



Informatikai technológiák laboratórium 2

Szolgáltatási szint menedzsment

Mérési útmutató

Készítette: Kocsis Imre, Paljak Gergely
ikocsis@mit.bme.hu , paljak@mit.bme.hu

2009. november 17.

Verzió: 1.3

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 A mérés menete

A mérésen az itt megadott feladatokat végezzük el. A mérési jegyzőkönyvben hangsúlyosan elvártan dokumentálandó és a mérés végén a mérésvezetőnek bemutatandó feladatokat külön kiemeljük. (A mérési jegyzőkönyvnek persze nem csak a jelölt elemeket szabad és kell tartalmaznia.)

A mérés megkezdése előtt gondosan olvassák végig a teljes útmutatót!

0. Felkészülés

- a. Tekintsük át a mérési elrendezés dokumentációját és általánosságban még egyszer a mérési segédletet!
- b. Gondoljuk át, hogy milyen feladatokat kell elvégeznünk és ezek hogyan épülnek egymásra!

1. Felügyelt infrastruktúra megismerése

- a. Indítsuk el a felügyelt virtuális gépeket és jelentkezzünk be rájuk.
- b. Próbáljuk ki a szempontunkból fontos szoftverkomponensek (MySQL, Tomcat, Apache) elindítását és leállítását!
- c. Ellenőrizzük a gazdagépről a szolgáltatásokat futó és nem futó állapotban (ügyelve a böngésző közttestárának ürítésére, ha böngészőt használunk)!

2. Ismerkedés az OMNIBus-szal

- a. Indítsuk el a TBSM virtuális gépet és jelentkezzünk be rá!
- b. Ellenőrizzük, hogy futnak-e a szükséges szolgáltatások (Netcool GUI Foundation Server, Netcool Postgres Database, Netcool Security Manager Server, Netcool/OMNIBus Object Server, NCO Flex License Manager, NCO Process Server)!
- c. Indítsuk el az Event List alkalmazást és ismerkedjünk meg a felhasználói felületével!
- d. Indítsuk el a Netcool/OMNIBus Administrator alkalmazást és keressük meg benne a mérési segédletben dokumentált, számunkra fontos elemeket!
- e. Navigáljunk el az `alerts.status` tábla adattartalom-nézetéig és injektáljunk egy tetszőleges tesztriast, majd figyeljük meg, hogy hogyan jelenik meg az üzenet az Event List-ben! **Ez az alfeladat dokumentálandó.**
- f. Hasonlóan tesztüzenet(ek) injektálásával mutassuk meg, hogy a mérési segédletben bemutatott beépített triggererek valóban működnek! **Ez az alfeladat dokumentálandó.**

3. Adatgyűjtés konfigurálása

- a. Nyissuk meg az Eclipse fejlesztőkörnyezetet és indítsuk el az adatgyűjtő alkalmazást! Győződjünk meg róla, hogy az alkalmazás a megfelelő adatokat írja a log állományba!
- b. Ellenőrizzük a GLF szonda `.props` állományát, majd először alakítsunk ki egy olyan `.rules` állományt, mely minden log bejegyzéshez külön riasztást generál! Ellenőrizzük az üzenetek megjelenését az Event List alkalmazásban! Mutassa be, hogy a naplófájl egyes adatait mely OMNIBus-beli attribútumhoz rendelte és miért. **Ez az alfeladat dokumentálandó.**
- c. Az Event List alkalmazásban hozzunk létre egy nézetet, melyben csak a „glf” Manager-től érkező üzenetek látszanak! **Ez az alfeladat dokumentálandó és bemutatandó.**

4. Rendszerelemek rendelkezésre állásának számítása

A feladat során a rendszer két logikai elemének (Apache HTTP szerver operációs rendszer, az egyik Tomcat szerver-szolgáltatása) rendelkezésre állását kell kiszámítani, és ezt eseményként megjeleníteni.

- a. Gondolja végig a feladatot, és szükség esetén módosítsa a korábban létrehozott `.rules` állományt. **Ez az alfeladat dokumentálandó.**
- b. Hozzon létre egy-egy olyan eseményt, amely nyilvántartja a rendelkezésre állást! Használja az `alerts.status` táblában tárolt adatokat és a temporal trigger lehetőségeit. A megoldást a következő módszer szerint javasoljuk:
 - i. Hozzon létre olyan SQL lekérdezést, amely kiszámítja a rendelkezésre állást a regisztrált események alapján.
 - ii. Illessze be ezt a lekérdezést egy új temporal triggerbe, és a lekérdezés eredménye alapján hozzon létre egy új eseményt, amely nyilvántartja a logikai elem rendelkezésre állását. Illessze be az eseményt az `alerts.status` táblába.
 - iii. A rendelkezésre állás függvényében különböző súlyosságú legyen a riasztás! A rendszergazda lássa, hogy mikor megfelelő a rendszer, mikor közelít az SLA megsértéséhez és mikor sértette már meg. Ossza három részre a rendelkezésre állás értékkészletét, és ehhez rendelje hozzá a különféle riasztásokat.

A rendelkezésre állás aktuális értéke legyen folyamatosan nyomon követhető Event List-ben! **Ez az alfeladat dokumentálandó és a különféle események között a változás demonstrálandó.**

- c. Módosítsa úgy az egyik temporal triggert, hogy egy csúszóablakot használjon a rendelkezésre állás kiszámításához; vagyis a ne a legelső méréstől kezdve vegye figyelembe a mért állapotokat, hanem csupán az utolsó néhány perc adatait. **Ez az alfeladat dokumentálandó.**
- d. Szorgalmi feladat: a fenti feladat nem a teljes szolgáltatás, csupán az egyes komponensek rendelkezésre állását vizsgálja, készítsen olyan eseményt, amely a teljes szolgáltatás rendelkezésre állását tartja nyilván!

A Java lekérdező program, amely a naplóállományt írja nem ellenőrzi, hogy az Apache HTTP szerver valóban a Tomcateken futó TPC-W alkalmazás oldalait szolgálja ki, vagy csupán a webszerver szokásos nyitóképernyőjét (ez akkor is elérhető, ha csupán az Apache HTTP szerver fut, semelyik másik)! A teljes szolgáltatás akkor működőképes, ha fut a MySQL, legalább egy Tomcat és az Apache HTTP.

5. IBM Tivoli Business Service Manager demo

2 A mérési infrastruktúra

A mérésen egy három rétegű tesztinfrastruktúrát használunk, amelyet az OMNIBus felügyel. A mérés során 5 virtuális gépre lesz szükség, ezek közül négy virtuális gép a tesztinfrastruktúra, míg az ötödik a rendszerfelügyetet valósítja meg. Először a tesztinfrastruktúrát mutatjuk be (lásd 1. ábra):

Mind a négy gép CentOS 5.3 operációs rendszert futtat, a négy gépet előrekonfiguráltuk, a mérés során csak elindítani és leállítani kell a kiszolgáló alkalmazásokat. Elterjedt szerveralkalmazások vannak telepítve: egy MySQL adatbázis szerver, két Apache Tomcat Java Servlet konténer, egy Apache HTTP szerver. Ezek egy közismert benchmarkalkalmazást, a TPC-W-t futtatják.

A TPC-W egy elektronikus könyvtárház specifikációja, ennek egy Java Servlet implementációját fogjuk most használni. A megfelelő `jar` fájl már fut a Tomcat alkalmazás szerveren, az adatbázist korábban létrehoztuk a MySQL szerveren. A web szervernek jelen konfigurációban annyi a feladata, hogy a telepített `mod_jk` plug-in segítségével úgy továbbítsa a kéréseket a két Tomcatnek, hogy kiegyenlítse a terhelést (load balancing), ennek az alapértelmezett stratégiája round robin elven működik. Tehát HTTP szerver nem végez tényleges kiszolgálást, csupán továbbítja a kéréseket.

A Linuxot futtató gépek úgynevezett kapcsolt klónok (linked clone), vagyis egy közös virtuális gép az alapjuk. Amennyiben induláskor nem találják a közös virtuális gépet a tallózás gomb segítségével

mutassuk meg, a template mappában található, először a CentOS-5.3-4GB-graph virtuális gépet kell kijelölni. Mivel a CentOS-5.3-4GB-graph maga is egy kapcsolt klón, a legelső indítás alkalmával a CentOS-5.3-4GB virtuális gépet is meg kell mutatni. A virtuális gépek első indításakor válassza az „I moved it” opciót!

A mérésen root felhasználóként fogunk végezni minden műveletet, a mérés időkorlátaira való tekintettel. Felhívjuk a figyelmet arra, hogy ez üzleti célokat kiszolgáló, a külvilágot (Internetet) elérő szerverek esetén szigorúan ellenjavallott és komoly biztonsági kockázat.

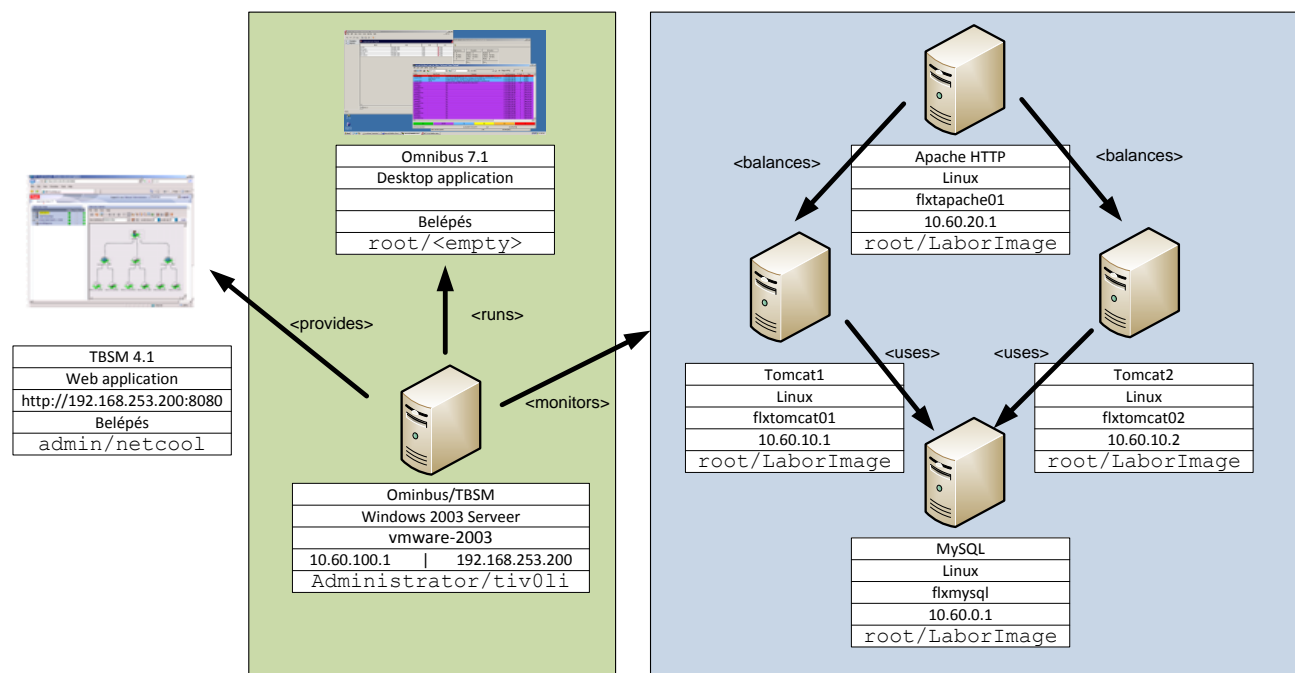
A Linuxot futtató gépekre a root felhasználóval LaborImage jelszóval tudnak belépni. A mérés során nem lesz szükség grafikus felület futtatására, ha mégis szeretnék futtatni, a startx paranccsal megtehetik (ebben az esetben érdemes növelni a virtuális gép számára allokkált memória mennyiségét).

A Linux virtuális gépeket a shutdown -h 0 paranccsal tudják leállítani. A futó folyamatokat a ps aux paranccsal tudják listázni, folyamatot a kill <folyamat azonosító> paranccsal tudnak terminálni.

Az ötödik virtuális gép Windows 2003 Servert futtat, ezen fut az OMNIBus 7.1. A gépre Administrator felhasználó, tiv01i jelszóval kell belépni. Figyeljünk rá, hogy a gép indításakor a Windows még angol (EN-US) billentyűzetkiosztás szerint várja jelszó beütését, de ezután már átáll magyar billentyűzetkiosztásra.

Az OMNIBus használatához a root felhasználó szükséges, a jelszó üres. A virtuális gépen indítható továbbá a Tivoli Business Service Manager 4.1, erre a mérés során nem lesz szükség.

A mérési elrendezést az 1. ábra szemlélteti, a fontosabb technikai adatokat a 2.1 fejezetben tekintjük át.



1. ábra Mérési elrendezés

2.1 Mérési elrendezés

A virtuális gépek hosztnevei és IP címei:

1. táblázat Hosztnevek és IP címek

Gépnév	Hosztnév	IP cím	IP cím2
MySQL	flxmysql	10.60.0.1	-
Tomcat1	flxtomcat01	10.60.10.1	-
Tomcat2	flxtomcat02	10.60.10.2	-

HTTP	flxapache01	10.60.20.1	-
Tivoli	vmware-2003	10.60.100.1	192.168.253.200

Fontosabb URL-ek a mérési környezetben:

2. táblázat Fontos URL-ek a mérési környezetben

Oldal neve	URL
Apache HTTP, TPC-W alkalmazás	http://10.60.20.1/tpcw/
Apache HTTP, nyitóoldal	http://10.60.20.1/
Apache HTTP, load balance státusz	http://10.60.20.1/jkstatus/
Apache HTTP, tesztoldal	http://10.60.20.1/test/stest.jsp
Tomcat nyitóoldal	http://10.60.10.1:8080
Tomcat menedzsment oldal	http://10.60.10.1:8080/manager/html belépés: tomcat/password
Tomcat, TPC-W alkalmazás	http://10.60.10.1:8080/tpcw/
Tomcat, tesztoldal	http://10.60.10.1:8080/test/stest.jsp

A két Tomcat konfigurációja teljesen szimmetrikus, a fenti URL-ekben az IP címeket átírva a másik szerver megfelelő oldalait is el tudjuk érni.

2.2 Alapvető szerveralkalmazás-kezelés

Röviden ismertetjük, hogy a virtuális gépekre belépve hogyan érhetik el a szerveralkalmazások alapvető funkcióit.

2.2.1 MySQL

Indítás: `service mysqld start`

Státusz lekérdezés: `service mysqld status`

Leállítás: `service mysqld stop`

Tesztelés:

Adatbázis kliens indítása: `mysql` (a további parancsok a kliens parancssorába gépelendők, figyelve a parancsot lezáró pontosvessző karakterre)

Adatbázis kiválasztása, a tesztkonfigurációban az `std` nevű adatbázist használjuk: `use std;`

Az kiválasztott adatbázis tábláinak megjelenítése: `show tables;`

Tetszőleges SQL parancsok kiadható (pl. `select * from orders;`)

Kilépés a kliensből: `quit;`

2.2.2 Tomcat

Indítás: `/usr/tomcat/apache-tomcat-5.5.26/bin/startup.sh`

Leállítás:

`telnet localhost 8005`

SHUTDOWN

Státusz lekérdezés: egy böngésző segítségével megnézhetjük az alapértelmezett nyitóoldalt (kiszolgálás a 8080-as alapértelmezett porton történik, további URL-ek lásd később) vagy: `telnet localhost 8080` (kilépés: `Ctrl+5`, `Enter` majd `quit`.)

2.2.3 Apache

Indítás: `service httpd start`

Státusz lekérdezés: `service httpd status`

Leállítás: `service httpd stop`

3 A tesztinfrastruktúra monitorozása

A mérés során két tesztet fogunk használni a monitorozás megvalósítására: a „pingelést” és a `http` szolgáltatás elérhetőségének a figyelését. Ez a két teszt a TBSM virtuális gépen, egy Java alkalmazásban került implementálásra. Az alkalmazás egy szöveges állományba naplóz, melyet a mérés során a Generic Logfile (GLF) szondával (lásd 4. fejezét és a korábban kiadott segédlet) fogunk megfigyelni. Életszerű szituációkban az ad-hoc Java alkalmazást a ping szonda, illetve egyéb, az OMNIBus-szal integrálható célalkalmazások (un. szolgáltatás-monitorok) helyettesítenék; a mérés viszonylagos rövidege miatt ezek alkalmazásától esetünkben eltekintünk.

Az Eclipse 3.4.1 fejlesztőkörnyezet, melyben az alkalmazás kialakításra került egy hivatkozáson keresztül elérhető a TBSM virtuális gép asztaláról. Igény esetén a kód módosítható (pl. a logállomány neve vagy a várakozási intervallumok értéke megváltoztatása céljából). Demonstrációs célú programról lévén szó, az alkalmazást a fejlesztőkörnyezetből javasoljuk indítani és leállítani (`CTRL+F11`, illetve a Console nézeten értelmezett „Terminate” gomb).

A szolgáltatás ellenőrzése egy egyszerű HTTP lekérdezés segítségével zajlik, az Apache HTTP szervert a 80-as, a két Tomcat szerver a 8080-as porton kérdezzük le. A MySQL nem nyújt HTTP prtokollon elérhető felületet, a MySQL szolgáltatást nem monitorozzuk. A monitorozás „UP” értéke nem jelenti azt, hogy a helyes oldal került kiszolgálásra, csupán azt, hogy kiszolgálás történt; pl. ha a MySQL szervert lekapcsoljuk, attól még a HTML oldalakat kiszolgálják a szerverek, de a dinamikus tartalom (pl. könyvek listája) üres lesz.

Az alkalmazás a `C:\javalog.txt` állományba ír (azt minden induláskor hozzáfűzésre megnyitva) a következő formátumban:

```
Mon Nov 23 16:02:42 CET 2009: Apache OS is DOWN
Mon Nov 23 16:02:42 CET 2009: Tomcat1 OS is DOWN
Mon Nov 23 16:02:42 CET 2009: Tomcat2 OS is UP
Mon Nov 23 16:02:42 CET 2009: MySQL OS is UP
Mon Nov 23 16:02:42 CET 2009: Apache Service is DOWN
Mon Nov 23 16:02:42 CET 2009: Tomcat1 Service is DOWN
Mon Nov 23 16:02:42 CET 2009: Tomcat2 Service is DOWN
Mon Nov 23 16:02:45 CET 2009: Apache OS is DOWN
Mon Nov 23 16:02:45 CET 2009: Tomcat1 OS is DOWN
Mon Nov 23 16:02:45 CET 2009: Tomcat2 OS is UP
Mon Nov 23 16:02:45 CET 2009: MySQL OS is UP
Mon Nov 23 16:02:45 CET 2009: Apache Service is DOWN
Mon Nov 23 16:02:45 CET 2009: Tomcat1 Service is DOWN
Mon Nov 23 16:02:45 CET 2009: Tomcat2 Service is UP
[...]
```

A TBSM gépre telepített, de automatikusan nem induló GLF szonda szolgáltatás `.props` állománya előre konfigurált, de az eseménygenerálás szabályai nem; azok kialakítása a mérés részét képezi.

4 OMNIBus Generic Logfile (GLF) szonda

A GLF szonda alapvető funkcióit a korábban kiadott mérési segédlet 4.4.2-es pontja ismerteti, itt csak néhány a méréshez különösen fontos funkcióra térünk ki. További, részletes információt a [4] és az [5] tartalmaz.

A mérési konfigurációban a GLF szolgáltatást pontos neve: NCO Generic Logfile Probe. Amennyiben megváltoztatjuk a szonda konfigurációját (`.rules` állomány), a szolgáltatást újra kell indítani, hogy életbe lépjen az új konfiguráció.

4.1 Az update függvény

A mérés során szükség lehet annak a finombeállítására, hogy azonos kulcsú (`Identifier`) esemény beszúrásakor mely mezőket írjuk felük, melyeket ne. Az OMNIBus parancsállomány-nyelvében elérhető `update` függvénnyel azt adhatjuk meg, hogy ha az aktuálisan összeállított, beszúrni kívánt üzenet (az `Identifier` mező már létező esemény `Identifier` mezőjével egyezése miatt) újra beszúrásnak minősül, akkor mely mezők kerüljenek mindenképpen frissítésre és melyek semmiképpen sem. Fontos: ezek a deklarációk felülbírálják a deduplication triggerben megvalósított logikát! Az `update` függvény szintaxisa:

```
update(fieldname [, (TRUE | FALSE) ] )
```

Alkalmazási példaként tekintsük a GLF szonda egy testre szabott `.rules` állományát. A monitorozott log állományban két, szóközzel elválasztott tokent várunk, a sorokat CR+LF választja el. Az első token mindig az érintett (logikai) komponenst azonosítja, a második pedig az „UP”, vagy a „DOWN” karaktersorozat.

[...]

```
@Class = 0
$tempdate = getdate
@Identifier = $FieldVal01
@Summary = $FieldVal02
@Node = $FieldVal01
@Manager = "glf"
@FirstOccurrence = $tempdate
@LastOccurrence = $tempdate
if(match($FieldVal02, "DOWN"))
{@Severity = 5
} else {
@Severity = 1
}
update(@Summary, TRUE)
update(@Severity, TRUE)
update(@FirstOccurrence, FALSE)
update(@LastOccurrence, TRUE)
}
```

A szabályállományban megfogalmazott fontosabb deklarációk a következők.

- Az üzenet az alapértelmezett osztályba tartozik.

- A `tempdate` segédváltozóhoz rendeljük az aktuális időbélyeget (a `getdate` az `update`-hez hasonlóan beépített függvény), melyet az üzenet első és legutolsó megjelenésének beállításához használunk.
- Az üzenet egyértelmű azonosítója a logikai komponens neve.
- Az üzenet összefoglalása a komponens állapota.
- Ha a logikai komponens állapota „DOWN”, akkor az üzenet súlyossága legyen 5 (`Critical`), minden egyéb esetben pedig 1 (`Indeterminate`).
- Ha deduplikációra kerül sor, úgy a `Summary`, `Severity` és `LastOccurrence` mezőket mindenképpen frissítsük, a `FirstOccurrence`-t azonban semmiképpen sem.

4.2 Segítség rules állományok hibakereséséhez

Hasznos tudni a következőket, ha OMNIBus Generic Logfile Probe rules állományt szerkesztünk:

- A szondák állományait a `$OMNIHOME/probes/arch/` elérési úton találják, ez a mérési környezetben: `C:\IBM\Netcool\omnibus\probes\win32`
- Debug módban indítható a szonda az alábbi paranccsal:
`$OMNIHOME/probes/arch/probename -messagelevel DEBUG -messagelevel
 STDOUT`
- Az `alerts.status` tábla `Identifier` attribútuma elsődleges kulcs. Gondoljuk át, mit jelent ez, amikor az eseményeket illesztjük be táblába.
- Az `alerts.status` a mérési feladatok során nyugodtan kiüríthetik, ha az aktuális feladat úgy kéri. Az összes sor kijelölése után (`Ctrl+A`) kattintsanak jobb klikkel a táblára, majd `Delete row`.

5 OMNIBus Időzített Triggerek használata

A triggereket a korábban kiadott segédlet 4.1-es pontja ismerteti, részletes leírásuk a [3]-ban található. Az időzített triggerek (temporal trigger) bizonyos időközönként automatikusan lefutó programok, a 4. feladat megoldásához szükség lesz triggerek használatára, ezért itt néhány gyakorlati, a mérés során is használatos beállítást mutatunk be:

- Triggert létrehozni az OMNIBus Administrator Automation menüjének Triggers menüpontjában lehet.
- Érdemes saját trigger létrehozása előtt áttekint a segédletben lévő példákat, illetve a hasonló beépített triggereket, pl: `mail_on_critical`, `expire`.
- Az „Evaluate” fülön SQL lekérdezéseket adhatunk meg. A lekérdezések eredménye egy tábla, amelynek a nevét a „Bind As:”mezőben kell megadni.
- Aggregáló függvényeket használhatunk (pl. `min()`, `max()`, `avg()`) az Evaluate fülön, de ez esetben ügyeljünk rá, hogy az „as” kulcsszóval adjunk nevet az oszlopnak, hogy később tudjunk rá hivatkozni az „Action” fülön.
- Az „Action” fülön kell megadni a konkrétan lefuttatandó programot, amelyet `begin` utasítás nyit és `end` zár. Az programot záró `end` után nem kell pontosvesszőt kitenni, egyéb esetben az utasításokat itt is pontosvessző zárja.
- Itt is használhatjuk szokásos SQL utasításokat, pl.
 - o `delete from alerts.status where Identifier = 'apache';`
 - o `insert into alerts.status (Identifier, Summary) values ('apache', 'apache availability');`

- Figyeljünk rá, hogy számot, csak integer típusú attribútum értékeként tudjuk megadni.
- Továbbá használhatóak egyszerű vezérlési szerkezetek, pl.

```
[..]
if (a.b > 900)
then
    [...]
elseif (a.b > 500)
then
    [...]
elseif (a.b > -1)
then
    [...]
end if;
[...]
```

- Az Action fülön hivatkozhatunk az Evaluate fülön előre kiszámított táblára. Tegyük fel, hogy a „Bind as” mezőben `availability` nevet adtunk:

```
[....]
for each row a in availability
begin
    [...]
end;
[...]
```

- A fenti `for` cikluson belül `a.<attribútum neve>` módon hivatkozhatunk az aktuális sor attribútumaira
- Figyeljen rá, hogy ha egész szám típusú attribútumra hivatkozik ilyen módon, akkor azt csak `integer` típusú attribútumba lehet beilleszteni, pl. `varchar[]`-ba nem.
- Ha szükségesnek érzi `System/Databases/status.alerts/Column Definitions` fülön jobb kattintás után az `Add column` gombbal adhat hozzá új attribútumokat a táblához.

6 További irodalom:

- [1] IBM Tivoli Netcool/OMNIBus 7.1 Infocenter (teljes dokumentáció)
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIBus.doc/welcome.htm
- [2] IBM Tivoli Netcool/OMNIBus 7. User's Guide
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/ug/om71ug.pdf
- [3] IBM Tivoli Netcool/OMNIBus 7.1 Administrator's Guide
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/ag/om71ag.pdf
- [4] IBM Tivoli Netcool/OMNIBus 7.1 Probe and Gateway Guide
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/pg/om71pg.pdf
- [5] IBM Tivoli Netcool/OMNIBus 7.1 Probe Rules File Syntax
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/pg/CHDHEFHB.html