SNMP-Traphandling für Nagios HowTo

1 Versionen

Version 0.1 - erstellt von Ingo Lantschner November 2006

Version 0.2 - Ergänzung BE Konfig./ SNMP-Traps von Scripts

2 Einleitung

Mit SNMP steht ein breit unterstütztes Informationssystem über Vorgänge im Netzwerk zur Verfügung. Neben den klassischen Netzwerkkomponenten wie Switches und Router unterstützen auch Serverhersteller (z. B. Fujitsu-Siemens mit FuSi ServerView) und PC-Applikationen wie zum Beispiel BackupExec SNMP. Letztere senden ausschließlich Traps, d.h. die Backupanwendung signalisiert von sich aus, wenn Backupjobs erfolgreich waren oder fehlschlugen.

Nagios liest im Allgemeinen mittels aktiver Checks die Zustände aus, das geht auch im SNMP-Kontext schön, wenn die Informationen vom SNMP-Agent des geprüften Systems auf jederzeitigen Abruf hin zur Verfügung gestellt wird. Die aktuelle Drehzahl eines Lüfters oder die Uptime des Servers kann so gut abgefragt werden. Dies ist bei SNMP-Traps naturgemäß nicht möglich; Traps werden gesendet und sind schon vergessen. Folgend die nötigen Schritte zum erfolgreichen Durchreichen solcher Traps an Nagios. Als Beispiel nehmen wir BackupExec auf einem Windows Server 2003. Sinngemäß das selbe können wir tun, wenn wir auf UNIX arbeiten.

3 Ziel

Im weiteren Verlauf wollen wir folgendes erreichen:

- Nagios soll alle Fehler des BE-Programms anzeigen.
- Nagios soll den letzten erfolgreichen Backupjob anzeigen. Wenn länger als eine Woche keine erfolgreichen Jobs signalisiert wurden, soll der Status von OK auf CRITICAL wechseln.

3.1 Konfiguration:

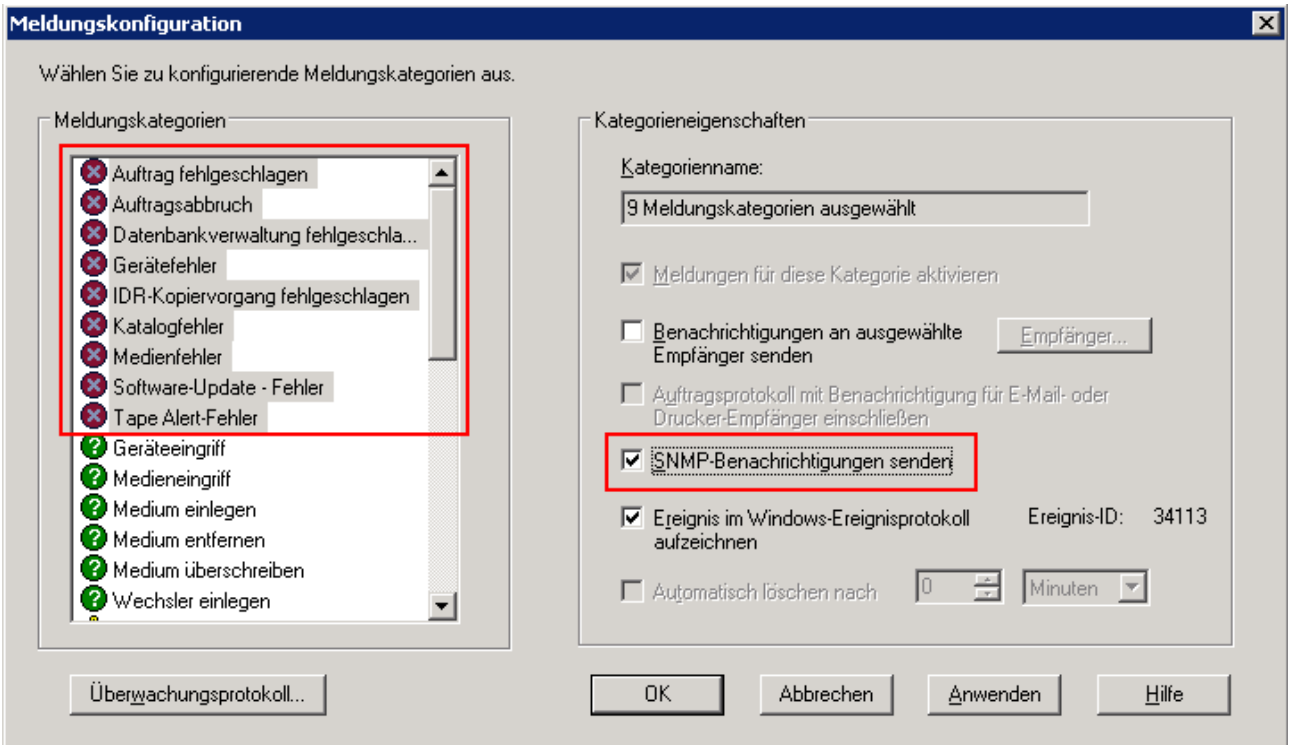
Wir gehen von einem Nagiosserver 10.10.4.11 und dessen Backup 10.10.4.12 aus.

Wichtige Voraussetzung: Die hier beschriebene Vorgangsweise setzt voraus, dass in den Hostdefinitionen am Nagiosservers der host_name gleich dem FQDN des Hosts ist.

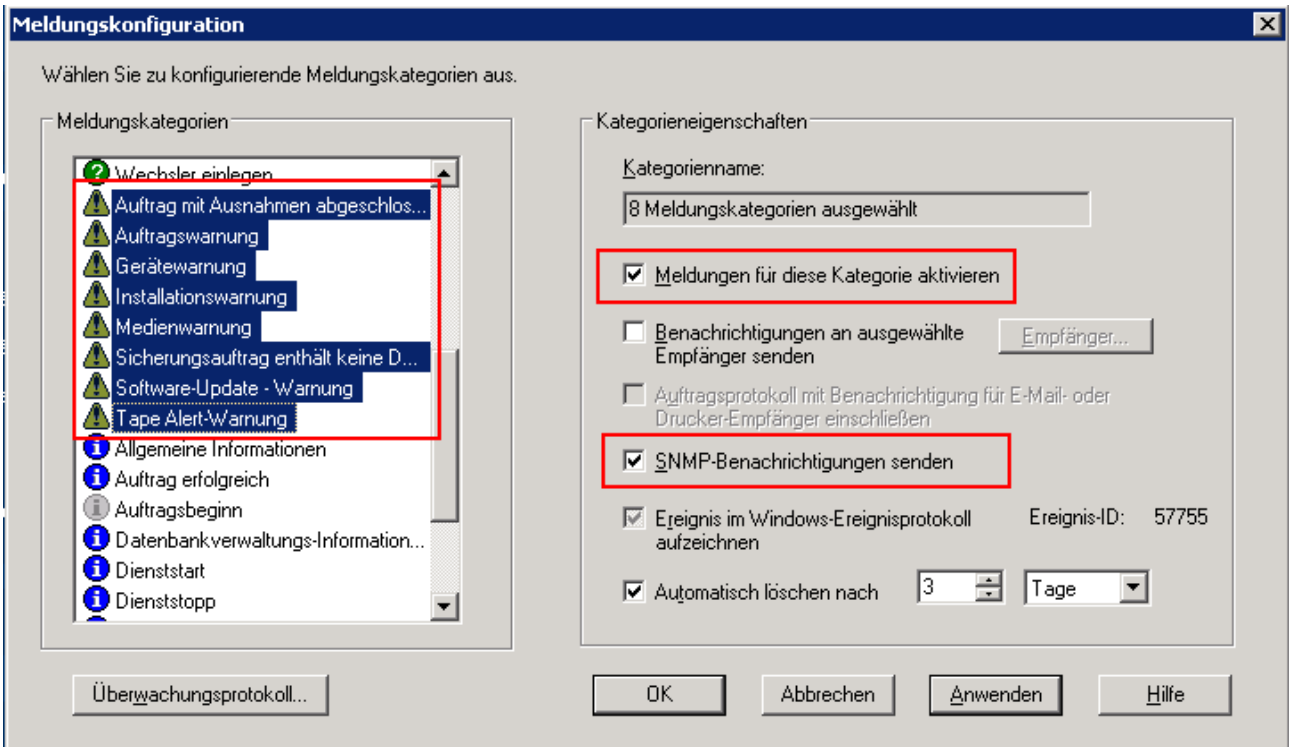
4 Konfiguration des Backupserver

4.1 Konfiguration der BE Software

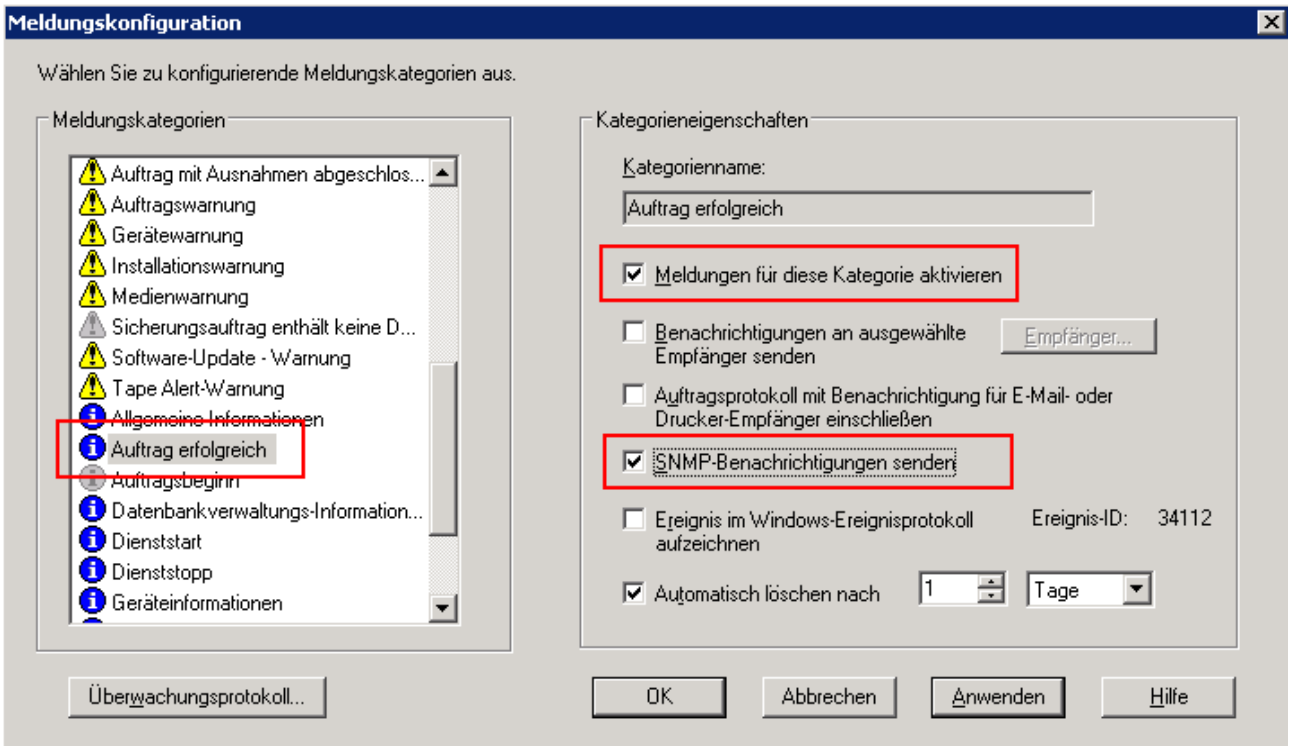
Die Basis ist natürlich ein korrekt konfiguriertes BackupExec. Die Meldungskategorien in BE sind zunächst so einzustellen, dass alle Fehler und Warnungen per SNMP signalisiert werden. Weiters ist die Kategorie für erfolgreiche Aufträge zu aktivieren und so einzustellen, dass per SNMP signalisiert wird. Das GUI von BackupExec ist dafür vollkommen ausreichend, die Konfiguration erfolgt im Menüpunkt *Meldungskategorien einrichten ...* .



Die Konfiguration der Fehlermeldungen ...



... der Warnungen ...



... und die der erfolgreichen Abschlüsse.

4.2 Konfiguration des SNMP-Dienstes

Der von MS für Windows Server 2003 mitgelieferte SNMP-Agent (SNMP-Dienst) ist ausreichend. GGf. muss er nachinstalliert werden. Jedenfalls ist er aber zu konfigurieren. Entscheidend ist, dass bei der richtigen Community die nötigen Trapdestinations eingetragen werden. In unserem Beispiel werden wir sowohl den Nagiosserver als auch sein Backup eintragen, also die IPs 10.10.4.11 und 10.10.4.12.

Somit stellen wir sicher, dass jeder von einer SNMP-fähigen Anwendung auf diesem Server generierte Trap an diese beiden Hosts gesendet wird. Da UDP ein sehr sparsames Protokoll ist, ist der Overhead durch die doppelte Eintragung vernachlässigbar.

Nach der Konfiguration ist der Service neu zu starten.

Für ein automatisiertes Rollout empfiehlt sich die Verwendung eines Scripts:

```
:: Konfiguriert den SNMP-Dienst

net stop snmp
regedit SNMP.reg
net start snmp
```

Die Datei SNMP.reg schaut zum Beispiel so aus:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters]
"EnableAuthenticationTraps"=dword:00000001

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\PermittedM
anagers]
"1"="10.10.4.11"
"2"="10.10.4.12"
"3"="localhost"
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\TrapConfig
uration]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\TrapConfig
uration\edvgsoerver]
"1"="10.10.4.11"
"2"="10.10.4.12"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\ValidCommu
nities]
"public"=dword:00000004
```

Tipp: Am einfachsten generiert man diese Datei mit der Anwendung `regedit.exe` aus einem bestehenden, korrekt konfigurierten Musterserver.

Somit sind die Konfigurationsarbeiten am zu überwachenden Backupserver beendet und wir können uns der etwas komplexeren Konfiguration am Nagiosserver widmen.

5 Konfiguration des Nagiosservers

Wir gehen davon aus, dass Nagios bereits installiert ist. Zu aller erst müssen wir sicherstellen, dass der Daemon für SNMP (snmpd) ebenfalls installiert ist. Dazu können wir zum Beispiel nach seinem init-script `/etc/init.d/snmpd` oder seinen Konfigurationsdateien (z.B. `/etc/snmp/snmptrapd.conf`) suchen. Auch ein erfolgreiches man snmptrapd deutet auf einen bereits installierten SNMP-Server hin.(Debian: `$ dpkg -l | grep snmpd`)

Ist er nicht installiert, so ist das nachzuholen. Auf den Debian-basierenden Linuxen geschieht dies z.B. durch

```
# apt-get install snmpd
```

So nun geht es an dessen Konfiguration ...

5.1 Konfiguration snmptrapd

In der Konfigurationsdatei des SNMP-Trap-Daemons wird diesem mitgeteilt, wie er mit bestimmten Traps zu verfahren hat. Wir benötigen nur eine Zeile, die alle Traps der zu BackupExec gehörenden OID (1.3.6.1.4.1.1302) einem Eventhandler von Nagios übergibt.

```
# /etc/snmp/snmptrapd.conf.

doNotLogTraps yes
ignoreAuthFailure yes

# BE Traps
traphandle 1.3.6.1.4.1.1302 /usr/local/nagios/libexec/eventhandlers/handle-BE-trap
```

Weiters ist festzulegen, dass der Trapdaemon vom init-script gestartet wird. Das ist nicht bei jedem UNIX selbstverständlich. In Ubuntu-Linux muss z.B. in der Datei `/etc/default/snmpd` die dafür vorgesehene Variable auf `yes` gesetzt werden.

5.2 Traphandler

Auf Grund der obigen Konfiguration übergibt der snmptrapd jene Traps die von BE stammen einem Skript `handle_BE-trap`. Dieses Script ist als Shellscript ausgeführt und verfügt über ein bisschen Intelligenz. Immerhin kümmert es sich darum, die Erfolgs- von den Fehlermeldungen zu trennen und mit dem jeweils richtigen Status versehen an Nagios weiter zu reichen. Es orientiert

sich dabei an der OID: Nur die OID 1302.3.1.2.8.0.3 (Auftrag erfolgreich abgeschlossen) wird als OK bewertet, alle anderen Traps sind für dieses Script Fehlermeldungen. Das erspart viel Aufwand, der nötig wäre um alle OIDs von BE einzupflegen, setzt aber einen wie oben beschrieben konfigurierten BE-Client voraus.

Beim Weiterreichen bedient es sich übrigens eines weiteren Scripts, `submit_check_result`, doch dazu später.

Die von BE via Trap gelieferte Information ist etwas zu technisch überladen und kryptisch um als schöne, sprechende Fehlermeldung im Nagios aufzuscheinen; daher kürzt und bereinigt das Skript diese, bevor es sie als $**s**be1 (das ***s*** steht dabei für *short*) weiterreicht.

```
#!/bin/sh
# handle-BE-trap
# Author: Ingo Lantschner
# Licence: GPL - http://www.fsf.org/licenses/gpl.txt

# This script receives SNMP-traps from snmptrapd as defined in
# snmptrapd.conf and sends them to submit_check_result for
# further processing in Nagios.
# The logic of this programm assumes, that all error-traps in
# BE-software are enabled plus the traps for successfull jobs.

PATH=/bin:/usr/bin

read host
read ip
read uptime
read trapid
read be1
read be2
read be3
read be4


## shortening the output
sbe1=$(echo $be1    | cut -d \" -f2)
sbe2=$(echo $be2    | cut -d \" -f2)
sbe3=$(echo $be3    | cut -d \" -f2)
sbe4=$(echo $be4    | cut -d \" -f2)

## check for success-Trap
echo $trapid | grep 1302.3.1.2.8.0.3 > /dev/null
if [ $? -eq 0 ]; then
    state=0
    /usr/local/nagios/libexec/eventhandlers/submit_check_result \
        $host "Bandsicherung" $state "$sbe1, Auftrag: $sbe3"
    exit 0
fi

# if the trap-oid is non-success continue here ...
state=2
/usr/local/nagios/libexec/eventhandlers/submit_check_result \
    $host "Bandsicherungsfehler" $state "$sbe1, $sbe2, $sbe3, $sbe4"
exit 0
```

Dieses Script übergibt letztendlich vier Parameter:

- Hostname
- Name des Servicechecks
- Status (0=OK, 1=Warning, 2=Critical, 3=Unknown)

- Meldungstext (bestehend aus den gekürzten Informationsfeldern des SNMP-Trap-Paketes)

Da bei der erfolgreichen Abarbeitung eines Auftrages nicht alle Trapinfos sinnvoll sind, werden bei einer Erfolgsmeldung nur $sbe1 und $sbe3 übergeben.

Die farbliche Markierung im obigen Script ist kein Zufall - deren Sinn wird in einem späteren Abschnitt noch klar.

5.3 Übergabe an Nagios

Für die Übergabe an Nagios, setzen wir das Script `submit_check_result` ein. Dieses ist sehr einfach gehalten und macht nichts anderes als die vier Befehlszeilenparameter in die von Nagios verstandene Form umzuwandeln und dann an das Nagios-Command-File anzuhängen. Dieses wird von Nagios in regelmäßigen Abständen abgefragt und die darin enthaltenen Befehle werden abgearbeitet. Diesem Prozess kann man im Allgemeinen am besten zusehen, wenn man ins syslog schaut.

```
#!/bin/sh
# submit_check_result

echocmd="/bin/echo"
CommandFile="/usr/local/nagios/var/rw/nagios.cmd"
datetime=`date +%s`
cmdline="[$datetime] PROCESS_SERVICE_CHECK_RESULT;$1;$2;$3;$4"
`$echocmd $cmdline >> $CommandFile`
```

5.4 Servicedefinition

Damit Nagios mit dem ihm übergebenen Befehl etwas anfangen kann, müssen entsprechende Servicedefinition, in unserem Beispiel *Bandsicherung* und *Bandsicherungsfehler* vorhanden sein. Fehlen sie, so wird das in das Nagios-Commandfile geschriebene Kommando ohne weiteren Hinweis verworfen. Hier kann man man sich schon mal vertippen und tut gut daran im Falle von nicht aktualisierten Checks nachzukontrollieren.

Folgende Service-Konfig sollte am Nagiosserver vorhanden sein:

```
## traps.cfg


## Templates
##

define service {
use generic-service
register        0
service_description     generic-service-traps
name    generic-service-traps
flap_detection_enabled  0
passive_checks_enabled  1
active_checks_enabled   0
check_command   service-is-stale
max_check_attempts      1
servicegroups   BE-Failure-Trap
check_freshness 0
notifications_enabled   1
}

## Servicechecks
##

define service {
hostgroup_name backup_server
```

```
service_description     Bandsicherung
use generic-service-traps
is_volatile     0
check_freshness 1
# 8 Tage = 691200 Sekunden
freshness_threshold     691200
check_command   service-is-stale!8d
}

define service {
hostgroup_name backup_server
service_description     Bandsicherungsfehler
use generic-service-traps
is_volatile     1
check_command   service-is-stale!noErr
}
```

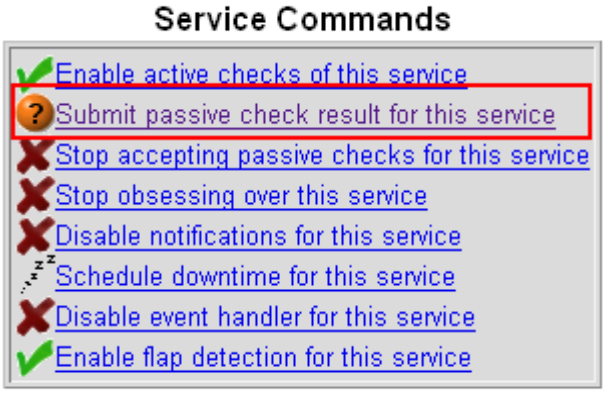
Diese Servicecheckdefinition verwendet Templates, die im Template generic-service-traps definierten Werte, gelten auch für die sich darauf beziehenden Servicechecks Bandsicherung und Bandsicherungsfehler. Natürlich kann man jeden Servicecheck auch einzeln definieren, dann sind die Zeilen des Templates ggf. nach unten zu den Servicecheckdefinitionen zu kopieren. Zur Funktionsweise von Templates bitte in der Doku von Nagios nachlesen.

Die Namensgleichheit beim Parameter *service_description* mit dem zweiten vom Script `handle-BE-trap` übergebenen Parameter ist weder zufällig noch verzichtbar - diese müssen exakt übereinstimmen. Die farbliche Markierung soll dies unterstreichen.

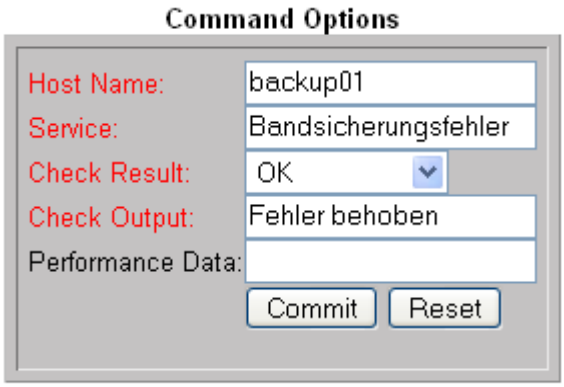
Notifications: Diese sind für Bandsicherung nicht aktiviert, da es meiner Meinung nach keinen Sinn ergibt, dass ich per Email benachrichtigt werde, wenn ein Job erfolgreich gelaufen ist. Was ich aber sofort wissen will, sind Fehler, die das Backupprogramm meldet. Daher ist bei *Bandsicherungsfehler* **notifications** aktiviert. Natürlich will ich auch über jede neu eintreffende Fehlermeldung informiert werden - auch dann, wenn dieser einen bereits bestehenden Fehler überschreibt. Dabei kommt es jedoch zu keinem State-Wechsel (Non-OK bleibt Non-OK), weshalb Nagios keine Fehlermeldung versendet. (Diese Logik ist normalerweise sinnvoll, Nagios bildet damit die Situation ab, dass ein Fehler noch immer besteht.) Mit dem Parameter ***is_volatile*** kann aber erreicht werden, dass jede Veränderung eines Servicechecks signalisiert wird.

Wichtig ist, dass für den Servicecheck *Bandsicherung* der freshnesscheck aktiviert wird. So erreichen wir, dass nach 8 Tagen ohne neue Erfolgsmeldungen von unseren Backupservern, das Script *`check_command service-is-stale!8d`* aufgerufen wird. Womit wir bei der nächsten Frage wären, was dieses denn nun tut ...

Zuvor noch kurz zum Servicecheck *Bandsicherungsfehler*: Dieser wird von Nagios niemals aktualisiert. Er wartet still und geduldig auf den Tag, an dem am Nagiosserver auf Port 162 (snmptrap) UDP-Pakete aufschlagen die Fehlermeldungen von BE enthalten. Diese werden dann über die oben beschriebenen Mechanismen vom snmptrapd über die eventhandlerscripts an Nagios übergeben, der dann die entsprechende Anzeige auf CRITICAL stellt und die Meldung anzeigt. Diese bleibt bestehen, bis entweder eine neue Fehler-Meldung eintrifft oder via Webinterface der Servicecheck auf OK zurückgesetzt wird.



Link zum folgenden Dialogfeld ...



... mit dem ein *Passive Check* manuell auf OK Retoure gesetzt werden kann.

5.5 Checkcommands

Laut den Servicecheckdefinitionen wird also ein check_command namens service-is-stale aufgerufen. Dieses ist denkbar einfach definiert:

```
define command {
command_name    service-is-stale
command_line    /usr/local/nagios/libexec/staleservice.sh $ARG1$
}
```

Aufgerufen wird also ein Shellscript mit dem sprechenden Namen `staleservice.sh` und ein Parameter wird übergeben. Dieser kann, wie wir den zuvor definierten Servicechecks Bandsicherung und Bandsicherungsfehler entnehmen konnten, die Werte *8d* und *noErr* annehmen. Schauen wir uns also dieses Script an:

```
#!/bin/sh
## staleservice.sh - Command for Passivechecks

case "$1" in
    8d)
        message="Seit letztem Check nicht mehr aktualisiert!"
        state=2
        ;;
    noErr)
        message="Keine Fehler seit letztem Check."
        state=0
        ;;
    *)
        message="Daten veraltet!"
        state=2
        ;;
esac
```

```
case "$state" in
    0) prefix=OK ;;
    1) prefix=WARNING ;;
    2) prefix=CRITICAL ;;
    *) prefix=UNKNOW ;;
esac

/bin/echo "$prefix - $message"
exit $state
```

Ich denke dieses Script ist klar: je nach Parameter werden passende Meldungen ausgegeben und der Status an Nagios gemeldet. Sollten neue Checks die auf SNMP-Traps beruhen hinzukommen, so ist dieses Script ggf. zu ergänzen.

6 Zusammenfassung

Wir sind nun mit der Konfiguration fertig. Ein Reload der Nagioskonfiguration ist nötig. Ab dann werden wir an Hand der beiden Checks eine relativ gute Überwachung unserer Backupservers haben. Ergänzende Checks, wie zum Beispiel ob die Services von BackupExec überhaupt laufen oder ob der SNMP-Agent am Backupserver aktiv ist sind sinnvoll. Die aktuelle Konfiguration signalisiert nämlich erst nach 8 Tagen, wenn ein Backupserver bzw. dessen ureigenste Services den Geist aufgeben - das kann dann schon ein wenig spät sein.

7 Ergänzung: SNMP-Traps von beliebigen Programmen aus senden

Wenn eine Infrastruktur wie zum Beispiel oben beschrieben existiert, die SNMP-Traps auswerten kann, ergeben sich rasch weitere Anwendungsmöglichkeiten. Doch nicht alle Programme und Tools sind SNMP-enabled. So zum Beispiel das Werkzeug Robocopy von Microsoft, das gerne verwendet wird um größere Datenmengen zu kopieren. Typischerweise wird Robocopy eingesetzt, um in einem zweistufigen Backupkonzept die Daten auf die Festplatte eines Backupservers zu kopieren. In einem zweiten Schritt, werden diese Daten dann in aller Ruhe auf ein Band gesichert; das Monitoring dieses zweiten Schrittes, haben wir oben beschrieben)

Auf Windows-Servern kommt uns ein Werkzeug wie TrapGen zu Hilfe, das unabhängig vom SNMP-Dienst des Betriebssystems SNMP-Traps an beliebig, über die Commandline definierbare Ziele sendet.

Die Scripts könnten zum Beispiel so aussehen:

7.1 Installation von TrapGen

TrapGen kommt als Zipdatei und muss nur entpackt werden. Danach finden wir 3 Dateien auf unserem Server vor:

- **`TrapGen.exe`**, das eigentliche Programm
- **`input.txt`**, eine Beispieldatei für die Übergabe von Parametern via Datei
- **`README`**, die Doku

Nach Lektüre der Doku sollte man gerüstet sein, um die nachfolgenden Skripts zu verstehen.

7.2 Batchfile zur Sicherung der Daten

Hier die wichtigste Zeile eins Sicherungsjobs. Diese ist natürlich individuell anzupassen, um

beispielsweise die Statusdatei rückzusetzen.

```
Robocopy.exe \\FILESERVER.example.org\d$ D:\Backup\Montag\File-Server\ *.* /E
/ZB /COPYALL /V /NP /LOG:Logfile.txt /XF Pagefile.sys
   echo #%errorlevel%# >> StatusRoboJob.txt
```

Diese Befehlszeile sichert die komplette Partition D: auf den lokalen Server und schreibt den Errorlevel-Code von Robocopy in eine Statusdatei namens `StatusRoboJob.txt`.

Diese sieht, wenn drei dieser o.g. Zeilen abgearbeitet wurden, dann zum Beispiel so aus:

```
#2#
#0#
#14#
```

7.3 Batchfile zur Auswertung der Errorlevel von Robocopy

Die folgende Datei wertet die oben erwähnte Statusdatei aus und sendet den Trap. Trapdestination und Communitystring werden über die Datei `input.txt` gesetzt.

```
:: Batchfile, evaluiert StatusRoboJob.txt und sendet den jeweils hoechsten
:: Fehlercode für die Nagios-Auswertung per SNMP Trap.
:: erstellt von: Ingo Lantschner am 29. 11. 2006
:: ---------------------------------------------------------------
:: Kurzbeschreibung:
::
:: Nagios liest die Datei Wochentag-STATUS.txt aus
:: Dieses Script durchsucht die von den Robojobs in der Datei
:: StatusRoboJob.txt gesammelten Exitcodes und sendet den jeweils hoechsten
:: Fehlercode als SNMP Trap.
:: Benoetigt Trapgen - Trapdestination und Communitystring siehe input.txt
::
::      Datei   SNMP    Bedeutung
::      0       OK      Fehlerfrei
::      1       WARNx   geringe/moegl. Fehler
::      2       CRITx   Schwerer Fehler
:: ---------------------------------------------------------------

:: Definition Pfade Statusdatei und Trap-Tool
set StatusRJob=C:\Batch\StatusRoboJob.txt
set TGPATH=C:\Batch\trapgen

:: Ruecksetzen des Status
set STATUS=CRIT

:: wenn Dateien kopiert werden (= 1), auch extra-Files(= 3) dann ist das gut
findstr #1# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=OK
findstr #3# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=OK


:: 4 bis 7 = mismatched Files - sollte man sich ansehen
findstr #4# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=WARN
findstr #5# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=WARN
findstr #6# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=WARN
findstr #7# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=WARN

:: 0 und 2 = keine Datein wurden kopiert, das ist seltsam
findstr #0# C:\Batch\StatusRoboJob.txt
```

```
   if %errorlevel% == 0 set STATUS=WARN
findstr #2# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=WARN

:: Fatale Fehlercodes sind 8 bis 16
findstr #8# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #9# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #10# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #11# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #12# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #13# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #14# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #15# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT
findstr #16# C:\Batch\StatusRoboJob.txt
   if %errorlevel% == 0 set STATUS=CRIT

:: Erzeugen der Nachricht fuer Trap
if %STATUS% == OK set MSG="Robocopy: Kopie erfolgreich"
if %STATUS% == WARN set MSG="Robocopy: mismatched Files entdeckt oder keine
Daten kopiert"
if %STATUS% == CRIT set MSG="Robocopy mit schwerwiegendem Fehler beendet."

:: SNMP-Trap senden
%TGPATH%\Trapgen -f %TGPATH%\input.txt -v 1.3.6.1.4.1.27440.123.1.1.0.1 STRING
"%STATUS%" -v 1.3.6.1.4.1.27440.123.1.1.0.2 STRING %MSG%

:ende
endlocal
```

Hinweis: Die hier verwendete **OID** ist bitte anzupassen. **Sie darf nicht in einem produktiven oder mit dem Internet verbundenen Umfeld verwendet werden**, da es sich um die dem Autor zugewiesene OID handelt und Kollisionen nicht auszuschliessen sind. Bitte ggf. bei der IANA eine eigene OID beantragen. Der Prozess ist einfach und kostenlos.

7.4 Scripts am Nagios-Server

Auf Nagiosseite wird zunächst von einem traphandle in `/etc/snmp/snmptrapd.conf` die für diese Jobs definiert OID an einen Eventhandler übergeben. Die Zeile lautet:

```
traphandle 1.3.6.1.4.1.27440.123  /usr/local/nagios/libexec/eventhandlers/handle-CP-trap
```

Das Script `handle-CP-trap` wiederum filtert und übergibt and as bereits bekannte `submit_check_result`:

```
#!/bin/sh
# handle-CP-trap
# Author: Ingo Lantschner
# Licence: GPL - http://www.fsf.org/licenses/gpl.txt

# This script receives SNMP-traps from snmptrapd as defined in
# snmptrapd.conf and sends them to submit_check_result for
# further processing in Nagios.
```

```
PATH=/bin:/usr/bin

read host
read ip
read uptime
read trapid
read cp1
read cp2

## shortening the output
scp1=$(echo $cp1    | cut -d \" -f2 | cut -c 1-4)
scp2=$(echo $cp2    | cut -d \" -f2)

case $scp1 in
    OK)         state=0 && pre=OK ;;
    WARN)       state=1 && pre=WARNING ;;
    CRIT)       state=2 && pre=CRITICAL ;;
    *)          state=3 && pre=UNKNOWN ;;
esac
/usr/local/nagios/libexec/eventhandlers/submit_check_result \
    $host "Kopierauftrag" $state "$pre - $scp2"
exit 0
```

Am Nagioshost ist nun noch ein Passive-Service-Check einzurichten:

```
define service {
hostgroup_name backup_server
service_description     Kopierauftrag
use generic-service-traps
check_freshness 1
# 8 Tage = 691200 Sekunden
freshness_threshold     691200
check_command   service-is-stale!8d
}
```

Das war's, wir bekommen nun bei jedem erfolgreichen oder nicht erfolgreichen Kopiervorgang am Backupserver einen Trap und somit eine aktualisiert Anzeige in unserem Nagios. Nach 8 Tagen ohne Nachricht, wird das Script `service-is-stale` aufgerufen und die Anzeige im Nagios mit einer entsprechenden Warnung versehen.

8 Literatur

Dieses HowTo setzt einige Grundkenntnisse voraus, folgend meine Empfehlungen:

- Nagios: Die mit Nagios im HTML-Format mitgelieferte Doku
- SNMP:
  - http://de.wikipedia.org/wiki/SNMP
  - http://www.linux-magazin.de/Artikel/ausgabe/2002/09/snmp/snmp.html
  - man-pages von snmptrapd und snmptrapd.conf