TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Elektroniikan suuntautumisvaihtoehto

Mikko Koskinen

LAITTEISTOTIETOJEN NOUTO Q1-RAJAPINTAA TUKEVILTA VERKKOELEMENTEILTÄ

Insinöörityö, joka on jätetty opinnäytteenä tarkastettavaksi insinöörin tutkintoa varten Tampereella 28.5.2007.

Työn valvoja: Yliopettaja Mauri Inha
Työn ohjaaja: R&D Manager Reijo Ropponen

Tekijä: Mikko Koskinen
Työn nimi: Laitteistotietojen nouto Q1-rajapintaa tukevilta verkkoelementeiltä
Päivämäärä: 28.5.2007
Sivumäärä: 22 sivua ja 26 liitesivua
Hakusanat: Q1, Laitteistonhallinta, NE3S, NetAct
Koulutusohjelma: Tietotekniikka
Suuntautumisvaihtoehto: Elektroniikka

Tyn valvoja: Yliopettaja Mauri Inha

Työn ohjaaja: R&D Manager Reijo Ropponen

Hardware Management on NetAct-verkonhallintajärjestelmän osa, jonka vastuulla on hankkia, ylläpitää ja esittää verkkolaitteiden laitteistotietoja. E3S_HW on Hardware Managementin osajärjestelmä, jonka vastuulla on hankkia laitteistotiedot NE3S-rajapintaa tukevilta verkkolaitteilta. Laitteistotiedot ovat xml-muotoisia tiedostoja ja ne tallennetaan Hardware Managementin laitteistorekisteriin. Q1-agentti on laite, joka osaa hakea laitteistotiedot Q1-verkkoelementeiltä. Q1-agentti tarjoaa NE3S-rajapinnan NetActin ja siten myös E3S_HW-osajärjestelmän käyttöön. NE3S-rajapinta tukee SNMP-protokollaa, jonka avulla E3S_HW-osajärjestelmä ohjaa Q1-agenttia hakemaan uusimmat laitteistotiedot verkkoelementeiltä ja tämän jälkeen hakee laitteistotiedot Q1-agentilta ftp:llä.

Ratkaisua, jossa E3S_HW-osajärjestelmän tuen piiriin lisätään jokin uusi verkkoelementti, kutsutaan adaptaatioksi. Adaptaation teko vaati tässä tapauksessa muutamien konfiguraatiotedostojen tekemistä. Valmis adaptaatio testattiin ja testeissä löydetyt viat korjattiin ja korjaukset testattiin toimiviksi.

E3S_HW-osajärjestelmään tehtyjen korjausten ansiosta laitteistotiedot sisältävä tiedosto saatiin ladattua Q1-verkkoelementeiltä onnistuneesti. Tehdyt korjaukset kuitenkin saattavat haitata myöhemmin tehtävien adaptaatioiden toimintaa ja samoihin ongelmakohtiin on hyvä kiinnittää huomiota tulevaisuudessakin.

Author:	Mikko Koskinen
Title:	Hardware Information Upload for network elements supporting Q1-interface
Date:	28.5.2007
Number of pages:	22 pages and 26 appendix pages
Key words:	Q1, Hardware Management, NetAct, NE3S
Program:	Computer Systems Engineering
Specialisation:	Electronics
Supervisor:	Senior Lecturer Mauri Inha
Instructor:	R&D Manager Reijo Ropponen

NetAct Hardware Management is a system that provides hardware management functionality for large range of network elements. Hardware Management functionality is to fetch, store and present hardware information of network elements. A part of the Hardware Management is E3S_HW subsystem that is capable of fetching hardware information files using NE3S-interface. Hardware Information files are in xml-format and those are stored in hardware repository. Q1-agent is a device that is able to fetch hardware information files from network elements using Q1-interface. Q1-agent also provides NE3S northbound interface. E3S_HW is able to utilize this interface in order to obtain the hardware information files of network elements under Q1-agent. E3S_HW uses SNMP-protocol to command Q1-agent and ftp to fetch the xml-files.

A solution that adds a new network element support to a subsystem is called an adaptation. In order to make an adaptation for Q1-agent and Q1-nodes, few configuration files were provided for E3S_HW subsystem and GEN_HW subsystem. After the adaptation was done, it was tested. During the testing phase, some faults were found and corrected. Correction were also tested and proved to be working.

Thanks to the corrections made, hardware information upload was working from Q1-agent. These corrections, however, brought up some concerns for the future adaptations. Same issues may rise again when another network element support is added to E3S_HW subsystem.

ALKUSANAT

Tutkintotyö toteutettiin projektina Nokia Oyj:lle. Työ on toteutettu vastaamaan Nokia Oyj:n tarpeita.

Tämän työn valvojana toimi yliopettaja Mauri Inha.

Tampereella 28.5.2007

_____

Mikko Koskinen

SISÄLLYSLUETTELO

TIIVISTELMÄ i
ABSTRACT ii
ALKUSANAT iii
SISÄLLYSLUETTELO iv
KÄYTETYT MERKINNÄT JA TERMIT v
1 JOHDANTO 1
2 SNMP-RAJAPINTA 2
2.1 Laitteistotietojen latausoperaation käynnistys verkkoelementillä 2
2.2 Laitteistotietojen latausoperaation seuranta 3
2.3 Laitteistotietojen latausoperaation onnistumisen tarkistus 4
2.4 Laitteistotietojen nouto 4
3 HARDWARE MANAGEMENT 5
3.1 Laitteistotietojen latauksen käynnistys 6
3.2 Laitteistotietojen latauspyyntöjen päivystäjä 7
3.3 Laitteistotietojen latausoperaatioiden valvonta 8
3.4 Laitteistotietojen varastointi 9
4 ADAPTAATION TEKO 9
4.1 Ne3sHWUpload-palvelun verkkoelementeittäinen konfigurointi 9
4.2 Ghwuldmx-prosessin tarvitsema konfiguraatiotiedosto 11
4.3 Ghwucomx-prosessin tarvitsema konfiguraatiotiedosto 11
4.4 ”Hardware Information Upload … ”-painike pääkäyttöliittymään 12
5 LAITTEISTOTIETOJEN LATAUKSEN TESTAUS 13
5.1 Testauksen valmistelut 13
5.2 Väärin asetettu noiHWUploadBaseObject 14
5.2.1 Testaus 14
5.2.2 Testin tulos 15
5.2.3 Vianetsintä ja korjaus 15
5.3 Väärä järjestys MIB-objektien asetuksessa 15
5.3.1 Testaus 15
5.3.2 Testin tulos 15
5.3.3 Vianetsintä ja korjaus 16
5.4 Ongelmia Q1-agentin antamassa hakemistopolussa 16
5.4.1 Testaus 16
5.4.2 Testin tulos 17
5.4.3 Vianetsintä ja korjaus 17
5.5 Ongelmia tiedoston nimeämiskäytännössä 18
5.5.1 Testaus 18
5.5.2 Testin tulos 19
5.5.3 Vianetsintä ja korjaus 19
6 YHTEENVETO 21
LÄHDELUETTELO
LIITELUETTELO

KÄYTETYT MERKINNÄT JA TERMIT

NetAct	Verkonhallintajärjestelmä.
Cron	Unix-komento, jolla voidaan ajastaa tehtäviä.
FTP	File Transfer Protocol. TCP/IP-protokollapinon protokolla, joka soveltuu tiedostojen siirtoon.
Q1	Sovituslaitteiden ja sovitustoimintoja tarvitsevien laitteiden välillä oleva Q-rajapinta.
SNMP	Simple Network Management Protocol. Verkonhallintaprotokolla.
NE3S	Nokia enhanced SNMP solution suite. Nokian patentoima rajapinta, joka tukee SNMP/XML-, SOAP/HTTP- ja FTP/XML-kuljetusprotokollia.
Adaptaatio	Toiminto, joka muuttaa asiakaskerroksen informaation esitystavaltaan palvelinkerroksella tapahtuvan tiedonsiirron edellyttämään muotoon tai palauttaa näin muutetun informaation alkuperäiseen tilaansa.
CORBA	Common Object Request Broker Architecture. OMG:n (Object Management Group) määritys, joka kuvaa toimittajariippumatonta tapaa toteuttaa hajautettuja oliopalveluita työasema-palvelin-ympäristössä.
DynCo	Ohjelma, joka luo tai muokkaa konfiguraatiotiedostoja useiden eri ohjelmien vaatimusten perusteella. DynCon avulla konfiguraatiotiedostoja voidaan päivittää, kun joku osaohjelma asennetaan tai poistetaan.
GID	Global ID. Numeerinen tunniste joka on jokaisella NetAct-tietokantaobjektilla.
DN	Distinguished name. Tunniste, joka on jokaisella NetAct-tietokantaobjektilla.
HP-UX	Hewlett-Packard Unix.
NetAct-objekti	Tarkoittaa tietokannassa olevaa riviä, joka sisältää NetActin tarvitsemat tiedot kyseisestä objektista. Kaikki NetActin tuntemat verkkolaitteet nähdään objekteina. Objekteilla on yleensä hierarkkinen rakenne ja objekteilla on keskenään lapsi-vanhempi-suhteita. Esimerkiksi tukiasemaohjain on loogisesti tukiaseman vanhempi.

MIB	Management Information Base eli hallintatietokanta. Hallintatietokanta on avoimeen järjestelmään sisältyvä joukko hallittavia objekteja. SNMP:ssä MIB on rajapintamäärittely.
E3S_HW	NetActin osajärjestelmä, joka on tehty yleiseksi toteutukseksi laitteistotietojen noutoa varten sellaisilta verkkolaitteilta, jotka tarjoavat SNMP-rajapinnan.
XML	Extensible Markup Language.
MIB-objekti	Hallintatietokannan objekti.

1 JOHDANTO

Nokia NetAct-verkonhallintajärjestelmä on kattava paketti erilaisia työkaluja, jotka auttavat ylläpitämään, pääosin Nokian laitteistoon perustuvia, toisen ja kolmannen sukupolven radioverkkoja. Yksi näistä työkaluista on Hardware Management, jolla verkko-operaattorin on mahdollista kerätä ja ylläpitää tietokantaa laitteistostaan. Tämä tietokanta on nimeltään laitteistorekisteri. Verkkojen ylläpitäjä pääsee käsiksi laitteistorekisterissä oleviin tietoihin Hardware Managementiin kuuluvien ohjelmien avulla. Näitä ohjelmia ovat Remote Inventory ja Hardware Browser.

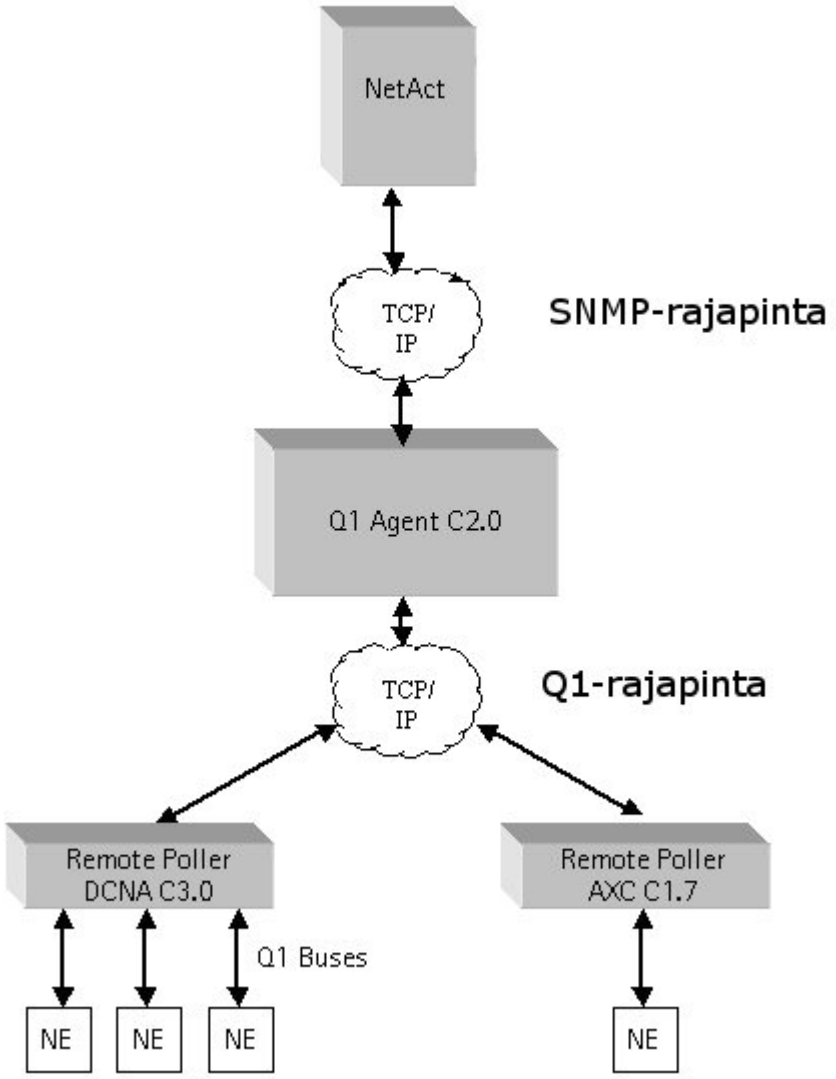
Laitteistorekisteriin tallennettava tieto saadaan laitteelta XML-tiedostona. Tiedostojen siirto laitteelta on toteutettu laitteen tukemien rajapintojen sallimissa rajoissa. Hardware Management tukee useita eri laitteita ja rajapintoja, joten tiedoston siirtoon käytettäviä toteutuksia on myös useita.

Tämän työn tarkoitus on tehdä toteutus, jossa kohdelaitteita ovat 3G-radioverkossa, esimerkiksi tukiasemien väliseen kommunikaatioon, käytettävät tiedonsiirtolaitteet. Yhteistä näillä laitteilla on Q1-rajapinta, joten niitä voidaan kutsua yhteisellä nimityksellä Q1-verkkoelementit. Tavoitteena on, että Q1-verkkoelementiltä saadaan ladattua sen laitetietoja kuvaava XML-tiedosto Hardware Managementin laitteistorekisteriin. Tämä toiminto on nimeltään Hardware Information Upload eli vapaasti suomennettuna laitteistotietojen nouto.

Laitteistotietojen nouto on mahdollista ajaa graafisesta käyttöliittymästä manuaalisesti yhdelle verkkoelementille kerrallaan tai automaattisesti käyttäen UNIXin cron-toimintoa. Laitteistotietojen nouto on monen ohjelman muodostama tapahtumaketju, jonka toiminta selvitetään tarkemmin myöhemmissä kappaleissa.

Hardware Managementissa ei ole valmista toteutusta, joka osaisi suoraan käyttää Q1-verkkoelementtien Q1-rajapintaa ja siten suorittaa laitteistotietojen noutoa. Q1-verkkoelementtien hallintaa varten on kuitenkin olemassa Windows-pohjainen, NetActiin kuulumaton ja siitä riippumaton ohjelma, Q1-agentti, jolla voidaan noutaa laitteistotiedot kyseisiltä Q1-verkkoelementeiltä. Q1-agentissa on ylemmän tason verkonhallintaratkaisuja, kuten NetActia, varten SNMP-rajapinta. Q1-agentin tarjoamat rajapinnat esitetään kuvassa 1. Vastaavasti, NetActin Hardware Managementiin kuuluu toteutus, E3S_HW, joka on tehty yleisesti kaikille niille verkkoelementeille, jotka tukevat Hardware Managementia SNMP-rajapinnan kautta. /2/

Työn alkuvaiheissa avoimia kysymyksiä oli toki monia. Oletus oli, että periaatteessa NetActista on mahdollista ladata Q1-verkkoelementtien laitteistotiedot käyttäen Q1-agentin tarjoamaa SNMP-rajapintaa. Kukaan ei kuitenkaan tiennyt käytännössä, miten Q1-agentti toimisi ja olisiko tämä oletettu ratkaisumalli oikea. Tässä vaiheessa minun tehtäväkseni muodostui ottaa näistä asioista selvää. Minun tulisi tehdä tarvittavat toimenpiteet, jotta kyseinen ratkaisumalli toimisi NetActissa ja testata toiminnallisuus laboratoriossa. /1/



Kuva 1 Q1-agentin tukemat rajapinnat /3/

2 SNMP-RAJAPINTA

Hardware Managementin käyttöön suunniteltu SNMP-rajapinta ei ole puhtaasti pelkkää SNMP:tä käyttävä ratkaisu. SNMP:tä käytetään kommunikointiin operaatioon osallistuvien osapuolten välillä. SNMP:tä käyttämällä voidaan siis luoda edellytykset tiedoston siirrolle, joka toteutetaan FTP:llä. Hardware Managementin SNMP-rajapintakuvauksen määrittämä laitteistotietojen latausoperaatio selvitetään tässä kappaleessa. Työn edetessä oli erittäin olennaista tuntea SNMP-rajapinnan toiminta, koska vian haku ja ohjelmakoodin korjaus olisi ollut mahdotonta ilman tätä tietoutta. Tässä työssä puhutaan SNMP-rajapinnasta, mutta käytännössä sillä tarkoitetaan NE3S-rajapintaa, joka tukee SNMP-protokollaa.

2.1 Laitteistotietojen latausoperaation käynnistys verkkoelementillä

Verkonhallintaohjelmisto, tässä tapauksessa NetAct, voi käynnistää laitteistotietojen latauksen verkkoelementillä asettamalla tietyt MIB-objektit. MIB-objekti asetetaan käyttämällä SNMP SET -komentoa. Käytännössä tätä ennen pitää tehdä tarkistuskysely SNMP GET -komennolla, jolla selviää, onko verkkoelementillä jo käynnissä olevaa laitteistotietojen lataustapahtumaa. Jos verkkoelementti on asettanut noiHWMOperation-objektin arvoon nolla, se tarkoittaa, että keskeneräisiä operaatioita ei ole ja uusi lataus voidaan käynnistää. Samalla kysytään ftp-palvelimen kansio, josta tiedosto löytyy ja se, onko agentti ftp-palvelin vai ftp-asiakas. Käytännössä Q1-agentti toimii aina ftp-palvelimena.

```
SNMP GET(
        noiHWMOperation{
                none (0)
                upload (1)
                fileGenerationInProgress (2)
        }
        noiHWMUploadDirectory{
                "/FOLDER/FOLDER"
        }
        noiHWMFtpMode{
                ftpClient (0)
                ftpServer (1)}
        }
)
```

Tarkistuksen jälkeen voidaan asettaa seuraavat MIB-objektit, jotta laitteistotietojen lataus käynnistyisi: haluttu tiedoston nimi, kohdeobjektin eli verkkoelementin nimi ja operaation nimi.

```
SNMP SET(
        noiHWUploadFile = 'xxx.xml'
        noiHWUploadBaseObject = "MED-5/NODE-1"
        noiHWMOperation = 'upload'
)
/4/5/6/
```

2.2 Laitteistotietojen latausoperaation seuranta

Kun laitteistotietojen lataustapahtuma on käynnistetty, sitä voidaan seurata tarkistamalla aika ajoin alla olevat MIB-objektit SNMP GET -komennolla. Kun noiHWMOperation on taas nolla, tiedetään operaation valmistuneen. NoiHWMOperationState antaa myös tiedon operaation valmistumisesta ja lisäksi onnistumisesta.

```
SNMP GET(
        noiHWMOperation{
                none (0)
                upload (1)
                fileGenerationInProgress (2)
```

```
                }
                    noiHWMOperationState{
                                idle (0),
                                uploadSucceeded (1),
                                uploadFailed (2)}
                    }
)
/4/5/6/
```

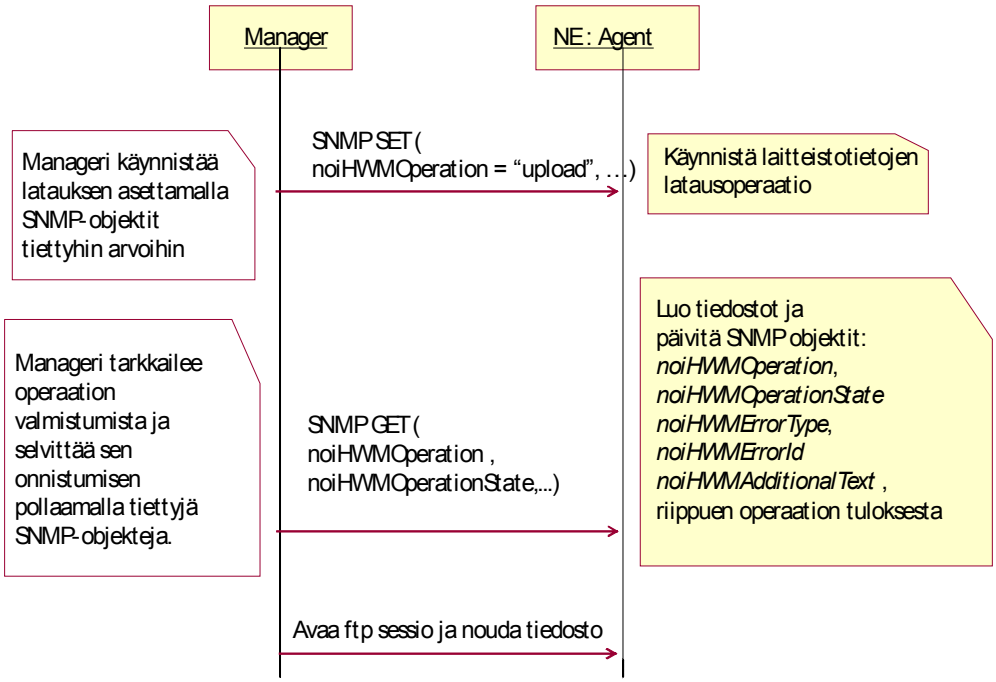
2.3 Laitteistotietojen latausoperaation onnistumisen tarkistus

Lopulta pitää tarkistaa, onko tiedoston generointi onnistunut. Jos noiHWNErrorId on nolla, niin se indikoi onnistunutta operaatiota. NoiHWMErrorType taas kertoo mahdollisen virheen laadusta ja noiHWMAdditionalText sisältää sanallisen kuvauksen virheestä, joka voidaan saattaa käyttäjän luettavaksi ja kirjata lokitiedostoon.

```
                    noiHWMOperationState{
                                idle (0),
                                uploadSucceeded (1),
                                uploadFailed (2)}
                    }
                    noiHWMErrorType{
                                none (0),
                                transient (1),
                                fatal (2)}
                    }
                    noiHWMErrorId{
                                no error (0)
                                errorId (any other)
                    }
                    noiHWMAdditionalText{
                                DisplayString (SIZE (0..255))
                    }
/4/5/6/
```

2.4 Laitteistotietojen nouto

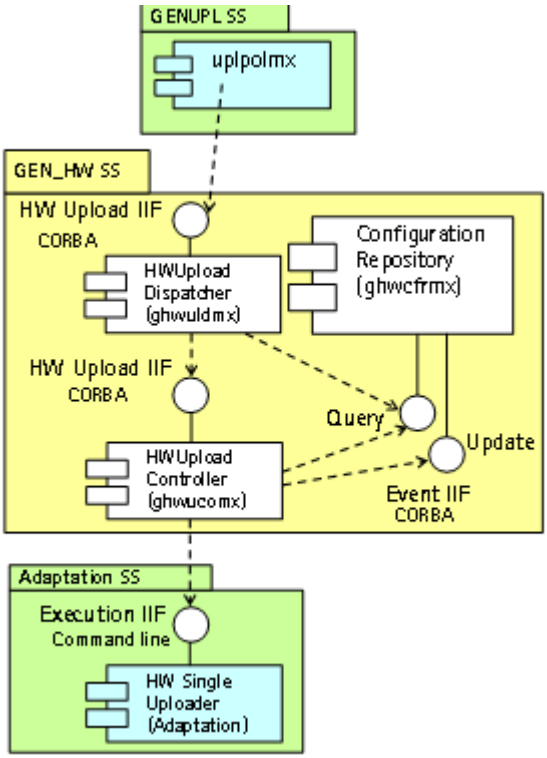
Jos agentti ei ole palauttanut virheitä, voidaan aloittaa tiedoston haku ftp:llä. Agentin ftp-palvelimen tunnukset ja yhteystiedot löytyvät NetActin tietokannasta ja ftp-moodi ja kohdehakemisto selvitettiin ensimmäisellä SNMP GET -haulla. Kun tiedosto on ladattu, sille voidaan suorittaa tarvittaessa jälkiprosessointia ja lopulta se tallennetaan laitteistorekisteriin. Edellä selostettu toiminta on kuvattu pelkistettynä seuraavassa kuvassa 2. /4/5/6/



Kuva 2 Laitteistotietojen lataus /5/

3 HARDWARE MANAGEMENT

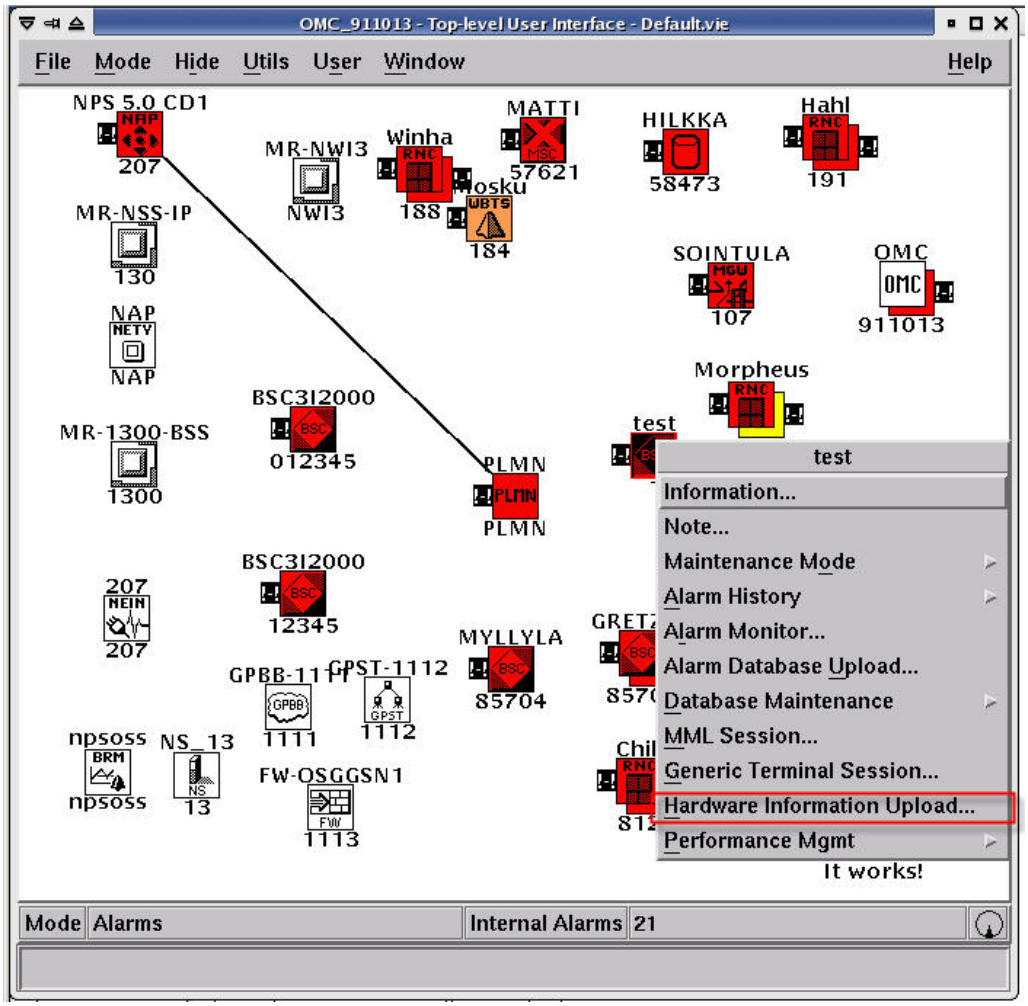
Kuten aiemmin totesin, laitteistotietojen lataus on monen ohjelman muodostama tapahtumaketju. Kuvassa 3 alimmaisena oleva lohko eli adaptaatio vaihtelee elementtityypeittäin. Tässä työssä kyseinen lohko on E3S_HW-osajärjestelmään kuuluva ohjelmalohko ne3sHWUPloader, johon on toteutettu edellisessä kappaleessa kuvatun kaltainen SNMP-rajapintaa käyttävä toiminnallisuus. Adaptaatio ohjelmalohko suorittaa kommunikaation verkkoelementtien kanssa ja lataa laitteistotiedot sisältävän tiedoston. Adaptaatio tarjoaa kuitenkin ainoastaan unix-komentoriviltä ajettavan komennon, johon on osattava antaa useita parametreja. Lisäksi komento voidaan ajaa ainoastaan yhdelle verkkoelementille kerrallaan. Näin ollen adaptaatioiden yläpuolelle tarvitaan muuta toiminnallisuutta, jolla voidaan tarjota käyttäjäystävällinen toiminnallisuus. Hardware Managementiin kuuluvat GEN_HW-osajärjestelmä ja GENUPL-osajärjestelmä tarjoavat tämän toiminnallisuuden ja ovat yhteisiä kaikille Hardware Managementin adaptaatioille. Tässä kappaleessa kuvataan Hardware Managementin yleinen toiminnallisuus.



Kuva 3 Hardware Management -komentoputki /7/

3.1 Laitteistotietojen latauksen käynnistys

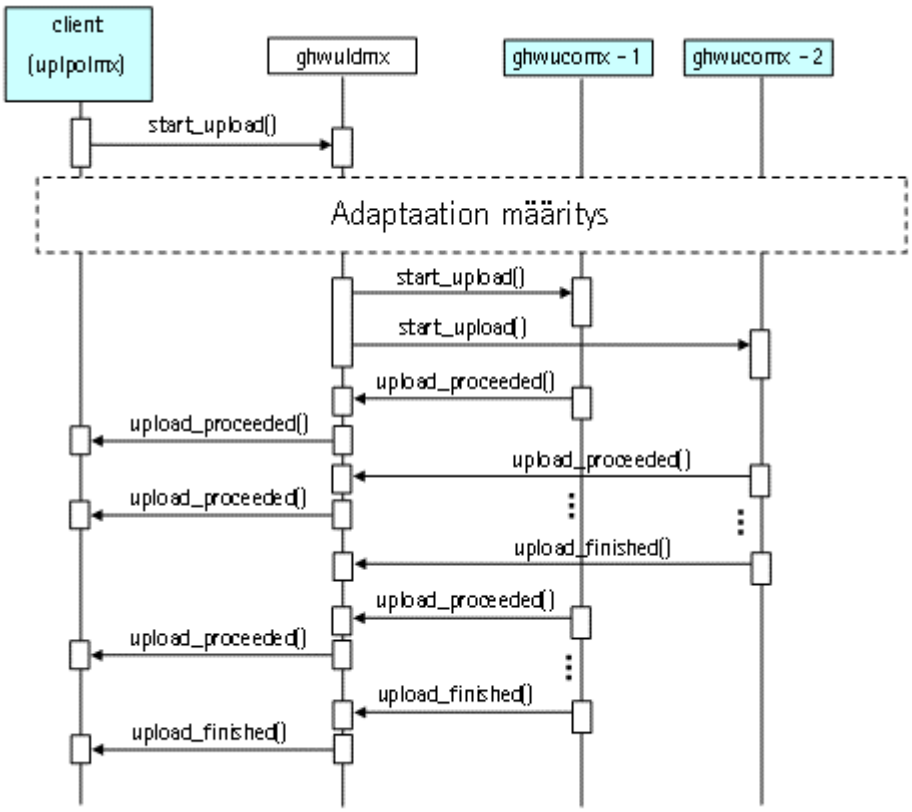
Uplpolmx on ohjelma, jonka tehtävänä on käynnistää laitteistotietojen latausoperaatioita, jotta laitteistorekisteri pysyisi ajanmukaisena. Uplpolmx saa parametrina objektityypin ja hakee topologiatietokannasta kaikkien tätä objektityyppiä olevien verkkoelementtien tunnisteet (GID), jotka se välittää ghwuldmx-prosessille. Uplpolmx käynnistetään usein ajastettuna cronista. Uplpolmx-komennolle voi myös antaa parametrina ainoastaan yhden verkkoelementin tunnisteen. Tätä optiota käytetään NetActin pääkäyttöliittymän (Top Level User Interface) tarjoamassa ”Hardware Information Upload … ”-napissa (katso kuva 4)./8/



Kuva 4 Hardware Information Upload pääkäyttöliittymästä (TLUI)

3.2 Laitteistotietojen latauspyyntöjen päivystäjä

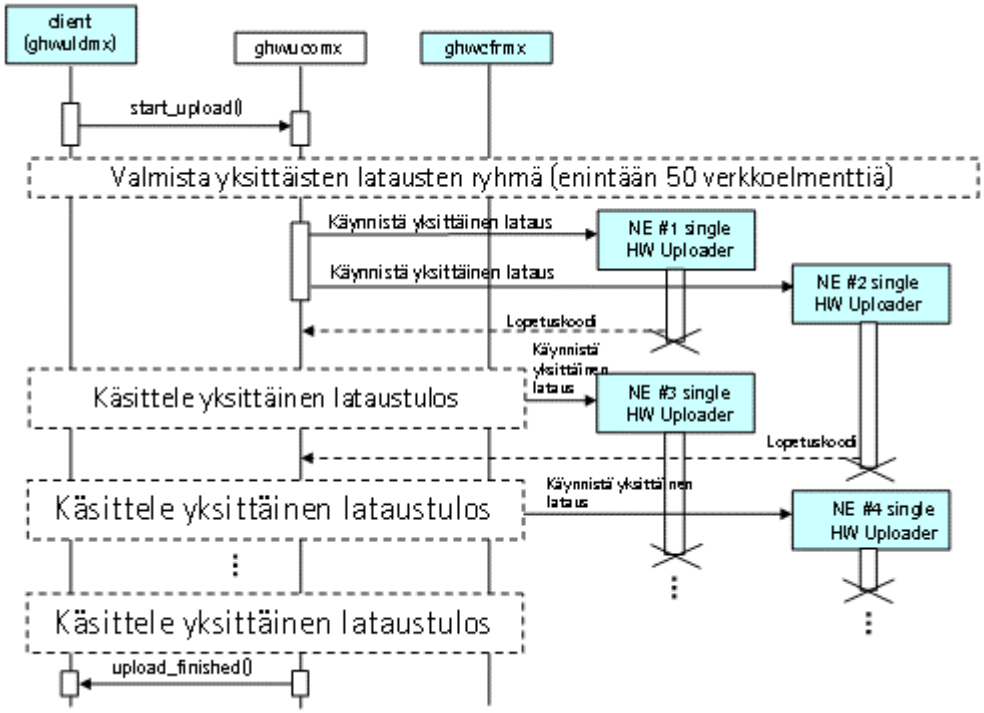
Ghwuldmx on päivystävä taustaprosessi, joka kuuntelee laitteistotietojen latauspyyntöjä. Saadessaan latauspyynnön se osaa valita verkkoelementtityypille tarkoitetun adaptaation. Tätä varten jokaisen adaptaation on tarjottava konfiguraatiotiedosto, jonka perusteella oikea adaptaatioittainen kontrolleri (ghwuco) löytyy. Kuvassa 5 esitetään laitteistotietojen latausoperaatioita ghwuldmx-prosessin näkökulmasta. /7/



Kuva 5 Laitteistotietojen lataus verkosta, ghwuldmx /7/

3.3 Laitteistotietojen latausoperaatioiden valvonta

Ghwucomx on kontrolleri ja siten myös taustaprosessi. Se vastaanottaa laitteistotietojen latauspyynnön ja muuttaa sen sarjaksi yksittäisiä yhtä verkkoelementtiä koskevia laitteistotietojen latauspyyntöjä sekä valvoo niiden suoritusta. Jokaisella adaptaatiolla on oma ghwucomx-prosessinsa. Ghwucomx voi myös tehdä päätöksen tarvitseeko uuttaa laitteistotietoa ladata verkosta vai kelpaako laitteistorekisterissä oleva tieto. Seuraavassa kuvassa 6 on kuvattu laitteistotietojen lataus ghwucomx-prosessin näkökulmasta. /7/



Kuva 6 Laitteistotietojen lataus verkosta, ghwucomx /7/

3.4 Laitteistotietojen varastointi

Kuvassa 3 esitetty ghwcfrmx on taustapalvelin, joka hallinnoi laitteistotietoja ja niiden tilaa. Laitteistotietokanta on toteutettu unixin tiedostojärjestelmässä siten, että yhdessä kansiossa ovat laitteistotiedot sisältävät tiedostot ja toisessa kansiossa ovat tiedostot, jotka kuvaavat edellisten tilaa. Tiedoston tila voi olla joko validi tai invalidi. Tämä tiedon perusteella ghwucom voi päättää, ladataanko verkosta uusi tiedosto. Tiedostojen nimistä selviää, mihin verkkoelementtiin ne viittaavat. Ghwcfrmx on ainoa prosessi, jolla on oikeudet koskea näihin tiedostoihin ja muut tahot joutuvat käyttämään sen tarjoamia palveluja. Laitteistorekisterin sijainti on /var/opt/nokiaoss/uma/uma_hw/vc_gen_hw/repository/. /7/

4 ADAPTAATION TEKO

Edellisissä kappaleissa on kerrottu, miten kaikki toimii teoriassa. Seuraavaksi piti selvittää, mitä pitää tehdä, että kaikki toimisi tämän teorian mukaisesti. Kuvassa 3 alimmaisena esitetty adaptaatio on Q1-verkkoelementtien tapauksessa nimeltään E3S_HW-osajärjestelmään kuuluva e3shwu-ohjelmalohko. E3shwu:n tarjoama CORBA-palvelu on nimeltään ne3sHWUpload. Ne3sHWUpload-palvelu pitää rekisteröidä Hardware Managementin eri prosessien tietoisuuteen. Lisäksi pitää määrittää mille verkkoelementtityypille ja millaisena tämä palvelu on tarjolla. Tätä varten pitää luoda ja editoida useita konfiguraatiotiedostoja.

4.1 Ne3sHWUpload-palvelun verkkoelementeittäinen konfigurointi

E3shwu-ohjelmalohkon tarjoama ne3sHWUpload-palvelu voidaan rekisteröidä mille tahansa NetActin topologiassa määritellylle objektille käyttäen servicesmx.cf-tiedostoa. Servicesmx.cf-tiedosto pitää löytyä ympäristömuuttujan $OMCCONFPATH määrittelemien tiedostopolkujen alta. Tässä työssä valittiin seuraava tiedostopolku:

/d/nmsopt/nokiaoss/uma/irp/conf/adaptations_nokia/

Servicemx.cf-tiedostossa määritellään muuttujia e3shwu-ohjelmalohkoa varten. Tämän konfiguraatiotiedoston on tarkoitus antaa adaptaation tekijälle mahdollisuus valita tiettyjä asioita ja näin ollen se laajentaa mahdollisten tuettavien verkkoelementtien joukkoa. Tässä työssä toteutettiin alla oleva servicesmx.cf-tiedosto.

```
objectClass "Q1A" # Objektin nimi
vendor "NOKIA"
version "C2.0" # Objektin versio. Q1A versio C2.0 tukee hw uploadia.
(services ""
        (ne3sHWUpload "on" # palvelun nimi
                #Pakolliset
                (ftpProtocol "ftp|sftp")
                (IROalgorithm "subTree|mf")
                (serviceKey "2207623066")
                #Vaihtoehtoiset
                (maxUploadDuration  "40")
                (postProcessingScript ”<file_name”)
```

ftpProtocol

Tällä määritellään protokolla, jota käytetään tiedostonsiirtoon. Vaihtoehtoina on ftp tai sftp. Tähän adaptaatioon valittiin käytettäväksi ftp, koska Q1-agentissa on ftp-palvelin. Ftp on myös NetActissä yleisesti käytetty tiedostonsiirtoprotokolla.

IROalgorithm

Tällä määritellään algoritmi, jota e3shwu käyttää löytääkseen juuriobjektin (IRO) tietylle verkkoelementille. Laitteistotietojen lataus käynnistetään kohteena jokin verkkoelementti, joten ohjelman tehtäväksi jää selvittää mitä objektia pitää kutsua. Tässä tapauksessa pitää siis löytää Q1-agentti. Tätä varten algoritmiksi valittiin subTree. Käytännössä tämä tarkoittaa, että ohjelma etsii objektihierarkiasta ensimmäisen kohdeverkkoelementtiä ylempänä olevan objektin, jolle on rekisteröity ne3sHWUpload-palvelu.

serviceKey

Tällä suojataan NetActissa olevien palvelujen luvatonta käyttöä. Asiakkaan pitää maksaa saadakseen käyttöön tiettyjä ominaisuuksia. Useat ohjelmat tulevat mukana perusasennuksessa, mutta niitä ei voida hyödyntää ilman Nokian kehitysympäristössä luotua palveluavainta.

maxUploadDuration

Tällä määritellään aika, jonka e3shwuploader odottaa elementillä tai agentilla tapahtuvaa laitteistotietojen latausprosessia. Jos tämä aika ylittyy, e3shwuploader ilmoittaa virheestä. Tähän adaptaatioon valittiin ajaksi 40 sekuntia, joka testausvaiheessa

osoittautui kuitenkin hieman liian lyhyeksi. Jos tämän jättää määrittelemättä, aika on oletusarvoisesti 30 sekuntia.

postProcessingScript
Tässä määritellään skripti, jolla NetActiin ladattua tiedostoa käsitellään. Jos tämä jätetään määrittelemättä, niin tiedostolle ajetaan oletusskripti. Tässä adaptaatiossa päädyttiin käyttämään oletusskriptiä.

4.2 Ghwuldmx-prosessin tarvitsema konfiguraatiotiedosto

Ghwuldmx-prossessia varten tarvitaan konfiguraatiotiedosto, jossa määritellään CORBA-palvelu, joka on tietylle verkkoelementtityypille tarjolla. Tämän tiedoston avulla ghwuldmx osaa yhdistää tietyn verkkoelementtityypin tiettyyn palveluun. Kyseinen konfiguraatiotiedosto pitää löytyä kansiosta: /m/ossetc/uma/irp/conf/gen_hw/ghwuld/. Alla on esimerkki tiedoston sisällöstä.

```
#e3s_hw.cf konfiguraatiotiedoston sisältö
(serviceRegistration "NE3S HW uploader" # Vapaavalintainen nimi
      (object_class "Q1N" # Tuettavan elementin objektiluokka
            (serviceType "ne3sHWUpload" # CORBA-palvelun nimi
                  (timeoutBetweenCallbacks   "450")
                  (totalTimeoutFactorPerNE   "150")
            )
      )
)
```

Olennaisimpia tietoja ovat object_class ja serviceType. Object_class on verkkoelementtityyppi jolle palvelu rekisteröidään ja serviceType on rekisteröitävän palvelun nimi.

4.3 Ghwucomx-prosessin tarvitsema konfiguraatiotiedosto

Ghwucomx-prosessia varten tarvitaan myös konfiguraatiotiedosto, joka kertoo ghwucomx_e3s_hw-prosessille esimerkiksi e3shwumx-prosessin käynnistyskomennon, joka tulee suorittaa, kun samassa tiedostossa määritettyä CORBA-palvelua kutsutaan. Ghwuco siis muuttaa CORBA-palvelun unix-komentorivikäskyksi e3shwumx.pl. Ghwucomx-prosessin konfiguraatiotiedosto on nimeltään ghwucomx_e3s_hw.cf ja se tulee sijoittaa e3s_hw osajärjestelmän omaan konfiguraatiotiedostot sisältävään kansioon /d/nmsopt/nokiaoss/ uma/umatkt/e3s_hw/conf/default/. Seuraavassa on kuvattu ghwucomx_e3s_hw.cf-tiedoston sisältö.

```
# my config
(processOptions          "ghwucomx.cf"
   (processDescription    "HW uploader for NE3S network elements")
   (corba "on"
      (serverPort       "8001")
      (callTimeout      "60")
   )
   (ghwuco "on"
```

```
        (childTimeout       "60")
        (maxParallelUploads  "1")
        (maxRetries         "1")
        (startCommand       "e3shwumx.pl")
        (uploadDirectory    "/var/opt/nokiaoss/uma/umatkt/e3s_hw/work")
        (serviceType        "ne3sHWUpload")
        (forceUploadAlways   "false")
    )
    (cpftrc "on"
        (TraceEnabled       "Default")
        (TraceDir           "/var/opt/nokiaoss/log")
        (TraceFile          "ghwucomx_e3s_hw.log")
        (TraceLevel         "5")
    )
)
```

Olennaisimpina tietoina voidaan mainita childTimeout, startCommand ja uploadDirectory. ChildTimeout on aika jonka ghwucomx antaa e3shwumx:lle suorittaa laitteistotietojen lataus. StartCommand on komento, jolla e3shwumx-prosessi käynnistetään. UploadDirectory on kansio, johon ghwucomx käskee e3shwumx lataamaan laitteistotiedot sisältävän tiedoston.

4.4 ”Hardware Information Upload … ”-painike pääkäyttöliittymään

Kuvassa 4 esitelty painonappi, josta laitteistotietojen lataus voidaan käynnistää, luodaan myös omalla konfiguraatiotiedostolla. Näitä tiedostoja on kaksi ja ne sijoitetaan e3s_hw-osajärjestelmän kansioon /d/nmsopt/nokiaoss/uma/umatkt/e3s_hw/conf/template/. Tiedostojen nimet ovat guimanmx.other.wco ja guimanmx.sysop.wco. Kummankin tiedoston sisältö on seuraava:

```
#/START
TAG:popupmenu|Q1N_POPUP/
IDENTIFIER:1
#Beginning of the addition of the subsystem e3s_hw
    (button "Hardware Information Upload..." "H"
GENUPL_HW_INFO_UPLOAD)
#End of the addition of the subsystem e3s_hw
#/END
```

Tiedostoja on kaksi, jotta sysop-ryhmän käyttäjille, jotka ovat eräänlaisia NetAct-pääkäyttäjiä, on tarvittaessa mahdollista sallia jotain toiminnallisuuksia joita muilla ryhmillä ei ole. Lisäksi samassa kansiossa olevaan resurssitiedostoon my_resources.txt tulee lisätä tiedot edellä mainituista tiedostoista. Tiedostoon lisättävät rivit ovat:

```
/DYN_CFG SC=uma SC=umatkt SUB=e3s_hw WCO=guimanmx.sysop.wco
/DYN_CFG SC=uma SC=umatkt SUB=e3s_hw WCO=guimanmx.other.wco
```

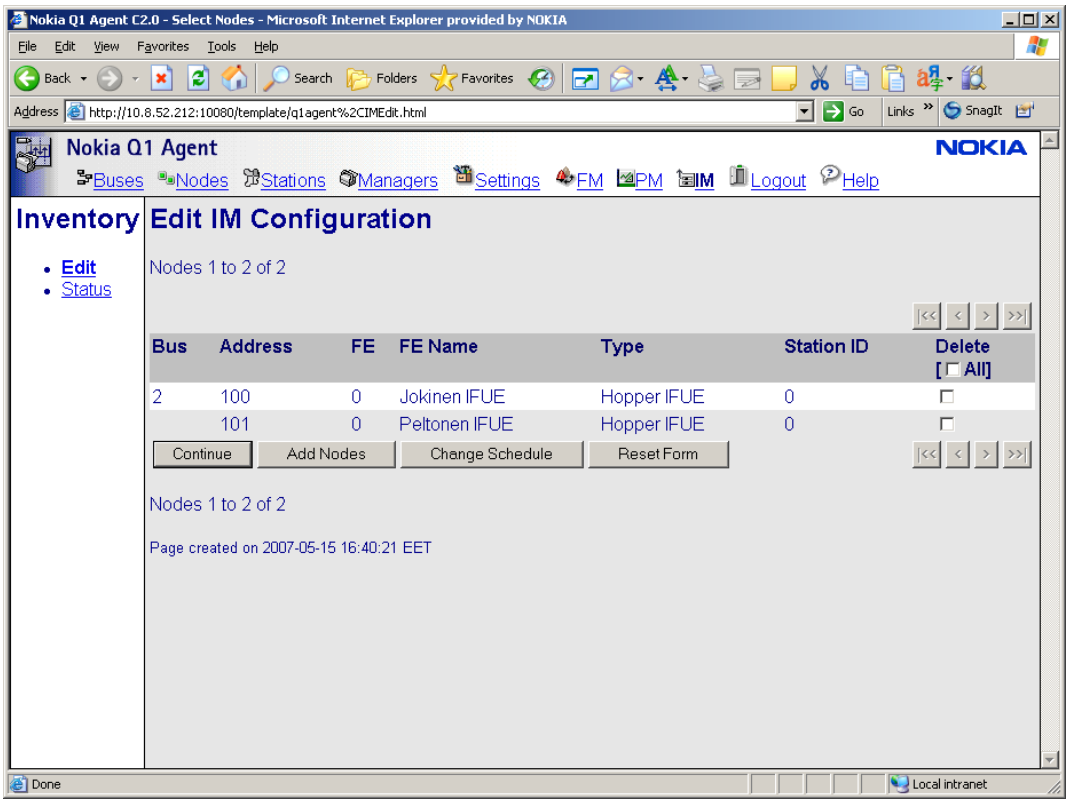
Näiden rivien avulla DynCo tietää luodessaan guimanmx.cf-tiedostoja, että näiden tiedostojen sisällöt pitää ottaa mukaan.

5 LAITTEISTOTIETOJEN LATAUKSEN TESTAUS

5.1 Testauksen valmistelut

Jos kyseessä olisi ollut jokin virallinen testiprojekti, testaus olisi ollut kontrolloidumpaa ja perustunut testitapauksiin. Lisäksi testitulosten dokumentointi olisi tapahtunut siihen tarkoitetulla verkkotyökalulla. Tässä työssä testaus tapahtui vapaamuotoisesti siihen perustuvana, että olin jo tutustunut aihealueeseen ja tiesin tarkkaan miten kyseisen toteutuksen tulisi toimia. Tässä kappaleessa kerrotaan testausta edeltävistä esivalmisteluista.

Aluksi piti valmistella käyttöön sopiva testausympäristö. Käyttötarkoitukseeni soveltui hyvin Sandbox-niminen NetAct-palvelin, joka on suunniteltu testipediksi kehityksen varhaisessa vaiheessa oleville ohjelmille. Sandbox NetAct kostuu ainoastaan yhdestä HP-UX-palvelimesta, kun normaalisti NetAct:iin kuuluu vähintään kaksi palvelinta, yksi Oracle tietokannalle ja yksi NetAct-ohjelmille. Sandboxissa Oracle-tietokanta on samalla palvelimella muiden ohjelmien kanssa. Lisäksi NetActiin piti liittää Q1-agentti ja sen alaisuuteen Q1-verkkoelementtejä. Laboratoriovastaavamme teki pyynnöstäni nämä toimenpiteet ja Sandboxiin saatiin liitettyä Q1-agentti ja Q1-verkkoelementit Jokinen IFUE ja Peltonen IFUE. Alla olevassa kuvassa 7 näkyy Q1-agentin käyttöliittymä ja sen alaisuudessa olevat Q1-verkkoelementit. Q1-agentti asennettiin vanhalle PC:lle, jossa oli käyttöjärjestelmänä Windows 2000 Server. Minun tehtäväni oli konfiguroida NetActiin tarvittavat muutokset aikaisemmissa kappaleissa esitettyjen tietojen mukaisesti.



Kuva 7 Q1-agentin käyttöliittymä

5.2 Väärin asetettu noiHWUploadBaseObject

5.2.1 Testaus

Testauksen ensimmäinen vaihe oli testata, olisiko ne3sHWUpload-palvelu hyväksytty NetActiin. Tämä tapahtui seuraavalla komennolla:

% netact_start.pl uma/umatkt/tktfmi – lstadpmx | grep ne3sHWUpload

Tämän jälkeen kokeilin ajaa komentoriviltä e3shwu-ohjelmalohkon komentoa e3shwumx. Tämä testi osoittaisi adaptaation toimivuuden. Hardware Managementin yleinen osuus ei ole mukana tässä operaatiossa, joten mahdollinen virheiden etsintä olisi helppo rajoittaa e3shwu-ohjelmalohkoon. E3shwumx-prosessi käynnistetään käynnistyskriprillä e3shwumx.pl. Sille annetaan parametreina kohteena olevan verkkoelementin tunniste (GID) ja tiedosto, johon tallennetaan verkosta haettu laitteistotieto. Lisäksi trace- ja debug-optiot on hyvä kytkeä päälle, jotta saadaan mahdollisimman paljon tietoa prosessin kulusta.

% netact_start.pl uma/umatkt/e3s_hw -- e3shwumx.pl -gid 911093000000397383 -file /m/home/omc/koskinen/tmp/Bus-2Node-100.xml -trace –debug

Onnistuessaan tämä testi tuottaisi 911093000000397383-tunnisteen omaavan verkkoelementin laitetiedot sisältävän tiedoston Bus-2Node-100.xml kansioon /m/home/omc/koskinen/tmp/.

5.2.2 Testin tulos

Testin aluksi kokeilemani palvelun rekisteröinti oli onnistunut, mutta itse laitteistotietojen lataus ei toiminut. Koska trace- ja debug-optiot olivat päällä, ohjelma tulosti näytölle riittävästi tietoa prosessin kulusta. Tästä tiedosta kävi ilmi, että ohjelma oli löytänyt oikean Q1-agentin käyttäen subTree-algoritmia. Q1-agenttiin oli saatu yhteys muodostettua ja laitteistotietojen latausoperaatio oli käynnistetty Q1-agentilla. Tämän jälkeen Q1-agentti kuitenkin ilmoitti virheestä. Kappaleessa 2.3 esitellyn SNMP-objektin, noiHWMAdditionalText, sisältö oli seuraava:

”noiHWMAdditionalText: variable noiHWUploadBaseObject is not properly set”

5.2.3 Vianetsintä ja korjaus

Näytölle tulostuvasta tiedosta kävi ilmi, että käynnistäessään laitteistotietojen latausta, kappaleessa 2.1 esitetyllä tavalla, e3shwumx asetti noiHWUploadBaseObject arvon väärin. Kyseinen MIB-objekti pitäisi asettaa kohteena olevan verkkoelementin DN-nimellä, joka on muotoa PLMN-PLMN/Q1A/Q1N. Jostain syystä tätä kohdeobjektin nimeä yritettiin selvittää eräällä algoritmilla, joka ei tässä tapauksessa kuitenkaan toiminut oikein. Ohjelma hukkasi nimen alusta ”PLMN-PLMN/”-kohdan ja siten noiHWUploadBaseObject:n arvoksi tuli ainoastaan Q1A/Q1N. Kun tämä kohta muutettiin ohjelmakoodista siten, että verkkoelementin nimen alku säilytetään, tämä ongelma oli ratkaistu. Korjaus on liitteenä olevassa lähdekoodissa riveillä 536-542.

5.3 Väärä järjestys MIB-objektien asetuksessa

5.3.1 Testaus

Edellä kuvatun korjauksen jälkeen päästiin jälleen testaamaan saman e3shwumx-ohjelman ajoa komentoriviltä samalla komennolla, kuin aiemmin:

```
% netact_start.pl uma/umatkt/e3s_hw -- e3shwumx.pl -gid
911093000000397383 -file /m/home/omc/koskinen/tmp/Bus-2Node-100.xml -
trace –debug
```

5.3.2 Testin tulos

Tällä kertaa e3shwumx-ohjelma pääsi välillä täysin samaan kohtaan kuin edellisessäkin testissä ja ilmoitti virheestä. Välillä taas tästä kohdasta päästiin

eteenpäin. Käytännössä keskimäärin joka toinen yritys pysähtyi virheeseen, jossa MIB-objektin noiHWMAdditionalText, sisältö oli:

”noiHWMAdditionalText : variable noiHWUploadFile is not properly set”

Tämä virheilmoitus tarkoitti, että e3shwumx-prosessin Q1-agentille antama tiedoston nimi oli väärä. Sama tulos saatiin vaikka Q1-agentille yritettiin antaa useita erilaisia tiedoston nimiä. Niillä kerroilla, kun tästä virhekohdasta päästiin ohi, ohjelma pysähtyi kappaleessa 5.4 esiteltävään vikaan.

5.3.3 Vianetsintä ja korjaus

Koska vikailmoitus ei tullut joka kerralla, vianetsintä muodostui hieman ongelmalliseksi ja näytti siltä, että koko ohjelman toiminnassa ei olisi mitään logiikkaa. E3shwumx-ohjelman lähdekoodia tutkimalla kuitenkin selvisi, mistä oli kysymys. Kappaleessa 2.1 esitetty laitteistotietojen latauksen käynnistys oli toteutettu väärässä järjestyksessä. Alkuperäisessä ohjelmakoodissa MIB-objektit asetettiin seuraavassa järjestyksessä:

noiHWMOperation = ‘upload’
noiHWUploadFile = ‘xxx.xml’
noiHWUploadBaseObject = "MED-5/NODE-1"

Kun noiHWMOperation asetettiin ensimmäisenä, Q1-agentti alkoi suorittaa laitteistotietojen latausta. Ensimmäiseksi se huomasi, että noiHWUploadFile on asettamatta, keskeytti operaation ja merkkasi tietyt SNMP-objektit määrättyihin arvoihin merkiksi virheestä. Tässä vaiheessa e3shwumx asetti vielä noiHWUploadFile:n ja noiHWUploadBaseObject:n arvot kohdalleen ennen kuin jatkoi eteenpäin ja huomasi, että operaatio on epäonnistunut. Kun seuraavan kerran e3shwumx asetti noiHWMOperation, edellisellä kerralla asetetut noiHWUploadFile ja noiHWUploadBaseObject olivat valmiiksi asetettu ja Q1-agentti ei ilmoittanutkaan virheestä. Näin ollen vaikutti siltä, että ohjelma olisi toiminut vain silloin tällöin.

Korjauksena MIB-objektien asettamisjärjestystä vaihdettiin siten, että noiHWMOperation asetettiin viimeisenä. Korjaus löytyy liitteenä olevasta ohjelmakoodista riveiltä 1106-1110.

5.4 Ongelmia Q1-agentin antamassa hakemistopolussa

5.4.1 Testaus

Kaksi vikaa oli saatu korjattua ja ohjelmaa testattiin edelleen ajamalla samaa komentoa:

```
% netact_start.pl uma/umatkt/e3s_hw -- e3shwumx.pl -gid
911093000000397383 -file /m/home/omc/koskinen/tmp/Bus-2Node-100.xml -
trace –debug
```

5.4.2 Testin tulos

Tällä kertaa tulostuksesta kävi ilmi, että vaihe jossa Q1-agentin kanssa keskustellaan SNMP-protokollalla, onnistuu joka kerralla. Tämän vaiheen jälkeen e3shwumx-prosessi käynnistää e3shwu_Transfer.pl-ohjelman, jonka tarkoituksena on hakea laitteistotiedot sisältävä xml-tiedosto Q1-agentin ftp-palvelimelta. Tiedostonsiirto ei kuitenkaan onnistunut, vaan e3shwu_Transfer.pl ilmoitti, että ei löydä määrättyä tiedostoa Q1-agentin levyltä.

5.4.3 Vianetsintä ja korjaus

Vianetsintä aloitettiin kokeilemalla e3shwu_Transfer.pl-ohjelman toimivuutta suorittamalla tiedostonhaku komentoriviltä e3shwu_Transfer.pl:n omalla käynnistyskomennolla:

```
/opt/nokiaoss/uma/umatkt/e3s_hw/bin/e3shwu_Transfer.pl
-file Bus-2Node-100.xml
-dir '\Program Files\Nokia\Q1Agent\data\q1\hw-xml'
-user q1pmuser
-passwd xxxxx
-gid 911093000000104983
-protocol ftp
-targetDir /m/home/omc/koskinen/tmp
-ip 10.8.52.212
-trace
-traceFile /var/opt/nokiaoss/tmp/trace/e3shwu_Transfer.trc
```

Edellä olevalla komennolla tiedosto saatiin ladattua onnistuneesti NetActin levylle. Tämän tuloksen myötä voitiinkin epäillä, että e3shwumx-prosessi ei osaa muodostaa e3shwu_Transfer.pl-ohjelman käynnistyskomentoa oikein. Ohjelmakoodia tutkittaessa tämä olettamus todettiin oikeaksi. Ongelmakohtana oli hakemistopolku, jonka Q1-agentti antaa e3shwumx-prosessille sen kysyessä MIB-objektia noiHMWUploadDirectory. Seuraavassa on alkuperäinen käynnistyskomennon muodostava ohjelmakoodi:

```
Q3FCOR::stringBuilder_c command;
command << ftpBinary
<< " -file " << _file
<< " -dir " << _noiHWMUploadDirectory
<< " -user " << user
<< " -passwd \"" << password << "\""
<< " -gid " << iroGid
<< " -protocol " << _noiHWMFtpProtocol
<< " -targetDir " << _targetDir
<< " -ip " << _ipAddress;
```

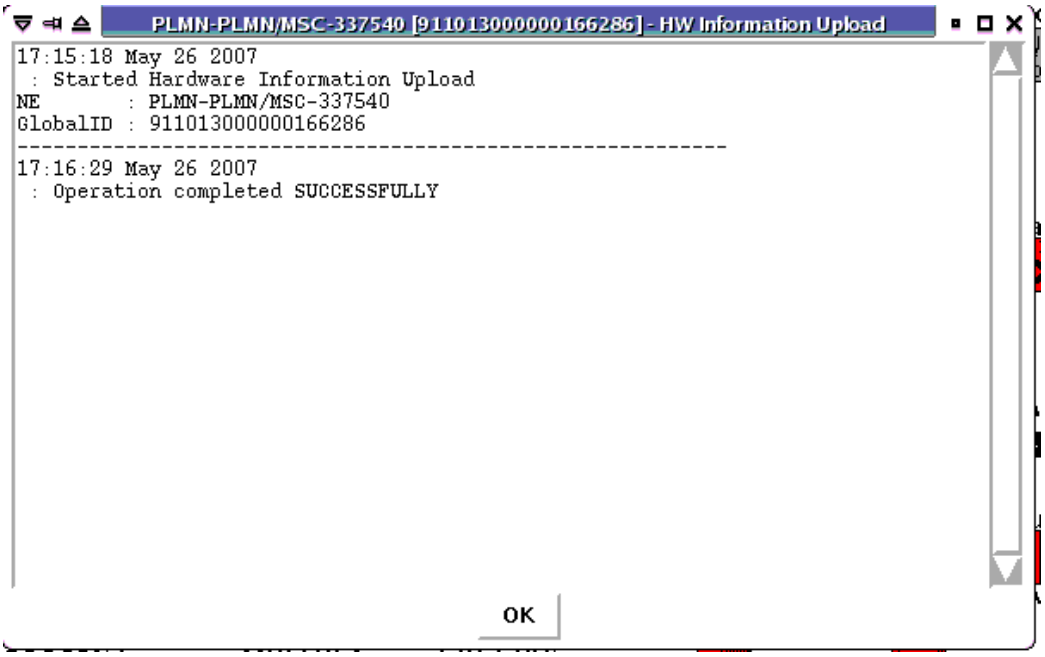
Ongelmakohta oli string-tyyppisen muuttujan _noiHWMUploadDirectory sisällössä. Q1-agentin antama MIB-objekti sisältää Windows-käyttöjärjestelmälle ominaisen hakemistopolun, C:\Program Files\Nokia\Q1Agent\data\q1\hw-xml, joka sisältää Windows-

käyttöjärjestelmälle ominaiseen tapaan asematunnisteen C: ja lisäksi myös välilyönnin. Asematunniste on ongelmallinen siksi, että ftp-palvelimen tiedostojärjestelmässä asematunnistetta ei käytetä. Tämän vuoksi ftp-asiakasohjelma ei onnistu yrittäessä siirtyä kyseiseen kansioon ftp-palvelimella. Itse asiassa tämä on vika Q1-agentissa, jonka tulisikin antaa ftp-palvelimen hakemistopolku \Program Files\Nokia\Q1Agent\data\q1\hw-xml. Välilyönti muodostuu ongelmaksi siinä vaiheessa, kun e3shwumx välittää tiedostopolun e3shwu_Transfer.pl-ohjelmalle unix-komentorivillä ilman heittomerkkejä. Option ”-dir” lukeminen katkeaa ensimmäiseen välilyöntiin ja sen vuoksi e3shwu_Trasfer.pl-ohjelman saama optio on ainoastaan C:\Program. Tiedostonhaku tuosta kansiosta ei onnistu. Tämä korjattiin siten, että e3shwumx-prosessin lähdekoodissa muuttujaan _noiHWMUploadDirectory lisättiin heittomerkit valmiiksi. Näin ollen optio välittyy välilyönteineen e3shwu_Transfer.pl-ohjelmalle. Tämä korjaus löytyy lähdekoodista riviltä 1448. Toinen korjattava asia oli tiedostopolun alussa oleva, Q1-agentin virheellisesti antama, asematunniste C:. Korjaus tehtiin siten, että ohjelma tarkistaa ensin onko alussa C: ja poistaa sen tarvittaessa muuttujan _noiHWMUploadDirectory sisällöstä. Tämä muutos löytyy lähdekoodista riveiltä 1436-1443. Tämän korjauksen jälkeen e3shwumx-ohjelma toimi moitteitta komentoriviltä testissä käytetyllä komennolla.

5.5 Ongelmia tiedoston nimeämiskäytännössä

5.5.1 Testaus

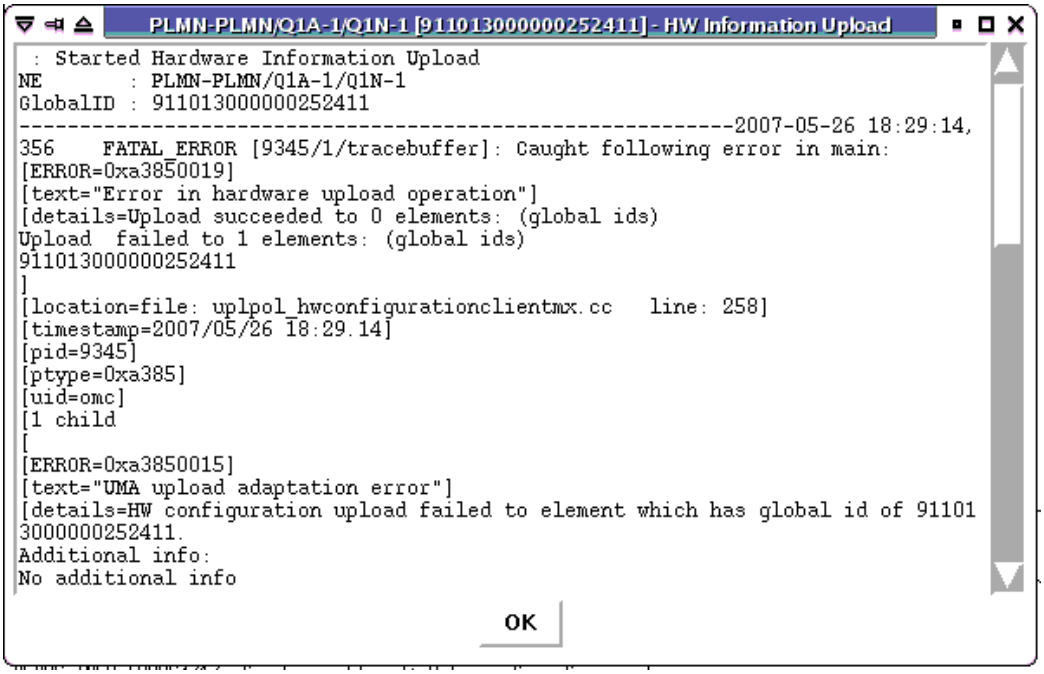
Kun adaptaatio oli saatu toimimaan komentoriviltä toivotulla tavalla, voitiin testauksessa siirtyä kokeilemaan laitteistotietojen latausta tavalla jolla se kuuluu tehdä, eli kuvassa 4 esitetyllä painikkeella. Latauksen ollessa valmis, alla olevan kuvan 8 kaltainen ponnahdusikkuna tulisi näkyä ja laitteistorekisterin kansiosta /var/opt/nokiaoss/uma/uma_hw/vc_gen_hw/repository/upload_data/ tulisi löytyä laitteistotiedot sisältävä tiedosto.



Kuva 8 Onnistunut laitteistotietojen lataus

5.5.2 Testin tulos

Tästäkään testistä ei selvitty ongelmitta. Pääkäyttöliittymässä ponnahdusikkuna ilmoittikin virheestä adaptaatiossa.



Kuva 9 Virhe laitteistotietojen latauksessa

5.5.3 Vianetsintä ja korjaus

Koska pääkäyttöliittymän virheilmoituksesta ei selvinnyt muuta tietoa, kuin että vika olisi adaptaatiossa, oli prosessista saatava lisää tietoa. Aikaisempien testien perusteella adaptaatio oli osoitettu toimivaksi joten epäillä saattoi, että vika olisi

tällä kertaa jossain toisaalla. Lisätiedon saamiseksi gwhucomx_e3shw käynnistettiin T 9 -optiota käyttäen, joka kytkee päälle tarkimman mahdollisen jäljityksen.

netact_start.pl -pkg osscore -- uma/umatkt/e3s_hw ghwucomx_e3s_hw -T 9

Ghwucomx-prosessin jäljitystiedostosta saatiin selvitettyä käynnistyskomento jota ghwucomx_e3s_hw käyttää käynnistäessään e3shwumx.pl-prosessin:

2007-05-26 18:36:44,242 MAJOR_INFO [18686/2/nw3n3t.cpftrc]: SubProcess_c::Spawn(commandline=
e3shwumx.pl
-gid 911013000000252411
-file /var/opt/nokiaoss/uma/umatkt/e3s_hw/work/hwdatAAAa18686
-dn PLMN-PLMN/Q1A-1/Q1N-1
-name "QYN"
-oc Q1N
-version C2.0
-vendor NOKIA
-subclass "Hopper IFUE"
-system 911013
-instance 1)

Kun tähän komentoon lisättiin aiemmissa testeissä käytetyt trace- ja debug-optiot, huomattiin, että Q1-agentti ilmoitti jälleen:

”noiHWMAdditionalText : variable noiHWUploadFile is not properly set”

Kokeilemalla huomattiin, että Q1-agentti hyväksyi ainoastaan tiedoston nimen joka päättyi ’.xml’. Ghwucomx taas käyttää väliaikaisia tiedostonnimiä, joissa ei ole xml-tarkennetta. Tämä ongelma oli siis ratkaistava e3shwumx-ohjelman sisällä. Lähdekoodia muutettiin siten, että alussa komentoriviltä saatu tiedoston nimi tarkistetaan ja lisätään siihen tarvittaessa ’.xml’-pääte. Lopuksi, kun tiedosto on haettu Q1-agentilta, tiedostolle annetaan sen alkuperäinen nimi takaisin. Tällä korjauksella laitteistotietojen lataus saatiin toimimaan myös pääkäyttöliittymästä. Korjaus on esitetty liitteenä olevassa lähdekoodissa riveillä 228-233 ja 1564-1565.

Samalla testillä huomattiin myös toinen vika, joka oli ollut alusta asti GEN_HW-osajärjestelmässä. Ghwucomx ei laittanut subclass-optiota lainausmerkkeihin muodostaessaan e3swumx.pl:n käynnistyskomentoa. Tämä tarkoitti sitä, että e3shwumx.pl ei aluksi käynnistynyt ollenkaan. Minulla ei ollut oikeuksia koskea ghwucomx:n koodiin, joten tästä ongelmasta tehtiin vikaraportti ja ghwucomx-ohjelmaan saatiin myöhemmin korjaus. Testejä pystyttiin jatkamaan tilapäisen korjauksen avulla, joka tehtiin e3shwumx:n käynnistyskriptiin. Sitä muutettiin siten, että subclass-option jälkeen luettiinkin kaksi seuraavaa merkkijonoa yhden sijaan. Näin pystyttiin käsittelemään kaksiosainen subclass-nimi.

Lopulta kaikki viat oli korjattu ja laitteistotietojen lataus toimi moitteettomasti komentoriviltä, pääkäyttöliittymästä ja ajastettuna cron-toimintona.

Testilaboratoriossa oli ainoastaan kaksi Q1-verkkoelementtiä, joten cron-toimintona käynnistyvä laitteistotietojen lataus kaikille Q1N-tyyppiä oleville objekteille ei ollut kovin kattava testi.

6 YHTEENVETO

Tavoitteena oli saada varmistettua, että Q1-agenttia ja NetActin E3S_HW-osajärjestelmää käyttämällä voidaan ladata laitteistotiedot Q1-verkkoelementeiltä ja tallettaa tiedostot Hardware Managementin laitteistotietokantaan. Työn edetessä todettiin, että E3S_HW-osajärjestelmään tarvitaan adaptaatio kyseiselle verkkoelementtityypille. Periaatteessa tämä tarkoitti muutaman konfiguraatiotiedoston tekoa ja editointia. Käytännössä konfiguraatiotiedostojen tarjoaminen ei kuitenkaan riittänyt toimivan ratkaisuun asti. Lopulta kuitenkin testauksella ja korjauksilla saavutettiin toimiva ratkaisu. E3shwuploader.cc lähdekoodiin tehtiin joukko korjauksia, joiden jälkeen toiminta yhdessä Q1-agentin kanssa sujui ongelmitta.

Ongelmana E3S_HW-osajärjestelmässä on sen yleispätevä luonne. Se on kehitetty alun perin mitä tahansa NE3S-rajapinnan tarjoavaa verkkoelementtiä varten ja sen testaukseen on käytetty oikeiden verkkoelementtien sijaan simulaattoria. Käytännön toteutuksissa saattaa kuitenkin olla eroja ja uutta adaptaatiota tehdessä saattaa aina löytyä uusia ongelmia. Tällä kertaa voitiin huoletta muuttaa e3shwumx-ohjelman lähdekoodia, koska kyseessä oli ensimmäinen sille tehty adaptaatio. Seuraavia adaptaatioita tehdessä muutoksen tekijän onkin huomioitava se, että nykyinen toteutus toimii jo Q1-agentin ja Q1-verkkoelementtien kanssa ja sen mukauttaminen johonkin toiseen ympäristöön saman laisilla koodi muutoksilla saattaa vaarantaa tämän toiminnallisuuden.

Ensimmäinen korjaus, joka esiteltiin kappaleessa 5.2, koski verkkoelementin DN-nimen laskennassa tapahtuvaa virhettä, jonka johdosta noiHWUploadBaseObject MIB-objekti asetetään väärin Q1-agentilla. Tässä tapauksessa tehty korjaus oli yksinkertaisesti nimeä laskevan algoritmin ohittaminen ja korvaaminen Q1-agentille sopivaksi. Yleispäteväksi tästä ratkaisusta ei kuitenkaan välttämättä ole. Missään vaiheessa ei selvinnyt miksi tätä algoritmia alun perin käytettiin ja onkin syytä epäillä, että tulevissa adaptaatioissa tässä samassa kohtaa saatta tulla ongelmia.

Toinen korjaus oli selvä vika e3shwumx:n ohjelmassa, mutta myös Q1-agentti olisi voinut toiminallaan vaikuttaa tilanteeseen. Jos Q1-agentin koodissa olisi määritelty laitteistotietojen latauksen käynnistäminen vasta sitten, kun kaikki tarvittavat MIB-objektit on asetettu, tätä korjausta ei olisi tarvittu. Tätä asiaa ei kuitenkaan ole tarkkaan kuvattu NE3S-rajapintamäärittelyssa, joten on ymmärrettävää, että tällaiset virheet ovat mahdollisia. Todennäköisesti verkkoelementtisimulaattori, jolla e3shwumx on aikanaan testattu toimi juuri tällä jälkimmäisellä tavalla. Muutoksen ei kuitenkaan olettaisi haittaavan tulevaisuuden adaptaatioiden toimintaa, vaikkakin jokainen on syytä tutkia tarkkaan myös tämän tapauksen kohdalla.

Kappaleissa 5.4 ja 5.5 esitettyjen vikojen ratkaisut olisi ehdottomasti pitänyt tehdä Q1-agentilla. Q1-agentilla ei kuitenkaan ollut tulossa uusia julkaisuja

lähiaikoina, joten ongelma päätettiin ratkaista paikallisesti. Kappaleessa 5.4 tiedostopolkuongelmaan tehty korjaus toimii kuitenkin siinäkin tapauksessa, että Q1-agentilla korjattaisiin tiedostopolku ftp-palvelimelle ominaisemmaksi ja poistettaisiin asematunniste ja turhat välilyönnit. Lisäksi muiden verkkoelementtien antamat tiedostopolut pitäisi nyt olla paremmin tuettuja riippumatta siitä onko niissä välilyöntejä vai ei. Asematunnisteista kuitenkin tarkistetaan ainoastaan 'C:', joten muut asematunnisteet saattavat aiheuttaa tulevaisuudessa ongelmia. Kappaleen 5.5 tiedosto nimeen liittyvä ongelma oli sisänsä erikoinen, että Q1-agentti ei hyväksynyt nimiä, jotka eivät pääty '.xml'. Tämä saataa liittyä, jotenkin Windows-käyttöjärjestelmän ominaisuuksiin. Tulevaisuudessa tämä muutos saattaa aiheuttaa ihmetyksiä, jos jokin verkkoelementti ei hyväksy xml-tiedostopäätteitä ja e3shwumx-ohjelma lisää päätteen automaattisesti. Parempi ratkaisu tähänkin ongelmaan olisi ollut korjata se Q1-agentilla.

Q1-verkkoelementtien osalta työtä voisi jatkaa siten, että testaus saataisiin kattamaan useampia verkkoelementtityyppejä. Tässä työssä testattavana oli ainoastaan kaksi Hopper IFUE -tyyppistä Q1-verkkoelementtiä vaikka Q1-agentti tukee kymmeniä eri elementtityyppejä. Lisäksi testien olisi hyvä kattaa myös Hardware Managementin laitteistorekisteriä lukevat ohjelmat Hardware Browser ja Hardware Inventory. E3S_HW-osajärjestelmään kuuluu myös e3shwcmx-ohjelma, jonka tehtävänä on vastaanottaa verkkoelementtien lähettämiä ilmoituksia laitteistotietojen vanhenemisesta. Tulevaisuudessa olisikin mahdollista vastaanottaa Q1-agentilta näitä ilmoituksia, joiden perusteella laitteistotietojen lataus voitaisiin automaattisesti käynnistää. Tämän toiminnallisuuden käyttöönotto vaatisi todennäköisesti samankaltaisen adaptaation teon ja testauksen kuin e3hwumx-ohjelmakin.

LÄHDELUETTELO

1 NE support effects to HW Management, [IAR], v1.1, 06.09.2005, Juha Kukkonen
Saatavissa:
http://pi.nokia.com/urn.htm?id=09006c3780487aea&DMW_DOCBASE=espoo11&auth=T&version=current&document_id=13-309915

2 E3S_HW Subsystem, [Technical Description], v1.1, 10.11.2004, Andreas Weber, Saatavissa:http://esdoc04nok.ntc.nokia.com/urn.htm?document_id=13-195521&version=current&auth=T&id=09006c37801fb840&dmw_docbase=espoo11

3 Q1 Agent C2.0, [Technical Requirement Specification], v.5.0, 09.06.2004, Ulla Tanskanen, Saatavissa:
http://esdoc04nok.ntc.nokia.com/urn.htm?document_id=13-17972&version=current&id=09006c3780076236&DMW_DOCBASE=espoo11

4 Q1Agent C2.0 PM and IM, [Design Specification], v3.1, 10.11.2004, Nisankarao Srikanth, Saatavissa:

http://esdoc04nok.ntc.nokia.com/urn.htm?document_id=13-21210&version=current&id=09006c37800806f4&DMW_DOCBASE=espoo11

5 NE3S Hardware Management Fragment, [Interface Specification], v1.0.3, 27.4.2003, M. Grosse-Kreul, Heinz-Günter Boettger.

6 SNMP Interface Subsystem, [Technical Description], Heinz-Günter Boettger, v1.0, 06.10.2000, Saatavissa: https://wwwshire.ntc.nokia.com/oss/OSS3/Projects/EMS/Subprojects/SNMP_MF/Products/SSTD_SNMP_Interface_1.0.doc

7 General HW Registry, [Subsystem Technical Specification], v1.2, 28.06.2004, Ramesh Nanjundaiah, Saatavissa: http://esdoc04nok.ntc.nokia.com/urn.htm?document_id=13-190849&version=current&id=09006c37801f333d&DMW_DOCBASE=espoo11

8 HW Configuration Upload Polling Process, [program block technical specification], v1.1, 3.12.2001, Jani Naukkarinen
Saatavissa: /vobs/subsys01/genupl/src/uplpol/doc/uplpolmx.doc

LIITELUETTELO

Liite 1 e3shwu_uploader.cc lähdekoodi

```
1
2 #include "e3shwu_Uploader.h"           // own class
3 #include "e3shwu_ServiceNe3sHWU.h"     // e3shwu_ServiceNe3sHWU_c
4
5 #include <dbomgrmx.h>                  // dbomgr_c
6 #include <fipscf_ServiceHandler.h>     // serviceHandler_c
7 #include <fiptoc_userAccessWrapper.h>  // userAccessWrapper_c
8 #include <q3fcoreascomx.h>             // q3fcorEasyConfiguration_c
9 #include <strstream>                   // ostrstream
10 #include <cpfslc_systemlog.h>          // CPFSLC::SystemLog_c
11 #include <libgen.h>                    // basename(), dirname()
12 #include <time.h>                      // localtime_r()
13
14
15 #include <stdlib.h>                      // mv
16
17
18
19 using namespace E3SHWU;
20 using namespace E3SHWL;
21 using namespace std;
22
23 const int E3SHWU_OPERATION_NONE     = 0;
24 const int E3SHWU_OPERATION_UPLOAD   = 1;
25 const int E3SHWU_OPERATION_FILE_GEN = 2;
26
27 const int E3SHWU_OPERATION_STATE_IDLE    = 0;
28 const int E3SHWU_OPERATION_STATE_SUCCESS = 1;
29 const int E3SHWU_OPERATION_STATE_FAILED  = 2;
30
31 const int E3SHWU_FTP_CLIENT_MODE = 0;
32 const int E3SHWU_FTP_SERVER_MODE = 1;
33
34 // constants for OIDs
35 const string noiHWOid = ".1.3.6.1.4.1.94.7.6";
36 const string noiHWMOperationOId       = noiHWOid + ".2.5.0";
37 const string noiHWMOperationStateOId  = noiHWOid + ".2.8.0";
38 const string noiHWUploadFileOId       = noiHWOid + ".2.6.0";
39 const string noiHWMUploadDirectoryOId = noiHWOid + ".2.4.0";
40 const string noiHWUploadBaseObjectOId = noiHWOid + ".2.7.0";
41 const string noiHWMFtpProtcolOid      = noiHWOid + ".2.2.0";
42 const string noiHWMFtpModeOId         = noiHWOid + ".2.3.0";
43 const string noiHWMErrorTypeOid       = noiHWOid + ".2.9.0";
44 const string noiHWMErrorIdOid         = noiHWOid + ".2.10.0";
45 const string noiHWMAdditionalTextOid  = noiHWOid + ".2.11.0";
46
47
48 // constants for error code
49 const int ERROR_IN_CONFIGURATION          = 1;
50 const int ERROR_OVERLOAD                  = 2;
51 const int ERROR_PARAMETERS                = 3;
52 const int ERROR_DB_ACCESS                 = 4;
53 const int ERROR_NE_TYPE_NOT_SUPPORTED     = 5;
54 const int ERROR_NO_ADDRESS                = 6;
55 const int ERROR_NO_USER                   = 7;
56 const int ERROR_NO_PASSWORD               = 8;
57 const int ERROR_NO_MEDIATOR_FOUND         = 9;
58 const int ERROR_NO_CONNECTION_TO_MEDIATOR = 10;
59 const int ERROR_NO_CONNECTION             = 11;
60 const int ERROR_NO_ACCESS                 = 12;
61 const int ERROR_UPLOAD_PROTOCOL           = 13;
62 const int ERROR_ERROR_IN_XML_FILE         = 14;
63 const int ERROR_ERROR_INTERNAL            = 15;
64
65
66 const int MAX_CONNECTION_RETIRES           =  3;
67 const int AGENT_AVAILABLE_WAIT_TIME        = 20;
68
69
70 string GetTimeString()
71 {
72     time_t timer;
73     tm tmStruct;
74
75     // get current time
76     time(&timer);
77
78     // fill the tmStruct in the initialization to current time;
79     localtime_r (&timer, &tmStruct);
```

```
80
81      // print to buffer
82      char printBuffer[40];
83      sprintf(printBuffer, "%.4d-%.2d-%.2d %.2d:%.2d:%.2d",
84              tmStruct.tm_year+1900, tmStruct.tm_mon+1, tmStruct.tm_mday,
85              tmStruct.tm_hour, tmStruct.tm_min, tmStruct.tm_sec);
86
87      return printBuffer;
88  }
89
90
91  //*****************************************************************
92  //  <PUBLIC> FUNCTION:  e3shwuFatalError_e::e3shwuFatalError_e()
93  //
94  //*****************************************************************
95  //
96  // Abstract: Constructor for Fatal Error exception
97  //
98  //*****************************************************************
99  e3shwuFatalError_e::e3shwuFatalError_e(std::string method,
100                                         std::string errReason,
101                                         int         exitCode)
102   : E3SHWL::e3shwlFatalError_e(method, errReason, exitCode)
103 {
104     // empty
105 }
106
107
108 //*****************************************************************
109 //  <PUBLIC> FUNCTION:  e3shwuTemporaryError_e::e3shwuTemporaryError_e()
110 //
111 //*****************************************************************
112 //
113 // Abstract: Constructor for Fatal Error exception
114 //
115 //*****************************************************************
116 e3shwuTemporaryError_e::e3shwuTemporaryError_e(std::string method,
117                                                std::string errReason,
118                                                int         exitCode)
119  : E3SHWL::e3shwlTemporaryError_e(method, errReason, exitCode)
120 {
121     // empty
122 }
123
124
125 //*****************************************************************
126 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::e3shwu_Uploader_c()
127 //
128 //*****************************************************************
129 //
130 // Abstract: Constructor
131 //
132 //*****************************************************************
133 e3shwu_Uploader_c::e3shwu_Uploader_c()
134  : _dbInterface(0),
135    _connector(0),
136    _moduleTest(false)
137 {
138     // empty
139 }
140
141
142 //*****************************************************************
143 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::~e3shwu_Uploader_c()
144 //
145 //*****************************************************************
146 //
147 // Abstract: Destrucor
148 //
149 //*****************************************************************
150 e3shwu_Uploader_c::~e3shwu_Uploader_c()
151 {
152     delete _dbInterface;
153     _dbInterface = 0;
154
155     delete _connector;
156     _connector = 0;
157 }
158
159
160 //*****************************************************************
```

```
161 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::Usage()
162 //
163 //*****************************************************************
164 //
165 // Abstract: Prints usage message
166 //
167 //*****************************************************************
168 void e3shwu_Uploader_c::Usage()
169 {
170   cout << "NE3S Hardware Uploader" << endl;
171   cout << "usage:" << endl;
172   cout << "e3shwumx -gid <gid> -file <file name> [-dn <DN>] [-name <name>] [-oc <object
class]" << endl
173    << "  [-vendor <vendor>] [-version <version>] [-subclass <subclass>] [-system <sys-
tem>]" << endl
174    << "  [-instance <instance>] [-trace] [-debug] [-verbose]" << endl;
175 }
176
177
178 //*****************************************************************
179 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::Initialize()
180 //
181 //*****************************************************************
182 //
183 // Abstract: Method checking passed commandling arrguments
184 //
185 //*****************************************************************
186 void e3shwu_Uploader_c::Initialize(int argc, char ** argv)
187 {
188     const char * method = "e3shwu_Uploader_c::Initialize()";
189
190     bool gidSet = false;
191     bool fileNameSet = false;
192
193     for ( int i = 1; i < argc; i++ )
194     {
195          if (strcmp(argv[i], "-gid") == 0)
196          {
197              i++;
198              if (i >= argc)
199              {
200                       throw e3shwuFatalError_e(method,
201                       "command line error: no gid specified with -gid.",
202                                                                ERROR_IN_CONFIGURATION);
203              }
204              string gidString = argv[i];
205
206              _gid = 0;
207              for (int x = 0; x < gidString.length() ; x++)
208              {
209                       _gid *= 10;
210                       _gid += gidString[x] - '0';
211              }
212              gidSet = true;
213          }
214          else if (strcmp(argv[i], "-file") == 0)
215          {
216              i++;
217              if (i >= argc)
218              {
219                       throw e3shwuFatalError_e(method,
220                       "command line error: no file name specified with -file.",
221                                                                ERROR_IN_CONFIGURATION);
222              }
223
224              _file = argv[i] ;
225
226              //ADDED START
227
228              //Add .xml if not already
229                       if ( strcmp(_file.substr((_file.length()-3),4),".xml") != 0 )
230                       {
231                                                    _file = _file + ".xml";
232                       }
233                       //ADDED END
234
235              _origfile = argv[i];
236
237              fileNameSet = true;
238          }
239          else if (strcmp(argv[i], "-dn") == 0)
```

```
240         {
241             i++;
242             if (i >= argc)
243             {
244                     throw e3shwuFatalError_e(method,
245                     "command line error: no DN specified with -dn.",
246                                             ERROR_IN_CONFIGURATION);
247             }
248             _distinguishedName = argv[i];
249         }
250         else if (strcmp(argv[i], "-name") == 0)
251         {
252             i++;
253             if (i >= argc)
254             {
255                     throw e3shwuFatalError_e(method,
256                            "command line error: no name specified with -name.",
257                                     ERROR_IN_CONFIGURATION);
258             }
259             _name = argv[i];
260         }
261         else if (strcmp(argv[i], "-oc") == 0)
262         {
263             i++;
264             if (i >= argc)
265             {
266                     throw e3shwuFatalError_e(method,
267                            "command line error: no object class specified with -oc.",
268                                     ERROR_IN_CONFIGURATION);
269             }
270             _oc = argv[i];
271         }
272         else if (strcmp(argv[i], "-version") == 0)
273         {
274             i++;
275             if (i >= argc)
276             {
277                     throw e3shwuFatalError_e(method,
278                            "command line error: no version specified with -version.",
279                                     ERROR_IN_CONFIGURATION);
280             }
281             _version = argv[i];
282         }
283         else if (strcmp(argv[i], "-vendor") == 0)
284         {
285             i++;
286             if (i >= argc)
287             {
288                     throw e3shwuFatalError_e(method,
289                            "command line error: no vendor specified with -vendor.",
290                                     ERROR_IN_CONFIGURATION);
291             }
292             _vendor = argv[i];
293         }
294         else if (strcmp(argv[i], "-subclass") == 0)
295         {
296             _subclass = "";
297             if (i + 1 < argc)
298             {
299                     string teststring = argv[i+1];
300                     if (teststring[0] != '-')
301                     {
302                         _subclass = argv[i];
303                         i++;
304                     }
305                     else
306                     {
307                   // assuming subclass ''
308                     _subclass = "";
309                     }
310             }
311         }
312         else if (strcmp(argv[i], "-system") == 0)
313         {
314             i++;
315             if (i >= argc)
316             {
317                     throw e3shwuFatalError_e(method,
318                           "command line error: no system specified with -system.",
319                                     ERROR_IN_CONFIGURATION);
320             }
```

```
321             _system = argv[i];
322         }
323         else if (strcmp(argv[i], "-instance") == 0)
324         {
325             i++;
326             if (i >= argc)
327             {
328                     throw e3shwuFatalError_e(method,
329                     "command line error: no instance number specified with -instance.",
330                                     ERROR_IN_CONFIGURATION);
331             }
332             _instance = argv[i];
333         }
334         else if (strcmp(argv[i], "-trace") == 0)
335         {
336             _trace = true;
337         }
338         else if (strcmp(argv[i], "-debug") == 0)
339         {
340             _debug = true;
341         }
342         else if (strcmp(argv[i], "-verbose") == 0)
343         {
344             _verbose = true;
345         }
346         else if (strcmp(argv[i], "-moduleTest") == 0)
347         {
348             _moduleTest = true;
349         }
350         else if ((strcmp(argv[i], "-help") == 0) || (strcmp(argv[i], "-h") == 0))
351         {
352             Usage();
353            exit (0);
354         }
355         else
356         {
357             Q3FCOR::stringBuilder_c error;
358             error << "command line error: unrecognized command line parameter '"
359                       <<  argv[i] << "'.";
360
361             throw e3shwuFatalError_e(method,
362                                      error.Get(),
363                                      ERROR_IN_CONFIGURATION);
364         }
365     }
366
367     if (gidSet == false)
368     {
369          throw e3shwuFatalError_e(method,
370                                   "command line error: gid not defined.",
371                                   ERROR_IN_CONFIGURATION);
372     }
373     if (fileNameSet == false)
374     {
375          throw e3shwuFatalError_e(method,
376                                   "command line error: file not defined.",
377                                   ERROR_IN_CONFIGURATION);
378     }
379
380     // read target directory
381     string fileName;
382     if (_moduleTest == true)
383     {
384         _myPBDir = "/vobs/src37/e3shwu";
385     }
386     else
387     {
388         _myPBDir = "/opt/nokiaoss/uma/umatkt/e3s_hw/bin";
389     }
390
391     _dcName = "NokiaOSS.local.services.SNMPInterface.vc_tktsni_nokia";
392     _dcName += ".ConnectionFactory_v2";
393
394     // split into file name intoe directory
395     char dup1[256];
396     char dup2[256];
397     strcpy (dup1, _file.c_str());
398     strcpy (dup2, _file.c_str());
399     char *fileBasename = basename(dup1);
400     char *fileDirname  = dirname(dup2);
401
```

```
402     _file = fileBasename;
403     _targetDir = fileDirname;
404 
405     // create data base interface
406     _dbInterface = new E3SHWL::e3shwl_DatabaseInterface_c;
407     if (_dbInterface == 0)
408     {
409          throw e3shwuFatalError_e(method,
410                                   "Can't create database interface",
411                                   ERROR_IN_CONFIGURATION);
412     }
413 
414     Q3fTrace("E3SHWU",
415               "Command line parameters successfully parsed",
416               ZMPBAS_DETAILED_TRACE);
417 }
418 
419 
420 //*********************************************************************
421 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::GetGid()
422 //
423 //*********************************************************************
424 //
425 // Abstract: Return the GID passed via command line
426 //
427 //*********************************************************************
428 NokiaOSS::Topology::GlobalId e3shwu_Uploader_c::GetGid()
429 {
430     return _gid;
431 }
432 
433 
434 //*********************************************************************
435 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::GetAccessInfo()
436 //
437 //*********************************************************************
438 //
439 // Abstract: Call the user access wrapper
440 //
441 //*********************************************************************
442 void e3shwu_Uploader_c::GetAccessInfo(NokiaOSS::Topology::GlobalId gid,
443                                       string & user,
444                                       string & password,
445                                       string & address,
446                                       string infoType)
447 {
448     const char * method = "e3shwu_Uploader_c::GetAccessInfo()";
449 
450     FIPTOC::userAccessWrapper_c myUserAccessWrapper;
451     try
452     {
453         myUserAccessWrapper.GetAccessInfo(gid,
454                                                                 user,
455                                                                 password,
456                                                                 address,
457                                                                 infoType.c_str());
458     }
459     catch ( const runtime_error )
460     {
461         Q3FCOR::stringBuilder_c error;
462         error << "Got runtime_error from UserAccess for gid / infoType: "
463               << gid << " / " << infoType;
464 
465         throw e3shwuFatalError_e(method,
466                                  error.Get(),
467                                  ERROR_NO_PASSWORD);
468     }
469 }
470 
471 
472 //*********************************************************************
473 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::GetIRObyMFsearch()
474 //
475 //*********************************************************************
476 //
477 // Abstract: Return the IRO gid found by the 'mf' algorithm
478 //           (objects are grouped via MF objects)
479 //
480 //*********************************************************************
481 void e3shwu_Uploader_c::GetIRObyMFsearch(NokiaOSS::Topology::GlobalId gid,
482                                                     string ocQuery,
```

```
483                                                     NokiaOSS::Topology::GlobalId & iroGid,
484                                                     string & systemDN)
485 {
486     const char * method = "e3shwu_Uploader_c::GetIRObyMFsearch()";
487
488     Q3fTrace("E3SHWU",
489               "Start searching IRO using 'mf' algorithm.",
490               ZMPBAS_DETAILED_TRACE);
491
492     iroGid = 0;
493
494     _dbInterface->GetIRObyMFsearch(gid, ocQuery, iroGid);
495
496     systemDN = _dbInterface->GetDN(iroGid);
497
498     Q3FCOR::stringBuilder_c trace;
499     trace << "GetIRObyMFsearch() found object." << endl
500            << "GID : " << iroGid << endl
501            << "DN  : " << systemDN;
502
503     Q3fTrace("E3SHWU",
504               trace.Get() );
505 }
506
507
508 //*****************************************************************
509 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::GetIRObySubTree()
510 //
511 //*****************************************************************
512 //
513 // Abstract: Return the IOR gid found by the 'subtree' algorithm
514 //
515 //*****************************************************************
516 void e3shwu_Uploader_c::GetIRObySubTree(NokiaOSS::Topology::GlobalId gid,
517                                                       string ocQuery,
518                                                       NokiaOSS::Topology::GlobalId & iroGid,
519                                                       string & systemDN)
520 {
521     const char * method = "e3shwu_Uploader_c::GetIRObySubTree()";
522
523     Q3fTrace("E3SHWU",
524               "Start searching IRO using 'subtree' algorithm.",
525               ZMPBAS_DETAILED_TRACE);
526
527     NokiaOSS::Topology::GlobalId testGid(gid);
528
529     while ((testGid != 0) &&
530            (_dbInterface->OcSupportsHWUpload(testGid, ocQuery) != true))
531     {
532          testGid = _dbInterface->GetParentGid(testGid);
533     }
534
535     if (testGid == 0)
536     {
537          throw e3shwuFatalError_e(method,
538                                   "No IRO found by 'subTree' algorithm.",
539                                   ERROR_NE_TYPE_NOT_SUPPORTED);
540     }
541
542     // set the out values
543     iroGid = testGid;
544
545     // compute system DN
546     string givenMoDN = _dbInterface->GetDN(gid);
547     string iroDN = _dbInterface->GetDN(iroGid);
548 //    string str1 = "/" + iroDN;
549 //    str1 = str1.substr(0, str1.rfind("/"));
550 //    systemDN = givenMoDN.substr(str1.length());
551     systemDN = givenMoDN; //ADDED
552
553     Q3FCOR::stringBuilder_c trace;
554     trace << "GetIRObySubTree() found object." << endl
555           << "GID : " << iroGid << endl
556 //ADDED DEBUG START
557             << "--------givenMoDN : " << givenMoDN << endl
558             << "--------iroDN : " << iroDN << endl
559 //ADDED DEBUG END
560             << "SDN(givenMoDN in this case): " << systemDN;
561
562     Q3fTrace("E3SHWU",
563               trace.Get() );
```

```
564 }
565
566
567 //*********************************************************************
568 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::ComputeIROandSystemDN()
569 //
570 //*********************************************************************
571 //
572 // Abstract: Computes the gid of the integration root object
573 //
574 //*********************************************************************
575 void e3shwu_Uploader_c::ComputeIROandSystemDN(NokiaOSS::Topology::GlobalId gid,
576                                               NokiaOSS::Topology::GlobalId & iroGid,
577                                               string & systemDN)
578 {
579     const char * method = "e3shwu_Uploader_c::ComputeIROandSystemDN()";
580
581     // build set that contains all object class numbers
582     string ocQuery;
583     serviceInfoHash_c::iterator iter;
584     set <long> ocIdSet;
585     for (iter = _FtpProtocolHash.begin(); iter != _FtpProtocolHash.end(); iter++)
586     {
587          FIPTOC::filterRule_c objectType = iter->first;
588          string actOc = objectType.GetMoc();
589          long actOcId = objectType.GetMocId();
590          ocIdSet.insert(actOcId);
591     }
592
593     // build sql clause for object class
594     if (ocIdSet.size() == 0)
595     {
596          throw e3shwuFatalError_e(method,
597                                   "There are no classes that support ne3sHWUpload",
598                                   ERROR_NE_TYPE_NOT_SUPPORTED);
599     }
600     else if (ocIdSet.size() == 1)
601     {
602          set <long>:: iterator element = ocIdSet.begin();
603          Q3FCOR::stringBuilder_c query;
604          query << " CO_OC_ID = " << *element << " ";
605          ocQuery = query.Get();
606     }
607     else
608     {
609          Q3FCOR::stringBuilder_c query;
610          query << " CO_OC_ID in ( ";
611          set <long>:: iterator x;
612          for (x = ocIdSet.begin(); x != ocIdSet.end(); x++)
613          {
614              query << *x << ", ";
615          }
616          ocQuery = query.Get();
617          ocQuery = ocQuery.substr(0, ocQuery.length() - 2);
618          ocQuery += ") ";
619     }
620
621     NokiaOSS::Topology::GlobalId myIROgid = 0;
622     string mySystemDN = "";
623
624     // first try the MF algorithem
625     try
626     {
627          GetIRObyMFsearch(gid, ocQuery, myIROgid, mySystemDN);
628
629          // check if IRO has MF algorithm configured
630          string actMoc = _dbInterface->GetOCAbbrev(myIROgid);
631          long actIntOC = _dbInterface->GetIntOC(myIROgid);
632          string actVendor = _dbInterface->GetVendor(myIROgid);
633          string actVersion = _dbInterface->GetVersion(myIROgid);
634          FIPTOC::filterRule_c objectType(actMoc, actIntOC, actVendor, actVersion);
635
636          // get ftp protocal that shall be used later on
637          serviceInfoHash_c::iterator actFtpProtocal = _FtpProtocolHash.find(objectType);
638          if (actFtpProtocal == _FtpProtocolHash.end())
639          {
640              _noiHWMFtpProtocol = "ftp";
641
642              Q3FCOR::stringBuilder_c traceMessage;
643              traceMessage << "No ftpProtocal information found for object type "
644                           << objectType << ".  Using default value 'ftp'.";
```

```
645
646             Q3fTrace("E3SHWU",
647                           traceMessage.Get());
648         }
649         else
650         {
651             _noiHWMFtpProtocol = _FtpProtocolHash[objectType];
652
653             Q3FCOR::stringBuilder_c traceMessage;
654             traceMessage << "The object class " << objectType << " uses ftp protocol '"
655                       << _noiHWMFtpProtocol << "'";
656
657             Q3fTrace("E3SHWU",
658                           traceMessage.Get());
659         }
660         serviceInfoHash_c::iterator actIROalorithm= _IROalagorithmHash.find(objectType);
661         if ((actIROalorithm == _IROalagorithmHash.end())
662            || (_IROalagorithmHash[objectType] != "mf"))
663         {
664             Q3FCOR::stringBuilder_c traceMessage;
665             traceMessage << "The object class " << objectType
666                       << " does not support IRO algorithn 'mf'.";
667
668             Q3fTrace("E3SHWU",
669                           traceMessage.Get());
670
671             myIROgid = 0;
672         }
673       else
674       {
675             _objectType = objectType;
676       }
677     }
678     catch(const E3SHWL::e3shwlFatalError_e &ex)
679     {
680         myIROgid = 0;
681
682           Q3FCOR::stringBuilder_c traceMessage;
683           traceMessage << "The 'mf' algorithm has failed. Details:" << endl
684                           << "Reason : " << ex.GetErrorReason() << endl
685                           << "Method : " << ex.GetMethod();
686
687           Q3fTrace("E3SHWU",
688                       traceMessage.Get() );
689     }
690
691     // try sub tree algorhitm
692     if (myIROgid == 0)
693     {
694          try
695          {
696             GetIRObySubTree(gid, ocQuery, myIROgid, mySystemDN);
697
698             // check if IRO has subTree algorithm configured
699             string actMoc = _dbInterface->GetOCAbbrev(myIROgid);
700             long actIntOC = _dbInterface->GetIntOC(myIROgid);
701             string actVendor = _dbInterface->GetVendor(myIROgid);
702             string actVersion = _dbInterface->GetVersion(myIROgid);
703             FIPTOC::filterRule_c objectType(actMoc, actIntOC, actVendor, actVersion);
704
705           // get ftp protocal that shall be used later on
706           serviceInfoHash_c::iterator actFtpProtocal =_FtpProtocolHash.find(objectType);
707             if (actFtpProtocal == _FtpProtocolHash.end())
708             {
709                 _noiHWMFtpProtocol = "ftp";
710
711                 Q3FCOR::stringBuilder_c traceMessage;
712                 traceMessage << "No ftpProtocal information found for object type "
713                           << objectType << ".  Using default value 'ftp'.";
714
715                 Q3fTrace("E3SHWU",
716                               traceMessage.Get());
717             }
718             else
719             {
720                 _noiHWMFtpProtocol = _FtpProtocolHash[objectType];
721
722                 Q3FCOR::stringBuilder_c traceMessage;
723                 traceMessage << "The object class " << objectType << " uses ftp protocol
'"
724                           << _noiHWMFtpProtocol << "'";
```

```
725
726                 Q3fTrace("E3SHWU",
727                                 traceMessage.Get());
728             }
729         serviceInfoHash_c::iterator actIROalorithm= _IROalagorithmHash.find(objectType);
730             if ((actIROalorithm == _IROalagorithmHash.end())
731                || (_IROalagorithmHash[objectType] != "subTree"))
732             {
733                 Q3FCOR::stringBuilder_c traceMessage;
734                 traceMessage << "The object class " << objectType
735                           << " does not support IRO algorithn 'subTree'.";
736
737                 Q3fTrace("E3SHWU",
738                                 traceMessage.Get());
739
740               myIROgid = 0;
741
742                 throw e3shwuFatalError_e(method,
743                      "The found IRO has wrong IRO alogrithm configured",
744                                       ERROR_NE_TYPE_NOT_SUPPORTED);
745             }
746           else
747           {
748                _objectType = objectType;
749           }
750         }
751         catch(const E3SHWL::e3shwlFatalError_e &ex)
752         {
753             Q3FCOR::stringBuilder_c traceMessage;
754             traceMessage << "The 'subTree' algorithm has failed. Details:" << endl
755                                  << "Reason : " << ex.GetErrorReason() << endl
756                                  << "Method : " << ex.GetMethod();
757
758             Q3fTrace("E3SHWU",
759                           traceMessage.Get() );
760         }
761     }
762
763     // if both algorithms failed we can't continue
764     if (myIROgid == 0)
765     {
766          throw e3shwuFatalError_e(method,
767                       "No IRO found. MF and sub tree alogrithm have failed",
768                                ERROR_NE_TYPE_NOT_SUPPORTED);
769     }
770
771     // set out values
772     iroGid = myIROgid;
773     systemDN = mySystemDN;
774
775     _ipAddress = _dbInterface->GetIPaddress(iroGid);
776     if (_moduleTest == true)
777     {
778          string ipAddress = getenv("E3SHWU_MT_MANAGER_FTP_IP_ADDRESS");
779          _ipAddress = ipAddress;
780
781          Q3FCOR::stringBuilder_c traceMessage;
782          traceMessage << "we are running in module test mode." << endl
783                       << "using IP address " << _ipAddress << " for ftp access.";
784
785          Q3fTrace("E3SHWU",
786                        traceMessage.Get(),
787                        ZMPBAS_DETAILED_TRACE);
788     }
789
790     Q3FCOR::stringBuilder_c traceMessage;
791     traceMessage << "IRO found : " << mySystemDN << " (" << myIROgid
792                          << " / " << _ipAddress << ")";
793
794     Q3fTrace("E3SHWU",
795                traceMessage.Get());
796 }
797
798
799 //*****************************************************************
800 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::CheckSNMPResponse()
801 //
802 //*****************************************************************
803 //
804 // Abstract: Checks if the SNMP response contains errors
805 //
```

```
806 //*********************************************************************
807 void e3shwu_Uploader_c::CheckSNMPResponse(NokiaOSS::SNMPInterface::GetResponse_var
&retVal,
808           string message,
809          NokiaOSS::Topology::GlobalId gid,
810         string callingMethod)
811 {
812     const char * method = "e3shwu_Uploader_c::CheckSNMPResponse()";
813
814     if ((retVal->err_index != -1) || (retVal->error != NokiaOSS::SNMPInterface::NOERROR))
815     {
816          Q3FCOR::stringBuilder_c error;
817          error << message << endl
818                << "(global id " << gid << ")" << endl
819                << "  err_index : " << retVal->err_index << endl
820                << "  error     : " << retVal->error;
821
822          _connector->CloseConnection();
823
824          throw e3shwuTemporaryError_e(callingMethod,
825                                       error.Get(),
826                                       ERROR_UPLOAD_PROTOCOL);
827     }
828 }
829
830
831 //*********************************************************************
832 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::CheckNumberOfVarbinds()
833 //
834 //*********************************************************************
835 //
836 // Abstract: Check that SNMP response has correct number of varbinds
837 //
838 //*********************************************************************
839 void e3shwu_Uploader_c::CheckNumberOfVarbinds(NokiaOSS::SNMPInterface::GetResponse_var
&retVal,
840          int expectedVarbindLength,
841          string message,
842          string callingMethod)
843 {
844     const char * method = "e3shwu_Uploader_c::CheckNumberOfVarbinds()";
845
846     if (retVal->var_binds.length() != expectedVarbindLength)
847     {
848          Q3FCOR::stringBuilder_c error;
849          error << message
850                << " (" << retVal->var_binds.length() << ")";
851
852          throw e3shwuTemporaryError_e(callingMethod,
853                                       error.Get(),
854                                       ERROR_UPLOAD_PROTOCOL);
855     }
856  }
857
858
859 //*********************************************************************
860 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::GetCommonOptions()
861 //
862 //*********************************************************************
863 //
864 // Abstract: Retries values from HW upload agent
865 //
866 //*********************************************************************
867 void e3shwu_Uploader_c::GetCommonOptions(NokiaOSS::Topology::GlobalId gid)
868 {
869     const char * method = "e3shwu_Uploader_c::GetCommonOptions()";
870
871     //
872     // get common options:
873     // reads noiHWMOperation, noiHWMUploadDirectory and noiHWMFtpMode
874     //
875     NokiaOSS::SNMPInterface::GetRequest_var getRequest =
876       new NokiaOSS::SNMPInterface::GetRequest;
877     getRequest->length(3);
878     getRequest[0] = noiHWMOperationOId.c_str();
879     getRequest[1] = noiHWMUploadDirectoryOId.c_str();
880     getRequest[2] = noiHWMFtpModeOId.c_str();
881
882     Q3FCOR::stringBuilder_c traceMessage;
883     traceMessage << "Reading common options. Executing SNMP get request to:" << endl
884                          << "noiHWMOperationOId       " << noiHWMOperationOId << endl
```

```
885                         << "noiHWMUploadDirectoryOId " << noiHWMUploadDirectoryOId << endl
886                         << "noiHWMFtpModeOId         " << noiHWMFtpModeOId << endl;
887     Q3fTrace("E3SHWU",
888               traceMessage.Get(),
889               ZMPBAS_DETAILED_TRACE);
890
891     NokiaOSS::SNMPInterface::GetResponse_var retVal = 0;
892     _connector->GetRequest(getRequest, retVal);
893
894     // check if SNMP response is ok
895     CheckSNMPResponse(retVal,
896                       "snmp get request to common options returns error:",
897                       gid,
898                       method);
899
900     CheckNumberOfVarbinds(retVal,
901          3,
902          "snmp get request to common options has wrong number of varbinds",
903          method);
904
905     _noiHWMOperation = retVal->var_binds[0].val.integer_val();
906
907  NokiaOSS::SNMPInterface::Asn1OctetString oct(retVal>var_binds[1].val.octet_string_val());
908     int len = oct.length();
909     char * charVal = new char[len+1];
910     charVal[len] = 0;
911
912     for (int i = 0 ; i < len; i++)
913     {
914          charVal[i] = oct[i];
915     }
916     _noiHWMUploadDirectory = charVal;
917     delete[] charVal;
918
919     _noiHWMFtpMode = retVal->var_binds[2].val.integer_val();
920
921     Q3FCOR::stringBuilder_c trace;
922     trace << "SNMP get request returned following values:" << endl
923             << "noiHWMOperationOId       : " << _noiHWMOperation << endl
924             << "noiHWMUploadDirectoryOId : " << _noiHWMUploadDirectory << endl
925             << "noiHWMFtpModeOId         : " << _noiHWMFtpMode << endl;
926     Q3fTrace("E3SHWU",
927               trace.Get(),
928               ZMPBAS_DETAILED_TRACE);
929 }
930
931
932 //*********************************************************************
933 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::SetUploadDirectory()
934 //
935 //*********************************************************************
936 //
937 // Abstract: Sets the HW upload directory in the HW upload agent
938 //
939 //*********************************************************************
940 void e3shwu_Uploader_c::SetUploadDirectory(NokiaOSS::Topology::GlobalId gid)
941 {
942     const char * method = "e3shwu_Uploader_c::SetUploadDirectory()";
943
944     if (_noiHWMFtpMode == E3SHWU_FTP_CLIENT_MODE)
945     {
946          Q3fTrace("E3SHWU",
947                        "set upload directory because the agent is using client mode");
948
949          NokiaOSS::SNMPInterface::SetRequest_var setDirectoryRequest =
950             new NokiaOSS::SNMPInterface::SetRequest;
951          setDirectoryRequest->length(1);
952
953          // set noiHWMUploadDirectory to the configured destination directory
954 setDirectoryRequest[0].var = CORBA::string_dup(noiHWMUploadDirectoryOId.c_str());
955          NokiaOSS::SNMPInterface::Asn1OctetString * tmp =
956           new NokiaOSS::SNMPInterface::Asn1OctetString;
957          tmp->length(_targetDir.length());
958          for (int i = 0; i < _targetDir.length(); i++)
959          {
960              (*tmp)[i]=_targetDir[i];
961          }
962          setDirectoryRequest[0].val.octet_string_val(*tmp);
963          delete tmp;
964
965          Q3FCOR::stringBuilder_c trace;
```

```
966           trace << "Executing SNMP set request to the following value" << endl
967                 << "noiHWMUploadDirectory : " << _targetDir << endl;
968
969           Q3fTrace("E3SHWU",
970                        trace.Get(),
971                        ZMPBAS_DETAILED_TRACE);
972
973           NokiaOSS::SNMPInterface::GetResponse_var retVal = 0;
974           _connector->SetRequest(setDirectoryRequest, retVal);
975
976         // check if SNMP response is ok
977         CheckSNMPResponse(retVal,
978                           "snmp set request returns error:",
979                        gid,
980                        method);
981
982         CheckNumberOfVarbinds(retVal,
983                        1,
984                        "Wrong number of varbinds in retrun value of snmp set request:",
985                        method);
986
987           Q3fTrace("E3SHWU",
988                        "The Upload Directory was successfully set",
989                        ZMPBAS_DETAILED_TRACE);
990     }
991 }
992
993
994 //*****************************************************************
995 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::CheckSystemResult()
996 //
997 //*****************************************************************
998 //
999 // Abstract: Checks if the return value of external command is ok
1000 //
1001 //*****************************************************************
1002 void e3shwu_Uploader_c::CheckSystemResult(int result,
1003                                                                 string command,
1004                                                                 string callingMethod)
1005 {
1006     const char * method = "e3shwu_Uploader_c::CheckSystemResult()";
1007
1008     if (result != 0)
1009     {
1010          Q3FCOR::stringBuilder_c error;
1011          error << "The command '" << command
1012                << "' failed with exit code " << result;
1013
1014          throw e3shwuFatalError_e(callingMethod,
1015                                   error.Get(),
1016                                   ERROR_IN_CONFIGURATION);
1017     }
1018 }
1019
1020
1021 //*****************************************************************
1022 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::DeleteFile()
1023 //
1024 //*****************************************************************
1025 //
1026 // Abstract: Delete the specified file
1027 //
1028 //*****************************************************************
1029 void e3shwu_Uploader_c::DeleteFile(string fileName)
1030 {
1031     const char * method = "e3shwu_Uploader_c::DeleteFile()";
1032
1033     ifstream file;
1034     file.open(fileName.c_str());
1035     if (file)
1036     {
1037         file.close();
1038
1039          Q3FCOR::stringBuilder_c traceMessage;
1040         traceMessage << "The file '" << fileName << "' has to be removed.";
1041
1042         Q3fTrace("E3SHWU",
1043                  traceMessage.Get());
1044
1045         Q3FCOR::stringBuilder_c command;
1046         command << "rm -f " << fileName;
```

```
1047
1048        int result = system(command.Get());
1049
1050         CheckSystemResult(result, command.Get(), method);
1051
1052         Q3FCOR::stringBuilder_c traceMessage2;
1053        traceMessage2 << "The file '" << fileName << "' was successfully deleted";
1054
1055         Q3fTrace("E3SHWU",
1056                     traceMessage2.Get(),
1057                     ZMPBAS_DETAILED_TRACE);
1058    }
1059 }
1060
1061
1062 //*********************************************************************
1063 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::TriggerUpload()
1064 //
1065 //*********************************************************************
1066 //
1067 // Abstract: Trigger the upload in the network element
1068 //
1069 //*********************************************************************
1070 void e3shwu_Uploader_c::TriggerUpload(NokiaOSS::Topology::GlobalId gid,
1071                                       string systemDN)
1072 {
1073     const char * method = "e3shwu_Uploader_c::TriggerUpload()";
1074
1075     //
1076     // check that agent has noiHWMOperation 'none'
1077     //
1078     time_t startTime = time(0);
1079     while ((_noiHWMOperation != E3SHWU_OPERATION_NONE)
1080           && (time(0) < startTime + AGENT_AVAILABLE_WAIT_TIME))
1081     {
1082          Q3fTrace("E3SHWU",
1083                       "rechecking agent is ready for HW upload request.",
1084                       ZMPBAS_DETAILED_TRACE);
1085
1086          GetCommonOptions(gid);
1087          sleep(2);
1088     }
1089
1090     if (_noiHWMOperation != E3SHWU_OPERATION_NONE)
1091     {
1092          // there is currently an other HW upload operation ongoing
1093          throw e3shwuTemporaryError_e(method,
1094          "there is currently an other HW upload operation ongoing",
1095                                       ERROR_UPLOAD_PROTOCOL);
1096     }
1097
1098     // remove the file that will be created in the destination directory
1099
1100     string fileName = _targetDir + "/" + _file;
1101     DeleteFile(fileName);
1102
1103     fileName = _targetDir + "/" + _file + ".gz";
1104     DeleteFile(fileName);
1105
1106
1107     //
1108     // trigger upoload
1109     //
1110     NokiaOSS::SNMPInterface::SetRequest_var setRequest =
1111        new NokiaOSS::SNMPInterface::SetRequest;
1112     setRequest->length(3);
1113
1114
1115 //ADDED. Changed noiHWMOperation = 'upload' to be the last thing to do i.e. setRequest[2]
1116
1117     // set noiHWMOperation = 'upload'
1118     setRequest[2].var = CORBA::string_dup(noiHWMOperationOId.c_str());
1119     setRequest[2].val.integer_val(E3SHWU_OPERATION_UPLOAD);
1120
1121     // set noiHWUploadFile to file name from command line
1122     setRequest[1].var = CORBA::string_dup(noiHWUploadFileOId.c_str());
1123     NokiaOSS::SNMPInterface::Asn1OctetString *tmp =
1124       new NokiaOSS::SNMPInterface::Asn1OctetString;
1125     tmp->length(_file.length());
1126     for (int i = 0; i < _file.length(); i++)
1127     {
```

```
1128         (*tmp)[i]=_file[i];
1129     }
1130     setRequest[1].val.octet_string_val(*tmp);
1131     delete tmp;
1132
1133     // set noiHWUploadBaseObject to systemDN
1134     setRequest[0].var = CORBA::string_dup(noiHWUploadBaseObjectOId.c_str());
1135     tmp = new NokiaOSS::SNMPInterface::Asn1OctetString;
1136     tmp->length(systemDN.length());
1137     for (int i = 0; i < systemDN.length(); i++)
1138     {
1139         (*tmp)[i]=systemDN[i];
1140     }
1141     setRequest[0].val.octet_string_val(*tmp);
1142     delete tmp;
1143
1144
1145
1146 /*
1147 //ADDED START
1148     // set noiHWUploadBaseObject to some working value :)
1149
1150     string ulBaseObj = _dbInterface->GetDN(gid);
1151
1152     setRequest[2].var = CORBA::string_dup(noiHWUploadBaseObjectOId.c_str());
1153     tmp = new NokiaOSS::SNMPInterface::Asn1OctetString;
1154     tmp->length(ulBaseObj.length());
1155     for (int i = 0; i < ulBaseObj.length(); i++)
1156     {
1157         (*tmp)[i]=ulBaseObj[i];
1158     }
1159     setRequest[2].val.octet_string_val(*tmp);
1160     delete tmp;
1161 //ADDED END
1162 */
1163     Q3fTrace("E3SHWU",
1164              "Triggering HW upload.");
1165
1166     Q3FCOR::stringBuilder_c trace;
1167     trace << "SNMP set request to the following values:" << endl
1168 //         << "noiHWMOperationOId       : " << E3SHWU_OPERATION_UPLOAD << endl
1169           << "noiHWUploadFileOId       : " << _file << endl
1170           << "noiHWUploadBaseObjectOId : " << systemDN << endl
1171           << "this one last---- noiHWMOperationOId:" << E3SHWU_OPERATION_UPLOAD << endl;
1172 //        << "noiHWUploadBaseObjectOId : " << ulBaseObj << endl; //ADDED
1173         Q3fTrace("E3SHWU",
1174              trace.Get(),
1175              ZMPBAS_DETAILED_TRACE);
1176
1177 //ADDED START
1178
1179 //    Q3fTrace("E3SHWU",
1180 //           "Waiting NE");
1181 //    Q3FCOR::stringBuilder_c trace2;
1182 //    trace << "nukutaan" << endl;
1183 //    Q3fTrace("E3SHWU",
1184 //       trace2.Get() );
1185
1186 //ADDED END
1187
1188     NokiaOSS::SNMPInterface::GetResponse_var retVal = 0;
1189     _connector->SetRequest(setRequest, retVal);
1190
1191     // check if SNMP response is ok
1192     CheckSNMPResponse(retVal,
1193                       "snmp set request to trigger upload returns error:",
1194                       gid,
1195                       method);
1196
1197     CheckNumberOfVarbinds(retVal,
1198                       3,
1199                       "snmp set request to trigger upload has wrong number of varbinds",
1200                       method);
1201
1202     sleep (1);
1203
1204     string actTime = GetTimeString();
1205     Q3FCOR::stringBuilder_c traceMessage;
1206     traceMessage << actTime << " HW upload was triggered successfully";
1207
1208     Q3fTrace("E3SHWU",
```

```
1209             traceMessage.Get(),
1210             ZMPBAS_DETAILED_TRACE);
1211 }
1212
1213
1214 //*********************************************************************
1215 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::WaitTillFileCreated()
1216 //
1217 //*********************************************************************
1218 //
1219 // Abstract: Wait until NE says it is ready. Throws an exception
1220 //           if the specified time is exceeded.
1221 //
1222 //*********************************************************************
1223 void e3shwu_Uploader_c::WaitTillFileCreated(NokiaOSS::Topology::GlobalId gid)
1224 {
1225     const char * method = "e3shwu_Uploader_c::WaitTillFileCreated()";
1226
1227     // get maxUploadDuration time form servicesmx.cf
1228     string tmp = _MaxUploadDurationHash[_objectType];
1229     int maxUploadDuration = atoi(tmp.c_str());
1230
1231     Q3FCOR::stringBuilder_c traceMessage;
1232     traceMessage << "Maximum upload duration for object type " << _objectType
1233                  << " is " << maxUploadDuration << " seconds." << ends;
1234
1235     Q3fTrace("E3SHWU",
1236              traceMessage.Get(),
1237              ZMPBAS_DETAILED_TRACE);
1238
1239
1240
1241     time_t actTtime = time(0);
1242     time_t timeOut = actTtime + maxUploadDuration;
1243     _noiHWMOperation = -1;
1244
1245     do
1246     {
1247          // poll operation and OperationState
1248         NokiaOSS::SNMPInterface::GetRequest_var getRequest =
1249           new NokiaOSS::SNMPInterface::GetRequest;
1250         getRequest->length(2);
1251         getRequest[0] = noiHWMOperationOId.c_str();
1252         getRequest[1] = noiHWMOperationStateOId.c_str();
1253
1254         string actTime = GetTimeString();
1255         Q3FCOR::stringBuilder_c traceMessage;
1256         traceMessage << actTime
1257          << " Checking if operation is finished. Executing SNMP get request to:" << endl
1258          << "noiHWMOperationOId       " << noiHWMOperationOId << endl
1259          << "noiHWMOperationStateOId  " << noiHWMOperationStateOId << endl;
1260         Q3fTrace("E3SHWU",
1261                  traceMessage.Get(),
1262                  ZMPBAS_DETAILED_TRACE);
1263
1264         NokiaOSS::SNMPInterface::GetResponse_var retVal = 0;
1265         _connector->GetRequest(getRequest, retVal);
1266
1267         // check if SNMP response is ok
1268         CheckSNMPResponse(retVal,
1269                           "snmp get request to check operation returns error:",
1270                           gid,
1271                           method);
1272
1273         CheckNumberOfVarbinds(retVal,
1274                               2,
1275                               "snmp get request to check operation has wrong number of varbinds",
1276                               method);
1277
1278         _noiHWMOperation = retVal->var_binds[0].val.integer_val();
1279         _noiHWMOperationState = retVal->var_binds[1].val.integer_val();
1280
1281          sleep(1);
1282
1283          // update time
1284          actTtime = time(0);
1285     }
1286     while ((_noiHWMOperation != E3SHWU_OPERATION_NONE) && (actTtime < timeOut));
1287
1288     Q3FCOR::stringBuilder_c traceMessage2;
1289     traceMessage2 << "The request returned the following results:" << endl
```

```
1290                        << "noiHWMOperationOId       " << _noiHWMOperation << endl
1291                        << "noiHWMOperationStateOId  " << _noiHWMOperationState << endl;
1292
1293     Q3fTrace("E3SHWU",
1294              traceMessage2.Get(),
1295              ZMPBAS_DETAILED_TRACE);
1296
1297     // check for time out
1298     if (actTtime >= timeOut)
1299     {
1300          Q3FCOR::stringBuilder_c error;
1301          error  << "The agent didn't finish HW upload operation in time." << endl
1302                 << "  globalId: " << gid;
1303
1304          throw e3shwuTemporaryError_e(method,
1305                                       error.Get(),
1306                                       ERROR_UPLOAD_PROTOCOL);
1307     }
1308     Q3fTrace("E3SHWU",
1309              "The Network element has finished HW upload",
1310              ZMPBAS_DETAILED_TRACE);
1311 }
1312
1313
1314 //*********************************************************************
1315 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::CheckOperationState()
1316 //
1317 //*********************************************************************
1318 //
1319 // Abstract: Store the state of the network element is the
1320 //           corresponding class variables
1321 //
1322 //*********************************************************************
1323 void e3shwu_Uploader_c::CheckOperationState(NokiaOSS::Topology::GlobalId gid)
1324 {
1325     const char * method = "e3shwu_Uploader_c::CheckOperationState()";
1326
1327     //
1328     // check operation state: read noiHWMOperationState, noiHWMErrorType,
1329     //                        noiHWMErrorId and noiHWMAdditionalText
1330     //
1331     NokiaOSS::SNMPInterface::GetRequest_var getRequest =
1332       new NokiaOSS::SNMPInterface::GetRequest;
1333     getRequest->length(4);
1334     getRequest[0] = noiHWMOperationStateOId.c_str();
1335     getRequest[1] = noiHWMErrorTypeOid.c_str();
1336     getRequest[2] = noiHWMErrorIdOid.c_str();
1337     getRequest[3] = noiHWMAdditionalTextOid.c_str();
1338
1339     Q3FCOR::stringBuilder_c traceMessage;
1340     traceMessage << "Checking operation state. Executing SNMP get request for:" << endl
1341                  << "noiHWMOperationStateOId  " << noiHWMOperationStateOId << endl
1342                  << "noiHWMErrorTypeOid       " << noiHWMErrorTypeOid << endl
1343                  << "noiHWMErrorIdOid         " << noiHWMErrorIdOid << endl
1344                  << "noiHWMAdditionalTextOid  " << noiHWMAdditionalTextOid << endl;
1345     Q3fTrace("E3SHWU",
1346              traceMessage.Get(),
1347              ZMPBAS_DETAILED_TRACE);
1348
1349     NokiaOSS::SNMPInterface::GetResponse_var retVal = 0;
1350     _connector->GetRequest(getRequest, retVal);
1351
1352     // check if SNMP response is ok
1353     CheckSNMPResponse(retVal,
1354                       "snmp get request to check operation state returns error:",
1355                  gid,
1356                       method);
1357
1358     CheckNumberOfVarbinds(retVal,
1359          4,
1360          "snmp get request to check operation state has wrong number of varbinds",
1361          method);
1362
1363     _noiHWMOperationState = retVal->var_binds[0].val.integer_val();
1364     _noiHWMErrorType = retVal->var_binds[1].val.integer_val();
1365     _noiHWMErrorId = retVal->var_binds[2].val.integer_val();
1366
1367NokiaOSS::SNMPInterface::Asn1OctetString oct(retVal->var_binds[3].val.octet_string_val());
1368     int len = oct.length();
1369     char * charVal = new char[len+1];
1370     charVal[len] = 0;
```

```
1371     for (int i = 0 ; i < len; i++)
1372     {
1373          charVal[i] = oct[i];
1374     }
1375     _noiHWMAdditionalText = charVal;
1376     delete[] charVal;
1377
1378     if (_noiHWMOperationState == E3SHWU_OPERATION_STATE_SUCCESS)
1379     {
1380          Q3FCOR::stringBuilder_c trace;
1381          trace << "SNMP get request returned following values:" << endl
1382                << "noiHWMOperationState       : " << _noiHWMOperationState << endl;
1383
1384          Q3fTrace("E3SHWU",
1385                        trace.Get(),
1386                        ZMPBAS_DETAILED_TRACE);
1387
1388          Q3fTrace("E3SHWU",
1389          "The network element has successfully finished the upload file generation.");
1390     }
1391     else
1392     {
1393          Q3FCOR::stringBuilder_c error;
1394          error << "The HW upload operation failed. Details:" << endl
1395                << "noiHWMOperationState : " << _noiHWMOperationState << endl
1396                << "noiHWMErrorType      : " << _noiHWMErrorType << endl
1397                << "noiHWMErrorId        : " << _noiHWMErrorId << endl
1398                << "noiHWMAdditionalText : " << _noiHWMAdditionalText;
1399
1400          _connector->CloseConnection();
1401
1402          throw e3shwuFatalError_e(method,
1403                                   error.Get(),
1404                                   ERROR_UPLOAD_PROTOCOL);
1405     }
1406
1407     // snmp connection is not needed any longer
1408     _connector->CloseConnection();
1409 }
1410
1411
1412 //*********************************************************************
1413 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::GetFile()
1414 //
1415 //*********************************************************************
1416 //
1417 // Abstract: Execute the command to retrieve the file form the
1418 //           network element and do post processing
1419 //
1420 //*********************************************************************
1421 void e3shwu_Uploader_c::GetFile(NokiaOSS::Topology::GlobalId iroGid)
1422 {
1423     const char * method = "e3shwu_Uploader_c::GetFile()";
1424
1425     //
1426     // get the file
1427     //
1428
1429     // 'MYTRACELOGDIR' exists (already checked in DoWork()
1430     string traceFileName = getenv("MYTRACELOGDIR");
1431     traceFileName += "/e3shwu_Transfer.trc";
1432
1433     if (_noiHWMFtpMode == E3SHWU_FTP_SERVER_MODE)
1434     {
1435          Q3fTrace("E3SHWU",
1436                        "The agent is running in server mode");
1437
1438          string user;
1439          string password;
1440          string address;
1441         GetAccessInfo(iroGid, user, password, address, "GENERIC_NE_FTP_ACCESS");
1442
1443          string ftpBinary = _myPBDir + "/e3shwu_Transfer.pl";
1444
1445          //ADDED START
1446          //We will remove C: if found
1447          if (_noiHWMUploadDirectory.substr(0,2) == "C:")
1448          {
1449              int i = _noiHWMUploadDirectory.length();
1450              _noiHWMUploadDirectory = _noiHWMUploadDirectory.substr(2,i-2);
1451          }
```

```
1452         //ADDED END
1453
1454         Q3FCOR::stringBuilder_c command;
1455         command << ftpBinary
1456                     << " -file " << _file
1457                     << " -dir " << "'"+_noiHWMUploadDirectory+"'" //ADDED ''
1458                     << " -user " << user
1459                     << " -passwd \"" << password << "\""
1460                     << " -gid " << iroGid
1461                     << " -protocol " << _noiHWMFtpProtocol
1462                     << " -targetDir " << _targetDir
1463                     << " -ip " << _ipAddress;
1464
1465         if (Q3fGetTraceLevel( "E3SHWU" ) >= 5)
1466         {
1467             command << " -trace"
1468                         << " -traceFile " << traceFileName;
1469         }
1470
1471 /*
1472
1473 #ftp transfer
1474 '/opt/nokiaoss/uma/umatkt/e3s_hw/bin/e3shwu_Transfer.pl -file Bus-2Node-100.xml -dir
'\Program Files\Nokia\Q1Agent\data\q1\hw-xml' -user q1pmuser
1475 -passwd xxxxx -gid 911093000000104983 -protocol ftp -targetDir /m/home/omc/koskinen/tmp
1476 -ip 10.8.52.212 -trace -traceFile /var/opt/nokiaoss/tmp/trace/e3shwu_Transfer.trc
1477
1478
1479 */
1480
1481
1482         Q3FCOR::stringBuilder_c commandPrintVersion;
1483         commandPrintVersion << ftpBinary
1484                     << " -file " << _file
1485                     << " -dir " << "'"+_noiHWMUploadDirectory+"'" //ADDED ''
1486                     << " -user " << user
1487                     << " -passwd " << "xxxxx"
1488                     << " -gid " << iroGid
1489                     << " -protocol " << _noiHWMFtpProtocol
1490                     << " -targetDir " << _targetDir
1491                     << " -ip " << _ipAddress;
1492
1493         if (Q3fGetTraceLevel( "E3SHWU" ) >= 5)
1494         {
1495             commandPrintVersion << " -trace"
1496                                                 << " -traceFile " << traceFileName;
1497         }
1498
1499         Q3FCOR::stringBuilder_c message2;
1500         message2 << "Executing command : " << commandPrintVersion.Get();
1501
1502         Q3fTrace("E3SHWU", message2, ZMPBAS_DETAILED_TRACE);
1503
1504         int result = system(command.Get());
1505
1506         CheckSystemResult(result / 256, commandPrintVersion.Get(), method);
1507
1508         Q3fTrace("E3SHWU",
1509                     "The ftp script was successful.",
1510                     ZMPBAS_DETAILED_TRACE);
1511     }
1512     else
1513     {
1514         Q3fTrace("E3SHWU",
1515                     "The agent is running in client mode");
1516
1517         Q3fTrace("E3SHWU",
1518                     "Checking if file needs to be decompressed",
1519                     ZMPBAS_DETAILED_TRACE);
1520
1521         // check if the file written by the agent has to be decompressed
1522         string fileName = _targetDir + "/" + _file + ".gz";
1523         ifstream file;
1524         file.open(fileName.c_str());
1525         if (file)
1526         {
1527             file.close();
1528
1529             Q3fTrace("E3SHWU",
1530                         "The file is gzipped. Unzipping it.",
1531                         ZMPBAS_DETAILED_TRACE);
```

```
1532
1533             Q3FCOR::stringBuilder_c command;
1534             command << "gunzip " << fileName;
1535
1536             int result = system(command.Get());
1537
1538             CheckSystemResult(result, command.Get(), method);
1539         }
1540     }
1541
1542     // check if uncompressed file is there
1543     string fileName = _targetDir + "/" + _file;
1544     ifstream file;
1545     file.open(fileName.c_str());
1546
1547     if (!file)
1548     {
1549         Q3FCOR::stringBuilder_c error;
1550         error << "The file '" << fileName << "' does not exist.";
1551
1552         throw e3shwuFatalError_e(method,
1553                                  error.Get(),
1554                                  ERROR_ERROR_IN_XML_FILE);
1555     }
1556     file.close();
1557
1558     Q3fTrace("E3SHWU",
1559              "The requested file was successfully transfered.");
1560
1561
1562 //MOVE
1563     //ADDED
1564      string origFileName = /*_targetDir + "/" +*/ _origfile;
1565      string comm = "mv " + fileName + " " + origFileName;
1566
1567     printf("\nE3SHWU : COMMAND FOR moving file WILL BE (%s)\n\n ", comm.c_str());
1568
1569
1570
1571     //
1572     // do the postporcessing
1573     //
1574
1575     // first execute the post processing script defined by the adaptation
1576     serviceInfoHash_c::iterator script = _PostProcessingScriptHash.find(_objectType);
1577     if (script != _PostProcessingScriptHash.end())
1578     {
1579         // now execute the standard NE3S post processing script
1580         string protprocessingSript = _PostProcessingScriptHash[_objectType];
1581         string myFile = _targetDir + "/" + _file;
1582
1583         string postProcessingTraceFile = getenv("MYTRACELOGDIR");
1584         postProcessingTraceFile += "/e3shwu_postProcessing.trc";
1585
1586         Q3FCOR::stringBuilder_c command;
1587         command << protprocessingSript
1588                 << " -file " << myFile
1589                 << " -gid " << _gid;
1590
1591         if (Q3fGetTraceLevel( "E3SHWU" ) >= 5)
1592         {
1593             command << " -trace"
1594                     << " -traceFile " << postProcessingTraceFile;
1595         }
1596
1597         Q3FCOR::stringBuilder_c trace;
1598         trace << "Executing command : " << command.Get();
1599
1600         Q3fTrace("E3SHWU", trace, ZMPBAS_DETAILED_TRACE);
1601
1602         // check if post processing script exists
1603         ifstream file;
1604         file.open(protprocessingSript.c_str());
1605         if (!file)
1606         {
1607             Q3FCOR::stringBuilder_c error;
1608 error << "The post-processing script '" << protprocessingSript << "' does not exist.";
1609
1610             throw e3shwuFatalError_e(method,
1611                                      error.Get(),
1612                                      ERROR_ERROR_IN_XML_FILE);
```

```
1613         }
1614         file.close();
1615 
1616         int result = system(command.Get());
1617 
1618         CheckSystemResult(result / 256, command.Get(), method);
1619 
1620         Q3fTrace("E3SHWU",
1621           "The adaptation specific post-processing script was successfully executed.");
1622     }
1623     else
1624     {
1625         Q3fTrace("E3SHWU",
1626                  "No adaptation specific post-processing script defined.",
1627                 ZMPBAS_DETAILED_TRACE);
1628     }
1629 
1630     // now execute the standard NE3S post processing script
1631     string protprocessingBinary = _myPBDir + "/e3shwu_Postporcessing.pl";
1632     string myFile = _targetDir + "/" + _file;
1633 
1634     Q3FCOR::stringBuilder_c command;
1635     command << protprocessingBinary
1636             << " -file " << myFile
1637             << " -gid " << _gid;
1638 
1639     string standardPostProcessingTraceFile = getenv("MYTRACELOGDIR");
1640     standardPostProcessingTraceFile += "/e3shwu_postProcessing.trc";
1641 
1642     if (Q3fGetTraceLevel( "E3SHWU" ) >= 5)
1643     {
1644         command << " -trace"
1645                 << " -traceFile " << standardPostProcessingTraceFile;
1646     }
1647 
1648     Q3FCOR::stringBuilder_c trace;
1649     trace << "Executing command : " << command.Get();
1650 
1651     Q3fTrace("E3SHWU", trace, ZMPBAS_DETAILED_TRACE);
1652 
1653     int result = system(command.Get());
1654 
1655     CheckSystemResult(result / 256, command.Get(), method);
1656 
1657     Q3fTrace("E3SHWU",
1658              "The standard post-processing script was successfully executed.");
1659 
1660 
1661     printf("\nE3SHWU : NOW moving file (%s)\n\n ", comm.c_str());
1662 
1663     system(comm.c_str());
1664 
1665     Q3fTrace("E3SHWU",
1666              "The error free process has completed!!");
1667 
1668 
1669 }
1670 
1671 
1672 //*********************************************************************
1673 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::DoUpload()
1674 //
1675 //*********************************************************************
1676 //
1677 // Abstract: Perform the HW upload operation on the network element
1678 //
1679 //*********************************************************************
1680 int e3shwu_Uploader_c::DoUpload(NokiaOSS::Topology::GlobalId gid, string systemDN)
1681 {
1682     const char * method = "e3shwu_Uploader_c::DoUpload()";
1683 
1684     //
1685     // open connection
1686     //
1687     int retryCount = 0;
1688     bool connectionOpend = false;
1689 
1690     try
1691     {
1692         do
1693         {
```

```
1694            try
1695            {
1696                string traceMessage("trying to open connection to ");
1697                traceMessage += _dcName;
1698
1699                Q3fTrace("E3SHWU",
1700                             traceMessage.c_str());
1701
1702                _connector = new e3shwu_SNMP_Connector_c(_dcName.c_str());
1703
1704                _connector->OpenConnection(gid);
1705
1706                Q3fTrace("E3SHWU",
1707                             "connection succesfully opened");
1708
1709                connectionOpend = true;
1710            }
1711            catch (const NokiaOSS::SNMPInterface::TemporaryError &ex)
1712            {
1713                Q3FCOR::stringBuilder_c error;
1714                error << "TemporaryError while opening connection." << endl
1715                           << "Details : " << ex.error;
1716
1717                Q3fTrace("E3SHWU",
1718                             error.Get() );
1719
1720                _connector->CloseConnection();
1721
1722                retryCount++;
1723                connectionOpend = false;
1724
1725                     // sleep some time
1726                     sleep(5);
1727            }
1728            catch ( const CORBA::Exception &e )
1729            {
1730               string info = CPFDEF::ToString(e);
1731
1732                Q3FCOR::stringBuilder_c trace;
1733                trace << "CORBA::Exception while opening connection. Details:"
1734                     << endl << info;
1735
1736                Q3fTrace("E3SHWU",
1737                             trace.Get() );
1738
1739                _connector->CloseConnection();
1740
1741                retryCount++;
1742                connectionOpend = false;
1743
1744                     // sleep some time
1745                     sleep(5);
1746            }
1747            // don't catch FatalError
1748        }
1749        while ((connectionOpend == false) && (retryCount < MAX_CONNECTION_RETIRES));
1750
1751        if (connectionOpend == false)
1752        {
1753         string errorStr ("Can not open connection to DataCollector: too many temporayEr-
          ror exceptions.");
1754
1755            throw e3shwuFatalError_e(method,
1756                                     errorStr.c_str(),
1757                                     ERROR_NO_CONNECTION_TO_MEDIATOR);
1758        }
1759
1760        GetCommonOptions(gid);
1761
1762        SetUploadDirectory(gid);
1763
1764        TriggerUpload(gid, systemDN);
1765 //sleep (15);
1766        WaitTillFileCreated(gid);
1767
1768        CheckOperationState(gid);
1769    }
1770    catch (const NokiaOSS::SNMPInterface::TemporaryError &ex)
1771    {
1772        Q3FCOR::stringBuilder_c error;
1773        error << "TemporaryError while opening connection." << endl
```

```
1774                << "Details : " << ex.error;
1775
1776         Q3fTrace("E3SHWU",
1777                  error.Get() );
1778
1779         _connector->CloseConnection();
1780
1781         throw e3shwuTemporaryError_e(method,
1782                                                 error.Get(),
1783                                       ERROR_NO_CONNECTION_TO_MEDIATOR);
1784     }
1785     catch ( const NokiaOSS::SNMPInterface::FatalError &ex )
1786     {
1787         Q3FCOR::stringBuilder_c error;
1788         error << "FatalError while opening connection." << endl
1789               << "Details : " << ex.error;
1790
1791         _connector->CloseConnection();
1792
1793         throw e3shwuFatalError_e(method,
1794                                             error.Get(),
1795                                    ERROR_NO_CONNECTION_TO_MEDIATOR);
1796     }
1797     catch ( const CORBA::SystemException &se )
1798     {
1799         ostrstream errorStream;
1800         errorStream << "" << se << ends;
1801
1802         _connector->CloseConnection();
1803         string error(errorStream.str());
1804         errorStream.rdbuf()->freeze(0);
1805
1806         throw e3shwuFatalError_e(method,
1807                                             error.c_str(),
1808                                    ERROR_NO_CONNECTION_TO_MEDIATOR);
1809     }
1810     catch ( const CPFDEF::Error_c &err )
1811     {
1812         ostrstream errorStream;
1813         errorStream << "got CPFDEF::Error_c exception:" << endl
1814                     << err << ends;
1815
1816         string errorText(errorStream.str());
1817         errorStream.rdbuf()->freeze(0);
1818
1819         _connector->CloseConnection();
1820
1821         throw e3shwuFatalError_e(method,
1822                                             errorText.c_str(),
1823                                    ERROR_NO_CONNECTION_TO_MEDIATOR);
1824     }
1825
1826     GetFile(gid);
1827
1828     return 0;
1829 }
1830
1831
1832 //*****************************************************************
1833 //  <PUBLIC> FUNCTION:  e3shwu_Uploader_c::DoWork()
1834 //
1835 //*****************************************************************
1836 //
1837 // Abstract: Do everithing
1838 //
1839 //*****************************************************************
1840 int e3shwu_Uploader_c::DoWork(int argc, char ** argv)
1841 {
1842     const char *method = "e3shwu_EventClient_c::DoWork()";
1843
1844     const char * omcLogDir = 0;
1845     const char * traceDir = 0;
1846     int myReturnCode = 15;
1847     zmpbasTraceTarget_t traceTarget = ZMPBAS_FILE;
1848
1849     try
1850     {
1851         //*************************************************************
1852         // Set process type, process name and enable tracing function
1853         //*************************************************************
1854         SetProcessType( 0xA508 );
```

```
1855        SetProcessName( argv[0] );
1856
1857        CPFDEF::ProcessProperties_c properties(0xA508,
1858                                               argc,
1859                                               argv,
1860                                               CPFDEF::PROCESS_IS_DEPLOYED_AS_CLIENT,
1861                                               CPFDEF::ORB_OBJECT_ADAPTER_SINGLE_THREADED);
1862
1863        CPFDEF::Process_c process(properties);
1864
1865        // initialize Uploader
1866        Initialize(argc, argv);
1867
1868        // initialize trace / debug
1869        int traceLevel = 0;
1870
1871        if (( _debug == true ) && ( _trace == true ))
1872         {
1873             traceTarget = ZMPBAS_CERR_AND_FILE;
1874            traceLevel = 6;
1875         }
1876        else if ( _debug == true )
1877        {
1878            traceTarget = ZMPBAS_CERR;
1879            traceLevel = 6;
1880        }
1881        else if ( _trace == true )
1882        {
1883            traceTarget = ZMPBAS_FILE;
1884            traceLevel = 6;
1885        }
1886
1887        if (( _verbose == true ) && ((( _debug == true ) || ( _trace == true ))))
1888        {
1889            traceLevel = 9;
1890        }
1891
1892        omcLogDir = getenv("MYERRLOGDIR");
1893        traceDir = getenv("MYTRACELOGDIR");
1894
1895        zmpbasTracer_c::InitializeTracing("e3shwumx",
1896                                          traceDir,
1897                                          "e3shwumx.trc",
1898                                          traceLevel,
1899                                          500000,
1900                                          traceTarget);
1901
1902        Q3fInitializeTrace( "ALL", zmpbasTracer_c::tracePtr, T, F);
1903
1904        // create start message in wcltoday.log
1905        CPFSLC::SystemLog_c::Send(5, "e3shwumx: NE3S HW uploader starting up");
1906
1907        string actTime = GetTimeString();
1908        Q3FCOR::stringBuilder_c traceMessage;
1909        traceMessage << actTime << " Starting NE3S Hardware Upload";
1910
1911        Q3fTrace(Q3FPRC_ID,
1912                 traceMessage.Get());
1913
1914        Q3FCOR::stringBuilder_c traceMessage2;
1915        traceMessage2 << actTime << " Trace level set to level " << traceLevel;
1916
1917        Q3fTrace(Q3FPRC_ID,
1918                 traceMessage2.Get(),
1919                 ZMPBAS_DEFAULT_TRACE);
1920
1921
1922        if ( omcLogDir == 0 )
1923        {
1924            throw e3shwuFatalError_e(method,
1925                                     "environment variable 'MYERRLOGDIR' not set.",
1926                                     ERROR_IN_CONFIGURATION);
1927        }
1928
1929        if ( traceDir == 0 )
1930        {
1931            throw e3shwuFatalError_e(method,
1932                                     "environment variable 'MYTRACELOGDIR' not set.",
1933                                     ERROR_IN_CONFIGURATION);
1934        }
1935
```

```
1936        // configure ServiceHandler for 'ne3sHWUpload'
1937        FIPSCF::serviceHandler_c serviceHandler;
1938        FIPSCF::serviceType_c *service = 0;
1939        service = new e3shwu_ServiceNe3sHWU_c(&_FtpProtocolHash,
1940                                              &_IROalagorithmHash,
1941                                              &_PostProcessingScriptHash,
1942                                              &_MaxUploadDurationHash);
1943        serviceHandler.AddService(service);
1944        serviceHandler.Configure();
1945
1946        // get gid from command line
1947        NokiaOSS::Topology::GlobalId orgGid = GetGid();
1948
1949        // find IRO
1950        NokiaOSS::Topology::GlobalId iroGid = 0;
1951        string systemDN = "";
1952        ComputeIROandSystemDN(orgGid, iroGid, systemDN);
1953
1954        // do HW upload from IRO
1955        myReturnCode = DoUpload(iroGid, systemDN);
1956    }
1957    catch(const E3SHWL::e3shwlFatalError_e &ex)
1958    {
1959        Q3FCOR::stringBuilder_c traceMessage;
1960        traceMessage << "e3shwumx has received a Fatal Error exception. Details:" << endl
1961                     << "Reason   : " << ex.GetErrorReason() << endl
1962                     << "Method   : " << ex.GetMethod() << endl
1963                     << "Exitcode : " << ex.GetExitCode();
1964
1965        Q3fTrace("E3SHWU",
1966                 traceMessage.Get() );
1967
1968        errAtom_c   *errorAtom;
1969        errorAtom = new errAtom_c(E3SHWU_FATAL_ERROR,
1970                                  ex.GetErrorReason().c_str(),
1971                                  ex.GetMethod().c_str());
1972
1973        errorAtom->WriteToStderr();
1974        errorAtom->WriteToLogFile();
1975
1976        delete errorAtom;
1977        errorAtom = 0;
1978
1979        myReturnCode = ex.GetExitCode();
1980    }
1981    catch(const E3SHWL::e3shwlTemporaryError_e &ex)
1982    {
1983        Q3FCOR::stringBuilder_c traceMessage;
1984        traceMessage << "e3shwumx has received a Temporary Error exception. Details:" <<
endl
1985                     << "Reason   : " << ex.GetErrorReason() << endl
1986                     << "Method   : " << ex.GetMethod() << endl
1987                     << "Exitcode : " << ex.GetExitCode();
1988
1989        Q3fTrace("E3SHWU",
1990                 traceMessage.Get() );
1991
1992        errAtom_c   *errorAtom;
1993        errorAtom = new errAtom_c(E3SHWU_TEMPORARY_ERROR,
1994                                  ex.GetErrorReason().c_str(),
1995                                  ex.GetMethod().c_str());
1996
1997        errorAtom->WriteToStderr();
1998        errorAtom->WriteToLogFile();
1999
2000        delete errorAtom;
2001        errorAtom = 0;
2002
2003        myReturnCode = ex.GetExitCode();
2004    }
2005    catch(...)
2006    {
2007        Q3FCOR::stringBuilder_c traceMessage;
2008        traceMessage << "e3shwumx has received an unknown exception.";
2009
2010        Q3fTrace("E3SHWU",
2011                 traceMessage.Get() );
2012
2013        errAtom_c   *errorAtom;
2014        errorAtom = new errAtom_c(E3SHWU_FATAL_ERROR,
2015                                  "exception is unknown",
```

```
2016                                    "main");
2017
2018         errorAtom->WriteToStderr();
2019         errorAtom->WriteToLogFile();
2020
2021         delete errorAtom;
2022         errorAtom = 0;
2023
2024         myReturnCode = 1;
2025     }
2026
2027     if (myReturnCode == 0)
2028     {
2029      CPFSLC::SystemLog_c::Send(5, "e3shwumx: NE3S HW upload was successfully finished.");
2030
2031         string actTime = GetTimeString();
2032         Q3FCOR::stringBuilder_c traceMessage;
2033         traceMessage << actTime << " The HW upload was successfully finished.";
2034
2035         Q3fTrace("E3SHWU",
2036                  traceMessage.Get());
2037     }
2038     else
2039     {
2040     CPFSLC::SystemLog_c::Send(5, "e3shwumx: NE3S HW upload has failed. For details check
$MYERRLOGDIR/werlogA508.log");
2041
2042         string actTime = GetTimeString();
2043         Q3FCOR::stringBuilder_c traceMessage;
2044         traceMessage << actTime << " The HW upload operatation has failed.";
2045
2046         Q3fTrace("E3SHWU",
2047                  traceMessage.Get());
2048     }
2049
2050    return myReturnCode;
2051 }
2052
2053
```