorban olvasható

```
pushd interface ip
```

parancs először lementi a jelenlegi kontextust, ezután beszáguld az INTERFACE IP alá, ott ügyködik néhány sort, majd a script legvégén POPD-vel visszaáll oda, ahol korábban voltunk. Ha a teljes Netsh hierarchiáról csinálunk egy dumpot, azt láthatjuk, hogy mindegyik ágba hasonlóan rohan be, ott teszvesz valamit, majd POPD-vel visszaugrik a gyökérbe, hogy a script következő parancsa a kályhától indulhasson el.

Hálózati beállítások mentése Netsh scriptbe

Elvileg a DUMP parancsnak van egy **[Filename]** paramétere, ahol megadhatjuk, hogy hova tegye a kész scriptet, de ez nem működik, ezért más módszert eszeltem ki a feladat sikeres végrehajtására. Parancssori mód + fájlbairányítás!

```
NETSH I I D >ipset.txt
```

Az eredmény *(az ipset.txt fájl tartalma)* szépen felkommentezve így néz ki:

```
pushd interface ip

# Interface IP Configuration for "Loopback"
set address name="Loopback" source=dhcp
set dns name="Loopback" source=dhcp
register=PRIMARY
set wins name="Loopback" source=dhcp

# Interface IP Configuration for "utp"
set address name="utp" source=dhcp
set dns name="utp" source=dhcp register=PRIMARY
set wins name="utp" source=dhcp

popd
```

Remélem mindenki felismeri már a fájl szerkezetét és belsejét: két hálózati kártyám van, az egyik neve „Loopback", a másiké „utp", és ezekkel dolgozik a script.
Futtatása pofonegyszerű:

```
Netsh -f ipset.txt
```

Az is megér néhány szót, miért neveztem el a hálókártyáimat, s miért nem hagytam meg az alapértelmezett „Local area..." nevet?

Hálózati kapcsolatok átnevezése

Mint azt korábban láttuk, a netsh parancsait extrém módon le lehet rövidíteni, és ilyen parancsokkal lehet szédíteni a tisztes kollégákat:

```
Netsh i i s a u d
```

Ezzel általában zajos sikertől elkezdve a néma hódolatig tetszőleges palettán arathatunk babérokat. *(Palettán babérokat? Iszonyú képzavar! Csak egy magyarszakos meg ne lássa!)*
Ez a parancs egyébként az utp nevű hálókártyát DHCP-re állítja *(INTERFACE IP SET ADDRESS UTP DHCP)*
Igen ám, de a rövidítés csak addig teszi helyesen a dolgát, amíg a kulcsszavak egyértelműen megkülönböztethetők egymástól. De miben különbözik a „Local area connection" a „Local area connection 2"-től? Csak az utolsó karakterben. Tehát az egyértelmű azonosítás kedvéért véges-végig ki kelleně írni a kártya nevét, különben cigányútra megy a parancs *(ugyanis az ABC-ben előbbre lévőn jut érvényre)*.
Ezzel kapcsolatban volt egy érdekes tapasztalatom: kiszerelt *(!!)* hálókártyára futott le a netsh, mert annak neve volt a legelső az ABC-ben! Mi meg csak lestük a meglévő kártyát, vajon miért nem engedelmeskedik a parancsnak? A kiszerelt bezzeg engedelmeskedett! ☺
Ennyi bevezetés után jöjjön a kártyák átnevezése:

```
Netsh interface set interface name="Local Area
Connection 2" newname=utp
```

Rövidítve:

```
Netsh i se i n="Local Area Connection 2" ne=utp
```

(Itt a SET parancsot SE-nek lehetett csak rövidíteni, mert különben a SHOW-t erőlteti. Mint ha a SH előbb lenne az ABC-ben, mint a SE :-O Vigyázzunk a kétértelmű rövidítésekkel, kiszámíthatatlanok!)

Hálózatdiagnosztika *(DIAG kontextus)*

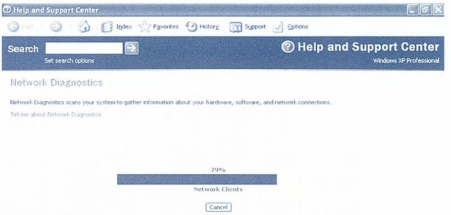
Érdekes lehetőséget rejteget a netsh a hálózatdiagnosztika tudományágából. A DIAG kontextus sokat mondó alparancsai a következők:

Parancs	Magyarázat
PING	Egy kijelölt elem megpingelése
CONNECT	TCP-kapcsolat építése és bontása egy kijelölt elemhez
SHOW	Egy kijelölt elem beállításának megtekintése
GUI	A fenti három parancs grafikus kivitelezése a Help és Support Center segítségével, HTML kimenettel

A kijelölhető elemeket csak felsorolom, annyi van, mint égen a csillag: client, computer, dhcp, dns, gateway, ieproxy, ip, mail, modem, news, os, version és wins. Például a

```
Netsh diag ping hcp
```

Parancs megpingeli a gép DHCP-kiszolgálóját. Mindez grafikusan is megy *(netsh diag gui)*:



netsh diag gui

A HTML-jelentést ki-ki nézze meg a saját gépén. Jó munkát!

Fóti Marcell
marcellf@netacademia.net

Terminálszolgáltatás a Windows Server 2003-ban

Alkalmazások a távolban

A terminálok segítségével elérhető központi kiszolgáló koncepciója napjainkban újra terjedőben van, mivel a központi felügyelet olcsóbbá teheti a nagy gépparkok üzemeltetését. A Windows Server 2003 Terminálszolgáltatás ennek megvalósításához kínál eszközöket.

Bevezetés

A Windows Server 2003 Terminálszolgáltatás segítségével elérhetővé tehetjük a Windows-alapú alkalmazásokat vagy a Windows Asztalt magát szinte bármilyen számítógépről, még azokról is, amelyek nem képesek a Windows futtatására. A Windows Server 2003 Terminálszolgáltatás a Windows 2000 Terminálszolgáltatás alkalmazás-kiszolgáló módján alapul, és tartalmazza a Windows XP-ben megjelent új tulajdonságok megvalósítását is.

A Terminálszolgáltatás csak a program felhasználói felületét továbbítja az ügyfélhez, az alkalmazások futtatása a kiszolgálón történik. Az ügyfél a billentyűzet- és az egérmozgatás jeleit küldi vissza a kiszolgálóra. Minden felhasználó csak saját munkamenetébe jelentkezhet be, és csak ezt látja. A kiszolgáló operációs rendszere a felhasználó számára láthatatlanul és egymástól függetlenül kezeli az ügyfélmunkameneteket. Az ügyfélszoftver számos hardvereszközön futhat, beleértve a számítógépeket és a Windows alapú terminálokat. Egy kiegészítő szoftver segítségével más eszközök, például Macintosh számítógépek vagy UNIX alapú munkaállomások is csatlakozhatnak a terminál-kiszolgálóhoz.

A Terminálszolgáltatás használata a következő előnyökkel jár:

- **Gyorsabban elérhetővé teszi az eszközök számára a Windows XP rendszert.** A Terminálszolgáltatás segítségével áthidalhatjuk azt az időtartamot, amíg a régi eszközöket a Windows XP Professional rendszerrel frissítik: „virtuálisan" biztosítja a Windows XP Asztalt a számítógépnek nem tekinthető eszközök, illetve az olyan számítógépek számára is, amelyek hardvere nem felel meg a teljes Windows XP operációs rendszer helyi futtatásához. A Terminálszolgáltatás ügyfélszoftvere számos különböző platform számára elérhető, beleértve az MS-DOS-t, a Windows alapú terminálokat, a Macintosh és a UNIX rendszereket. *(Az MS-DOS, a Macintosh és a UNIX alapú számítógépekhez kiegészítő szoftver szükséges.)* A Terminálszolgáltatás segítségével a Windows Server 2003 számítási teljesítménye és szolgáltatásai válnak elérhetővé az újonnan megjelent, kisméretű és korlátozott kapacitású eszközökről is *(például Pocket PC)*
- **Az adatokhoz való hozzáférés kisebb sávszélességet igényel.** Terminálkiszolgáló használatával, a nagy adatmennyiséggel dolgozó alkalmazásokat korlátozott sávszélességet biztosító környezetben is futtathatjuk *(telefonvonal vagy megosztott WAN kapcsolat)*, mivel így az adatok helyett csak azok képernyőn megjelenő képét kell átvinnünk a hálózaton.
- **Biztosítja a meglévő hardvereszközök által nyújtott összes lehetőséget.** A Terminálszolgáltatás kiterjeszti az elosztott feldolgozás modelljét, hiszen a számítógépek működését egyszerre sovány ügyfél módban, illetve az összes szolgáltatást futtató személyi számítógép módban is lehetővé teszi. A számítógépek a megszokott módon használhatók a meglévő hálózatokban, de sovány ügyfélként is képesek működni és a Windows XP Professional Asztalt emulálni.
- **Központosítja a programok kezelését.** A Windows Server 2003 rendszert használó számítógépen futó Terminálszolgáltatás az összes programfuttatást, adatfeldolgozást és adattárolást a kiszolgálón hajtja végre, központosítva ezzel a programok kezelését. A Terminálszolgáltatás minden ügyfél számára biztosítja a programok aktuális verzióihoz való hozzáférést. A szoftvert csak a kiszolgálóra kell telepíteni, nem pedig a szervezet összes számítógépére, csökkentve ezzel az egyes számítógépek frissítésével járó költségeket.
- **Távfelügyeletet biztosít.** A Terminálszolgáltatás lehetővé teszi a Windows Server 2003 rendszert futtató kiszolgálók távoli felügyeletét, biztosítva ezzel a rendszergazdák számára a kiszolgáló bármely ügyfélről történő távoli kezelését WAN hálózaton vagy telefonos kapcsolaton keresztül is.

> A Windows Server 2003 Terminálszolgáltatások használata sok előnnyel jár: segítségével központosítható az alkalmazások felügyelete, és a Windows alkalmazások számos platformról elérhetővé válnak.

A Windows Server 2003 Terminálszolgáltatás számos új tulajdonsággal és funkcióval rendelkezik a korábbi verzióhoz képest, a cikk további részében ezeket fogjuk sorra venni.

Az ügyfélprogram újdonságai:

- Továbbfejlesztett felhasználói felület
- Ügyféleröforrások átirányítása
- Ügyfelek telepítésével kapcsolatos tulajdonságok

A kiszolgáló újdonságai:

- Továbbfejlesztett kiszolgáló-felügyeletet
- Biztonsági fejlesztések
- Munkamenet-nyilvántartás

Az ügyfélprogram tulajdonságai

A Terminálszolgáltatás ügyfélprogramja számos újdonságot tartalmaz a korábbi kiadásokhoz képest.

- Távoli asztali kapcsolat *(Remote Desktop Connection, RDC)* – Az RDC program segítségével csatlakozhatunk mind a Windows XP Professional gépeken futó Távoli asztal szolgáltatáshoz, mind pedig a Terminálszolgáltatás régebbi verzióihoz *(Windows NT 4.0 Terminal Server Edition és Windows 2000)*. Az RDC program az

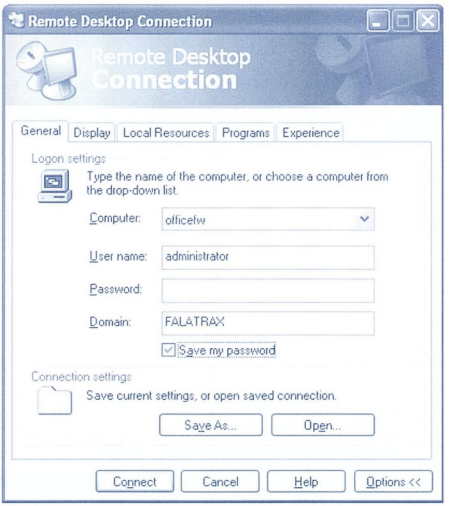
```
C:\mstsc.exe
```

parancs beírásával, vagy a Start menüből indítható:
Start → Programok → Kellékek → Kommunikáció → Távoli asztali kapcsolat

Csatlakozás távoli számítógéphez az RDC program segítségével

A távoli kapcsolatok beállításai

A távoli kapcsolatok különféle opciói a Beállítások gomb segítségével érhetők el. A megnyíló tulajdonságlapon megadhatjuk a bejelentkezéssel, a megjelenítéssel, a helyi erőforrásokkal és a munkamenet létrehozáskor automatikusan elinduló programokkal kapcsolatos adatokat.



A távoli kapcsolat beállításai

Az önálló Kapcsolatkezelő *(Connection Manager)* már nem létezik, minden funkcióját közvetlenül az RDC valósítja meg. Segítségével a felhasználók és rendszergazdák elmenthetik és betölthetik a kapcsolatok beállításait tartalmazó, rdp kiterjesztésű fájlokat. Az elmentett jelszavak biztonságos tikosítással kerülnek tárolásra, és csak azon a számítógépen használhatók, amelyen mentésre kerültek.

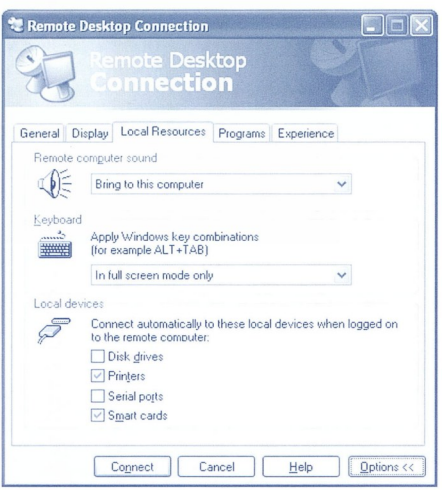
Ügyfél erőforrások átirányítása

A Távoli asztali kapcsolat program segítségével számos adattípus átirányítását megvalósíthatjuk. Biztonsági okokból minden átirányítás az ügyfélen és a kiszolgálón is letiltható. Figyelmeztető üzenet jelenik meg a fájlrendszerre, valamely portra vagy smart cardra vonatkozó átirányítási kérelemkor; a felhasználó megszakíthatja a kapcsolatot, vagy letiltathatja az átirányítást.

A következő átirányításokat állíthatjuk be:

- **Fájlrendszer** – Az ügyfél meghajtói *(a hálózati meghajtók is)* elérhetők a kiszolgálói munkamenetből. Az engedély egyszerre az összes meghajtóra vonatkozik, ráadásul alapértelmezés szerint a kiszolgálói munkamenetben Everyone > Full Control jogosultsággal jelennek meg az ügyfélgép meghajtói, így az átirányítás használatához némi óvatosság szükséges.
- **Portok** – Az ügyfél soros portjai elérhetővé tehetők a kiszolgálói munkamenetből, így a kiszolgálón futó szoftverek hozzáférhetnek az ügyfél bizonyos hardvereszközeihez.
- **Nyomtatók** – Az ügyfél valamennyi telepített nyomtatója *(a hálózati nyomtatók is)* elérhető a kiszolgálói munkamenetből *(a Windows 2000 Terminálszolgáltatás csak a helyi nyomtatók átirányítását tette lehetővé)*. Az átirányított nyomtatók könnyen értelmezhető nevet kapnak.

- **Hangok** – A hibaüzenetekhez kapcsolódó hangjelzések, vagy például az új elektronikus levél érkezését jelző hangok átirányíthatók az ügyfelekre.
- **Smart Card bejelentkezés** – A Windows rendszer bejelentkezési adatait tartalmazó smart card használható a Windows Server 2003 távoli munkamenetbe történő bejelentkezéshez is. A funkció használatához olyan ügyfélrendszer szükséges, amely önállóan is képes a smart card kezelésére *(Windows 2000, Windows XP, Windows CE .NET)*
- **Windows billentyűkombinációk** – Az ügyfél a Windows billentyűkombinációkat *(Alt-tab, Ctrl-esc stb.)* alapértelmezés szerint továbbítja a távoli munkamenetnek. A Ctrl-alt-del billentyűkombinációt azonban biztonsági okokból mindig az ügyfél dolgozza fel. Az átirányítás működik Windows 2000 terminál kiszolgáló esetén is, de csak NT alapú ügyfél operációs rendszerrel *(Windows 9x-szel nem)*.
- **Időzóna** – Az RDC ügyfél képes az időzónára vonatkozó adatok automatikus átadására, illetve a felhasználók manuálisan is beállíthatják a megfelelő időzónát. Így a különböző időzónában lévő felhasználók egyetlen kiszolgálót használhatnak.
- **Virtuális csatornák** – A virtuális csatornákon keresztül adatokat továbbíthatunk az ügyfél és a kiszolgáló között.



Átirányítások

Ügyfelek telepítése

A Távoli asztali kapcsolat *(RDC)* program a Windows XP és a Windows Server 2003 része. Más ügyfélplatformokra történő telepítéshez a következő lehetőségek valamelyikét választhatjuk:

- A Microsft Systems Management Server *(SMS)* vagy a csoportos házirendek segítségével publikálhatjuk a Windows Installer alapú RDC csomagot.
- A Windows Server 2003 számítógépen *(vagy a Windows 2000 kiszolgálón)* megosztást hozunk létre a telepítőcsomag számára.
- Telepíthetünk közvetlenül a Windows XP vagy a Windows Server 2003 CD-ről, ha a telepítőmenüből az Egyéb feladatok végrehajtása *(Perform Additional Tasks)* pontot választjuk.
- Az RDC letölthető az **[RDC]** webhelyről.

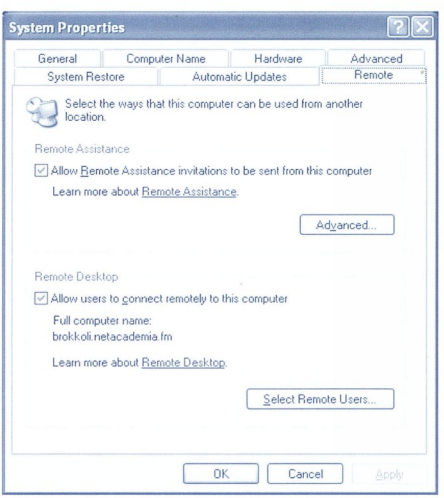
Az RDC Windows CE verziója

A Windows CE .NET Platform Builder tartalmazza az RDC Windows CE verzióját, így a hardvergyártók beépíthetik azt termékeikbe.

A kiszolgáló újdonságai

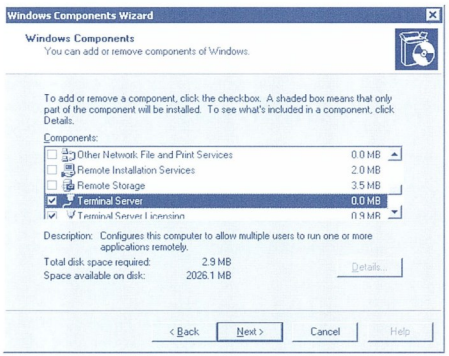
A Terminálszolgáltatás kiszolgálói oldala is számos újdonságot nyújt, a legfontosabbak a következők:

- **Távoli felügyelet *(Remote Desktop for Administration)*** – A távoli felügyelet a Windows 2000 Server Terminálszolgáltatás távfelügyeleti módján alapul. A Windows 2000 Server esetében rendelkezésre álló két virtuális munkamenet mellett lehetőség van arra is, hogy a konzol *(session 0)* munkamenethez csatlakozzunk távolról. A konzol munkamenethez egyidőben csak egyetlen felhasználó csatlakozhat, ha ezt egy távoli munkamenet foglalja el, akkor lokálisan már nem lehet a gépre bejelentkezni. A konzol munkamenethez a Remote Desktop MMC beépülő modul segítségével, vagy az mstsc program /console kapcsolójának felhasználásával csatlakozhatunk.
- **A távoli asztal és a Terminálszolgáltatás aktiválása** – A Windows 2000 Server esetében a terminálszolgáltatást két üzemmódban futtathattuk *(alkalmazás-kiszolgáló és távfelügyeleti mód)*, míg a Windows Server 2003-ban a távfelügyelet és a Terminálszolgáltatás két elkülönített, önállóan beállítható komponens. A távoli felügyelet a Rendszer tulajdonságpanel Távoli használat *(Remote)* lapján aktiválható.



A távfelügyelet engedélyezése

A Terminálszolgáltatás aktiválásához pedig a Terminal Server komponenst kell feltelepítenünk a Programok telepítése és törlése *(Add/Remove Programs)* varázsló segítségével.



A Terminal Server telepítése

További felügyeleti eszközök

- **Csoportos házirend** – A Terminálszolgáltatás tulajdonságainak beállításához használható a csoportos házirend. Ennek segítségével lehetőség van kiszolgálócsoportra vonatkozó paraméterek együttes beállítására is.
- **Active Directory Service Interfaces** – Az ADSI csatolófelület segítségével programból férhetünk hozzá a Terminálszolgáltatás felhasználóira vonatkozó beállításokhoz *(például home mappa, hozzáférési jogok)*.
- **Single Session Policy** – A Single Session Policy beállításával a rendszergazdák egyetlen munkamenetre korlátozhatják a felhasználók hozzáférését, még kiszolgálócsoportok esetében is.
- **Hibaüzenetek** – Több mint 40 új hibaüzenet segíti az ügyfélkapcsolatokkal összefüggő problémák gyors diagnosztizálását.
- **Biztonsággal kapcsolatos fejlesztések** – A Terminálkiszolgáló biztonsági modellje jobban illeszkedik a Windows kiszolgálóknál megszokott rendszerbe.
- **Remote Desktop felhasználói csoport** – A Remote Desktop Users felhasználói csoport tagjai jogosultak a Terminálszolgáltatáshoz való csatlakozásra.
- **128 bites titkosítás** – Alapértelmezés szerint a terminálkiszolgálóval való kapcsolat 128 bites, kétirányú RC4 titkosítással biztosított, ha az ügyfél támogatja a 128 bites titkosítást *(az RDC alapértelmezés szerint 128 bites)*. Lehetséges azonban a régebbi alacsonyabb titkosítási szintet biztosító ügyfelek csatlakozása is, ha ezt le nem tiltjuk.
- **Szoftverkorlátozó házirendek** – A szoftverkorlátozó házirendek segítségével a rendszergazdák beállíthatják, hogy a terminálkiszolgálón *(vagy bármely más Windows 2003 kiszolgálón)* a megadott felhasználók csak bizonyos alkalmazásokat futtathassanak.
- **Munkamenet nyilvántartás *(Session Directory)*** – A terminálkiszolgálók farmokba szervezhetők, így a terheléskiegyenlítéssel működő fürt hibatűrő szolgáltatást biztosíthat a felhasználóknak. A munkamenet nyilvántartás biztosítja a felhasználóknak, hogy a farmon belüli elhagyott munkamenetükhöz csatlakozhassanak vissza.

Összefoglalás

A Windows Server 2003 Terminálszolgáltatás segítségével megbízható, jól skálázható, és könnyen felügyelhető kiszolgáló alapú rendszer építhető fel. Az új opciók segítségével jobban kihasználhatjuk a kis sávszélességű kapcsolatokat, így a mobil, illetve kis teljesítményű eszközök is jól használhatóak. A továbbfejlesztett ügyfél csatolófelület számos átirányítástípust támogat, fejlődött a kiszolgálók felügyelete és biztonsága.

Szerényi László
szerenyi.l@met.hu

Hagyományos rendszermenedzsment

...ami már nem is annyira hagyományos

Napjaink IT infrastruktúráinak hatékony üzemeltetése csak egy rendszermenedzsment eszköz által támogatva képzelhető el. De be kell-e érnünk egy HW-SW leltárral, egy szoftverterítési eljárással és a többi jellemző funkcióval? Mi lesz például az októberi számban bemutatott 800 gépes feladattal, ha nincs a közelben az akkori szerző? Kerüljünk a közelébe egy Tivoli rendszernek!

Emlékeztetőül: a feladat az volt, hogy operációs rendszer upgrade első lépéseként a munkaállomások C:\ meghajtóján lévő *.DOC és *.XLS fájlokat *(továbbiakban Office állományok)* át kell másolni a D:\ meghajtóra. Akkor egy scriptes megoldást olvashattunk, ami a Windows Scripting Host *(WSH)* környezetet felhasználva végzi el a mentést. A feladat rendszermenedzsmentes megoldását az IBM Tivoli **[sysmgmt]** rendszerének használatával mutatom be. Pontosabban a Tivoli Configuration Manager modul az, ami segítségünkre lesz, ugyanis a teljes körű megoldást kínáló rendszereknek csupán egy részterülete cégeink munkaállomásainak konfigurációmenedzsmentje.

Az infrastruktúra

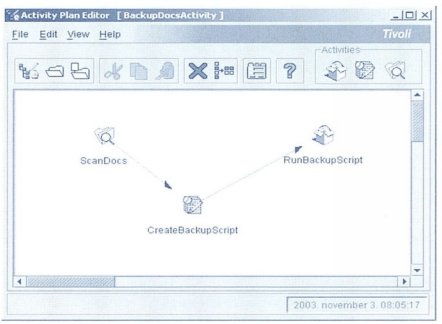
Sok mindent meg lehet oldani Group Policyvel, láthattuk, hogy a WSH mire képes, vannak eszközök távoli telepítésre és képernyőátvételre is, viszont ha nem rendelkezünk minden egyes komponensből *(munkaállomás operációs rendszertől az adatbáziskezelőig)* az aktuális évszámnak megfelelő verzióval, vagy éppen sokvégpontos környezetre kell felügyeletet gyakorolnunk, netalán platformjaink sokszínűsége a tavaszi rét színkavalkádjával vetekszik – teljes körű rendszermenedzsment nélkül nem ússzuk meg.
Az előző, bekezdésnyi mondatot kipihenendő folytassuk a következő szakaszt a megvalósítás áttekintésével.

A megoldás

A kezdeti döbbenet után logikusan három részfeladatra bonthatjuk teendőinket:

1. Leltározás: ahol összegyűjtjük az Office állományokat, azok pontos elérési útjával.
2. Másolási feladat kijelölése: nem a felhasználó által létrehozott állományok kiszűrése, majd a fájlmásolás definiálása
3. Másolás: hadd menjen!

Több eltérő feladat, amelyek lehetőleg egyből kövessék egymást a végpontokon, ugyanis egy kéthetes állományleltár alapján végrehajtott mentés várhatóan nem erősíti az IT üzemeltetési csapat pozícióját a cégen belül. Tivolink Activity Plannerre keresztelt szolgáltatása megkímél bennünket a „szőnyeg szélétől", ugyanis a Configuration Manager és a Tivoli Framework keretrendszer által biztosított minden eszközt képes egy activityvé vagy planné vagy tervvé összefogni – kinek ahogy tetszik. Az egyes részfeladatok között sorrendiséget, kapcsolatot definiálhatunk, majd az egészet ráengedhetjük akár az összes munkaállomásunkra: a sorrendet és kapcsolatot minden egyes végponton tartani fogja.



Tervünk, a maga vizuális valóságában

A terv elemei – sorrendben balról-jobbra:

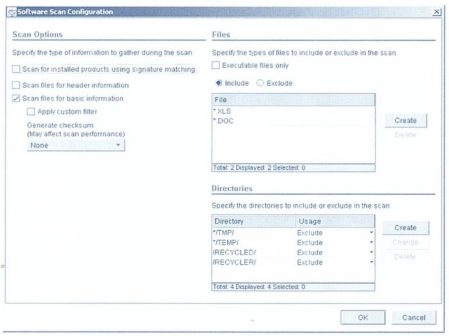
(1) InventoryScan: a fájlok leltárjának előállítója
(2) Task: a leltár alapján készít egy .BAT állományt, ami a mentést fogja végezni
(3) SoftwarePackage: elindítja a létrehozott script-et

Kaméleon van a kezünkben?

Szeretném hangsúlyozni, nem ez az egyetlen módja annak, hogy teljesítsük a küldetést és vezetőnktől, megrendelőnktől megkapjuk a hivatalos, ám mégis elismerő üzenetet: „MISSION COMPLETED SUCCESFULLY". A Tivoli eszköztárában épp az a csodálatos, hogy utat enged a kreativitásnak, teljesen testre szabhatjuk, többféle megoldási lehetőséget kínál, még az ilyen nem szokványos feladatokra is. A lehetséges megoldási alternatívák között még talán egyszerűbb is akad: amikor a kezdeti leltározás után, a másolandó fájlok listáját közvetlenül a végponton generáljuk a Tivoli Inventory által készített állományból. Ez utóbbi a Korn Shell igen gazdag szövegkezelési képességeit igényli, és annak ellenére, hogy egy taskkal még ilyen shell scriptet is futtathatunk *(Windows-on!!!)*, mégsem ezt a rövidebb változatot használtam, hanem inkább egy látványosabbat.

Első lépés részletesen

Az alábbi ábra magáért beszél: már ebben a lépésben lehetőség van alkönyvtárak kizárására és természetesen fájlmaszkok definiálására. A leltározás befejeztével RDBMS adatbázisunk gyarapodik, esetleg módosul az Office állományok adataival: név, méret, teljes elérési út, állománylétrehozási és -hozzáférési információk. Ellenőrzőösszeget is kérhettem volna, de arra most nincs szükség. Ezt SQL utasításokkal és a Tivoli eszközeivel lekérdezhetjük, de ne szaladjunk előre, beszéljen most a látvány:



Szoftverleltár testreszabása

Második lépés részletesen

Ennek végeztével addig kell eljutni, hogy a célgépeken ott legyen egy futtatható állomány, – célszerűen egy .BAT fájl, – amit majd lefuttatva abszolváljuk a feladatot. Itt jön képbe a Tivoli, „task" nevezetű eszköze, ami egy kamikaze elszántságával felvértezve lefuttat bármit, bármely menedzselt erőforráson. Le kell tehát kérdeznünk az adatbázist, és abból elő kell állítani a .BAT állományt. Ezt nem célszerű a végponton csinálni, a feladatra központi Tivoli szerverünk örömmel vállalkozik, viszont így nem szabad elfelejteni az előállt fájl végpontra való juttatását sem.

Írjunk tehát egy kis scriptet, amit a szerveren fog a taskunk futtatni:

```
#!/bin/ksh
# Source environment variables for Tivoli
. /etc/Tivoli/setup_env.sh
ENDPOINT=$1
BATFILE=/tmp/copytod.bat
rm -f $BATFILE

wsetquery -w "TME_OBJECT_LABEL='$ENDPOINT'"  <-1
↳ DocsOnC

wrunquery -n DocsOnC | while read line       <-2
do
  newline=$(echo $line | sed 's/\//\\\\/g')
  echo "copy \"$newline\" d:\\ " >>
↳ $BATFILE
done                                         <-3

wadminep $EP send_file $BATFILE "c:\copytod.bat"
RC=$?
exit $RC
```

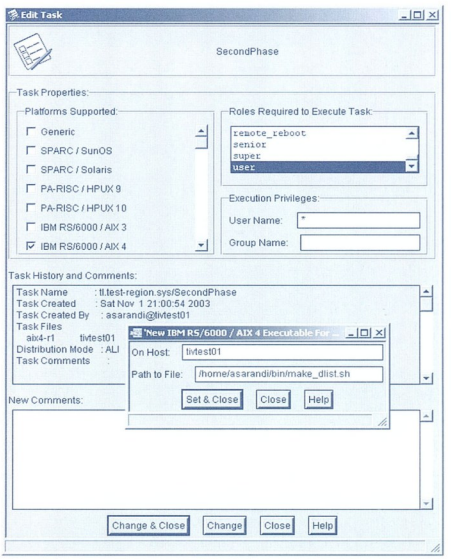
Magyarázat:

(1) „DocsOnC" lekérdezés szűrése az aktuális végpontra, amit paraméterként kap meg automatikusan a task.

(2) Lekérdezés kieresztése a karámból, hogy aztán soronként végighaladva rajta – némi konverzió után – előállítsa a másolást végző állományt.

(3) Az előállt copytod.bat elküldése a munkaállomásra.

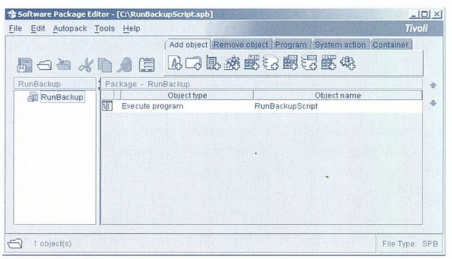
Kész a script, adjuk oda a task-nak:



Milyen platform is a szerver? Ki sem fér a választék...

Harmadik lépés részletesen

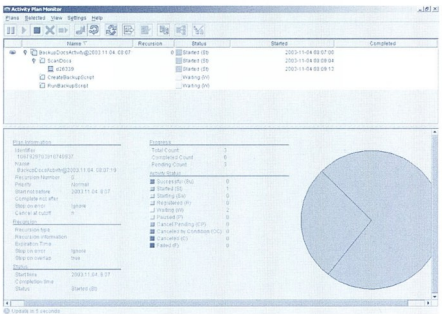
Ha valakinek hiányérzete támadt volna a Tivoli eszközeinek számát illetően, azt nem hagyom tovább keseregni, engedjék, engedjétek meg, hogy bemutassam a „Software Distribution"-t. Ő fogja Nekünk lefuttatni a másoló .BAT programot. Készítünk egy ún. SoftwarePackage–et, aminek semmi egyebet nem mondunk, minthogy a C:\copytod.batot futtassa le:



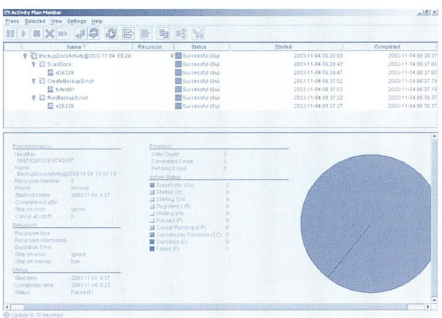
Software Package Editor madártávlatból

Essünk neki a munkaállomásoknak

Hátradőlés előtt már csak egyet kell tennünk, ráereszteni a gépekre a tervet és nyomon követni hogyan terjed szét cégünk hálózatán művünk. Mint ahogy eddig mindent, ezt is megtehetjük parancssorból és grafikus felületről *(Activity Planner Monitor)* is. Ez utóbbit választottam, ahol demonstrálom a megoldás talán egyik legnagyobb előnyét, a NYOMONKÖVETHETŐSÉGET. Pontosan tudjuk, hol tartunk, mely gépeken, mely fázisban jár a végrehajtás.



Elindítás után. A scan már elkezdődött



A végén. Ezt jól megcsináltuk...

Amit nem láttunk

Néhány olyan funkció, ami a háttérben zajlik a csomag terítése alatt és szót érdemel:

- MDIST2: segítségének köszönhetően nem kell minden végpontnak a kiküldés pillanatában bekapcsolva lennie, ha a beállított lejárati időn belül feljelentkezik a hálózatra, a Tivoli automatikusan leküldi rá az akár napokkal korábban elengedett csomagot.
- Wake-On-Lan: biztosíthatja pl. az éjszakai terítést.

Haladó tanfolyamosoknak

Amennyiben abban a szerencsés helyzetben talál bennünket ez a feladat, hogy nem magasodik fölénk vezetőnk – esetleg igen tisztelt Főszerkesztőnk – egyik kezében a befejezés – a cikkleadás – legkésőbbi dátumát mutató táblával, másikban egy ostorral, finomíthatjuk munkánkat az alábbiakkal:

- D:\ ellenőrzése: Létezik? Helyi meghajtó? Írható? Van elég hely? Ezekre a Tivoli HW-SW leltára választ ad.
- Felhasználói értesítés: feldobhatunk a könyvelés kedvenc Gizikéjének egy ablakot, hogy végeztünk az elmúlt 5 évben készített Office állományainak – annak a háromnak – lementésével, bármelyik percben megérkezhet a Takarító Brigád a maga kissé nyers, de hasznos eszközével, a „format c:" –vel.
- Újabb leltározás a folyamat végén, hogy teljesen biztosak legyünk a sikerességben.
- A teljes folyamathoz feltételeket is definiálhatunk, így néhány kattintással kizárhatjuk a körből a már upgrade-elt munkaállomásainkat.
- Tömörítés: mielőtt másolnánk, tömörítsük össze az Office állományokat, így nem zúdítunk, akár több száz állományt a D:\-be.
- Egy központilag létrehozott Windowsos megosztásra való másolás, gépnevenként külön alkönyvtárba.

Jöhet a karosszék?

Ha az ilyen vagy hasonló feladatot, könnyen, rugalmasan, külső erőforrás nélkül kipipálhatjuk, hátradőlhetünk a rendszerüzemeltetők képzeletbeli karosszékében és elismerően tekinthetünk a kis piros Tivoli ikonra a taskbaron. Ha nem rendelkezünk rendszermenedzsment eszközzel – az pótolható. Annál viszont kevés elkeserítőbb helyzetet tudok felsorolni, ha van ilyen segédünk, viszont az nem használható kellőképpen rugalmasan az ilyesfajta feladatok elvégzéséhez. Ugyanis 6 éves „rendszermenedzsmentes" múlttal és három gyermekkel a hátam mögött bizton állíthatom: ha elképzeljük a lehető legvadabb üzleti, vezetői igényt, az elkövetkezendő 1 évben a nagybetűs ÉLET túltesz majd ezen is.

Sárándi Attila
asarandi@index.hu
Magyar Tivoli Felhasználók Egyesülete
Hungarian Tivoli User Group
hungariantug@lotus.extranet.com

Hálózati Kábelezés

UTP, CAT5, RJ45

Ha szükségünk van egy speciális méretű hálózati kábelre... Szerelőt hívunk, mert saját magunk félünk elkészíteni? Inkább megvesszük az üzletben? Esetleg egy „szaki" ismerősünk készíti el? Ha ezekre a kérdésekre igen a válasz, a cikk végére talán majd nem az lesz...

Aki elméleti *(szoftver)* síkon már magas szintre képezte ki magát a hálózati rendszereket illetően, ám az idevágó fizikai *(hardver)* megoldásokhoz még nem áll olyan közel, mint szeretné... nos, akikre igaz eme állítás, azoknak szeretnék egy kis információt nyújtani az egyik legalapvetőbb és legelterjedtebben használt hálózati hardvereszközről: az UTP kábelről.
Hogy miért pont ezt választottam? Mert ez jutott éppen az eszembe, és mert nemrégiben többször is előjött egy-két, a kábelezésre visszavezethető probléma.

Miről is lesz szó?

Az Ethernet alapú hálózati megoldást követve, a CAT5-ös *(CATegory 5)* UTP *(Unshielded Twisted Pair)* kábelek, szabványos RJ45-ös csatlakozóval *(szakzsargon szerint: 8-as pucukával)* szerelt változatával foglalkozom.

Ugrás a „mélyvízbe"...

A „8-as pucuka" talán elegendően informatív ahhoz, hogy kiderüljön belőle, hogy egy nyolc eres kábeltípusról van szó, amiből az említett hálózati szabványban mindössze két érpárt, azaz négy vezetéket használunk. Az UTP betűszóból kiemelten fontos a T, azaz a Twisted *(csavart)* kifejezés. Mivel nagyfrekvenciájú jeleket viszünk át a kábelen, elengedhetetlenül fontos, hogy az egyes erek ne viselkedjenek hosszú kifeszített antennaként – mindenféle környezeti zavart összeszedve és sugározva –, illetve egymást se zavarják a közösen, „egymás mellett" megtett út során.
Erre a legegyszerűbb és legjobb megoldás, ha összecsavarjuk őket. Ez egészen pontosan úgy néz ki, hogy a nyolc vezetékből, kettesével véve, négy pár van képezve, amely párokban egy-egy vezetéket egymással összetekerünk, és végül az így létrejött négy pár egymással is össze van tekergetve a tökéletes hatás érdekében. Ezzel minimálisra csökken az egyes erek közti interferenciahatás és jelkisugárzás sem lép fel. A CAT3 szerint 10-15 csavarás kell méterenként, míg ez szám a CAT5-nél 20-30. Felmerülhet a kérdés, hogy összecsavarva vajon miért nem zavarják egymást? *(Egyébként zavarják, de csak a megengedett szint alatt.)*
Azért nem, mert az „egyenes antennaszakaszok" gyakorlatilag megszűnnek, azaz nincs egyetlen „egyenes" pontja sem a kábel ereinek. Szabályosan elvégzett sodással elérhető, hogy az elemi kis csavarokban fellépő indukciók kioltsák egymást. Emiatt erősen érzéketlenné válik a környezeti zavarforrásokra nézve, illetve alkalmatlanná válik a rajta futó jel sugárzására is. Ez persze csak egy bizonyos frekvenciahatáron belül érvényesül. Van azonban még egy érdekes „zavartényező" is, ami viszont nem küszöbölhető ki. Ez pedig abból adódik, hogy a kábelen átjutó jel folyamatosan ellenállásba ütközik *(kábelünk ebben a példában nem szupravezető)* így a két vége között ellenállás mérhető. Emellett a két szigetelt kábel-ér *(szorosan egymás mellett)* megvalósítja a „két vezető fegyverzet között elhelyezkedő szigetelő réteg"-et, ami nem más, mint a kondenzátor. Ezek így együtt – végig a kábelen – egy jó hosszú sor, elemi RC tagokból álló láncolatot alkotnak. Az RC tag *(Resistor+Capacitor)* pedig az elektronikában használt egyik szűrőkapcsolás.
Emiatt van, hogy nem lehet végtelen hosszú UTP kábel két hálózati eszközöm között, hanem százméterenként csinálnom kell valami forradalmat, különben a jel torzul, vagy elvész. A kábel csillapítása erősen függ az alkalmazott anyagoktól: A legjobb minőségű jeltovábbítást a merevszálas tiszta rézkábellel érhetjük el. A szabvány kitalálói úgy gondolták, hogy egyezményes színjelekkel látják el az egyes ereket, így segítve a pontos kábelezés kialakítását. A színpárok a következők:

- Narancssárga és narancssárgacsíkos-fehér
- Zöld és zöldcsíkos-fehér
- Kék és kékcsíkos-fehér
- Barna és barnacsíkos-fehér

A párok tagjai természetesen egymással vannak összetekerve, és színről könnyen felismerhető, hogy melyik melyikhez tartozik.
Visszatekintve a múltba látható, hogy többféle – egyre erősödő – igényt támasztottunk az UTP kábelekkel szemben az idők folyamán. A specifikáció együttes családneve: 10BaseT, de csak CAT5-ig. Régebben még elég volt, ha a 10 Mbit/secos *(max. 16 MHz-es jel)* hálózatot átvitte, de mára már a 100 Mbit/sec *(100 MHz-es jel)* vált általánossá. Sőt, az 1000 Mbitet teszik rézkábelre is, azaz van gigabites „nem-üveg" kábel. *(Meg persze egy sor új szabvány: CAT5e, CAT6, CAT7, stb.)*
A kis sebességnek még a CAT3-as is megfelelt, a CAT5-ös viszont magasabb követelményi szintet teljesít, ezért pontosan meg van határozva, hogy mely ereket milyen kötési rendben használhatunk.
Ezzel válik lehetővé, hogy a „100 Mega full duplex" azaz effektív 200 Mbit/sec hiba nélkül átmenjen rajta. Le van ám írva szépen, hogy milyen színű ereket, milyen sorrendben, hogyan tehetünk bele az RJ45-ös csatlakozóba. Még az is meghatározott, hogy mekkora maximum hosszban tekerhetjük szét az egymással összesodort ereket a csatlakozó felhelyezése előtt, hogy minél rövidebbek legyenek a kialakult egyenes szakaszok. *(Persze nem kell azért szőrszálhasogatni, a lényeg, hogy a lehető legkevesebb „szétcsavarást" alkalmazzuk.)*
De mielőtt felhelyezzük a „pucukát" a „krimpelő" fogóval a kábel végeire, nézzük meg, hogy mit mivel kell összekötni, hogy kétirányú kapcsolatot teremthessünk két eszköz között. Van ugye egy vevő meg egy adó mindkét oldalon. Jelük

egyezményesen: RX és TX. R mint Receive és T mint Transmit. Mindkettő két vezetéket használ, hogy az áramkör létrejöjjön, így összesen négy ér kell ahhoz, hogy valakinek az RX-ét a másik TX-éhez, illetve annak RX-ét ennek a TX-éhez kössem. RX hallgat tehát TX-re, és ha válaszolni szeretne, akkor a hasonló, de másik irányú csatornát kell használnia. Itt jön a kérdés, hogy miként valósul meg a páros RX-TX összekötés két eszköz között, ha a kábel két vége teljesen egyforma?

Nos, erre azt a megoldást találták ki a szabványkialakítók, hogy egyes hálózati eszközök, a számítógépek hálózati kártyáihoz képest maguktól „fordítanak", azaz sima kábelt használva jó helyen várják, illetve küldik az adatot. Ilyenek például a HUB-ok, Switch-ek. A hálózati kártyák viszont mind egyformán nem fordítanak. Azaz mindben fixen normál sorrendben van az adó és vevő. Így az egyforma kategóriába eső eszközöket mindig fordítós kábellel kell összekötni, ha közöttük kapcsolatot szeretnénk teremteni, mert valakinek mindenképpen muszáj elvégeznie azt a feladatot, hogy egy RX mindig TX-el találkozzon, és viszont.

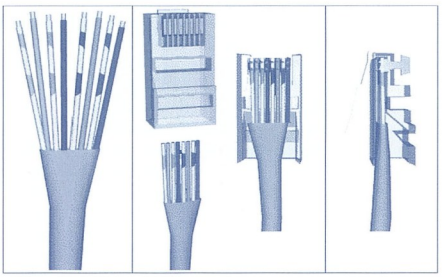
Az aktív *(abb)* eszközökön, egy kiemelt „Uplink" porton többnyire van egy átkapcsolási lehetőség is, hogy ne kelljen mégsem fordítós *(nem egyforma végű)* UTP kábelt gyártanunk, ha vele egy kategóriába eső eszközzel szeretnénk őt összekapcsolni. Sőt, olyan is van már, hogy automatikusan átkapcsol az eszköz portja a megfelelő státuszba, ha szükségét érzi.

Kábelvariációk két témára

Van tehát kétféle kábel, a sima és a fordítós. Ha képletesen egyszerre kézbeveszünk egy ilyen és egy olyan kábelt, akkor összesen négy végük van, amiből három vég teljesen egyforma. Csupán a fordítós kábel egyik végében fel van cserélve az RX a TX-szel. *(Ha mindkét végén fordítós végződés lenne, akkor fordítanánk a fordítást és végül nem történne semmi, csak lenne egy szabványtalan „CAT akárhányas" sima kábelünk.)*

Ezek után lássunk szín szerint, hogy mi is a helyes – nem fordítós – sorrend. Az RJ45 nézete szemből, érintkezőkkel felénk fordítva *(rögzítő-csattintója hátul)*, balról jobbra:

- Narancssárgafehér – narancssárga – zöldfehér – kék – kékfehér – zöld – barnafehér – barna.



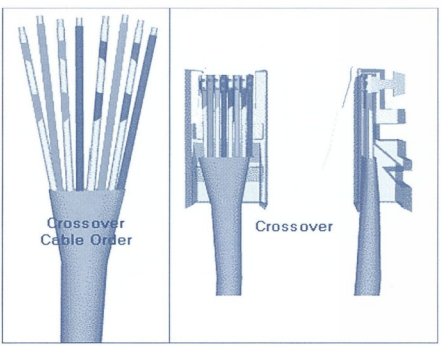
A CAT5-ös UTP kábel színsorrendje (ez csak a weben színes, az újságban minden kék-ezüst)

Mint már említettem, csak négy eret használunk a nyolcból, ezek az 1-es, 2-es, 3-as és 6-os jelűek, amik színre a narancssárgafehér – narancssárga – zöldfehér – zöld, a szabvány szerint. Használhatnánk más színűeket is, de a szabály azért szabály, hogy betartsuk.

Egyébként itt az a fontos, hogy az 1-es és 2-es legyen egymással összetekerve, illetve a 3-as és a 6-os szintén. Érdekességképpen elárulom, hogy a középső kettő – a kékek – a telefonhálózatokban használatosak, ezzel megoldható az is, hogy egyetlen UTP kábelen egyszerre utazzon az ügyfél hálózati- és telefonforgalma is. Már csak egy RJ45-RJ11 Splitter nevű filléres kis csatlakozóra van ilyenkor szükség, ami leválasztja a kettőt egymásról. Mindezt úgy, hogy egy RJ45-ös és egy RJ11-es *(vékonyabb, 2-eres)* aljzatot ad egyszerre. Egyikből aztán megy a számítógép, a másikból meg a telefon: analóg, vagy digitális is akár. A barnákról még nem beszéltünk: őket ebben a környezetben, pl. az ISDN adta lehetőségek kihasználására tartották fenn. Látható, hogy ez az összecsavarás egész jól működik, egyáltalán nem zavarják egymást az így egymás mellett haladó jelek.

A fordítós színsorrend pedig a következő:

- Zöldfehér – zöld – narancssárgafehér – kék – kékfehér – narancssárga – barnafehér – barna.



A CAT5-ös fordítós UTP kábel színsorrendje

Szembeötlő, hogy csak a zöldek és a narancssárgák helyzete változott, a többiek érintetlenül maradtak. Ezzel megoldottá vált a manuális RX-TX párosítás problémája. A szereléshez használt RJ45-ös krimpelő fogót bátran kezeljük! Az ábrákon látható módon előkészített kábelvégeket egy határozott erős mozdulattal szorítsuk meg vele. A kis érintkezők majd átszúrják az egyes ereket és jó, hibamentes kötést biztosítanak.

Ha kábelre van szükség, készítsük magunk

Nem nagy tudomány szépen és pontosan vágni, illeszteni a kábelvégeket. Egy kis gyakorlással bárki nekiállhat. Nem kell majd ezután egy hirtelen támadt extra igény esetén a szerelőkre várni, hogy ők csinálják a fentebb leírtakat. *(Azért ötven kábel elkészítése után már érezni fogjuk, hogy nem is olyan könnyű azt a fogót szorítani.)* Marad még munka a szerelőknek is bőven, például patch-portok kifejtése, illetve a falban futó kábelek elhelyezése, rendezése, stb.

Füzesi Szabolcs
fuzesisz@osi.hu
MCSA

Felhasznált internetes szakirodalom:

http://www.netspec.com/helpdesk/wiredoc.html

Univerzális Plug and Play

Az önműködő hálózat

A UPnP hálózat segítségével megvalósítható a hálózati eszközök automatikus csatlakoztatása és eltávolítása, a műveletek semmilyen emberi beavatkozást nem igényelnek; az összes szükséges információt maguk az eszközök hordozzák és hirdetik meg a hálózaton.

Mit jelent az univerzális Plug and Play?

A különféle hardvereszközök és az operációs rendszerek Plug and Play képességekkel való felruházása jelentősen megkönnyítette a hardvereszközök telepítését és beállítását. Az univerzális Plug and Play *(UPnP)* ezt az egyszerűséget terjeszti ki a teljes hálózatra, lehetővé téve a hálózati eszközök *(hálózati nyomtatók, átjárók, különféle elektronikai eszközök)* automatikus felderítését és használatba vételét.

A UPnP több, mint a Plug and Play modell kiterjesztése, segítségével megvalósítható a beállítást nem igénylő, „láthatatlan" hálózat, amely képes a különféle gyártóktól származó különféle eszközök automatikus érzékelésére és integrálására.

A UPnP segítségével az eszközök dinamikusan csatlakoznak a hálózathoz, meghirdetik képességeiket, és információt gyűjtenek más eszközök jelenlétéről és képességeiről.

A UPnP képességekkel felruházott hálózat szolgáltatásait a legkülönfélébb eszközök vehetik igénybe, felhasználási területe pedig igen sokféle lehet, például:

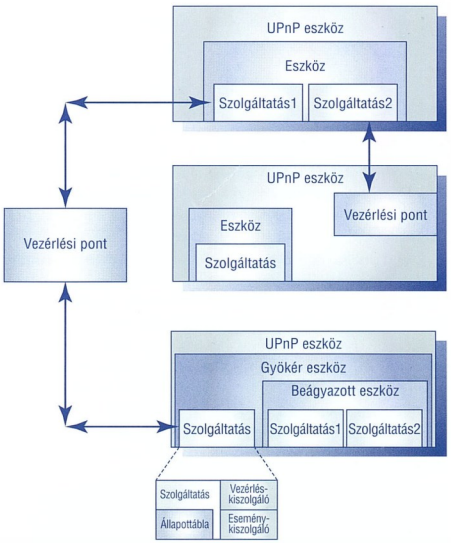
- Háztartás-automatizálás
- Nyomtatás és képfeldolgozás
- Audio és videoeszközök
- Konyhai gépek
- Gépjárművek fedélzeti eszközei

A UPnP a szabványos TCP/IP protokollt használja, így könnyen beilleszthető a meglévő hálózatokba. Mivel a UPnP nyílt, elosztott hálózati architektúra, amelyet a felhasznált protokollok határoznak meg, független lehet bármely adott operációs rendszertől, programozási nyelvtől, vagy fizikai médiumtól *(akárcsak az Internet)*. A UPnP nem definiálja az alkalmazások által használandó API-kat, így az operációs rendszerek gyártói hozhatják létre a megrendelőik igényeinek leginkább megfelelő API-t.

A UPnP eszközöket és szolgáltatásokat *(Device and Service Descriptions)* a Universal Plug and Play Forum definiálja a Microsoft által kifejlesztett eszközarchitektúrának megfelelően. A Universal Plug and Play Forum az iparág azon vállalataiból álló csoport, amelyek szerepet kívánnak játszani a UPnP eszközök és szolgáltatások kialakításában. A szervezet 1999. október 18-án alakult, és jelenleg több mint 625 gyártó tartozik hozzá.

A UPnP hálózat komponensei

A UPnP hálózat három komponens-típust tartalmaz: eszközöket, szolgáltatásokat és vezérlési pontokat.



UPnP vezérlési pontok, eszközök és szolgáltatások

- **Eszközök** – A UPnP eszköz szolgáltatásokból és beágyazott eszközökből épül fel. Például egy videomagnó szalagtovábbító, TV vevő, óra, stb. szolgáltatásokból áll, míg a TV/videomagnó kombináció a szolgáltatásokon kívül több együttműködő eszközt is tartalmaz. Az eszközre vonatkozó valamennyi információ az eszköz részét képező XML formátumú leírásban található meg. Minden eszköz több szolgáltatást is tartalmazhat.
- **Szolgáltatások** – A UPnP hálózatban a vezérlés legkisebb egysége a szolgáltatás. Minden szolgáltatáshoz műveletek és állapotváltozók tartoznak. Például az óra szolgáltatás modellezhető egy állapotváltozó *(aktuális_idő)* és két művelet *(idő_beállítás, idő_lekérdezés)* felhasználásával. Az eszközök leírásához hasonlóan ezek az információk is a UPnP Forum által definiált XML formátumú szolgáltatás leírásban találhatók. Minden szolgáltatás állapottáblából, vezérléskiszolgálóból és eseménykiszolgálóból áll. Az állapottáblában történik a szolgáltatás állapotváltozóinak nyilvántartása. A vezérléskiszolgáló fogadja a műveleti kérelmeket *(pél-*

dául idő_beállítás), végrehajtja azokat, frissíti az állapottáblát, és válaszüzeneteket küld. Az eseménykiszolgáló a szolgáltatás állapotváltozásairól szóló üzeneteket juttat el a megfelelő címzetteknek.

- **Vezérlési pontok** – Az UPnP hálózat vezérlési pontja képes más eszközök felderítésére és irányítására. A sikeres felderítés után a vezérlési pont képes:
  1. Az eszköz, és az általa nyújtott szolgáltatások leírásának bekérésére
  2. Az eszköz szolgáltatásaihoz kapcsolódó műveletek kezdeményezésére
  3. A szolgáltatások által küldött eseményüzenetek fogadására

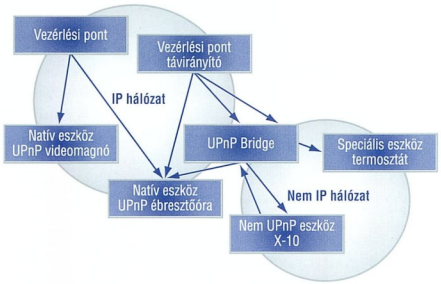
A valódi egyenrangú hálózat megvalósításának érdekében célszerű, ha az eszközök egyben a vezérlési pont funkciókat is tartalmazzák.

A UPnP protokoll áttekintése

Hálózati média

A UPnP hálózat komponenseit bármilyen kommunikációs csatornával összekapcsolhatjuk, a média lehet telefonvonal, IrDA, Ethernet, IEEE1394 stb. Bármilyen médium, amely alkalmas a hálózati eszközök összekötésére, megfelelő a UPnP hálózat számára is. Az egyetlen megkötés, hogy a médiának biztosítania kell a szükséges sávszélességet.

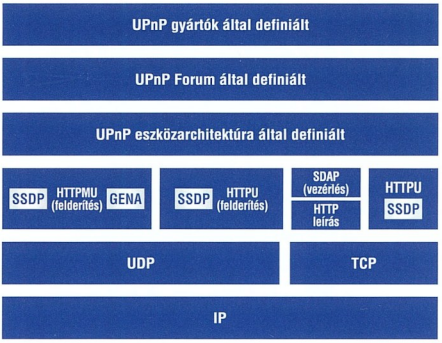
A UPnP nyílt, szabványos protokollokat használ *(TCP/IP, HTTP, XML)*, képes azonban más technológiákkal *(például HAVi, CeBus, LonWorks, EIB, X10 stb.)* való együttműködésre is UPnP bridge, vagy proxy közbeiktatásával.



Bridge-t tartalmazó UPnP hálózat

Protokollok

A szabványos protokollok használata biztosítja az egyes gyártók implementációinak együttműködését. A UPnP által felhasznált protokollok igen elterjedtek az Interneten és a lokális hálózatokon is, így széles körben megvannak a szükséges ismeretek az ezekre alapuló megoldások kifejlesztéséhez és bevezetéséhez. A továbbiakban az UPnP implementálásához felhasznált protokollok összefoglaló leírása következik:



Az UPnP protokoll-verem

- **TCP/IP** – A UPnP-t alkotó protokollok alapját a TCP/IP protokollkészlet képezi. A UPnP eszközök a TCP/IP számos protokollját *(TCP, UDP, IGMP, ARP, IP)* és szolgáltatását *(DHCP, DNS)* használják.
- **HTTP, HTTPU, HTTPMU** – A HTTP protokoll és variánsai szintén alapvető részei a UPnP-nek. A HTTPU *(unicast)* és a HTTPMU *(multicast)* a HTTP protokoll variánsai, amelyek TCP helyett UDP csomagokat használnak az üzenetek közvetítésére. A protokollok által használt üzenetformátum alapvetően megegyezik a HTTP által használttal. Az ilyen típusú üzenetek a multicast kommunikációban, és a megbízhatóságot nem igénylő üzenetek közvetítésében játszanak szerepet.
- **SSDP** – Az egyszerű szolgáltatás-felderítési protokoll *(Simple Service Discovery Protocol)*, mint neve is mutatja a hálózati szolgáltatások felderítéséért felelős. Az SSDP a HTTPU és HTTPMU protokollokra épül, és meghatározza a vezérlési pontok által az erőforrások megtalálásához szükséges metódusokat. Az SSDP protokollt használják továbbá a különféle eszközök, hogy jelenlétüket meghirdessék a hálózaton.

Az UPnP vezérlési pontok aktiválásuk után SSDP keresési kérelmet küldenek szét *(HTTPMU protokoll fölött)*, hogy a hálózaton elérhető eszközöket és szolgáltatásokat felderítsék. A vezérlési pont képes keresési kritériumok megadására is, például a keresési kérelem vonatkozhat egy adott eszköztípusra, vagy szolgáltatásra.

A UPnP eszközök a multicast portot figyelik, és keresési kérelem érkezése esetén az eszköz megvizsgálja a keresési kritériumot. Ha az eszköz megfelel a kérelemben szereplő kritériumnak, unicast SSDP *(HTTPU felett)* választ küld a vezérlési pontnak.

Hasonló módon, ha egy adott eszköz csatlakozik a hálózathoz, SSDP jelenléti értesítőket küld szét, amelyekben meghirdeti az általa nyújtott szolgáltatásokat.

- **GENA** – A GENA *(Generic Event Notification Architecture, általános esemény-értesítési architektúra)* biztosítja a TCP/IP és multicast UDP feletti HTTP értesítések küldését és fogadását. A vezérlési pont, amely esemény-értesítéseket szeretne kapni egy adott szolgáltatás állapotváltozásairól, igénybejelentést

(subscription request) küld az adott eszköz eseménykiszolgálójának, amely tartalmazza a szolgáltatás azonosítását, a saját címét, és azt az időtartamot, ameddig az értesítéseket szeretné megkapni. Az értesítések küldésére vonatkozó kérelmet periodikusan meg kell újítani, illetve a GENA segítségével le is lehet mondani.

- **SOAP** – A SOAP *(Simple Object Access Protocol, egyszerű objektum-hozzáférési protokoll)* definiálja az XML és a HTTP használatát a távoli eljáráshívások *(RPC)* végrehajtásában. A SOAP az Interneten keresztüli RPC kommunikáció szabványává vált. A UPnP a SOAP használatával továbbítja a vezérlőüzeneteket az eszközök felé, és a vezérlési pontok ennek segítségével kapják vissza a válaszokat és a hibaüzeneteket.
- **XML** – Az XML *(Extensible Markup Language, kiterjeszthető leírónyelv)* a strukturált adatok továbbításának univerzális webes formátuma. Az XML segítségével csaknem bármilyen strukturált adathalmaz egyszerű szövegfájllá alakítható. Az XML-t a UPnP a szolgáltatás-leírások tárolásához, a vezérlési üzenetekhez és az eseménykezeléshez használja.

A UPnP működése

A UPnP működése az alábbi hat lépésre tagolható:

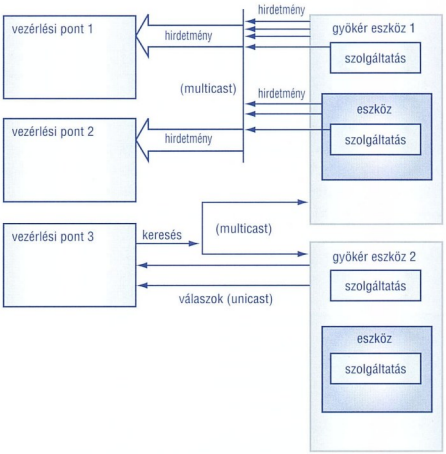
- Címkiosztás *(addressing)*
- Felderítés *(discovery)*
- Leírás *(description)*
- Vezérlés *(control)*
- Eseménykezelés *(eventing)*
- Megjelenítés *(presentation)*

Címkiosztás *(addressing)*

A UPnP hálózat működésének alapja az IP-címek kiosztása. Minden eszköz DHCP ügyfélszoftvert tartalmaz, és a hálózathoz való első csatlakozáskor DHCP kiszolgálótól próbál IP-címet kérni. Ha a hálózaton működik DHCP kiszolgáló *(felügyelt hálózat)*, az eszköz a megkapott IP-címet fogja használni. Ha nincs elérhető DHCP kiszolgáló *(felügyelet nélküli hálózat)*, az eszköz automatikusan választ magának IP-címet *(Auto-IP)*.

Felderítés *(discovery)*

Ha új eszköz csatlakozik a hálózathoz az UPnP felderítési protokollja *(SSDP)* teszi lehetővé, hogy az eszköz meghirdesse az általa biztosított szolgáltatásokat a hálózat vezérlési pontjai felé. Csatlakozáskor az eszköz multicast felderítési üzeneteket küld szét a hálózaton. Hasonlóan, ha vezérlési pont csatlakozik a hálózathoz, az SSDP protokoll használatával keresi meg azokat az eszközöket, illetve szolgáltatásokat, amelyeket vezérelni fog.
Ha valamely eszközt eltávolítunk a hálózatból, szintén multicast üzenetekben közli, hogy beágyazott eszközei és szolgáltatásai már nem érhetők el.



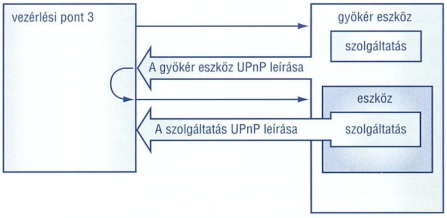
Felderítés a UPnP hálózatban

Leírás *(description)*

Miután a vezérlési pont felderített egy adott eszközt, még igen kevés információval rendelkezik róla. Csak a felderítési üzenetben szereplő információk jutottak el hozzá:

- az eszköz *(vagy szolgáltatás)* UPnP típusa
- az eszköz egyedi azonosítója
- URL az eszköz UPnP leírásához

Hogy a vezérlési pont többet is megtudhasson az eszközről és képességeiről, meg kell szereznie a felderítési üzenetben szereplő URL által megcímzett leírást.



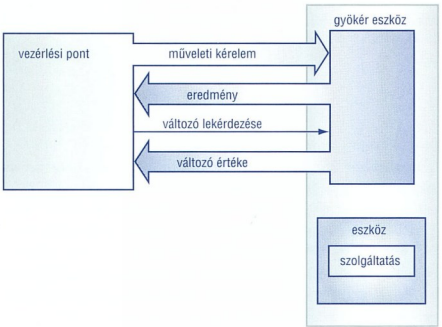
A leírások eljuttatása a vezérlési pontokhoz

Az eszközök UPnP leírása logikailag két részre tagolódik: az eszközleírásra, ami a megvalósított logikai vagy fizikai tárolók leírása, és a szolgáltatásleírásokra, amelyek az eszköz képességeiről tartalmaznak információt.
A UPnP eszközleírások gyártóspecifikus információkat is tartalmaznak, például az eszköz nevét, szériaszámát, a gyártó nevét, weboldalának URL-jét, stb. Az eszköz által megvalósított minden egyes szolgáltatásra vonatkozóan a leírás tartalmazza a szolgáltatás típusát, nevét, valamint URL-t a szolgáltatás leírásához, vezérléséhez és eseménykezelő rendszeréhez.
Ha az eszközt eltávolítjuk a hálózatból, az multicast felderítési üzenetek formájában közli a vezérlési pontokkal, hogy

beágyazott eszközei és szolgáltatásai a továbbiakban nem lesznek elérhetőek.

Vezérlés *(control)*

Az eszközről és szolgáltatásairól szóló információk birtokában a vezérlési pont különféle műveletek elvégzésére szólíthatja föl a szolgáltatásokat, és lekérdezheti azok állapotváltozóinak értékét. A műveletek kezdeményezése távoli eljáráshívásként fogható fel; a vezérlési pont elküldi a kívánt műveletre vonatkozó kérést, a művelet befejezése *(vagy sikertelensége)* után pedig a szolgáltatás visszaküldi az eredményt, illetve a hibaüzeneteket.



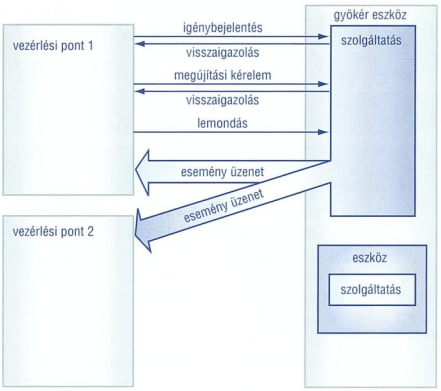
Eszközök vezérlése

Az eszköz által biztosított szolgáltatás vezérlése úgy történik, hogy a vezérlési pont a szolgáltatás leírásában szereplő vezérlési URL-re küldi a megfelelő üzenetet. Az üzenet hatására elvégzett művelet megváltoztathatja a szolgáltatás állapotváltozóit, ez eseményt vált ki, amiről az eseménykezelő rendszer értesíti a megfelelő vezérlési pontokat.
A vezérlési pont az állapotváltozók aktuális értékeit is lekérdezheti a szolgáltatástól. A lekérdezés a vezérlési URL-re küldött megfelelő formátumú lekérdezési üzenet segítségével történhet, amelyre válaszként a szolgáltatás elküldi az állapotváltozó értékét.

Eseménykezelés *(eventing)*

Miután a vezérlési pont felderítette az eszközt, és lekérte annak leírását, már képes az eszközzel kapcsolatos események kezelésére. Hogy az eseményekről szóló üzeneteket megkaphassa, be kell jelentenie igényét *(subscription request)* a szolgáltatásnak.

A szolgáltatás az állapotváltozók értékének megváltozásáról eseményüzeneteket küld a megfelelő vezérlési pontoknak. Az üzenet az állapotváltozók nevét és aktuális értékét tartalmazza XML formátumban.
A vezérlési pontoknak bizonyos időközönként meg kell újítaniuk bejelentkezésüket az adott esemény értesítéseire.



Eseménykezelés

Megjelenítés *(presentation)*

Ha az eszköz rendelkezik a megjelenítést biztosító URL-lel, a vezérlési pont képes az adott oldal letöltésére és böngészőben való megjelenítésére. A felhasználó az oldal tulajdonságaitól függően vezérelheti az eszközt és/vagy megtekintheti annak állapotát.

Szerényi László
szerenyi.l@met.hu

A cikkben szereplő URL-ek:
[1] http://upnp.org

A DHCP rejtett szépségei – VII.

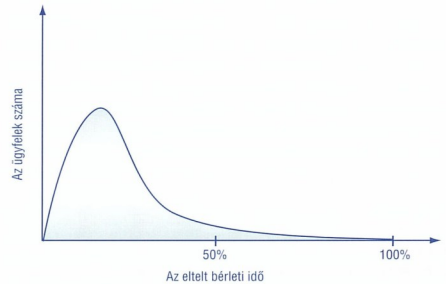
A cikksorozat befejező részében három fontos témakört elemzünk. Megvizsgáljuk, milyen lehetőségeink vannak a DHCP-szolgáltatás rendelkezésre állásának növelésére, áttekintjük a beépített monitorozó eszközöket, végül néhány hasznos fogást ismertetünk, amelyek a DHCP-adatbázis karbantartását segíthetik.

A rendelkezésre állás növelése

Akik figyelemmel kísérték a tech.net magazin DHCP-cikksorozatát, pontosan tudják, hogy a címkiosztó szolgáltatás kritikus funkció, s bár nem kell minden másodpercben működnie, a hibás szerver lassan, de biztosan megbénítja az egész hálózat életét. Szerencsére számos lehetőség van a kezünkben, hogy a rendelkezésre állást növeljük. Az alábbi pár módszer közül a vékonyabb pénztárcájú és a professzionális szolgáltatók is megtalálhatják a nekik megfelelő megoldást.

Rendelkezésre állás növelése „józan ésszel"

Szőr mentén már említettük, most újra elővesszük az egyik legegyszerűbb és legkézenfekvőbb megoldást, amellyel akár napokat is nyerhetünk a DHCP helyreállításához. A címkérés módszeréből, valamint az Windowsos ügyfelek alapértelmezett beállításából következik, hogy a szolgáltatást csak rövid időre igénylik az IP-címmel dolgozó eszközök. Egy ügyfél akkor kerül kapcsolatba a DHCP-szerverrel, amikor először címet kér, újraindul, amikor meg kell újítani a bérletet a bérletidő felénél, sikertelenség esetén a háromnegyedénél, végül a bérlet tényleges lejáratakor. A kontaktusok közül csak az első és az utolsó kritikus, vagyis ha a kliensnek eleve nem volt címe, vagy végleg lejárt a bérlete. Minél hosszabb a két időpont közötti idő – vagyis a bérletidő, annál nagyobb a valószínűsége, hogy nem ilyen kritikus helyzet áll fenn.
Vagyis a bérletidő növelése önmagában a „rendelkezésre állást" növelő tényező. Persze csak statisztikai alapon. A bérletidő függvényét vizsgálva azt tapasztalhatnánk, hogy az ügyfelek legtöbbje a bérlet egynegyedénél jár. Miért? Mert ha jól működik a DHCP-szolgáltatás, akkor a gépek túlnyomó többsége félidőben megújítja a bérletét, vagyis a bérletidő 0 és 50%-a között egy normális jellegű eloszlás jellemzi a gépek címbérleteit.



A ügyfélszámítógépek eloszlása az eltelt bérletidő függvényében

Tökéletes normális eloszlásról azért nem beszélhetünk, mert mindig lesznek olyan gépek, amelyek kikapcsolt állapotban „átalusszák" a félidőt, és megjósolhatatlan, hogy mikor „ébrednek". Ezzel együtt egy kevés gépből *(20-60)* álló hálózatban, naponta használt állomásokkal a normális eloszlás nagyon jó közelítés. Belátható, hogy egy 90 napos bérletperiódus esetén majdnem 45 nap áll rendelkezésre a DHCP-szolgáltatás megjavításához – megint csak statisztikai alapon. Akkor javasolt ehhez a taktikához folyamodni, ha:

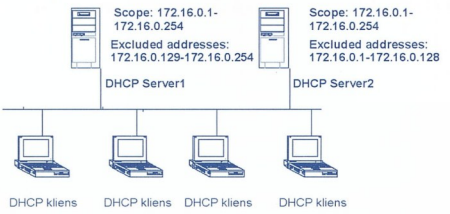
- Egyetlen szerverünk van.
- Viszonylag kicsi, jól átlátható hálózattal dolgozunk.
- Stabil a környezetünk, tehát nem konfiguráljuk át gyakran a DHCP-kiszolgálót, nem vásárolunk éppen most gépeket, nem adunk új IP-cím tartományt a hálózatnak.
- Nincsenek olyan ügyfeleink, amelyek egy másik hálózatban is rendszeresen megfordulnak, és ott címet igényelnek.
- Nincsenek BOOTP ügyfeleink. *(Ők újrainduláskor mindenképp igényelnek IP-címet.)*
- Nem konfiguráltuk úgy az ügyfeleket, hogy leálláskor „engedjék el" a címüket.

Bár kemény korlátok a fentiek, azért szép számmal akad olyan hálózat, amely minden kritériumot teljesít. Fontos megemlíteni, hogy a több telephely nem akadály, hisz az ügyfél számára transzparens, vajon egy valódi címkiszolgáló szervertől kapja a címet, vagy egy Relay Agent közvetíti azt.
Nem ajánlott a megoldás extrapolálása a bérletidő végtelenre állításával. Igaz ugyan, hogy ekkor nincs félidő sem, tehát a haranggörbe a fenti ábra bal tengelyére „vasalódik", ám a megoldásnak több a hátulütője, mint az előnye. A végtelen bérletperiódussal ellátott ügyfél csak akkor reflektál a DHCP-opciókban bekövetkezett változásokra ha újraindítják, ám ez még csak a kisebbik baj.
A nagyobb gond, hogy a DHCP-kiszolgáló nem értesül arról, ha egy gépet végképp kivonnak a hálózatból. A kiadott cím „örökre" az adott MAC-Address-hez tartozik, tehát elvész. A cím visszanyeréséhez pontos nyilvántartással kell rendelkeznünk a hálózatunk található valamennyi DHCP ügyfél MAC-címéről. Ez túl sok többletmunka a végtelen bérletért cserébe. Igazából épp ezt a nyilvántartást teszi feleslegessé a címkiosztás.
A fenti módszer, bár működik, igazából a szolgáltatás tehermentesítésével teremt „rendelkezésre állást". Ha a statisztikai elv nem fogadható el, vagy nem alkalmazható, egy komolyabb és megbízhatóbb megoldást kell keresünk.

80-20-as *(50-50-es)* megoldás

Ha nem csak egyetlen kiszolgálóval rendelkezünk, hanem legalább kettővel, egy ügyes trükkel biztosíthatjuk, hogy az ügyfelek mindig hozzájussanak a mindennapi IP-címükhöz.

A módszer igen egyszerű. Hozzunk létre két DHCP kiszolgálót, és konfiguráljuk mindegyiken egy-egy azonos címtartományra vonatkozó bérlettartomány. Ezután az egyik kiszolgálón zárjuk ki a címek felső húsz *(vagy ötven)* százalékát. Végezzük el ugyanezt a műveletet a másik szerveren is, de ott az alsó 80 *(vagy 50)* százaléknyi címet zárjuk ki. Az eredményt a második ábra mutatja.



A 80-20-as (50-50-es) eljárás alkalmazása

A végeredmény két, azonos címtartományt kezelő, de pontosan azonos címeket soha ki nem osztó szerverpár lesz. Az eljárás alkalmazásakor a következőre kell figyelni:

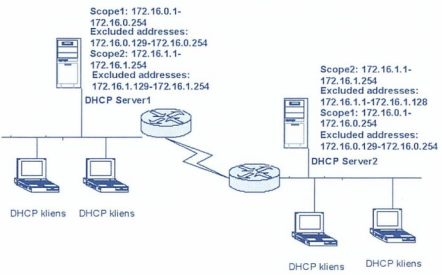
- Véletlenül sem fordulhat elő, hogy azonos dinamikus címet mindkét szerver kiadhasson, tehát a címek között nem lehet átfedés.
- A két scope opcióinak teljesen azonosnak kell lennie, beleértve a bérletidőt és a speciális opciókat is.
- A lefoglalt címeket *(reserved addresses)* mindkét kiszolgálón be kell állítani, beleértve a kiegészítő opciókat is, ha vannak ilyenek.
- A BOOTP címeket a lefoglalt címekhez hasonlóan kell kezelnünk, vagyis kétszer kell felvennünk minden rekordot.
- A címtartomány nagyságát úgy érdemes megválasztani, hogy akár egy kiszolgáló is képes legyen valamennyi potenciális ügyfelet kiszolgálni. Ha tehát van 300 gépünk, és 150-et szolgál ki egy DHCP-szolgáltatás, akkor a bérlettartomány tartalmazzon legalább 300 címet. Manapság, a privát címtartományok használatakor ez ritkán probléma. *(Ugyanakkor gyakoribb az 50-50%-os megosztása a bérlettartománynak, ezért szerepel ez az ábrán.)*
- Stabil a környezetünk, tehát nem konfiguráljuk át gyakran a DHCP-kiszolgálót, nem vásárolunk éppen most gépeket, nem adunk új IP-cím tartományt a hálózatnak.

Nézzük, mit nyerünk a konfigurációval. A módszer valódi rendelkezésre-állás növekedést jelent, mert az egyik kiszolgáló kiesése esetén is elláthatók az ügyfelek IP-címekkel. Akár kézzel is nekiállhatunk egy teljesen új DHCP kiszolgáló felállításának, ha kiesne az egyik, hiszen van egy „tükörképünk", amit csak le kell másolnunk. Nem akadályoznak az előző megoldásnál felsorolt kritériumok sem, lehetnek vándorló gépeink, BOOTP klienseink, tetszőlegesen konfigurálhatjuk az ügyfeleket, skálázható megoldáshoz jutunk – amely ráadásul mindig működik. Mindezt úgy, hogy nem kellett extrán konfigurált hardvert vásárolnunk, sem új technológiát bevezetnünk, nincs szükség a Windows Server drágább változatára. Olyannyira nincs, hogy a „tükörszerver" akár másik operációs rendszert is futtathat, például Linuxot. A feltétel csupán annyi, hogy a DHCP-szabvány pontosan kövesse a másik rendszer is, legalábbis mindazt tudja, amit mi használunk a szabványból *(esetleg: superscope, különleges opciók* (class, vendor stb.)*)*

A magasabb színvonalú szolgáltatásnak azonban ára van. Lényegében kétszer kell elvégeznünk minden konfigurációs műveletet. Ha ritkán változó scope-pal dolgozunk ez kevésbé fájdalmas, ám ha sok lefoglalt címünk van, netán még BOOTP ügyfeleket is kiszolgálunk, akkor nehéz két szerveren precízen azonos adatbázist fenntartani. *(Ilyenkor javasolt minden műveletet parancssorból végezni, mert a megkonstruált parancssort könnyű két szerverre lefuttatni.)*

Katasztrófatűrő megoldás

Az 50-50-es módszert továbbfejleszthető valódi katasztrófatűrő megoldássá. A következő ábra két telephelyből felépülő hálózatot mutat az előbb ismertetett DHCP-konfigurációval.



Katasztrófatűrő megoldás

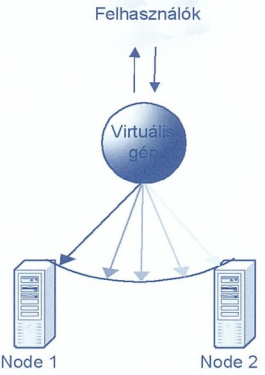
Az eredeti módszert ki kell egészíteni az útválasztók konfigurálásával *(vagy egyéb módon kell biztosítani DHCP Relay Agenteket)*. A rajzról leolvasható, hogy mindkét telephelyen mindkét DHCP-szerver megkapja az ügyfelek kérését. Ha azonban a továbbító ügynökök némi késéssel indítják el a címigénylő csomagokat a távoli hálózat felé, normál működéskor a helyi kiszolgáló „győz". Meghibásodáskor viszont a távoli rendszer mindenképp kiszolgálja az ügyfeleket. A módszer katasztrófatűrő, amennyiben az egyik szerver fizikai megsemmisülése esetén is működőképes marad a címkiosztás. Egyéb vonatkozásaiban a megoldás pontosan ugyanazokat az előnyöket és hátrányokat hordozza, mint az „egy telephelyes" kialakítás.

A fürtözés

A Windows 2000 Advanced Server az első olyan Windows kiszolgáló változat, amely lehetővé teszi, hogy megfelelően kialakított hardverrel a DHCP szolgáltatást fürtözhessük.

A megoldás ezúttal a virtualizálás. A fürtök állomásokból *(node)* és virtuális szerverekből állnak. A virtuális szerverek erőforrásokat tartalmaznak. Ilyen erőforrás lehet egy fizikai le-

mez, egy IP cím, egy hálózati név, vagy éppenséggel egy DHCP-szolgáltatás. A fürt elvi működését mutatja a következő ábra:



A DHCP-szolgáltatás a virtuális szerveren található

A kliensek sohasem az egyik, vagy másik állomástól kapják a fürtözött szolgáltatást, hanem mindig egy virtuális szervertől. Ez a szerver szükség szerint költözhet egyik állomásról a másikra. *(Windows Server 2003 négy, vagy akár 8 node egyikére.)* Az átköltözési idő az erőforrások számától és terheltségétől függ. Ha a DHCP-szolgáltatás számára egy saját virtuális szervert definiáltunk, akkor az átköltözés 6-10 másodperc. Ennyi tehát a kiesés – tulajdonképpen semennyi.

A fürtözött DHCP-kiszolgáló, a virtuális erőforrás létrehozását leszámítva, semmiben sem különbözik egy „normális" címkiosztó szervertől. A fürt nem ad hozzá és nem vesz el a teljesítményből vagy a funkcionalitásból, csupán a rendelkezésre állást növeli.

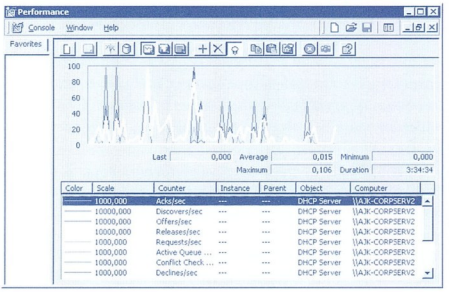
A megoldás előnye az előzőekhez képest az egyszeres konfiguráció. Előnyös lehet sok *(max: 10000!)* scope kezelésekor, nagyszámú lefoglalt cím nyilvántartásakor, vagy BOOTP ügyfelek kiszolgálása esetén. Akárcsak az előző két módszer, ez is valódi rendelkezésre állást növelő eljárás.

Persze a „tökéletes" megoldásnak ára van: a fürt kialakítása host-bus adaptereket, közös háttértárat, speciális HCL listán hitelesített konfigurációt feltételez. Az operációs rendszer vagy két Windows 2000 Advanced Server, vagy *(legalább két)* Windows 2003 Server Enterprise Edition lesz fürtözéskor. Ez jelentős többletköltség, mert a standard változatokhoz képest a licencár legalább tízszeres *(ha mindkét állomást számolom.)* Meggondolandó, hogy a fürtök felépítése, működtetése szakértelmet kíván, a hibajavításkor szakértői költségek merülhetnek fel, hacsak nem profik az üzemeltetők.

Láthatjuk, hogy a rendelkezésre állást segítő technológiák igen sokfélék. A különbözőségük lehetőséget teremt arra, hogy ötvözzük őket, ha van rá lehetőségünk és a feltételek adottak. A fürtözés nem zárja ki a hosszú bérletidő használatát, ha pedig két fürtünk is futtat DHCP-kiszolgálót különböző telephelyen, földrajzilag elosztott megoldást alakíthatunk ki az 50-50-es megoldással. Az egyes módszereket építőelemnek tekinthetjük, és tetszés szerint készíthetünk akár nagyon bonyolult címkiosztó architektúrákat is. Csupán a fantázia *(és a pénz)* szab határt az elképzeléseinknek. Egy fontos tanácsot azonban érdemes megfogadni: lehet bonyolult, amit elkészítünk, csak legyen jól dokumentált. A rendelkezésre állás egyik halálos ellensége a hiányos dokumentáció. A másik a felügyelet hiánya.

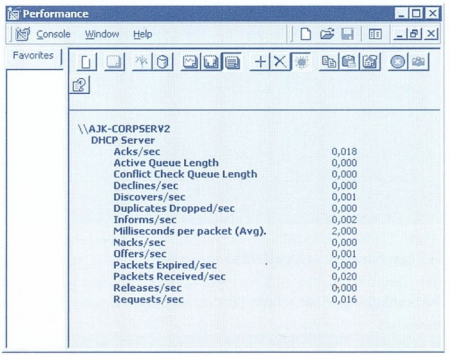
A DHCP kiszolgálók monitorozása

A kiszolgálók figyelése a rendszergazdák napi feladatai közé tartozik. Ha figyelembe vesszük a DHCP fontosságát, ez még inkább így van. Amikor azonban az ember szembesül a teljesítménynapló számaival, hajlamos kissé visszavenni a lendületből.



A helyi tömegközlekedés menetrendje DHCP számlálókba csomagolva

A fenti diagram az egyik nagyobb telephelyünk DHCP-kiszolgálójának teljesítményadatait mutatja. A napló eredetileg a teljes 24 órát rögzítette, én azonban csak az 5.30 és 9.00 közötti időszakot ábrázoltam. Mivel kezdetben csupa kisimított vonalakat láttam, elkezdtem „nagyítani" a diagramot, jól látszik, hogy ezer és tízezerszeres „nagyítás" hozott eredményt. Végül pusztán az adatok felé fordulva *(és tovább szűkítve az időt 5.30 és 8.00 közé)* a következőt mondhatjuk a telephelyről.

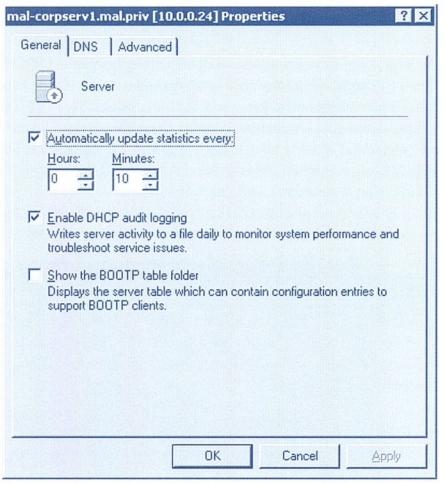


Néhány adat a reggeli indulásról

Vannak kimutathatatlanul alacsony értékek, például nem alakult ki „sor", hogy a DHCP leellenőrizze keletkezne-e konfliktus egy cím kiadásakor. Ugyancsak nem történt elutasítás *(Nack/sec)*, ami leginkább azt jelenti, hogy nem érkezett olyan notebook a hálózata, amely korábban más alhálózatban kapott címet. A címjóváhagyások *(Ack/sec)* átlagos értéke 0,018 másodpercenként, tehát 1,08 percenként 64,8 óránként. A két és fél óra alatt tehát nagyjából 162 gépet kapcsoltak be, amelyek azután jóváhagyott címmel elindultak. Persze a többség eleve létező bérlettel indult, a discover csomagot is kibocsátó gépek száma *(0,001x60x60x2,5)* mindössze 9. Ha elárulom, hogy kb. 10 gép állandóan üzemel, 10-et pedig átlagosan nem kapcsolnak be, máris megkaptuk, hogy kb. 190 gépet lát el a címkiosztó rendszerünk. A DHCP kiszolgáló „teljes terheltsége" 0,020 csomag másodpercenként, vagyis csak 1,2 csomag percenként. Ez a szolgáltatás vélhetően nem fogja megizzasztani a legkisebb szervereket sem.
Tudom, hogy vannak ennél jóval nagyobb implementációk is. Egy Internetszolgáltató akár százezer címet is kioszthat egyetlen este – nagyvállalati környezetben azonban a DHCP szerver méretezése és terhelése egyszerűen nem lehet probléma.

A Diagnostic log

A mindennapi hibaelhárítás során a teljesítménynaplónál is fontosabb lehet a Windows 2000-ben bevezetett diagnostic vagy más néven audit log.
Az audit log használata nem kötelező, a szerver tulajdonságai párbeszédpanelen lehet bekapcsolni, az advanced fülön pedig a naplóállományok helyét is megadhatjuk.



Egy hasznos szolgáltatás – az audit

A következő pár sor a fenti szerver naplója ugyanaznap.

```
            Microsoft DHCP Service Activity Log

Event ID   Meaning
00    The log was started.
01    The log was stopped.
02    The log was temporarily paused due to low
disk space.
10    A new IP address was leased to a client.
11    A lease was renewed by a client.
12    A lease was released by a client.
13    An IP address was found to be in use on the
network.
14    A lease request could not be satisfied
because the scope's address pool was exhausted.
15    A lease was denied.
16    A lease was deleted.
17    A lease was expired.
20    A BOOTP address was leased to a client.
21    A dynamic BOOTP address was leased to a
client.
22    A BOOTP request could not be satisfied
because the scope's address pool for BOOTP was
exhausted.
23    A BOOTP IP address was deleted after
checking to see it was not in use.
50+    Codes above 50 are used for Rogue Server
Detection information.

ID Date,Time,Description,IP Address,Host Name,MAC
Address
51,11/03/03,00:35:43,Authorization
succeeded,,mal.priv,
57,11/03/03,00:35:43,Server found in our
domain,10.5.0.14,mal.priv,
11,11/03/03,00:47:07,Renew,10.5.1.32,AJK-
056.mal.priv,00A0C9417B89
51,11/03/03,01:36:15,Authorization
succeeded,,mal.priv,
57,11/03/03,01:36:15,Server found in our
domain,10.5.0.14,mal.priv,
11,11/03/03,02:06:04,Renew,10.5.1.38,AJK-
099.mal.priv,00A0C9419FB2
51,11/03/03,02:36:47,Authorization
succeeded,,mal.priv,
```

A napló szerkezete meglehetősen egyszerű, táblázatba foglalva a következő:

Mező	Leírás
ID	Az eseményt azonosító kód
Dátum	A bejegyzés dátuma
Idő	A bejegyzés pontos időpontja
Leírás	Az esemény leírása
IP-cm	Az ügyfél IP-címe
Host név	Az ügyfél host neve
MAC cím	Az ügyfél hálózati kártyájának MAC címe

A napló elején – angolul – felsorolják a legfontosabb eseménykódokat. Ilyen lehet az indulás, egy címbérlet kiadása, egy cím megújítása stb. Az 50 feletti kódok a szerverek felhatalmazásával kapcsolatosak, a DHCP súgójában minden kódot részletesen leírnak.
Az audit log hallatlan előnye, hogy minden eseményt feljegyez, ha tehát a címkiosztással, az ügyfelekkel való kommunikációval vagy a felhatalmazással bármi problémánk lenne, ez lesz az első számú segédletünk a probléma elhárításakor.

Nem részletezzük, csak megemlítjük, hogy a DHCP természetesen a Windows 2000 eseménynaplójába is képes elhelyezni bejegyzéseket. Akinek a fenti megoldások egyike sem nyújtja azt, amit ő elvár, akkor még mindig adott a lehetőség, hogy SNMP alapú megfigyelést végezzen *(a DHCP külön MIB adatbázissal rendelkezik)* vagy akár a Microsoft Operation Manager beépített tudástárát is használhatja. A naplózó és monitorozó rendszerek további boncolgatása helyett áttérünk egy másik nagyon fontos témára, a DHCP adatbázis kezelésére és karbantartására.

A DHCP kiszolgálók adatbázisa

Volt olyan kollégám, aki kétkedését fejezte ki, amikor elmeséltem neki, hogy a DHCP-adatbázisokról akarok írni. Szerinte, ha baj van, egyszerűen „csinálunk egy új kiszolgálót" és kész. Nos, valóban, az esetek nem elhanyagolható részében ez egy járható út. Igaz, előfordulhat esetlegesen címütközés, de kisebb hálózatoknál ez is ritka. Vannak azonban esetek, amikor egy DHCP-adatbázis fontossá válhat. Ha nagyszámú lefoglalt címünk van, vagy BOOTP ügyfelünk, ha több scope-ot is működtetünk *(akár több tucatot)*, ha különleges paraméterekkel, osztályazonosítókkal láttuk el az ügyfeleinket, talán nem mindegy, hogy újabb hosszú napokat töltünk el az adatbázisunk kialakításával, hagyjuk, hogy záporozzon a felhasználók panaszáradata a címütközések miatt, vagy némi erőfeszítéssel megmentjük az adatbázisunkat az enyészettől.
A legfontosabb fogásokat tekintjük át: megvizsgáljuk az adatbázis működési elvét, majd néhány kritikus műveletet is ismertetünk, mint például az adatbázis mozgatása, javítása, mentése és helyreállítása.

A DHCP-adatbázis szerkezete

A DHCP a Microsoft által több termékben is alkalmazott JET adatbázismotort használja. *(Ez az alapja többek között az Access, az Exchange 5.5 és a WINS szolgáltatások.)* A hasonlóság olyan fokú, hogy aki jártasságot szerzett már egy másik termék adatbázisának menedzselésében, könnyedén megbirkózik a DHCP adataival is.
A JET adatbázis nem egyetlen állományból áll. Ha megvizsgáljuk a DHCP könyvtárat, akkor a következő fájlokat fedezhetjük fel:

- J50.log *(J50xxx.log)* – tranzakciós állományok. A JET adatbázismotor nem azonnal az adatbázisba írja a változásokat, hanem ún. tranzakciós állományokba. Egy tranzakciós állomány mérete állandó, a használat során az adatbázismotor csak kitölti. Amikor a fájl megtelt, egy újat kezd a rendszer mindaddig, amíg egy sikeres mentés nem történik, vagy szabályosan le nem állítjuk a DHCP-szervert.
- J50.chk – Checkpoint állomány. Annak az információnak a helyét jelzi az utolsó tranzakciós állományban, amelyet sikeresen beírt a rendszer az adatbázisba.
- DHCP.MDB – A tényleges adatbázis.
- Dhcptmp.mdb – Ideiglenes állomány az indexek karbantartásakor keletkezik. Hibás leálláskor a könyvtárban maradhat.
- Resx.log – Egy „helyfenntartó" állomány. Ha véletlenül elfogyna a hely a DHCP adatbázis kötetén, a motor ezeket az állományokat felhasználva fejezné be a tranzakciót, majd leállítaná a szolgáltatást.

A teljes adattartalom ezek szerint az adatbázis, a chk állomány, valamint az adatbázisba be nem írt tranzakciókat tartalmazó állományok együttesével írható le.

Az adatbázis mentése

Az adatbázis mentéséről maga a rendszer is gondoskodik. A regisztrációs adatbázis szerint az alapértelmezett backup útvonal a %SystemRoot%\System32\dhcp\backup. A mentés gyakorisága 60 perc. Ezt az

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Servic
es\DHCPServer\Parameters\backupInterval
```

értékkel módosíthatjuk. Ugyanitt változtatható meg a mentés útvonala. Ha az útvonal más meghajtóra mutat, mint az eredeti adatbázis helye, máris megtettük az első lépést a DHCP szerverünk megmentése érdekében.

Az adatbázis tömörítése és javítása

A mentésünk megvan, de ez nem jelenti azt, hogy rögtön használnunk kellene. Ha az eseménynaplóban piros x-szel jelzett JetErr hibákat találunk, a rendszerhez mellékelt Jetpack.exe segédprogram segítségével megpróbálkozhatunk az adatbázisunk kijavításával. A műveléthez le kell állítanunk a DHCP kiszolgálót, majd a következő módszert használhatjuk:

1. Navigáljunk a DHCP-adatbázis könyvtárához.
2. Adjuk ki az alábbi parancsot:

```
Jetpack dhcp.mdb temp.mdb
```

A művelet megpróbálja orvosolni az adatbázis logikai hibáit, továbbá egy offline tömörítést végez.
(A fenti műveletet fürtözött DHCP-nél is elvégezhetjük, feltéve, hogy előtte az erőforrást Offline állapotba állítottuk. A munkakönyvtár ekkor természetesen az erőforrás által használt közös lemezen lesz. A művelet befejezése után az erőforrást újra el kell indítanunk.)

Az adatbázis helyreállítása

Amennyiben nem boldogulunk az adatbázis javításával, kétféle módszerrel is helyreállíthatjuk a szolgáltatásunkat.
Az első esetben ragaszkodunk az eredeti adatbázishoz. Ekkor a következő teendőink vannak:

1. Mentsük el a régi – nem használható – adatbázist.
2. Ha megvan a legutóbbi automatikus mentés, másoljuk az eredeti DHCP adatbázis helyére.
3. Tömörítsük a Jetpack.exe programmal az adatállományt.

Ezután a szolgáltatásnak már el kell tudnia indulni. Ha nem találjuk a scope-jainkat, helyre kell állítanunk a DHCP konfigurációt tartalmazó regisztrációs ágat is. Az automatikus backuppal együtt a Windows 2000 ezt is lementi egy DHCPCfg nevű állományba. *(Ez valójában a HKLM\Software\Microsoft\DHCPServer\Configuration kulcs tartalma).* Ha ezzel nem rendelkezünk, a legutóbbi mentésből kell megszereznünk az állományt.
Akárhogy is, amikor a szerverünk elindul, egészen biztosan nem a legfrissebb állapottal rendelkezünk. Szükségünk lesz a

regisztrációs kulcs és az adatbázis közötti konzisztencia megteremtésére, továbbá biztosítani kell, hogy az ügyfelek véletlenül se kapjanak duplán IP címet. Az első feladatról úgy gondoskodunk, hogy a szerverre jobb oldali egérgombbal kattintunk, majd a „Reconcile" menüpontot választjuk. Ezután be kell kapcsolni a szerver tulajdonságai közt az Advanced fülön található „Conflict detection" eljárást, így a kiszolgáló egy cím kiadása előtt - szórt üzenettel – meggyőződik arról, hogy van-e másik állomás, amely ezt a címet használja már. Ez persze némi teljesítmény-visszaeséssel jár.
Van egy másik járható út a szerver helyreállításához, ennek menete a következő:

1. Egy új adatbázis hozunk létre úgy, hogy elmozgatjuk az eredeti adatbázist, és üresen hagyjuk a könyvtárát. A DHCP-szolgáltatást elindítva egy új – üres – adatbázis keletkezik
2. Helyreállítjuk a DHCPCfg állomány segítségével a regisztrációs adatbázis DHCP-szerverre vonatkozó részét
3. Elvégezzük a „Reconcile" műveletet.
4. Bekapcsoljuk a „Conflict detection" eljárást.

Az adatbázis mozgatása

A fenti tudásunkat arra is felhasználhatjuk, hogy a DHCP kiszolgálót egyik szerverről a másikra költöztessük. *(Tegyük fel, hogy a Server1-ről a Server2-re mozgatjuk az adatbázist.)* Ehhez a lépések a következők:

1. Állítsuk le a Server1-en a DHCP szolgáltatást.
2. Mentsük le a DHCPCfg állományt a regisztációs adatbázis fent ismertetett ágából. Másoljuk az állomány a Server2-re.
3. A Server2-n telepítsük a DHCP szolgáltatást, majd állítsuk le a szervizt.
4. Töröljük a frissen telepített adatbázisfájlokat a %systemroot%\system32\dhcp könyvtárból és másoljuk oda a Server1 DHCP adatbázisát
5. Állítsuk helyre a DHCPCfg regisztrációs ágat a Server1-ről lementett fájl segítségével
6. Indítsuk el a Server2-n a DHCP-kiszolgálót
7. Végezzünk „Reconcile" műveletet.

Zárszó

A DHCP-t ismertető cikksorozat lezárul. Úgy érzem, hogy az elmúlt több mint fél év során egy jó alapos távtanfolyamot sikerült alkotni. Azt remélem, hogy az olvasók többségének átfogó képet adtam erről a meghúzódó, ámde fontos rendszerelemről. Talán voltak kezdők, akik most már „majdnem mindent" tudnak, s voltak haladók, akik imitt-amott kiegészíthették a tudásukat. Habár az IPv6 hosszabb távon csökkenteni fogja témánk fontosságát, úgy gondolom, még hosszú ideig hasznos lesz a megszerzett tudás.
Végezetül egy apró titok: a Windows Server 2003 címkiosztó szolgáltatása és a Windows 2000-hez képest minimális változtatáson esett át, az itt megszerzett ismerethalmaz tehát jó eséllyel 2006-ig, vagyis a következő Windows Server verzióig is friss maradhat. Bár az is igaz, hogy ez esetben kevésbé az operációs rendszer, mint inkább a szabványváltozás lehet fontosabb. Akárhogy is: legyünk résen, legyünk naprakészek.

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

Felhasznált és ajánlott irodalom

Q173396 How to Restore a Corrupted DHCP Database File
Q130642 How to Move a DHCP Database to Another Windows Server
Q145881 How to Use Jetpack.exe to Compact a WINS or DHCP Database
Microsoft Systems Architecture: Enterprise Data Center
Reference Architecture Guide – Chapter 6 – Network Services

Felhasználói bizalom, mint a fejlődés kulcsa

Az információs társadalom fejlesztésén munkálkodó Magyarországnak szembesülni kell azzal a problémával, hogy a fejlődés gátja ma már nem a rendelkezésre álló hardverek szűkössége, és nem is alapvetően a világhálóhoz való csatlakozás drágasága, hanem egyrészt az Internettel elérhető szolgáltatások kínálatának szűkössége, másrészt az a felhasználói bizalmatlanság, amely legfőbb akadálya a szolgáltatások bővülésének.

A kulcs tehát egy olyan környezet megteremtése, amely az információs társadalmi szolgáltatások felhasználóiban bizalmat gerjeszt, amely bővíti a szolgáltatások igénybevevőinek körét, és ezzel mintegy spirális kölcsönhatást beindítva az elérhető szolgáltatások területén is jelentős fejlődést érünk el. A fogyasztóvédelem hagyományos állami és civil fórumai fontos szerephez jutnak az online környezetben is, azonban a kívánt bizalmi szint megteremtéséhez nem elégséges e szervezetek hagyományos fellépése. A fogyasztóvédelem csak egy, kétségkívül igen jelentős szegmense az információs társadalomnak, azonban itt jóval szélesebb területről van szó. A felhasználó csak szűk értelemben véve fogyasztó. Az „e" korszak sajátossága leginkább éppen az, hogy a felhasználó rendszerint szolgáltató is, az aktív szerep egybeolvad a passzívval, és így gyarapodik, épül az információtömeget egységes virtuális térré szervező Internet.

Az internet sajátosságából adódóan az állami norma szigorú rendelkezései sem elegendőek ahhoz, hogy a kérdést megnyugtatóan rendezzék. Éppen ezért van szükség egy public private partnership jellegű megoldásra, mely az állami szervek és civil szervek szoros együttműködésén, az önszabályozás és alternatív vitarendezés eszközrendszerén nyugszik, és amely egyik legfontosabb feladatának a tájékoztatást, az informálást tekinti.

A világ azon részein, ahol az online kereskedelem, az elektronikus kormányzati szolgáltatások, az emberek közötti online kommunikáció különféle formái igen elterjedtek, már kialakultak azok a sajátos megoldások is, amelyek a fenti kívánalmaknak megfelelnek.

Az Európai Unió hagyományosan igen fontosnak tekintette a fogyasztóvédelem erősítését, hiszen ez az egységes belső piac bővítésének egyik záloga. Ugyanez a szemlélet az e-Europe programokban is megmutatkozik, ahol kiemelt jelentőséggel szerepel a fogyasztóvédelem kérdése. A dokumentumokon kívül az európai rangsorolást maga az uniós szervezet is tükrözi, hiszen a fogyasztóvédelemnek külön biztosa van, és a Bizottság igen szoros együttműködést tart fenn a különféle fogyasztóvédelmi egyesületek hálózataival. Ugyanakkor az Unióban azt is felismerték, hogy az információs társadalomban a jelen cikk elején kifejtettek értelmében más jellegű probléma kezelésre is szükség van, így több példamutató speciális intézményt is létrehoztak erre a célra.

Az Információs Társadalom Bizottság sajátos kezdeményezése a Dr. eCommerce. A sajátos szolgáltatás célja a fogyasztók tájékoztatása, a felvilágosítás. A hozzá intézett kérdéseket megválaszolja, és a kérdéseket *(az azokat feltevő személy azonosítása nélkül)*, valamint a válaszokat nyilvánosan megjelenteti. Dr. eCommerce 1999. óta működik. A hozzá intézett kérdések variációja igen széles, az elektronikus fizetési módok jogi helyzetétől az Internet és a magánszféra védelmének viszonyát firtató kérdésen át az EU elektronikus kereskedelemre irányuló marketingjén keresztül minden, amely az Internettel, és az információs társadalmi technológiákkal kapcsolatos, már napirendre került.

Az EEJ-NET *(European Extra Judicial Network)* szolgáltatásai már jóval túlmutatnak a felvilágosításon és tanácsadáson. Ez a szervezet már nem csak a fogyasztók, hanem a felhasználók szélesebb köre érdekében tevékenykedik, és egyfajta adatbázisaként szolgál az elérhető gyors, bíróságon kívüli *(alternatív)* vitarendezési megoldásoknak.

A CECUA *(Confederation of European Computer User Association – Európai Számítógép Felhasználók Szövetsége)* 12 Európai Uniós illetve EEA és EFTA tagállam területén véd több mint fél millió felhasználót, aki tagjai a nemzeti szervezeteknek. 1982-ben alapították, és azóta szorosan együttműködik az Európai Bizottsággal, és így a CECUA az egyik meghatározó társaság a felhasználók érdekeinek védelmére valamint az információs társadalom előmozdítására. E független szervezet nevéhez kötődik az Internetre kidolgozott „Bill of Rights" kidolgozása is.

A civil és állami együttműködésen alapuló megoldások területén a tagállamokban is érdekes és igen jól működő példákat találhatunk. Így Svédországban négy magánszemély 1999-ben megalakította az Internet Ombudsman intézményét kifejezetten azzal a céllal, hogy a felhasználók számára nyújtsanak segítséget. A non-profit szervezet a felhasználók részére nyújt ingyenes tanácsadást, bármely az Internet használatához kapcsolódó kérdésben, technikai, jogi vagy etikai jellegű kérdések esetén. Működését anyagilag egyrészt a svéd Kormány Kereskedelmi Minisztériuma *(Government Trade Department)*, a svéd Királyi Bíróság és a svéd parlament által alapított KK-alapítvány *(The Knowledge Foundation – Tudás Alapítvány)* támogatja.

Nem kell azonban ilyen messzire mennünk, hiszen a Lajtán túl, a szomszédos Ausztriában az ÖIAT *(Austrian Institute for Applied Telecommunication)*, és a VKI *(Consumer Information Association)* – amely Ausztria fő fogyasztóvédelmi szervezete – közösen létrehozta és működteti az Internet Ombudsman intézményét.

Az Internet Ombudsman munkáját az Osztrák Szövetségi Munkaügyi Kamara, a Szövetségi Igazságügyi Minisztérium, a Gazdasági és Munkaügyi Szövetségi Minisztérium, az

Internet Szolgáltatók Tanácsa, az Osztrák Fogyasztói Tájékoztató Szervezet és az Osztrák Szövetségi Kereskedelmi Kamara támogatja anyagi és más eszközökkel.

Az Ombudsman szerepe elsősorban az elektronikus kereskedelem fejlesztéséhez kötődik, így feladata az Interneten való vásárlás új aspektusáról általános információkat adni, fő célja az e-kereskedelem biztonságát növelni a széleskörű tájékoztatással, és a biztonsági alapkövetelménynek meghatározásával. Konkrét problémák felmerülése esetére emellett gyors, alternatív vitarendezési eljárásokat kínálnak.

Működésének hatékonyságát és elismertségét bizonyítja, hogy 2000-ben a Gazdasági és Munkaügyi Minisztérium által alapított PR díjat az Internet Ombudsmannak ítélték oda. Talán zavaró, hogy az említett intézmények visszaköszönő megnevezése az ombudsman. A kifejezés a magyar olvasó számára nyilván zavaró, amikor az országgyűlési biztosok intézményére gondol. Ugyanakkor az ombudsman lényegében egy közbenjáró, szószóló, többnyire vitarendező fórum, amely a világ számos országában, így elsősorban az angolszász nyelvterületeken civil kezdeményezésre áll fel és működik.

A felhasználók védelmének társadalmi keretekben történő megoldása azért alkalmasabb a bizalom megteremtésére, mint a szigorú normatív megközelítés és az állami szervek hatáskörébe utalás, mert ha a szolgáltatók csak szankciók által kényszerülnek a magas fokú bizalmat kiérdemlő minőség biztosítására, az információs társadalom világában az államhatárokat nem ismerő virtuális térben az előírásokra könnyen fittyet hánynak. Ám ha az érdekük azt diktálja, hogy önként feladva szuverenitásuk egy részét, meghatározott önszabályozó csoportba tömörülve annak szabályait magukra nézve kötelezőnek fogadják el, azt az állami végrehajtás fenyegető veszélye nélkül is betartják.

Jó példa erre a TRUSTe. Megalakításának ötlete 1996-ban pattant ki egy számítástechnikai konferencián, ahol éppen a magánszféra védelmének lehetőségein töprengtek egyesek. A TRUSTe – ahogy önmagát hirdeti – eszköz az önszabályozáshoz: nonprofit alapon, önkéntesen létrehozott és globálisan működő, mind az államoktól, mind az ipari szereplőktől független kezdeményezés.

A TRUSTe olyan mechanizmust alakított ki, amely a közbizalmat növeli a fogyasztókban a web site-ok iránt. Ha az adott site megfelel a TRUSTe szabta feltételeknek, akkor ennek igazolásául jogosult a megfelelő vizuális jel elhelyezésére. A jellegzetes embléma bizonyítja, hogy olyan oldalon jár a felhasználó, ahol személyes adatait a legnagyobb gondossággal kezelik.

Arra az esetre, ha a felhasználó ennek ellenére a TRUSTe emblémával ellátott web site-on személyes adatainak jogosulatlan felhasználásával találkozik, megfelelő alternatív vitarendezési eljárást is felkínálnak, a házőrző kutya után elnevezett ún. „Watchdog Dispute Resolution" megoldást.

Az eljárás a szabályok szerint ingyenes, és nyitva áll minden magánszemély számára, aki úgy véli, hogy a TRUSTe embléma alatt működő szolgáltató személyes szférájában megsértette. A vita kérelemre indul, az eljárás online zajlik. Ha a panaszos ügyben eljáró „bíró" a kérelmet megalapozottnak találja, kötelezheti az érintett szolgáltatót, hogy módosítsa a feltüntetett adatokat, vagy törölje közülük azokat, amelyek más jogát sértik; módosítsa az adatkezelési szabályzatát; illetve más szükséges intézkedéseket megtegyen.

Sor kerülhet továbbá az illetékes állami felügyeleti szerv értesítésére, vagy a TRUSTe embléma eltávolítására, és a szolgáltató és a TRUSTe közötti szerződés felbontására is. Az első fokú döntéssel szemben a TRUSTe fellebbezési fórumához lehet fordulni, az eljárás tehát két fokon zajlik.

A szabályok betartása tehát önkéntes ugyan, de ki is kényszeríthető az alternatív eljárással. Az eljárás során alkalmazható legsúlyosabb szankció a TRUSTe szerződés felbontása, vagyis a csoport tagjai közül történő kizárás. Ez azért jelenthet komoly hátrányt az Internetes kereskedőnek, mert a bizalmat hozó embléma elvesztése az üzletmenetére is hátrányos lehet. Az önkéntes alávetést biztosító érdek tehát a TRUSTe vonatkozásában is tetten érhető.

Magyarországon is tapasztalhatók már előrelépések e kérdésben. Az október 1-én kihirdetett kormányrendelet, amely az informatikai és hírközlési miniszter hatás- és feladatkörét határozza meg, előírja, hogy a miniszter – a felhasználói és szolgáltatói érdekképviseleti szervek útján – online ügyfélszolgálatot üzemeltessen. Az Országgyűlés előtt folyamatban lévő, az elektronikus kereskedelmi szolgáltatásokról szóló 2001. évi CVIII. Törvény módosításával Magyarországon először az állami norma szintjén kerül elismerésre az önszabályozás, mint az azzal egyenrangú, ahelyett egyes esetekben hatékonyabb szabályrendszer, valamint az alternatív vitarendezési eljárás, amely megfelelő forma lehet az információs társadalom gyors választ igénylő problémáira.

Az ugyancsak a t. Ház előtt lévő elektronikus hírközlési törvénybe is beépült egy intézmény, a Hírközlési Fogyasztói Jogok Képviselője. A jogalkotó kétségtelenül előremutató szándékát reprezentáló megoldás legnagyobb hibája az, hogy az irodavezetői beosztást kapó köztisztviselő és irodája lényegében nem rendelkeznek majd bővebb jogosítványokkal, mint amelyek bármely civil szervezetet megilletnek, ezért az intézmény igen sok kritikát kapott már eddig is.

A most készülő Magyar Információs Társadalmi Stratégiában is kiemelt kérdésként jelenik meg a bizalom és biztonság kérdése, olyannyira, hogy a készülő programfüzetek közül az egyik e kérdésnek lett szentelve.

Az Informatikai Érdekegyeztető Fórum, mint a magyar információs társadalom szolgáltatóinak és felhasználónak teljességét reprezentáló legnagyobb szervezet, már 2002. nyarán javasolt az előbbiekben kifejtetteknek megfelelő érdekvédelmi szolgáltatás felállítását. A társadalmi és állami együttműködést reprezentáló public private partnership megvalósításának előkészületei végre megkezdődtek, és várható, hogy 2004-ben Magyarország élenjáró példával járhat a csatlakozó országok számára is a felhasználók érdekeinek védelmére kifejlesztett megoldással, amely számottevően járul majd hozzá az oly fontos bizalom megteremtéséhez.

Dr. Mayer Erika

Tanúsítványkiadók a Windowsban

Qualified Subordination

Sorozatunk utolsó része az altanúsítványkiadókról szól.

A kiadható tanúsítványok névterének meghatározása

Az altanúsítványkiadók működési köre korlátozásának következő lépése az lehet, hogy meghatározzuk, KI kérhet és kaphat tanúsítványt az adott tanúsítványkiadótól. Ehhez különféle korlátozó szabályokat *(angolul „constraint"-eket)* definiálunk. Elsőként ismerkedjünk meg a lehetséges szabályokkal!

Az igénylő „nevének" korlátozása *(Name Constraints)*

A címben szereplő „név" nem biztos, hogy egyértelműen utal a Name Constraintek szerepére. Ezek a szabályok arra valók, hogy az igénylő Active Directory-beli vagy DNS neve, URI-ja, e-mail címe vagy IP-címe alapján korlátozhatjuk a CA-hoz való hozzáférést. Ennek megfelelően többfajta Name Constraint létezik, amelyeket néhány sor múlva egyenként bemutatunk. Fontos, hogy ha a tanúsítványkiszolgálón definiáltunk Name Constraintet, a beérkező tanúsítványkérésben szereplő nevek mindegyikének engedélyezve kell lennie, különben a CA a kérést visszautasítja.

Létrehozhatunk engedélyező és tiltó típusú Name Constrainteket is. A tiltó szabályok minden esetben nagyobb erejűek, mint az engedélyezők, tehát ha a tanúsítványkérés olyan nevet tartalmaz, amelyre mind engedélyező, mind pedig tiltó szabályt létrehoztunk, a kérést el kell utasítani. Alapértelmezésben szabály az is, hogy ha szabályok között egy adott típusú *(LDAP, DNS, stb.)* névre vonatkozó tag – akár engedélyező, akár tiltó típusú – szerepel, a tanúsítványkérésben is kell, hogy legyen egy megfelelő típusú név. Ha nincs, – függetlenül attól, hogy tiltás esetleg nem vonatkozik rá, – a tanúsítvány nem adható ki.

A Name Constraintek kiértékelésénél figyelembe kell venni a tanúsítványt igénylő nevét *(Subject mező)*, valamint az összes létező alternatív nevet is *(Subject Alternative Name bővítmény)*.

Lássuk most a Name Constraintek típusait:

1. Relative Distinguished Name Constraint

RDN Constraintben az objektum Active Directory-beli LDAP nevét adhatjuk meg (pl. „CN=Computers, DC=falatrax, DC=hu"), a következő formátumban:

```
DIRECTORYNAME = "CN=Computers, DC=falatrax, DC=hu"
```

A fenti példa a falatrax.hu tartomány minden olyan számítógépére utal, amelyek a Computers tárolóban vannak. Ha tehát az RDN Constraint egy objektumokat tartalmazó címtárobjektumra *(azaz, „konténerre")* utal, a szabály az objektum minden gyermekére vonatkozik.

```
DIRECTORYNAME = "OU=Könyvelés, DC=falatrax, DC=hu"
```

```
DIRECTORYNAME = "CN=Gizike, OU=Recepció,
DC=falatrax, DC=hu"
```

Az első példa tehát a Könyvelés szervezeti egység minden tagjára *(felhasználókra és számítógépekre egyaránt)*, a második példa pedig kifejezetten a recepciós Gizikére vonatkozik.

2. DNS Name Constraint

A DNS Name Constraint az igénylő *(ezesetben nyilván számítógép vagy hálózati eszköz)* DNS nevére utal. Itt is meghatározhatunk konkrét nevet, de hozhatunk szabályt egy DNS zóna minden tagjára is. Példák:

```
DNS = falatrax.hu
```

A falatrax.hu DNS zóna minden tagja (például a www.falatrax.hu, de önmagában a falatrax.hu is)

```
DNS = .falatrax.hu
```

A falatrax.hu DNS zóna gyermekei *(Figyeljünk a pontra!)*. Ilyen a www.falatrax.hu, de NEM ilyen maga a falatrax.hu.

```
DNS = szerver.falatrax.hu
```

A szerver.falatrax.hu *(és gyermekei, ha vannak, pl.: www.szerver.falatrax.hu)*.

Uniform Resource Identifier *(URI)* Constraint

Az URI egy szöveges azonosító, amely tartalmazza egy hálózati erőforrás pontos elérési útját, beleértve az eléséhez szükséges protokollt is, a következő formátumban:

<protokoll>://<szerver DNS név>[:<port>]/<fájl elérési út>

Ilyen URI például egy URL is:

http://www.falatrax.hu/default.htm, ahol a protokoll a http, a szervernév a www.falatrax.hu, a fájl elérési útja pedig /default.htm. Protokoll lehet még például az ftp, telnet, mailto, gopher, stb. URI Constraintek segítségével tipikusan webkiszolgáló-tanúsítványkéréseket engedélyezhetünk, illetve tilthatunk, az alábbi formátumban:

```
URL = http://www.falatrax.hu
```

E-mail, UPN Constraint

E-mail, illetve UPN *(User Principal Name)* Constraint segítségével az igénylő e-mail címére, illetve Active Directory-beli UPN nevére hivatkozhatunk. A kettő között technikailag az a

különbség, hogy az UPN nevet UTF-8 kódolással kezeljük, míg az E-mail ellenőrzésénél a tanúsítványkezelésnél szokásos *(IA5)* karakterkódolást használ a CA *(ez nem tartalmaz ékezetes karaktereket, így az ellenőrzés is ékezetfüggetlen lesz)*.

Formátuma:

```
EMAIL = gizike@falatrax.hu
EMAIL = @falatrax.hu
```

illetve

```
UPN = @falatrax.hu
UPN = .falatrax.hu
```

Az e-mail példák közül az első a már ismert Gizikére, a második a falatrax.hu tartomány minden felhasználójára vonatkozik. UPN neveket mindig kettesével kell létrehoznunk: egyszer a szokásos módon, egyszer pedig úgy, hogy a @ karaktert ponttal helyettesítjük.

IP Address Constraint

Az IP Address Constraint a tanúsítványt igénylő ügyfél IP címe alapján segít eldönteni, hogy a tanúsítvány kiadható-e. A Constraintben meg kell adnunk az alhálózat, illetve az ügyfél IP címét, majd / jel után az alhálózati maszkot. Ha a maszk 255.255.255.255, a Constraint konkrét IP címre, különben az adott alhálózatra vonatkozik:

```
IPADDRESS = 192.168.1.0/255.255.255.0
IPADDRESS = 192.168.1.113/255.255.255.255
```

A Name Constraint használata

Name Constrainteket altanúsítványkiadó telepítése esetén a CAPolicy.inf, „idegen" tanúsítványkiadó beléptetése *(azaz a konkrét értelemben vett qualified subordination)* esetén pedig a tanúsítványkérés előállításához használt policy.inf fájlban kell megadnunk, a következő módon:

```
[NameConstraintsExtension]
Include = NameConstraintsPermitted
Exclude = NameConstraintsExcluded
Critical = True

[NameConstraintsPermitted]
DIRECTORYNAME = "DC=falatrax, DC=hu"
EMAIL = @falatrax.hu

[NameConstraintsExcluded]
DIRECTORYNAME = "OU=kulsosok, DC=falatrax, DC=hu"
```

A fenti példa esetén a tanúsítványkiadó a falatrax.hu Active Directory-tartomány felhasználóinak *(kivéve a külsősök szervezeti egység tagjait)* adna tanúsítványt, amennyiben a kérésben található e-mail cím a @falatrax.hu tartományba esik. Ha a tanúsítványkérés nem tartalmaz e-mail címet vagy UPN hivatkozást, a jogosultság nem ellenőrizhető, ezért a CA visszautasítja azt.

Az igénylő nevének korlátozása a gyakorlatban

Itt az ideje, hogy készpénzre váltsuk a tudásunkat. A példában a falatrax2003.hu tartományban, a Falatrax Enterprise Root CA tanúsítványkiadó „alá" beléptetjük a Falatrax Issuer CA-t, amely csak a tartomány Kőművesek szervezeti egységének tagjai részére ad ki tanúsítványt, persze csak akkor, ha a kérésben a megfelelő e-mail cím *(felhasználónév@komuvesek.falatrax2003.hu)* szerepel. A példa kedvéért a @burkolok.falatrax2003.hu e-mail címeket explicit megtiltjuk.

```
[Version]
Signature="$Windows NT$"

[RequestAttributes]
CertificateTemplate=SubordinateCertificationAuthor
ityv2.0

[NameConstraintsExtension]
Include=NameConstraintsPermitted
Exclude=NameConstraintsExcluded
Critical=True

[NameConstraintsPermitted]
DIRECTORYNAME="OU=komuvesek,DC=falatrax2003,
DC=hu"
EMAIL=@komuvesek.falatrax2003.hu

[NameConstraintsExcluded]
EMAIL=@burkolok.falatrax2003.hu
```

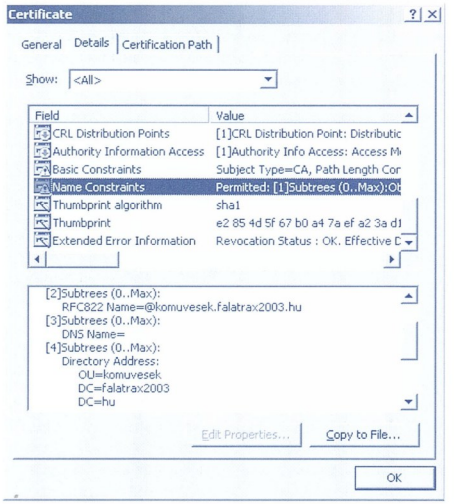
Most is az előző részben létrehozott új tanúsítványsablont használjuk *(hiszen a „gyári", 1.0-s verzió nem alkalmas a Qualified Subordination-re)*. Ha kitöltöttük a CAPolicy.inf fájlt, a Certification Authorities konzolban kérjünk új tanúsítványt a gyermek CA-nak *(All tasks → Renew CA Certificate…)*. A tanúsítványkérést most se hagyjuk elküldeni *(mert az MMC-s varázsló nem venné figyelembe a Name Constraints mezőket)*, hanem mentsük el fájlba. A kérést tartalmazó .req fájlt pedig „oltsuk" be a CAPolicy.inf-fel, majd ne felejtsük el digitálisan aláírni se! Adjuk ki a következő parancsot:

```
certreq -policy
```

Ezután válasszuk ki az előbb elmentett .req fájlt, majd a CAPolicy.inf-et, a digitális aláírás tanúsítványát *(a Qualified Subordination Request típusú aláírótanúsítvánnyal)*, az eredményként keletkezett kérést pedig mentsük el új néven *(ez is egy .req fájl lesz)*. Ezt a .req fájlt kell eljuttatnunk a vállalati gyökér CA-hoz, a webes felületen, vagy egy újabb paranccsal:

```
certreq -submit
```

A parancs kiadása után válasszuk ki a gyökérkiszolgálót, majd adjuk meg a fájlnevet, amibe a kész tanúsítványt mentheti az eszköz *(.cer)*! Ezután már csak be kell importálnunk a gyermek CA-ba a kész tanúsítványt *(Certification Authorities konzol, All Tasks → Import CA Certificate…)*, és már készen is vagyunk.



A kész CA tanúsítvány, és benne a Name Constraints szabályaink

Amint látható, ha a policy-ben egy-egy Name Constraints típust nem definiálunk, az üres értékkel kerül bele a tanúsítványba. Ez a joker, azt jelenti, hogy az adott típusú név bármi lehet *(esetünkben ilyen pl. a DNS Name is)*.

Tanúsítványkérés a gyermekkiadótól

Próbáljuk ki, működnek-e a korlátozások. Kérjünk tanúsítványt egy felhasználóval, akinek az e-mail címe megfelelő ugyan *(teszt@komuvesek.falatrax2003.hu)*, de nem a megfelelő Active Directory szervezeti egységbe tartozik! A felhasználó tanúsítványkérésére hibaüzenet a válasz:

„A kérést a tanúsítványkiadó elutasította. A tanúsítvány érvénytelen nevet tartalmaz. A név nem található az engedélyezettek között, vagy kifejezetten tiltva van”

A felhasználónak ennyi talán elég is. Ha pontosan tudni szeretnénk, hogy mi a probléma, nézzük meg a kérést a Certification Authority MMC konzolban:

Az elutasított tanúsítványok listájában látszik az indoklás is

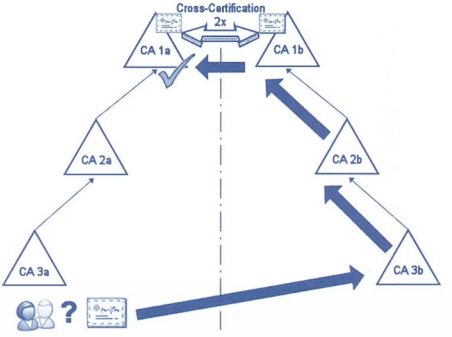
A Failed Requests listában keressük meg a tanúsítványkérést. Itt a Request Status Code oszlopban ugyanazt az üzenetet látjuk, mint a felhasználó; a Request Disposition Message viszont már bővebb információkkal szolgál:

Error Constructing or Publishing Certificate. No Permitted Name Constraint for <Directory Address: E=teszt@komuvesek.falatrax2003.hu, CN = Teszt Elek, CN = Users, DC = falatrax2003, DC = hu>.

Helyezzük át most a felhasználót a „Komuvesek” szervezeti egységbe, és tegyünk újabb próbát! A kérés ezúttal már sikeres lesz.

Két, egymástól független CA hierarchia összekapcsolása

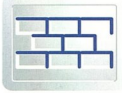
A következő eset az, amikor a qualified subordination célja két, egymástól független tanúsítványkiadó hierarchia összekapcsolása *(cross-certification, azaz keresztbehitelesítés)*. Ennek több formája van: összekapcsolhatjuk a két hierarchia csúcsát, gyökér-tanúsítványkiadóit, kapcsolatot hozhatunk létre az egyik hierarchia gyökere és a másik hierarchia egyik gyermek CA-ja, vagy akár a két hierarchia gyermek CA-i között is.



A keresztanúsítás lényege

A lényeg minden esetben az, hogy a „saját” felhasználóink elfogadnak bizonyos célokra alkalmas, az „idegen” hierarchia CA-iból származó tanúsítványokat, anélkül, hogy ehhez az idegen gyökér CA-t fel kellene vennünk a megbízott gyökér-tanúsítványkiadóink *(Trusted Root Certification Authorities)* listájába.

Vegyünk egy példát! A fenti ábrán két, egymástól független tanúsítványkiadó-hierarchia látható *(CA 1a-3a illetve CA 1b-3b)*. Tegyük fel, hogy az „a” hierarchia felhasználója digitálisan aláírt levelet kap a „b” hierarchia egyik felhasználójától. Az aláíráshoz használt tanúsítvány a CA 3b tanúsítványkiadótól származik. A tanúsítvány ellenőrzése *(a tanúsítványlánc felépítése)* során a vastag fekete nyilak mentén eljutunk a „b” hierarchia csúcsáig, a CA 1b gyökér-tanúsítványkiadóig. Normális esetben az ellenőrzés itt véget ér, és mivel a CA 1b nem szerepel a felhasználónk megbízott gyökér CA-inak listájában, a tanúsítvány érvénytelen lenne.



A keresztbetanúsításnak köszönhetően a két hierarchia gyökér CA-i egymás minősített „gyermekeinek" számítanak *(ezt jelzi az ábra tetején látható kettős fekvő nyíl)*. Az ellenőrzés ezért a CA qualified subordination tanúsítványa mentén folytatódik – a CA 1b keresztanúsítványát viszont a CA 1a adta ki, aki nem más, mint a felhasználónk saját gyökér CA-ja!

A valóságban gyökér CA-k között ritkán hoznak létre keresztanúsítást, de az ábra alapján könnyű belegondolni a tanúsítványlánc felépítésébe akkor is, ha a keresztanúsítás egy gyermek és egy gyökér, vagy akár két gyermek CA között áll fenn.

Issuance és Application Policies

A tanúsítványok mezőinek bemutatása során szerepelt két speciális, a tanúsítvány kezelésére, használatára utaló mező:

- **Issuance Policy:** meghatározza, hogy az adott tanúsítvány kiadása során milyen biztonsági előírásoknak megfelelően kell eljárnunk *(a különböző eljárásoknak saját OID-jük van, de generálhatunk saját eljárást is)*
- **Application Policy:** meghatározza, hogy az adott tanúsítvány milyen célra használható fel *(minden felhasználási területnek saját OID-je van, és természetesen itt is generálhatunk saját felhasználási területet, saját OID-vel)*

A Qualified Subordination tanúsítvány kiadása *(egészen pontosan kérése)* során a nevek korlátozása mellett a fenti két policy is megadható, így elérhetjük, hogy egy altanúsítványkiadó *(vagy egy keresztbehitelesített CA hierarchia)* például csak titkosított e-mail küldésére alkalmas tanúsítványokat adhat ki *(illetve csak azokat tekintjük érvényesnek)*.

Az Issuance Policy meghatározása

Amikor a Qualified Subordination során a „gyermek" CA-nak új tanúsítványt kérünk, - mint eddig is – egy .inf fájl határozza meg a tanúsítvány különféle jellemzőit. A továbbiakban ezt a fájlt policy.inf-nek nevezzük, de a fájl neve tulajdonképpen nem lényeges, csak az, hogy a tanúsítványkérés generálása során a megfelelő fájlt adjuk meg *(ugyanúgy, mint néhány bekezdéssel korábban, a Name Constraints definiálása esetén)*. Ha tehát egy tanúsítványkérésben korlátozni szeretnénk a használható Issuance Policy-k értékeit, a következő sorokat kell elhelyeznünk a policy.inf fájlban:

```
[PolicyStatementsExtension]
Policies = Policy1, Policy2, Policy3 ; akármennyi
Critical = False

[Policy1]
OID = 1.3.6.1.4.1.311.21.7... ; Policy1 OID-je

[Policy2]
OID = 1.3.6.1.4.1.311.21.8... ; Policy2 OID-je

[Policy3]
OID = 1.3.6.1.4.1.311.21.9... ; Policy3 OID-je
```

Emlékezzünk, hogy az Issuance Policy értékek minden Active Directory-ban mások és mások, mert véletlenszerűen generált értéket tartalmaznak. Ha olyan CA hierarchiákat akarunk összekapcsolni, amelyek különböző tartományban működnek, a különböző OID-ket egymás mellé kell rendelnünk, ezt nevezik Policy Mapping-nek. Innen tudja majd az egyik CA, hogy mit jelent egy adott Issuance Policy a másik CA-ban:

```
[PolicyMappingsExtension]
1.3.6.1.4.1.311.21.8.1.124 = 1.3.6.4.1.442.21.23.2
...
```

A **[PolicyMappingsExtension]** részben soronként egy-egy OID-rendelünk össze. Előre kerül a saját OID-nk *(az, ami a **[PolicyStatementExtension]** szakaszban is szerepel)*, majd egyenlőségjel után a partner CA hierarchia értékei.

Unexpected Trust

Azaz „váratlan bizalom". Tegyük fel, hogy Qualified Subordination-t hozunk létre a partnervállalatunk egy adott gyermek CA-jával. Elfogadjuk az általa kiadott tanúsítványokat, mert megbízunk benne. Ha azonban ez a tanúsítványkiadó újabb gyermeket fogad maga alá, akkor – váratlanul – elfogadjuk majd e gyermek *(sőt, unoka)* CA által kiadott tanúsítványokat is, hiszen a tanúsítványlánc felépítése során így is eljutunk az általunk megbízott gyermek CA-hoz. Ezt nevezik Unexpected Trust-nak, hiszen egyáltalán nem biztos, hogy ezt mi a Qualified Subordination kiépítése pillanatában előre tudjuk, és támogatjuk.

A dolog kivédése érdekében két beállítással élhetünk:

```
[PolicyConstaintsExtension]
RequireExplicitPolicy = 1
InhibitPolicyMapping = 1
```

- **RequireExplicitPolicy** – a Qualified Subordination akkor érvényes, ha a felépített tanúsítványláncban a Qualified Subordination tanúsítvány „alatt" legfeljebb x „idegen" CA található, ahova bele kell számolni azt a gyermek CA-t is, amellyel a Qualified Subordination-t létrehoztuk.
  Például: vegyük a keresztanúsítást bemutató ábrát. A kapcsolat a két gyökérkiadó *(CA 1a, CA 1b)* között áll fenn, korlátozások nélkül tehát az „a" cégben megbízunk a teljes „b" CA hierarchiában. Ha a RequireExplicitPolicy értékét itt például 2-re állítanánk, akkor a bizalom csak a CA 1b és a CA 2b-re terjedne ki, a CA 3b által kiadott tanúsítványokra már nem.
- **InhibitPolicyMapping** – az előzővel teljesen azonos elven működik, csak ez az érték nem a Qualified Subordination-ra általában vonatkozik, hanem a Policy Mapping-ban meghatározott összerendelésekre. Ha például az „a" cég „x" policy-jét összerendeljük a „b" cég „y" policy-jével, az InhibitPolicyMapping 2-es értéke esetén az összerendelés a CA 1b és CA 2b által kiadott tanúsítványok esetén igen, a CA 3b által kiadott tanúsítványokra viszont már nem lenne érvényes.

Az Issuance Policy tehát azt határozza meg, hogy a Qualified Subordination „hidat" milyen módon kiadott tanúsítványok járhatják át.

Az Application Policy meghatározása

Teljesen ugyanúgy működik, mint az Issuance Policy, csak eb-

ben az esetben a tanúsítványok felhasználási módjait korlátozzuk *(a hidas példánál maradva: azt, hogy a Qualified Subordination „hídat" milyen célokra kiadott tanúsítványok járhatják át)*. Az Application Policy-k meghatározása tehát *(ne feledjük, a policy.inf-be kerül)*:

```
[ApplicationPolicyStatementExtension]
Policies = PolicyEmail, PolicyCodeSign, PolicyAuth
; stb.

[PolicyEmail]
OID = 1.3.6.1.5.5.7.3.4 ; Secure Email

[PolicyCodeSign]
OID = 1.3.6.1.5.5.7.3.3 ; Code Signing

[PolicyAuth]
OID = 1.3.6.1.5.5.7.3.2 ; Client Authentication
```

Az Issuance Policy-vel ellentétben az Application Policy-k OID-i általában általánosan ismertek és használtak, a Policy Mapping-re tehát ritkán van szükség. Ettől függetlenül előfordulhat, hogy az „a" cég saját Application Policy-t hoz létre, saját OID-vel, és azt összerendeli a „b" cég OID-ivel:

```
[ApplicationPolicyMappingsExtension]
1.3.6.1.4.1.311.21.64 = 1.2.3.4.5.6.7
1.3.6.1.4.1.311.21.65 = 1.2.3.4.5.6.8
```

Természetesen itt is fennál az Unexpected Trust, és az eszközeink is megvannak a kivédésére:

```
[ApplicationPolicyConstraintsExtension]
RequireExplicitPolicy = 4
InhibitPolicyMapping = 3
```

Két, különálló CA hierarchia összekapcsolása

Próbáljuk ki az olvasottakat! A két összekapcsolni kívánt CA hierarchia a Falatrax Enterprise Root CA által, valamint az EladLak Enterprise Root CA által képviselt „hierarchia" lesz. *(Miután mindkét hierarchia mindössze egy-egy gyökér CA-ból áll, értelemszerűen gyökér CA-kat kapcsolunk majd össze)*. A Qualified Subordination során meghatározzuk, hogy a kapcsolat csak felhasználói tanúsítványokra érvényes *(Secure Email Application Policy, OID: 1.3.6.1.5.5.7.3.4)*. Korlátozzuk még a váratlan bizalmat is, bár két érték közül az InhibitPolicyMapping beállítására *(Policy Mapping híján)* nem lesz szükség. A teljes – kölcsönös – bizalom érdekében a most következő műveleteket kétszer kell végrehajtani *(egyszer a Falatrax, másodszor pedig az Eladlak tanúsítványkiadóin)*. Mielőtt belekezdenénk a keresztanúsítás létrehozásába, ellenőrizzük, hogy a tanúsítványkiadók CDP *(a CRL elérési útja)* és AIA *(a tanúsítványkiadó saját tanúsítványának elérési útja)* jól vannak-e beállítva *(azaz, az „idegen" szervezetből is elérhetők-e)*!

Keresztanúsítás-kérés létrehozása

A keresztanúsítás létrehozásához az „idegen" CA saját tanúsítványára, valamint a qualified subordination korlátozásait tartalmazó policy.inf fájlra lesz szükségünk. Magát a keresztanúsítást tehát mindenki a saját háza táján kezeli, a létrehozásához pedig nem kell kiadni semmi olyan információt, ami egyébként nem lenne nyilvános *(hiszen a CA tanúsítványa az)*. Üljünk le képzeletben a Falatrax tanúsítványkiadója elé, és egy könyvtárba mentsük el az Eladlak Enterprise Root CA tanúsítványát! *(eladlakca.cer)*. A policy.inf fájl tartalma:

```
[Version]
Signature="$Windows NT$"

[RequestAttributes]
CertificateTemplate=SubordinateCertificationAutho
rityv2.0

[ApplicationPolicyStatementExtension]
Policies = PolicyEmail, PolicyEFS, PolicyAuth

[PolicyEmail]
OID = 1.3.6.1.5.5.7.3.4 ; Secure Email

[PolicyEFS]
OID = 1.3.6.1.4.1.311.10.3.4 ; Encrypting File
System

[PolicyAuth]
OID = 1.3.6.1.5.5.7.3.2 ; Client Authentication

[ApplicationPolicyConstraintsExtension]
RequireExplicitPolicy = 1
InhibitPolicyMapping = 1
```

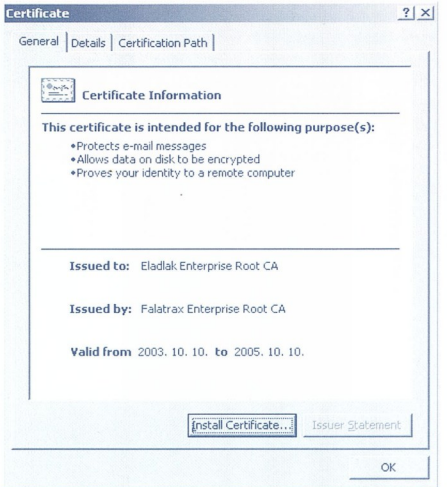
Első lépésként hozzunk létre egy tanúsítványkérést az Eladlak CA meglévő tanúsítványa alapján:

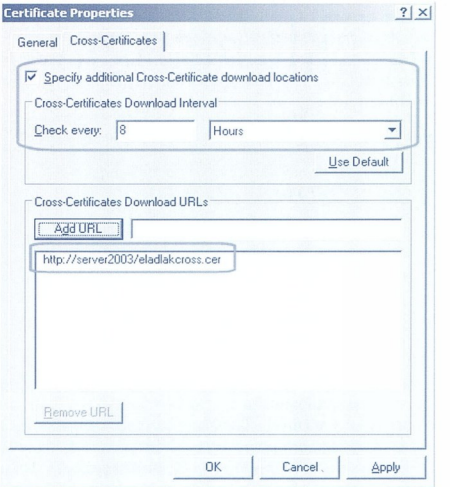
```
certreq.exe -policy
```

A varázsló először egy kérésfájl megnyitását ajánlja fel, de mi válasszuk a Certificate Files *(*.cer)* mezőt, és nyissuk meg az eladlakca.cer tanúsítványt. Ebből készül a kérés. Ezután a policy.inf fájlt kell megadnunk, majd a kérés digitális aláírásához használt tanúsítványt *(emlékszünk, Qualified Subordination Request Signing)*. Végül a kész kérést mentsük fájlba, mondjuk crosscert.req néven.

A kereresztanúsítvány kiadása

Ez egy nagyon egyszerű művelet: a Falatrax CA Certification Authority MMC konzoljában jelöljük ki a CA-t majd a menüből válasszuk az All tasks → Submit New Request... parancsot, és adjuk meg az előbb elmentett fájl nevét *(crosscert.req)*! A kész tanúsítványt mentsük el *(pl. eladlakcross.cer)*!





A kész tanúsítvány: a Falatrax CA adta ki, az Eladlak CA részére, és csak kívánt három dologra érvényes!

A tanúsítvány létrehozásával egyidőben a Falatrax CA a saját tartományában *(az Active Directory-ban)* is közzétette a kereszttanúsítás tényét *(valójában magát a tanúsítványt)*. A Falatrax tartomány felhasználói ezért ettől fogva – a meghatározott célokra – elfogadják az Eladlak CA tanúsítványait *(ha a kölcsönösség feltétel, ne felejtsük el a műveletet megismételni az Eladlak CA-nál!)*

Kereszttanúsítás Active Directory nélkül

Előfordulhat azonban, hogy olyan környezetben dolgozunk, ahol az Active Directory nem érhető el, a kereszttanúsításra azonban mégis szükség lenne. Ha publikusan elérhetővé tesszük a kereszttanúsítványt *(például egy webkiszolgálón – ebből sem lesz baj, hiszen a kereszttanúsítványban sincs privát adat –)*, akkor megtehetjük, hogy az ellenőrizni kívánt tanúsítványban megjelöljük a kereszttanúsítvány elérési útját. Ehhez nyissuk meg az ellenőrizni kívánt tanúsítványt *(fontos, hogy a .cer fájl ekkor nem jó, csakis a store-unkban már eltárolt tanúsítványokhoz rendelhetünk ugyanis kereszttanúsítványt)*, majd a tanúsítvány tulajdonságlapjának Details oldalán kattintsunk az Edit Properties gombra. *(Ha ez a gomb nem kattintható, a megnyitott tanúsítványt még nem importáltuk a store-unkba).*

A tanúsítványokba „kézzel" is bevéshetjük a CA-k közötti kapcsolatot igazoló kereszttanúsítvány elérési útját

Epilógus

Ezzel elértünk a Windows tanúsítványkiadókat bemutató sorozatunk végére *(is)*. Én a magam részéről szeretném megköszönni az Olvasók kitüntető figyelmét, és remélem, az elmúlt évek során sok új és hasznos információval lettek gazdagabbak írásaim által. A legközelebbi viszontlátás – olvasás? – reményében búcsúzom, mindenkinek jó és sikeres munkát kívánok:

Fülöp Miklós
MCSE+Security, MCT
mick@inetcom.hu

Objektumorientált alkalmazásfejlesztés

Záró gondolatok

Sokat gondolkoztam azon, hogy mit lehetne írni a cikksorozat befejezéseként. Az elmúlt hónapokban áttekintettük az objektumorientált fejlesztés alapfogalmait, néhány alapelvét, valamint emberi tényezőkről is írtam egy-két gondolatot.

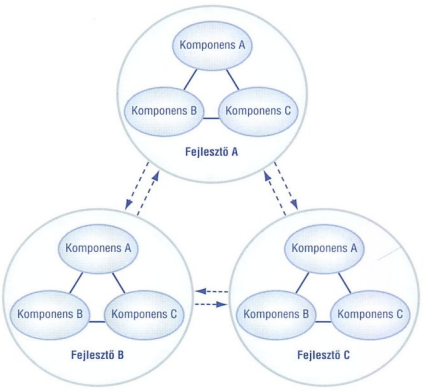
Most ismét a folyamat technikai oldalához térnék vissza, pontosabban azokhoz a dolgokhoz, amelyek szükségesek ahhoz, hogy egy szoftverfejlesztési projekt sikeres legyen. A szakmai hozzáértés és a megfelelő emberek kiválasztása mellett ugyanis még egy sor módszertan, eszköz és praktika alkalmazására szükségünk van.

Eszközök terén rendkívül széles a kínálat, szinte el lehet veszni a sokféle megoldás között. Az egyes rendszerek különböző követelményeknek felelnek meg, így érdemes jó alaposan körülnézni, mielőtt eldöntjük, mit fogunk használni. Egy rosszul megválasztott eszköz ugyanis súlyos veszteségeket okozhat a projektnek. A „jó ez nekünk, de…" kezdetű mondatok soha nem vezettek még jóra.

Lássuk először az egyik legalapvetőbb igényt, egy vállalati kódcentrum létrehozását. Ennek funkcionalitása többféle lehet. Elsődleges cél természetesen az, hogy a programkódokat egyetlen központi helyen tároljuk, ahova mindenki feltölti a saját részét, így mindenki számára elérhetővé válik a mindenkori legfrissebb verzió. Természetesen a feltöltést kellő körültekintéssel kell végezni, hiszen egy rosszul megírt komponens az egész alkalmazás működését felboríthatja.

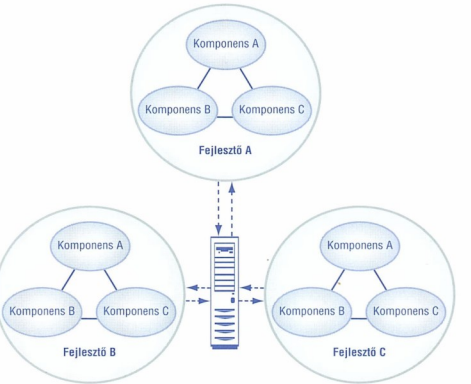
Lássunk egy példát. Tegyük fel, hogy a projekten három fejlesztő *(vagy három fejlesztőcsoport, a példa szempontjából lényegtelen)* dolgozik. Az alkalmazás komponensei egymással mind kommunikálnak, tehát interfészt kell biztosítanunk bármely kettő között.

Ez azt jelenti, hogy mindhárom komponensnek biztosítania kell két interfészt, egyet-egyet mindkét másik komponens felé. Amennyiben ezek a komponensek lokálisak, azaz egy gépen kell elhelyezkedniük, nem pedig hálózati kommunikáció zajlik közöttük, mindhárom fejlesztő tárol a saját gépén egy-egy példányt a másik két fejlesztő kódjaiból is. Természetesen az együttműködés így nagyon nehézkes: mindig mindenkinek el kell juttatni a legújabb frissítéseket, ügyelni kell arra, hogy az egyes verziók kompatibilisek legyenek egymással, mindenki gépén ugyanaz a változat legyen stb. Az alábbi ábra ezt az esetet szemlélteti: minden fejlesztő minden fejlesztőnek küldi a saját komponenseinek verzióit, és minden fejlesztő gépén megtalálható minden komponens:



Minden fejlesztő minden fejlesztővel kapcsolatban áll

A következő lépés egy központi számítógép kinevezése, ahova mindenki feltöltheti a saját komponensét, így a többszöri másolgatásból eredő problémákat kiküszöböljük. Helyette azonban szembesülünk azzal, hogy így mindenkinek oda kell figyelnie arra, hogy ha kerül a szerverre újabb verzió valamely komponensből, azt frissítenie kell lokálisan is, hiszen máskülönben az egyes fejlesztőknél lévő verziók nem lesznek kompatibilisek egymással. Az alábbi ábra ezt a helyzetet szemlélteti:



Központi szerver beiktatása

Ráadásul arra is oda kell figyelni ebben az esetben, hogy ha a szerverre felviszünk egy kódrészletet, az felülírja a korábbiakat, így azok sem hibajavítás, sem egyéb célokból nem lesznek hozzáférhetők a későbbiekben. Erre pedig időnként nagy szükség lehet. Gondoljunk arra például, hogy egy nagy rendszert fejlesztünk, amelyet már a fejlesztés fázisában is elkezd tesztelni egy teljesen különálló csapat. Miközben ők a rendszer üzleti logikáját, megvalósítását, külsejét stb. próbálgatják, találnak valamilyen hibát, és ezt jelzik felénk. Nyilvánvalóan ennek a folyamatnak időigénye van, ami alatt mi is végezzük saját munkánkat, azaz a rendszer fejlődik, bővül, változik. Az is elképzelhető, hogy a régi verziónak már mi is megtaláltuk a gyermekbetegségeit, így mire a visszajelzés érkezik a tesztelő csapattól, mi már egy másik rendszerrel dolgozunk „odabenn", amely esetleg már nem is rendelkezik az adott hibával, és nem utolsó sorban: belső szerkezete, felépítése is nagyban eltérhet a tesztelőknél lévőnél.
Hasonló gondokat okozhat az is, ha különböző felhasználónál különböző verziói vannak a rendszernek. Ha ugyanis valamely felhasználó jelzi felénk, hogy a rendszer nála nem megfelelően működik, nem biztos, hogy ugyanaz a verzió nálunk is megtalálható, hiszen a komponenseket mindig felülírjuk a központi táron.

Újabb előrelépést jelenthet, ha a szerveren verziónként új helyre mentjük komponensünket, így a legfrissebb változat is hozzáférhető mindenki számára, és a korábbiak is elérhetők maradnak. A file-rendszerben ez tehát valahogy így fog kinézni:

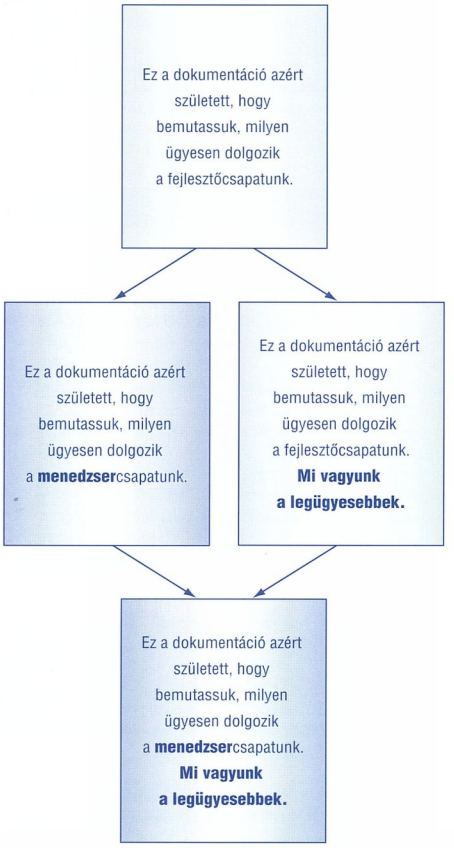
A szerveren az egyes verziókat külön-külön mappákban tároljuk

Ebben az esetben is az egyes fejlesztők külön-külön felelőssége, hogy a változásokat figyeljék a szerveren, és saját gépükön frissítsék a komponenseket, illetve az is, hogy a szerveren minden verzió a megfelelő helyre kerüljön. Megoldható viszont a régebbi verziók elérése, szükség esetén bármikor „visszagörgethetjük" és igény szerint dolgozhatunk a régebbi komponensekkel.

Az, hogy a fejlesztőktől igényel figyelmet a verziók kezelése, rengeteg hiba forrása lehet. Egy elgépelt könyvtárnév, egy véletlenül felülírt régi verzió beláthatatlan következményekkel járhat.
Erre a problémára jelentenek megoldást az úgynevezett kódcentrumok vagy kódmenedzsment eszközök, amelyek automatikusan elvégzik a verziókezeléssel járó dolgokat. A fejlesztőknek csak be kell regisztrálniuk kódjukat, és innen kezdve a verziók mentése, a verziószám növelése és egyéb teendők is automatikusan végrehajtódnak, ezek nem igényelnek külön figyelmet.

Hasonló problémákat vet fel a projekthez kapcsolódó különféle dokumentumok kezelése is: mind a menedzsment, mind a tervezés és fejlesztés oldalán születnek olyan specifikációk, elemzések, iratok, amelyek historikus rögzítése elengedhetetlen. Ehhez is rendelkezésre állnak különféle dokumentumkezelő rendszerek.

A fenti megoldások azonban felvetnek egy igen fontos kérdést: mi történik akkor, ha ugyanazon a kódon vagy dokumentumon többen is dolgoznak egyszerre? Ha ezt a problémát nem kezeljük megfelelően, előfordulhat, hogy a későbbi mentések mindig felülírják a korábbiakat, így bizonyos változtatások elvesznek.
Amennyiben a file-hoz egyetlen ember férhet csak hozzá, a munka akadozik, sőt az is előfordulhat, hogy hosszabb időre is leáll, például mert valaki elfelejtette „elengedni" az adott dokumentumot, és közben elment nyaralni Java szigetére. *(Elnézést kérek mindenkitől, de .NET sziget sajnos még nincs... ☺)*
A kész rendszerek ezekre a problémákra is megoldást nyújtanak az úgynevezett check in – check out segítségével. Amikor a felhasználó *(fejlesztő, rendszertervező, menedzser stb.)* egy file-on szeretne dolgozni, annak egy lokális példányán megteheti azt. Amikor munkájával végzett, és be szeretné regisztrálni a módosításokat, a rendszer megnézi, más is változtatott-e az adott file tartalmán az eltelt idő alatt.
Amennyiben nem történt változás, nincs probléma, a dokumentum vagy kód egyszerű bemásolása és a verziószám lekezelése elegendőnek bizonyul. Gond akkor adódik, ha közben valaki más már módosított bizonyos dolgokat. Az összefésülés már nem bízható teljesen a rendszerre, hiszen ennek logikája minden egyes alkalommal más és más lehet, feltétlenül szükség van az emberi közreműködésre.
Az összefésülés előnye, hogy minden módosításról külön-külön eldönthetjük, hogy megtartjuk-e avagy eldobjuk. Így egy dokumentum élettörténete például az alábbiak szerint alakulhat:



A dokumentumok összefésülésének eredménye

Mind a dokumentumokhoz, mind a programkódokhoz szervesen kapcsolódik a dokumentációgenerálás. Ez többnyire úgy történik, hogy vagy a tervező- vagy a fejlesztőkörnyezet lehetőséget biztosít automatikus dokumentumgenerálásra *(többnyire HTML és/vagy RTF formátumban)*. A dokumentumok különböző UML diagramokból, hozzájuk tartozó kódokból, paraméterekből és szöveges megjegyzésekből állnak, formátumuk és pontos tartalmuk rendszerfüggő.

Ha már a kód- és dokumentumkezelést megoldottuk és gördülékenyen halad a fejlesztés, illik odafigyelni a rendszerrel kapcsolatos hibák kezelésére és feldolgozására is. A rendszer tesztelése alapvetően három szinten zajlik:

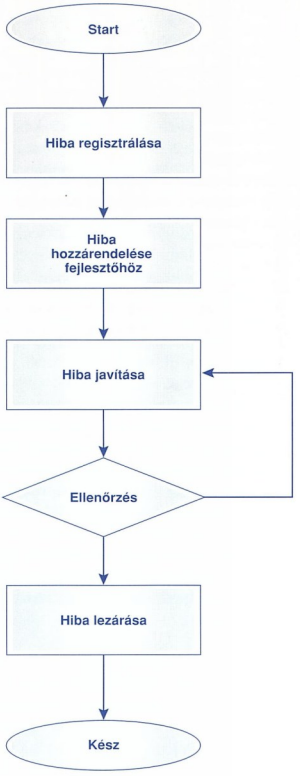
1. fejlesztői tesztek
2. üzleti és üzemeltetői tesztek
3. felhasználói tesztek

A fejlesztői teszteket értelemszerűen a fejlesztők végzik, a rendszer finomítása közben. Enélkül nem is fejlesztés a fejlesztés, hiszen ki látott már olyan kódot, ami elsőre fordult és hibátlanul futott volna?...

Az üzleti és üzemeltetői teszteket már külön tesztelésre szakosodott csapat végzi, a felhasználói tesztek pedig a végfelhasználónál történnek: vagy dedikáltan tesztelés céljából átadjuk a rendszert, vagy az éles átadás után, a használat során derül fény néhány hibára, hiányosságra.

A teszteléshez valamennyi fázisában szükségszerű, hogy legyen egy központi hely, ahol a teszteredményeket, hibajelentéseket tárolhatjuk. Ehhez tehát a fent említett valamennyi szereplőnek hozzá kell férnie, természetesen a megfelelő jogosultságokkal. Így aztán mindenki egységes felületen dolgozhat: itt rögzíthetők a tesztesetek, a hibajelenségek, itt lehet a hibát delegálni az illetékes fejlesztőnek vagy fejlesztőcsoportnak, itt lehet rögzíteni a hibajavítás tényét stb. Így aztán maguknak a hibáknak is saját életciklusuk van, amely rendszerről rendszerre változik, az alapmotívumok azonban általában ugyanazok: a hibát észlelő személy beregisztrálja a hibát, amelynek státusza ekkor még nyitott. A vezető fejlesztő *(vagy az ezért felelős személy)* hozzárendeli azt az érintett fejlesztőhöz, akinek a felelőssége a hiba javítása. A fejlesztő erről értesítést kap, majd a hiba javítása után bejelöli ennek tényét a rendszerben. Újabb teszteléssel ellenőrzésre kerül, hogy a hiba valóban kijavításra került-e, és ha igen, a hiba lezárható. Amennyiben mégsem tökéletes a javítás, újabb körök kezdődnek.

Honnan tudhatjuk azonban, hogy mi számít hibának és mi nem? Természetesen a technológiai bukások minden esetben *(a rendszer lefagyása, rendellenes viselkedése stb.)*, az üzleti hibázásokkal azonban sajnos nem ilyen egyértelmű a helyzet. Itt ugyanis egyértelműen tudni kell, mik a rendszerrel szemben támasztott igények, pontosan milyen üzleti folyamatokat valósít meg. Sajnos sok esetben a nem elég precíz specifikáció rengeteg félreértéshez vezet, a megrendelő ugyanis beleért olyan dolgokat is, amelyek a rendszer tervezőinek és fejlesztőinek eszébe sem jutottak. *(Lásd előző havi cikkem.)*



A hibajavítás folyamata

A fent említett problémákra természetesen megoldást jelenthet egy jó specifikáció. Ha ugyanis a specifikáció kellően pontos és részletes, nincs lehetőség arra, hogy belemagyarázzunk dolgokat. A jó specifikáció ugyanis nemcsak azt tartalmazza, hogy mit kell tudnia a megvalósítandó rendszernek, hanem azt is, hogy mi nem célja a futó fejlesztési projektnek. Így *(optimális esetben)* a szállító és a megrendelő között nem merülhetnek fel félreértések.
A jó specifikáció alapfeltétele a megrendelő *(a későbbi felhasználó)* igényeinek pontos rögzítése. Ennek egyik oldalát, a Use Case-ek *(használati esetek)* alkalmazását korábban már bemutattam. Most a technológiai háttérre, a különböző eszköztámogatottságra térnék ki.
Maguk a használati esetek, és általában az UML modellek kezelésére számtalan eszköz áll rendelkezésünkre. Az igényfelmérés azonban sokkal több UML diagramok rajzolásánál. Az igényeket rendezetten, könnyen elérhető, hozzáférhető módon kell tárolni. Ez alatt azt értem, hogy ha a követelményeket áttekinthető, rendszerezett formában tároljuk, a későbbiekben arra rendkívül egyszerűen hivatkozhatunk a fejlesztés bármely fázisában.
Szokás szerint valamilyen hierarchikus sorszámozást alkalmazunk. Ennek felhasználása a későbbiekben úgy történik, hogy egyrészt a dokumentációkban is eszerint hivatkozunk rá *(így egyértelmű, mire gondolunk, nem kell barokk körmondatokkal körülírni az adott követelményt)*, másrészt a kódok kommentezésében, ezáltal az egyes kódrészletek megértésében is sokat segíthet, ha szerepeltetjük, hogy az adott eljárás pontosan melyik funkciót valósítja meg.
Az igények pontos rögzítése a tesztelésnél is nagy segítséget nyújthat. A „hibagyanús" üzleti momentumokról pillanatok alatt kiderülhet, hogy valóban hibáról van-e szó, vagy pusztán olyan dologról, amit mi nem így csinálnánk, de a specifikáció szerint a megvalósítandó rendszer célja ez volt. Így a tesztesetek és a hibák rögzítésénél egyaránt hivatkozhatunk a követelmény- illetve a funkcióspecifikáció megfelelő pontjaira, ezzel is megkönnyítve kollégáink dolgát.

És ha mindezen feltételek teljesülnek? Akkor minden bizonnyal olyan körülmények között dolgozhatunk, amelyek nagyban elősegítik a sikeres fejlesztési folyamatot. A fentieket kiegészíthetjük még különböző tervezési illetve fejlesztési ajánlások bevezetésével, alkalmazásával. Ilyenek lehetnek például az MSF *(Microsoft Solutions Framework)* vagy a RUP *(Rational Unified Process)* stb.

A fentiek bevezetéséhez azonban a fejlesztő cég belső kultúráját, működését kell először is feltérképezni, majd meghatározni a szükséges előrelépéseket. Ehhez azonban alapos körültekintésre és felkészülésre van szükség, hiszen a rosszul előkészített változtatások ellenérzést váltanak ki munkatársainkból, és az elérni kívánt célok meghiúsulnak.

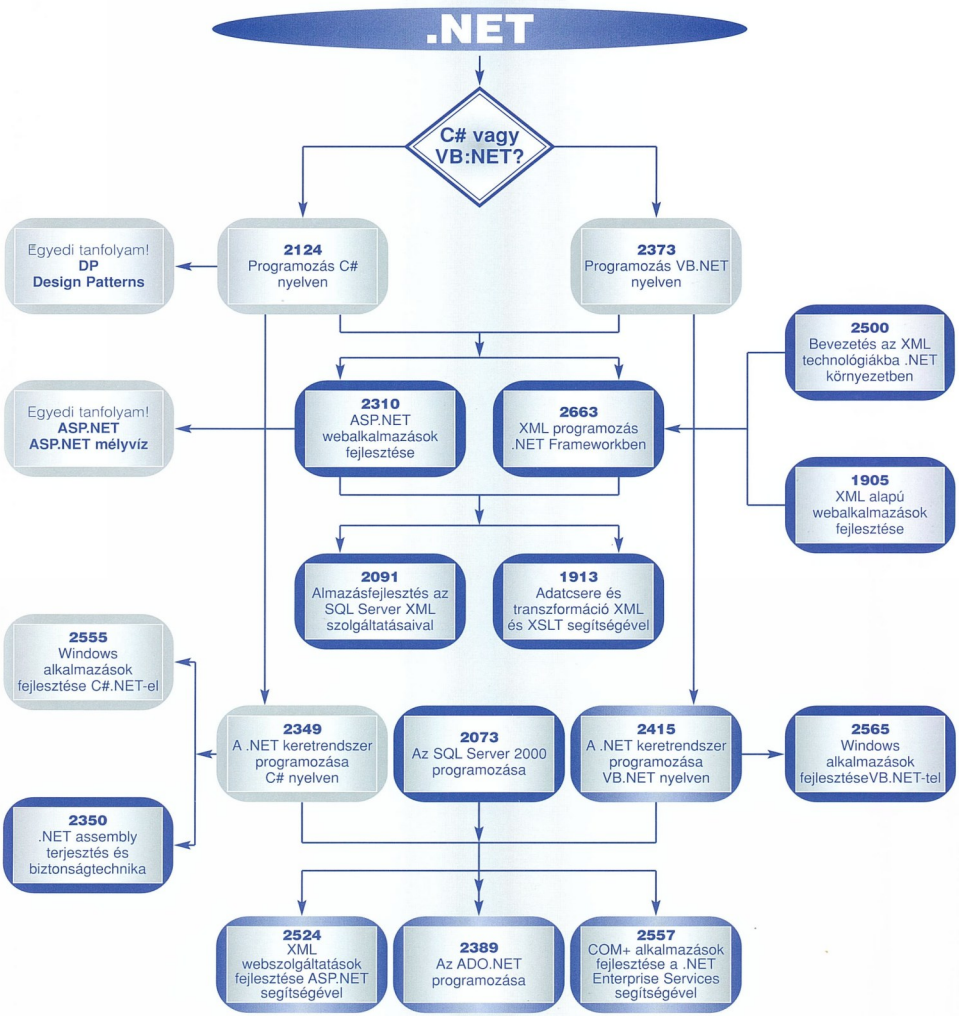
Ehhez kívánok mindenkinek sok sikert és kitartást!

Molnár Ágnes
Molnar.Agnes@cleware.hu

netACADEMIA

A LEGJOBBAKAT TANÍTJUK

Fejlesztői tanfolyamok térképe



www.netacademia.net • 1062 Budapest, Andrássy út 62. • Telefon: (1) 472-1214 • Fax: (1) 472-1215

A 2004-es év újdonságai

A SZÁMALK Továbbképzés a 2004-es évben is számos új képzéssel és szolgáltatással várja ügyfeleit. A tanfolyamokról, oktatási konstrukciókról további információkat weboldalunkon talál.

Cisco hálózati rendszermérnök képzés

Windows Server 2003 és szerveroldali tanfolyamok

Office 2002/2003 képzések, megújult ECDL tanfolyamok

Kedvezményes Windows 2003 rendszeradminisztrátor és rendszermérnök (MCSA/MCSE) képzések

Megújult .NET fejlesztői képzési csomagok

Microsoft CERTIFIED Professional

Microsoft GOLD CERTIFIED Partner

Microsoft CERTIFIED Technical Education Center

Minden tisztelt ügyfelünknek, partnerünknek ezúton is kellemes Karácsonyt és eredményekben gazdag, boldog új esztendőt kívánunk. Reméljük, hogy a 2004-es évben is igénybe veszik majd szolgáltatásainkat.

SZÁMALK

www.szamalk.hu/tisza

Tel.: 203-0304/3050 (Simon Fere
1115 Budapest, Etele út 68.