**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# LDAP COMO SUPORTE NORMALIZADO PARA MAPAS DE DESCOBERTA

Lino Miguel Martins Tinoco

Dissertação submetida para satisfação parcial dos requisitos do
grau de Mestre em
Redes e Serviços de Comunicação

Porto, Março de 2005

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# LDAP COMO SUPORTE NORMALIZADO PARA MAPAS DE DESCOBERTA

Lino Miguel Martins Tinoco

Licenciado em Engenharia Electrónica Industrial pela
Escola de Engenharia da Universidade do Minho

Dissertação submetida para satisfação parcial dos requisitos do grau de Mestre em
Redes e Serviços de Comunicação

Dissertação realizada sob a supervisão do
Professor Doutor Raul Teixeira de Oliveira,
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia do Porto

Porto, Março de 2005

*À minha mulher e filha,*
*pelas horas que não foram nossas.*

## ABSTRACT

Every day now, network management plays an important and bigger role. The number of applications dedicated to network management grows, as well as the complexity of the networks to manage.

Companies have an ever growing number of networks with different topologies, addresses and also a huge number of active nodes in each network. To ease up the management, applications provide several tools to easily discover networks, network elements and help on its administration.

A problem rises though. Each application it's a close universe where the discovered information that is provided to the manager on discovery maps is kept under proprietary formats, making impossible to share that same information with other applications and centralize it.

This work intends to show the advantage of having a normalized support for discovery maps that are generated by different applications, being LDAP that support. Through this work will be pointed the constrains of the present methods of support, the reasons for choosing LDAP as support, the techniques used for discovering networks and their elements, the model adopted for the LDAP directory service and examples of advantages of having LDAP directory service as a repository for all the information obtained during the discovery process.

On the final part of this paper a prototype developed to serve as proof-of-concept will be presented, helping to better understand the concept presented on previous chapters. This prototype will demonstrate the possibility of sharing information between different applications with different capabilities, using LDAP as support for the shared information.

# RESUMO

Cada vez mais nos nossos dias a gestão de redes assume um papel de destaque e importância. O número de aplicações dedicadas à gestão de redes cresce, bem como a complexidade das redes a gerir.

As empresas têm cada vez mais redes com diferentes topologias, endereçamentos e também um grande número de máquinas activas em cada rede. Para facilitar a tarefa de gestão, as aplicações oferecem facilidades na descoberta de redes e novos elementos de rede e na sua administração.

Um problema levanta-se no entanto. Cada aplicação é um universo fechado, onde a informação descoberta e oferecida ao gestor em mapas de descoberta é guardada sob formatos proprietários, impossibilitando a partilha dessa mesma informação por outras aplicações e a sua centralização.

Este trabalho pretende mostrar a vantagem de se ter um suporte normalizado para os mapas de descoberta gerados por diferentes aplicações e desse mesmo suporte ser o LDAP. Ao longo do trabalho serão apresentados os constrangimentos dos métodos actuais, as razões da escolha do LDAP para suporte, os actuais métodos utilizados para descobrir a informação sobre as redes e os seus elementos para que esta seja disponibilizada em mapas de descoberta, o modelo adoptado para o serviço de directório LDAP e exemplos de vantagens em ter um serviço de directório LDAP como repositório da informação obtida no processo de descoberta.

No final do trabalho é apresentado um protótipo que ilustra de forma mais clara uma aplicação do conceito aqui defendido, onde é mostrada a possibilidade de partilha de informação entre diferentes aplicações com potencialidades diferentes utilizando o LDAP como meio de suporte à informação partilhada.

## AGRADECIMENTOS

Agradeço ao Professor Doutor Raul Filipe Teixeira Oliveira por ter aceite ser o meu orientador, pelo trabalho proposto e pelo apoio prestado durante a elaboração do mesmo.

Agradeço também ao colega de doutoramento Bruno Filipe Marques, que me foi guiando e prestando auxílio e que criou as bases necessárias para que o protótipo no final deste trabalho pudesse ser implementado.

# ÍNDICE

**ABSTRACT** i

**RESUMO** iii

**AGRADECIMENTOS** v

**ÍNDICE** vii

ÍNDICE DE TABELAS ix
ÍNDICE DE FIGURAS ix

**NOMENCLATURA** xi

ABREVIATURAS xi

**1 INTRODUÇÃO** 1

1.1 ÂMBITO DA TESE 1
1.2 PROBLEMA 1
1.3 ABORDAGEM 2
1.4 ESTRUTURA 3

**2 DESCOBERTA** 5

2.1 COMO SE FAZ 5
*2.1.1 Ping* 5
*2.1.2 Broadcast Ping* 6
*2.1.3 Tabelas de ARP* 6
*2.1.4 Traceroute* 7
*2.1.5 SNMP* 7
*2.1.6 Varrimento de Portos* 8
*2.1.7 Transferência de Zona do DNS* 8
2.2 QUEM FAZ 10
2.3 PERFORMANCE 10
*2.3.1 Network Node Manager* 12
*2.3.2 Scotty/Tkined* 12
2.4 MODELO DE DADOS 13
*2.4.1 Network Node Manager* 14
*2.4.2 Scotty/Tkined* 15

**3 MAPEAMENTO DAS DESCOBERTAS EM LDAP** 17

3.1 MODELO LDAP 17
*3.1.1 Topologia* 18
*3.1.2 Eventos* 19
*3.1.3 Tendências* 20

**4 VANTAGENS DE UM SUPORTE NORMALIZADO** 23

4.1 NAGIOS 23

4.2 AGENTES 24
**5 PROTÓTIPO 27**
**6 CONCLUSÕES 31**
**ANEXO A 33**
FICHEIRO MANIA_OBJECTS.SCHEMA 33
FICHEIRO MANIA_ATTRIBUTE.SCHEMA 38
**ANEXO B 65**
SCRIPT PARA CRIAR LDIF DOS EVENTOS 65
SCRIPT PARA CRIAR LDIF DAS TENDÊNCIAS 75
SCRIPT PARA CRIAR LDIF DA TOPOLOGIA 84
**ANEXO C 95**
SCRIPT PARA CRIAR FICHEIRO DE HOSTS DO NAGIOS 95
**BIBLIOGRAFIA 99**

## ÍNDICE DE TABELAS

Tabela 1 – Prós e contras dos métodos de descoberta 9
Tabela 2 – Definição de um Objecto do Mapa de Descoberta do Scotty 16

## ÍNDICE DE FIGURAS

Figura 1 – Relação entre a *data warehouse* e as BD's de Tendências, Topologia e Eventos 14
Figura 2 – Estrutura do serviço de directório LDAP até aos Objectos 18
Figura 3 – Estrutura da BD deTopologia mapeada em LDAP 19
Figura 4 – Estrutura da BD de Eventos mapeada em LDAP 20
Figura 5 – Estrutura da BD de Tendências mapeada em LDAP 21
Figura 6 – Sequência de realização das diferentes fases de implementação do protótipo 27
Figura 7 – Pesquisa da máquina que suporta o serviço de directório LDAP de um domínio 30

# NOMENCLATURA

## ABREVIATURAS

| | |
|---|---|
| ARP | *Address Resolution Protocol* |
| BD | *Base de Dados* |
| DEEC | *Departamento de Engenharia Electrotécnica e de Computadores* |
| DHCP | *Dynamic Host Configuration Protocol* |
| DNS | *Domain Name Service* |
| DSML | *Directory Service Mark-up Language* |
| FEUP | *Faculdade de Engenharia da Universidade do Porto* |
| FTP | *File Transfer Protocol* |
| HP | *Hewlett-Packard* |
| HPOV NNM | *HP OpenView Network Node Manager* |
| HTTP | *Hypertext Transfer Protocol* |
| ICMP | *Internet Message Control Protocol* |
| IP | *Internet Protocol* |
| ISP | *Internet Service Provider* |
| LAN | *Local Area Network* |
| LB | *Largura de Banda* |
| LDAP | *Lightweight Directory Access Protocol* |
| LDIF | *Lightweight Directory Interchange Format* |
| MAC | *Media Access Control* |
| MIB | *Management Information Base* |
| OSI | *Open Systems Interconnect* |
| SNMP | *Simple Network Management Protocol* |
| SQL | *Structured Query Language* |
| TCP | *Transmission Control Protocol* |
| TTL | *Time To Live* |

# 1 INTRODUÇÃO

Existem diversas aplicações comerciais e de código livre capazes de realizar a descoberta de redes, elementos de rede e serviços e apresentar o respectivo mapa de descoberta dos elementos encontrados. A informação contida nesses mapas de descoberta é de uma forma geral armazenada sob um formato proprietário e não está disponível para outras aplicações.

Neste capítulo vão ser abordados quatro tópicos distintos que pretendem apresentar de forma sumária quais as limitações relacionadas com as escolhas feitas pelas diversas aplicações existentes no que toca ao suporte oferecido para a informação descoberta, bem como a forma de as resolver.

## *1.1 ÂMBITO DA TESE*

O trabalho apresentado nesta tese decorre da inexistência de um suporte normalizado no que toca ao armazenamento e disponibilização da informação contida em mapas de descoberta de redes e serviços.

Pretendem-se apresentar os problemas existentes e explorar possíveis soluções, bem como apresentar as vantagens e desvantagens de sistemas de descoberta comerciais e de código livre existentes mo mercado. Será também abordada a forma como estes dois tipos de aplicações endereçam o problema da descoberta (*discovery*) e apresentado no final um exemplo (protótipo) de uma solução possível para um suporte à informação disponível em mapas de descoberta e as vantagens deste tipo de suporte, com a partilha com outra aplicação da informação contida em mapas de descoberta.

## *1.2 PROBLEMA*

As redes actuais são complexas e vão sendo alteradas com bastante frequência. Para manter um registo preciso da topologia de rede e do estado dos seus elementos (nós) são frequentemente utilizadas ferramentas de descoberta e gestão.

Após a descoberta de uma rede e dos seus serviços por uma determinada aplicação, a informação obtida é guardada tipicamente num formato proprietário, num formato não optimizado ou mesmo apenas em memória. Embora os formatos proprietários possam facilitar o armazenamento e, em alguns casos, serem mais optimizados, eles não podem ser utilizados por outras aplicações.

Com a multiplicidade de aplicações existentes e as que vão sendo apresentadas de novo, vai sendo acumulada informação referente a diversos aspectos de uma mesma rede em diversos formatos e torna-se cada vez mais difícil o processo de gestão da rede e da informação e a integração e partilha de informação obtida pelas diferentes aplicações.

A existência de informação de rede replicada em diversos formatos e por diversas aplicações implica também um aumento dos recursos de rede dispendidos com tarefas de actualização ou verificação da informação existente por parte das diversas aplicações.

## *1.3 ABORDAGEM*

A solução para o problema apresentado passa pela uniformização do formato utilizado para guardar a informação obtida no processo de descoberta de rede e de serviços pelas diversas aplicações.

Se a informação for disponibilizada num formato conhecido, seria possível utilizar a informação obtida por um programa que seja bastante eficiente no processo de descoberta por outro que seja mais eficiente noutros processos (p. ex. a monitorização). Seria ainda possível disponibilizar a informação obtida num local público para que outros tivessem acesso a ela via Internet e dessa forma partilhar parte do conhecimento da estrutura e facilitar processos de monitorização que envolvessem mais que um domínio de gestão, i. e., redes pertencentes a entidades diferentes.

A forma de armazenamento da informação presente em mapas de descoberta proposta e abordada neste trabalho recai na utilização de um serviço de directório LDAP[I] [1]. A escolha desta solução prende-se com diversos aspectos, sendo eles enumerados de seguida:

- O LDAP é um standard já amadurecido e com provas dadas no que toca à fiabilidade, segurança e escalabilidade;
- A capacidade de um directório LDAP poder conter diversos atributos associados a um só nome (um objecto) e poder conter múltiplos nomes;
- Facilidade em criar um directório LDAP, importando todos os objectos a povoar o directório através de um ficheiro (LDIF[II] [2] ou DSML[III]);

.....
[I] LDAP – Lightweight Directory Access Protocol
[II] LDIF – Lightweight Directory Interchange Format
[III] DSML – Directory Service Mark-up Language

- Possibilidade de visualização e actualização de todo o directório utilizando os exploradores (*browsers*) existentes para o efeito;
- Capacidade de acréscimo de mais informação a um determinado nome (objecto) de forma a valorizar mais e completar a informação obtida pelos programas de descoberta;
- Fácil partilha da informação e interligação com outras aplicações
- Partilha selectiva de informação, de acordo com uma política de utilizadores previamente definida;

## 1.4 ESTRUTURA

Este documento está dividido em seis capítulos em que este é o primeiro e tem como objectivo apresentar o trabalho elaborado.

O segundo capítulo trata do processo de descoberta, dando uma visão sobre a forma como é feito, quais são as entidades (empresas) que neste momento estão na vanguarda desta tecnologia, qual a performance das aplicações escolhidas e que serviram de base prática e prova de conceito a este trabalho e o seu modelo de dados.

O terceiro capítulo aborda a forma de se realizar o mapeamento da informação disponível nos mapas de descoberta no serviço de directório LDAP, as dificuldades no processo de mapeamento e o modelo de dados adoptado.

O quarto capítulo foca as vantagens de se ter um suporte normalizado para a informação presente nos mapas de descoberta, apresentando dois casos em que se confirma esta vantagem.

O quinto capítulo descreve o protótipo implementado que serve como prova de conceito ao trabalho apresentado e as suas diferentes fases de implementação e resultados.

O sexto capítulo contém as conclusões do presente trabalho.

# 2 DESCOBERTA

Actualmente as topologias de rede variam com bastante frequência tornando a sua gestão manual bastante difícil e complexa. É neste contexto que se recorre à de descoberta dos elementos e serviços de uma rede.

A descoberta é o processo pelo qual se torna possível saber de forma automática que máquinas e serviços se encontram disponíveis num determinado domínio e o respectivo espaço de endereços IP[IV] [3], bem como a topologia de rede.

As diversas aplicações utilizadas nos nossos dias possuem formas diferentes de realizar a descoberta, com maior ou menor grau de eficiência. Estas aplicações são tanto do universo do código livre como de cariz comercial.

## *2.1 COMO SE FAZ*

Existem diversas formas de descobrir que nós estão activos num domínio ou numa rede, serviços disponibilizados por esses mesmos nós e a topologia de rede [4,5,6].

Há, no entanto, prós e contras nos métodos actualmente utilizados que serão apresentados de forma sucinta nos parágrafos seguintes, bem como uma descrição do seu funcionamento.

### 2.1.1 PING

O ping [7] é uma ferramenta ou aplicação que permite verificar se uma máquina com um determinado IP está ou não activa na rede, utilizando pacotes `IP/ICMP`[V] [8] `ECHO_REQUEST` e `ECHO_REPLY` para questionar a máquina e para a resposta, respectivamente. A conectividade é assim verificada até ao nível 3 do modelo OSI[VI] [9,10,11].

Como os pacotes se podem perder ou não chegar em tempo útil para serem considerados uma resposta válida (existe um intervalo de tempo em que o pacote da resposta tem que chegar para ser validado), são normalmente enviados vários pacotes

.....
[IV] IP – Internet Protocol
[V] ICMP – Internet Message Control Protocol
[VI] OSI – Open Systems Interconnect

para testar a conectividade. Uma máquina é considerada inatingível ou inactiva se todos os pacotes enviados para ela não obtiverem resposta.

### 2.1.2 BROADCAST PING

É semelhante ao método anterior e utiliza a mesma ferramenta (o ping) tendo apenas como diferença o endereço IP de destino utilizado. Neste caso, o endereço de destino vai ser o endereço de difusão (*broadcast*) da rede. Desta forma, o pacote de ping é dirigido a todas as máquinas que estão no domínio de difusão de uma rede e todas elas devem responder ao ping.

Este método não é no entanto suportado em todas as redes. Em alguns casos apenas responde a este tipo de ping a máquina que conecta a rede de destino do ping com as outras redes (o *gateway*). Noutros casos e por razões de segurança, o ping de difusão não funciona de todo.

### 2.1.3 TABELAS DE ARP[VII]

A consulta às tabelas de ARP de um *gateway* pode revelar que máquinas estão activas numa determinada rede e que já comunicaram com o *gateway*. As tabelas de ARP contêm os endereços MAC[VIII] e respectivos endereços IP das máquinas que recentemente enviaram pacotes para o *gateway* para que ele os encaminhasse para as respectivas redes de destino.

Como as entradas numa tabela de ARP não permanecem lá indefinidamente (o tempo de permanência é da ordem de alguns minutos), é necessário forçar todas as máquinas da rede a comunicar com o *gateway* e imediatamente de seguida ler a tabela de ARP do router.

Isto é conseguido enviando de uma rede diferente e para a rede que se quer descobrir um *broadcast* ping. Desta forma, as máquinas irão responder à máquina que enviou o pacote de ping e a resposta passará pelo *gateway* a consultar, que colocará na sua tabela de ARP os endereços das máquinas que responderem.

Podem no entanto existir constrangimentos relacionados com o *broadcast* ping (já mencionados no tópico anterior) e caso não se utilize este para contornar os possíveis

.....
[VII] ARP – Address Resolution Protocol
[VIII] MAC – Media Access Control

problemas podem-se não obter os endereços IP de todas as máquinas presentes na rede local.

### 2.1.4 TRACEROUTE

O traceroute permite que se descubra qual a rota (caminho) que é tomado para que um pacote IP vá desde a sua origem até ao destino [7]. Este resultado é conseguido incrementando gradualmente o campo TTL[IX] do pacote ICMP. As máquinas que se encontram ao longo do caminho vão verificando que receberam um pacote com um TTL igual a zero vão responder com um pacote `ICMP TIME_EXCEEDED`.

É por vezes difícil de descobrir o caminho com este método, visto as máquinas que fazem parte deste caminho podem deliberadamente não responder a este tipo de pacote por razões de segurança. A acrescentar há ainda a questão do *overhead* do pacote, que é superior ao do ping, e a demora do processo que é também considerável, para evitar a sobrecarga de processamento nas máquinas existentes no caminho.

### 2.1.5 SNMP[X]

Se as máquinas sobre as quais queremos obter informação suportarem e tiverem configurados agentes de SNMP, é possível saber uma grande quantidade de informação sobre elas, para além de se realizar a sua descoberta [12].

O agente de SNMP responde aos diversos pedidos autenticando o requerente recorrendo a uma palavra-chave (*community*) e recolhe a informação necessária da máquina onde está instalado, enviando-a para o requerente. Pode também enviar informação que não foi requisitada (como por exemplo alarmes) se para isso for configurado (envio de *traps*).

No entanto é um método com algumas fragilidades de segurança que podem permitir um acesso rápido a informação importante e em grande quantidade. Revela também pouca flexibilidade visto não poder ser facilmente alterado para fazer frente a possíveis falhas na sua implementação.

.....
[IX] TTL – Time To Live
[X] SNMP – Simple Network Management Protocol

### 2.1.6 VARRIMENTO DE PORTOS

O varrimento de portos (*port scan*) é uma técnica que permite descobrir de uma forma rápida quais os portos e respectivos protocolos disponíveis numa máquina. Isto é conseguido tentando estabelecer uma conexão para a máquina remota e no porto de protocolo que se deseja verificar. Se a máquina aceitar a conexão, então esse protocolo está activo na máquina remota.

Este método permite, além de descobrir quais os portos activos, atestar o estado da máquina remota (activa ou inactiva). Caso a máquina remota aceite uma conexão num qualquer porto, ela está activa. Isto é particularmente interessante nos casos em que os ISP's[XI] ou as próprias máquinas remotas não permitem utilizar as técnicas anteriores.

Este método pode tornar-se moroso consoante o número de portos activos que se queiram descobrir. Quanto maior o número de portos que se tente descobrir, maior será o tempo gasto a inquirir uma máquina, podendo esse tempo escalar para a ordem dos minutos.

### 2.1.7 TRANSFERÊNCIA DE ZONA DO DNS[XII]

Para uma maior facilidade de administração, as máquinas têm usualmente associado ao seu endereço IP um nome único. Este nome torna mais fácil para os gestores das redes a tarefa de gestão de rede, visto um nome ser para os humanos mais fácil de lidar/decorar que um endereço IP.

Quer se trate de uma máquina na Internet, quer seja uma máquina numa rede privada, a associação do seu nome ao endereço IP é gerida por um servidor de nomes (DNS Server) [13]. No caso de uma rede privada, este servidor não requer conectividade com outros servidores da Internet ao passo que um servidor que esteja a gerir nomes de máquinas visíveis da Internet terá que estar em comunicação e em sincronia com outros servidores de DNS presentes na Internet.

Ambos estes tipos de servidores respondem usualmente a um comando de transferência de zona com uma lista de todas as máquinas no seu domínio (administradas por ele). Desta forma pode-se obter rapidamente o nome e endereço IP das máquinas do domínio, com um baixo *overhead* (uma só pergunta retorna todas as máquinas).

.....
[XI] ISP – Internet Service Provider
[XII] DNS – Domain Name Service

É de mencionar que nem todos os servidores de DNS se comportam desta maneira. Por razões de segurança, existem servidores de DNS que não autorizam a transferência de zona. Chama-se também a atenção para o facto de existem máquinas cujos endereços IP não estão presentes nos servidores de DNS, que são as máquinas com endereços atribuídos por DHCP[XIII] [14].

**Tabela 1 – Prós e contras dos métodos de descoberta**

| Método | Prós | Contras |
|---|---|---|
| **Ping** | - Pacotes pequenos<br>- Pouco *overhead*<br>- Pings para máquinas activas são rápidos | - É necessário mais que um pacote<br>- Pings para máquinas inactivas é bastante lento<br>- Pode ser barrado pelos operadores (ISP's) ou pelas máquinas |
| **Broadcast Ping** | - Pacotes pequenos<br>- Pouco *overhead*<br>- Pings para máquinas activas são rápidos<br>- Um pacote pode descobrir várias máquinas | - Pouco suportado<br>- Pode ser barrado pelos operadores (ISP's) ou pelas máquinas |
| **Tabelas de ARP** | - Pouco *overhead*<br>- Podem ser descobertas várias máquinas | - Podem-se não obter toda as máquinas |
| **Traceroute** | - Traça o caminho entre a fonte e o destino | - *Overhead* elevado<br>- Pode ser barrado pelos operadores (ISP's) ou pelas máquinas<br>- É demorado |
| **SNMP** | - Obtém-se uma grande quantidade de informação<br>- Envio de *traps*<br>- Não necessita de ter as máquinas visíveis ao ICMP | - Necessita de ser suportado nas máquinas a inquirir<br>- Possíveis falhas de segurança<br>- Pouco flexível |
| **Varrimento de Portos** | - Tem a maior capacidade de descoberta<br>- Descobre serviços activos e informações relacionadas | - Lento para descobrir um elevado número de portos |
| **Transferência de Zona do DNS** | - Rápido<br>- Baixo *overhead* | - Não disponível em todos os servidores<br>- IP's atribuídos por DHCP não são descobertos |

Existem ainda outros métodos que são híbridos implementados com base em vários destes métodos e também métodos que recorrem a implementações de protocolos proprietários. Entre eles encontra-se o método de descoberta utilizado pela HP[XIV] na sua ferramenta OpenView Network Node Manager, que recorre a algoritmos proprietários que minimizam o tempo de descoberta dos métodos SNMP e ping. De igual forma, também o Scotty/Tkined utiliza os métodos de descoberta SNMP e ping.

.....
[XIII] DHCP – Dynamic Host Configuration Protocol
[XIV] HP – Hewlett-Packard

## 2.2 QUEM FAZ

Existem diversas empresas que desenvolveram *software* de descoberta e gestão de redes, sendo algumas aplicações vocacionadas para LAN's[XV] ou redes com um número de máquinas reduzido e outras para situações em que existe um grande número de máquinas e redes. Há também *software* de código livre (*open-source* [15]) criado com os mesmos objectivos.

Como o objectivo deste trabalho incide sobre os mapas de descoberta, vão ser discutidas apenas as ferramentas capazes de realizar a descoberta de rede e seus elementos e a descoberta da topologia de rede.

Das ferramentas mais utilizadas e mais capazes destas tarefas foi escolhido o HPOV NNM[XVI] [16,17] e o Scotty/Tkined [18,19,20]. O HPOV NNM foi escolhido por ser a ferramenta comercial mais capaz e utilizada e o Scotty/Tkined foi escolhido por ser uma ferramenta de código livre, gratuita e capaz de cumprir com os objectivos do trabalho. Serão estas duas ferramentas que serão analisadas em seguida e comparadas na sua performance e modelo de dados.

Existem mais ferramentas capazes de realizar a descoberta de máquinas e redes (algumas descobrem também serviços), criar os mapas de descoberta e monitorizar, principalmente de cariz comercial e para ambientes Windows como o Solarwinds [21], o What's UP [22], o SNMPc [23], o LAN Surveyor [24] ou o ManageEngine OpManager [25]. No universo das aplicações de código livre existe o OpenNMS [26] e o Cheops-ng [27] mas o primeiro não cria mapas de descoberta com a topologia das redes e máquinas descobertas e o segundo não suporta monitorização.

## 2.3 PERFORMANCE

Mais do que uma boa apresentação e facilidade de utilização, as ferramentas de descoberta e mapeamento devem ser rápidas precisamente a descobrir máquinas e redes. Um utilizador não deve ter que esperar muito tempo para ter as suas máquinas e redes prontas a gerir.

Neste campo entram os algoritmos proprietários que tratam de fazer o melhor uso possível das ferramentas utilizadas para descoberta e optimizam o processo de descoberta com os parâmetros fornecidos.

.....
[XV] LAN – Local Area Network
[XVI] HPOV NNM – HP OpenView Network Node Manager

A rapidez de descoberta está também associada à configuração do programa utilizado, fornecendo ele diversos parâmetros que podem ser afinados e que vão assim permitindo um melhor desempenho dos algoritmos de descoberta.

Tradicionalmente, esses parâmetros prendem-se com quais os métodos a utilizar para a descoberta (ping, SNMP ou conexão a um porto TCP[XVII] [28]), se vão ser utilizados mais do que um caso o método preferido falhe e quais os tempos de espera para declarar uma máquina inactiva.

A performance está também bastante ligada à forma como é armazenada a informação obtida durante a descoberta. Dependendo se é armazenada em ficheiros de texto, BD[XVIII] ou em memória, a rapidez na pesquisa da informação armazenada pode variar bastante e sendo essa diferença tanto mais notória quanto maior for o número de elementos de rede guardados.

Outro aspecto que altera a rapidez de descoberta é a quantidade de elementos existentes numa rede e o número de redes interligadas e que se querem descobrir, sendo o tempo de descoberta necessário para gerar os mapas de descoberta seja tanto maior quanto maior seja o número de máquinas por rede e o número de redes.

A topologia das diversas redes a descobrir, a LB[XIX] das interligações entre elas, a sua estabilidade e a capacidade de processamento dos *gateways* que interligam as diversas redes podem também tornar a descoberta mais demorada. Topologias em estrela favorecem o processo de *discovery*, bem como interligações de maior LB entre redes e a presença de *gateways* com elevadas capacidades de processamento de pacotes a interligar as diversas redes.

Capazes de influenciar a performance são ainda os recursos da máquina onde está alojado o programa que irá realizar a descoberta e onde serão guardados os respectivos mapas de descoberta. Os recursos a ter em conta é a capacidade de processamento da máquina, o seu espaço em disco e a sua memória.

Essa máquina deve ter os recursos dimensionados de acordo com as recomendações fornecidas com cada aplicação para que não se torne um elemento estrangulador do desempenho da solução adoptada, quer seja ela com o HPOV NNM ou com o Scotty/Tkined.

.....
[XVII] TCP – Transmission Control Protocol
[XVIII] BD – Base de Dados
[XIX] LB – Largura de Banda

### 2.3.1 NETWORK NODE MANAGER

O HPOV NNM utiliza como métodos de descoberta o ping, as tabelas de ARP e o SNMP. Associado a estes métodos existem vários parâmetros configuráveis que permitem ajustar a eficiência do discovery tais como a frequência com que são enviados os pacotes de *discovery*, o seu número, o tempo necessário para considerar que um pacote foi perdido ou quantos pacotes são necessários para considerar uma máquina activa.

Para maximizar a descoberta com um menor número de pacotes, são empregues algoritmos proprietários que permitem verificar situações potencialmente causadoras de atrasos como uma máquina ter mais que um endereço, mais que um interface ou pertencer a mais que uma rede.

De forma a contornar o problema de interligações de baixa LB e *gateways* de menor capacidade de processamento, o intervalo de tempo utilizado por defeito entre o envio de cada um dos pacotes de *discovery* é já adequado à maioria das redes que se podem encontrar.

Para permitir uma grande quantidade de informação guardada de forma a ser acedida sem perdas de performance, o HPOV NNM pode utilizar uma BD interna ou externa. Esta forma de armazenamento é bastante escalável e traz melhorias significativas de desempenho tanto maiores quanto maior for a quantidade de informação utilizada nos mapas de descoberta.

Em termos de recursos necessários da máquina onde vai residir o HPOV NNM, no caso de uma plataforma Windows ela terá que ter um mínimo de 100MB de memória livres, 550MB de espaço em disco e um espaço em *swap* recomendado de 512MB. A capacidade de processamento recomendada para mapas de descoberta até aproximadamente 2000 elementos é a de um processador Pentium 4 a 3,2GHz [29].

O resultado final dos métodos, técnicas e tecnologias adoptadas pelo HPOV NNM traduz-se numa capacidade de descoberta de aproximadamente 1000 elementos de rede (ou nós) em 90 segundos na maioria das plataformas e desde que estas respeitem os requisitos mínimos em termos de recursos recomendados pela HP.

### 2.3.2 SCOTTY/TKINED

Os métodos utilizados pelo Scotty/Tkined para realizar a descoberta dos elementos de rede e redes são o ping, tabelas de ARP e SNMP. Existem parâmetros de

descoberta que podem ser ajustados para permitir uma maior performance como o número de pacotes de ICMP enviados para verificar se uma máquina está activa, o intervalo de tempo entre cada pacote e o tempo que leva a considerar inválido um pacote sem resposta.

A eficiência do processo de descoberta está também dependente de alguns factores já mencionados anteriormente no caso do HPOV NNM, tais como as topologias de rede, LB das interligações entre redes e a capacidade de processamento dos *gateways* que interligam as diversas redes. As topologias em estrela favorecem o processo de descoberta, bem como as interligações entre redes com grandes LB e *gateways* com elevada capacidade de processamento de pacotes.

No caso do Scotty/Tkined os resultados da descoberta são armazenados num único ficheiro de texto, que pode ser visualizado e editado manualmente. Embora isto possa trazer algumas vantagens no que toca a partilhar a informação de forma fácil, não é escalável e o acesso à informação guardada no ficheiro vai sendo cada vez mais lento à medida que a quantidade de informação guardada no ficheiro for aumentando.

No que toca aos requisitos necessários relacionados com os recursos físicos (capacidade de processamento, espaço em disco e memória) para ter o Scotty/Tkined a funcionar numa máquina, eles são inferiores aos necessários para o HPOV NNM (apenas 20MB de memória livre e 15MB de espaço em disco numa plataforma Windows), o que torna possível instalar esta aplicação em máquinas com poucos recursos disponíveis. Em relação à capacidade de processamento um processador Intel Pentium 3 a 400 MHz será suficiente para os mapas de descoberta do Scotty/Tkined.

## 2.4 MODELO DE DADOS

Após obtenção dos dados necessários a gerir uma rede e a criação do seu mapa, é necessário que eles sejam guardados segundo uma estrutura específica que seja percebida pelo programa de gestão. As soluções podem várias, quer em relação à estrutura adoptada, quer em relação à forma de armazenamento dos dados obtidos.

Em relação aos programas analisados e experimentados em laboratório, os autores das respectivas aplicações optaram por soluções diferentes no que toca à estrutura adoptada para armazenar os dados mas ambos escolheram o armazenamento dos mesmos em disco.

O armazenamento de dados em disco vai permitir a ambos terem um histórico da informação obtida, podendo o utilizador guardar o estado da rede para futuras verificações, mesmo que o programa ou a máquina onde reside o mesmo sejam reiniciados.

### 2.4.1 NETWORK NODE MANAGER

O modelo de dados do HPOV NNM utiliza 5 BD's [16,30], sendo duas delas relacionadas com o programa em si e o seu funcionamento e o posicionamento dos objectos nos mapas gráficos e as outras três contém a informação da rede que pode fornecer informação acerca de problemas da rede, a sua frequência, utilização dos elementos de rede e as tendências da performance de uma rede. São elas a BD de Topologia, a BD de Tendências e a BD de Eventos.

Na BD de Topologia pode ser encontrada a informação referente a todos os nós (máquinas) e respectivos interfaces presentes na rede. Nesta BD é possível encontrar informação como os nomes dos nós e respectivos endereços IP.

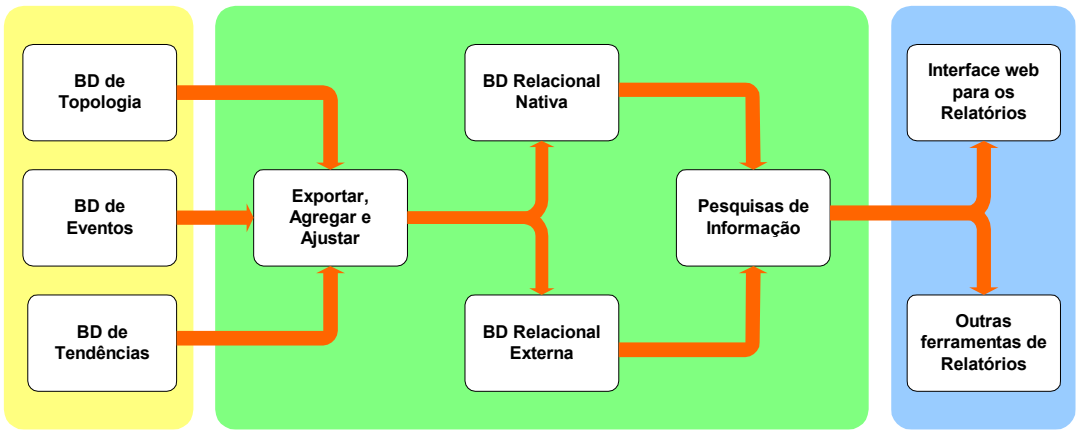
Na BD de Eventos pode ser encontrada a informação referente aos alarmes e acontecimentos importantes em no que toca aos nós e à rede e estados associados às informações apresentadas.

Na BD de Tendências pode ser encontrada toda a informação que é recolhida pelo SNMP e é depois utilizada por diversas facilidades existente no HPOV NNM.

Cada uma destas BD's tem por sua vez vários objectos definidos e cada um dos objectos tem diversos atributos. Poderá dizer-se que a unidade básica de cada uma destas BD's é o objecto, que por sua vez pode ser caracterizado pelos atributos definidos para ele. Estas três BD's podem ser agrupadas no que a HP denomina de *data warehouse*, que não é mais que um repositório de informação relacionada com os objectos geridos.

Esta *data warehouse* pode utilizar a BD relacional interna do HPOV NNM ou em alternativa podem ser utilizadas BD's externas sendo suportadas BD Oracle (para servidores Unix ou Windows) e SQL (apenas para servidores Windows). No trabalho elaborado no laboratório foi utilizada a BD interna do HPOV NNM.

**Figura 1 – Relação entre a *data warehouse* e as BD's de Tendências, Topologia e Eventos**



Para melhor entender a relação entre as diferentes BD's e a *data warehouse*, é apresentada uma figura ilustrativa da respectiva relação na Figura 1.

A BD interna é bastante robusta e dinâmica, crescendo à medida que os dados vão sendo inseridos. Sendo relacional, as pesquisas são também mais fáceis. Esta é a BD recomendada pela HP e foi também a BD utilizada no ambiente de testes montado.

Ter uma BD destas como suporte à informação encontrada oferece vantagens sobre outros métodos de armazenamento de dados. A forma de armazenamento em ficheiros de texto adoptada por alguns fabricantes, embora mais fácil de implementar e mais fácil para um utilizador ter acesso e manipular, revela-se bastante ineficiente quando o número de objectos começa a crescer.

A questão da segurança é outro ponto a não descurar e com uma BD deste tipo é possível proteger a informação, visto se poderem definir permissões para realizar algumas alterações.

Para se ter acesso a toda a informação colocada na *data warehouse* é utilizado SQL[xx], que é uma ferramenta e uma linguagem para interagir com BD's relacionais. Ao ser usado SQL vai ser possível aos utilizadores e administradores da BD terem um conjunto de comandos para interagir com a BD e ser distribuída a informação por outras máquinas e servidores.

Além da facilidade oferecida de se poderem escrever *scripts* de pesquisa em SQL, o HPOV NNM disponibiliza também uma interface *web* onde é possível verificar os relatórios que são gerados automaticamente. Com esta facilidade, é o HPOV NNM que pesquisa a *data warehouse* e os apresenta ao utilizador.

### 2.4.2 SCOTTY/TKINED

O modelo de dados deste programa é bastante simples, não utilizando nenhuma BD para armazenar a informação obtida como o HPOV NNM. Tudo o que é descoberto sobre a rede e as funcionalidades de monitorização adicionadas a qualquer máquina da mesma é guardado num ficheiro único em formato de texto.

Este ficheiro está por sua vez organizado em pequenos grupos, sendo cada um dos grupos um dos objectos do mapa de descoberta. Por sua vez, cada um destes objectos tem as suas propriedades definidas dentro desse grupo, propriedades como o endereço IP, o nome ou os serviços associados.

.....
[xx] SQL – Structured Query Language

Este modelo de dados torna mais fácil para o utilizador verificar e modificar o mapa de descoberta de uma forma imediata mas torna-se também mais vulnerável a falhas, bastando um problema num único ficheiro para que se perca toda a informação sobre a rede.

Este tipo de armazenamento de informação é também pouco escalável para um número elevado de elementos de rede, visto a estrutura não ter mecanismos de procura e indexação eficientes, tendo o ficheiro que ser totalmente criado a cada alteração que se seja feita num objecto do mapa de descoberta.

Um exemplo de alguns objectos do mapa de descoberta e da sua definição pode ser visto adiante (Tabela 2). O primeiro objecto é uma máquina sem quaisquer serviços associados. O segundo objecto define a ligação do primeiro a uma determinada rede. O terceiro objecto tem um serviço de monitorização associado (máquina com o endereço IP 169.254.36.129).

**Tabela 2 – Definição de um Objecto do Mapa de Descoberta do Scotty**

```
set node52 [ ined -noupdate create NODE ]
ined -noupdate move $node52 42.00 250.00
ined -noupdate icon $node52 node
ined -noupdate font $node52 fixed
ined -noupdate name $node52 bl5-126-179.dsl.telepac.pt
ined -noupdate address $node52 82.154.126.179
ined -noupdate label $node52 name

set link68 [ ined -noupdate create LINK $network0 $node52  ]

set stripchart1 [ ined -noupdate create STRIPCHART ]
ined -noupdate move $stripchart1 802.01 625.62
ined -noupdate font $stripchart1 fixed
ined -noupdate scale $stripchart1 100.00
ined -noupdate size $stripchart1
ined -noupdate name $stripchart1 pc700633
ined -noupdate address $stripchart1 169.254.36.129
ined -noupdate attribute $stripchart1 {round trip time} {rtt 1 ms}
ined -noupdate label $stripchart1 {round trip time}
```

Como se pode verificar, a informação necessária para desenhar o mapa de descoberta também se encontra neste ficheiro. No caso do HPOV NNM existe uma BD específica que guarda a informação necessária para desenhar o mapa de descoberta.

# 3 MAPEAMENTO DAS DESCOBERTAS EM LDAP

Apresentados os modelos de dados utilizados pelos programas que fizeram parte de protótipo e expostas as diferenças entre ambos, é necessário falar sobre o modelo de dados adoptado para o serviço de directório LDAP e os pontos fulcrais para o mapeamento da informação obtida dos mapas de descoberta do HPOV NNM no modelo de dados do serviço de directório LDAP.

Existem diferenças entre os dois modelos que são necessárias anular para que a informação possa representar e informar o seu utilizador da mesma forma, quer ela esteja representada de uma ou outra forma. Essas diferenças prendem-se sobretudo com a estrutura do serviço de directório LDAP ser hierárquica, ao passo que a estrutura do modelo de dados HPOV NNM é plana.

Outra questão a ter em conta é a inexistência de uma forma conhecida para retirar a informação contida na *data warehouse* do HPOV NNM e colocá-la no serviço de directório LDAP. Essa solução teve que ser pensada e implementada, sendo ela a criação de um *script* em Perl [31,32] que toma como argumento um ficheiro que é o *dump* de um dos objectos de uma das BD's e retorna um ficheiro com toda a informação do objecto passado como argumento. Esse ficheiro está num formato reconhecido e aceite pelo serviço de directório LDAP, podendo ser importado e assim capaz de colocar toda a informação do objecto em questão no serviço de directório.

## *3.1 MODELO LDAP*

O serviço de directório LDAP adopta uma estrutura hierárquica, à semelhança dos directórios num sistema de ficheiros de um computador. Como o modelo de dados do HPOV NNM é plano e composto por várias BD's, foram identificadas quais as BD's com informação relevante e que fosse importante partilhar através do serviço de directório LDAP. Essas BD's foram as de Topologia, Eventos e Tendências.

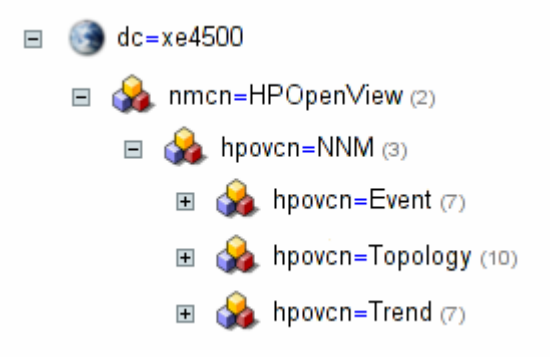
Nestas três BD's estão contidas as informações referentes às máquinas e redes encontradas, como estão dispostas e como se interligam, os acontecimentos que se deram nas máquinas e redes descobertas e informação mais detalhada sobre as máquinas e um histórico com informações obtidas delas.

Por sua vez, cada uma destas BD's tem vário objectos, cada um deles com os respectivos atributos. Como não existe mais informação para além dos atributos (são como o ficheiro no último directório de um sistema de ficheiros), a hierarquia ficava definida como BD, seguida de objecto e atributos. Acima da BD ficaria a estrutura

necessária para que o serviço de directório funcionasse correctamente e que cada objecto ou atributo pudesse ser identificado de forma unívoca dentro do serviço de directório.

Uma figura ilustrativa com a estrutura adoptada ara o serviço de directório LDAP pode ser observada de seguida (Figura 2). Esta figura contém apenas a estrutura até às BD's, sendo a dos objectos apresentada mais adiante.

**Figura 2 – Estrutura do serviço de directório LDAP até aos Objectos**




Na figura acima pode verificar-se a estrutura básica do serviço de directório LDAP que foi criado no laboratório para servir de protótipo. O nome da máquina onde foi instalado foi xe4500 e é o primeiro elemento da estrutura hierárquica do serviço de directório LDAP.

### 3.1.1 Topologia

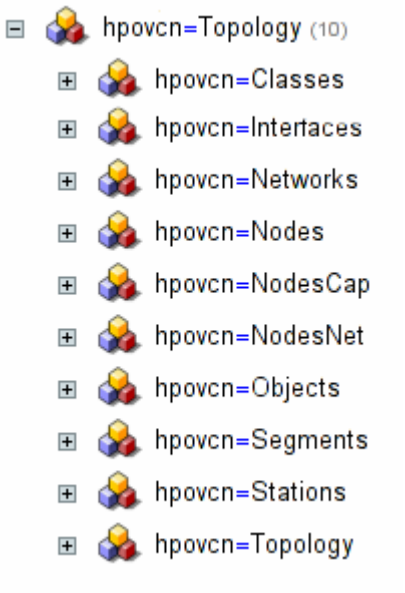
A BD de Topologia é onde o HPOV NNM guarda a informação referente aos nós, segmentos de rede, a topologia de rede e redes descobertas e é composta por 10 objectos diferentes. Cada objecto da BD vai ser mapeado numa entrada no serviço de directório LDAP e por sua vez cada objecto terá atributos que o caracterizarão.

A estrutura do serviço de directório do protótipo que foi adoptada e implementada para a topologia é apresentada na Figura 3.

Esta estrutura reflecte de forma fidedigna o que existe na BD de Topologia do HPOV NNM, tendo os objectos os mesmos nomes no serviço de directório LDAP e na BD. Isto permite que haja uma maior facilidade de automatização do mapeamento, futuras actualizações e que um utilizador possa facilmente utilizar uma e outra forma de armazenamento e reconhecer os objectos num e noutro lado.

Os atributos contidos em cada um dos objectos também foram todos mapeados e são possíveis de encontrar nos mesmos sítios quer na BD do HPOV NNM ou no serviço de directório.

**Figura 3 – Estrutura da BD deTopologia mapeada em LDAP**




### 3.1.2 Eventos

Todos os acontecimentos ou alterações de estado que se dêem nos elementos pertencentes a um mapa de descoberta e *traps* recebidos por SNMP são catalogados e registados na BD de Eventos.

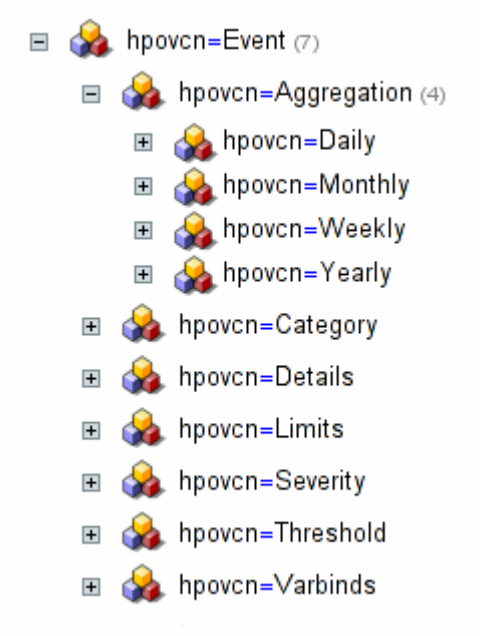
Esta BD contém também o estado da informação apresentada ao utilizador, visto este poder atribuir estados diferentes às informações apresentadas, como por exemplo verificada, por verificar ou a escalar/distribuir.

Toda a informação contida nesta BD é importante e por isso foi feito o seu mapeamento no serviço de directório LDAP. O mapeamento foi executado nos mesmos moldes do utilizado para a BD de Topologia.

Todos os objectos existentes na BD têm um objecto com o mesmo nome no serviço de directório sob a entrada correspondente aos eventos. Por sua vez, cada objecto vai ter todos os seus atributos iguais quer na BD do HPOV NNM, quer no serviço de directório LDAP.

O resultado final do mapeamento da BD de Eventos pode ser observado na Figura 4, que representa a estrutura criada no serviço de directório LDAP para os eventos.

**Figura 4 – Estrutura da BD de Eventos mapeada em LDAP**




### 3.1.3 TENDÊNCIAS

A BD de Tendências contém toda a informação recolhida de agentes SNMP, quando presentes nos nós que foram descobertos. Esta informação a retirar é a que está descrita na MIB[XXI] [33] do equipamento onde o agente está colocado.

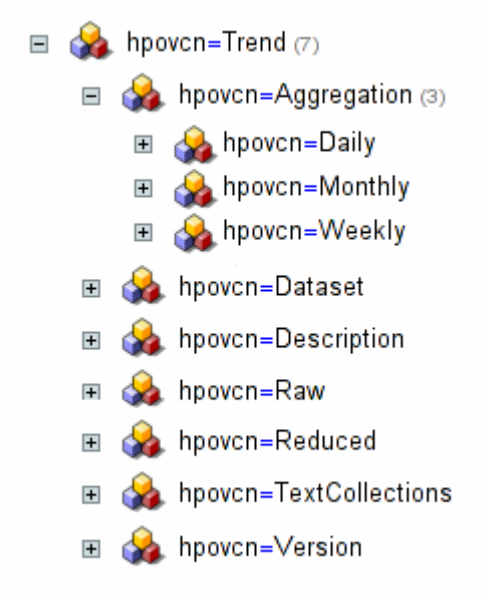
Esta BD contém também a informação referente aos limiares adoptados para fazer despoletar os alarmes. A importância dessa informação ficar guardada prende-se com o facto de que o operador pode personalizar quais os limiares acima dos quais deve ser enviado um alarme, sendo esses limiares diferentes para os diversos a que se aplicam. Com a possibilidade dessa informação ser guardada, ela irá ser mantida mesmo quando o HPOV NNM ou a máquina onde reside sejam reiniciados.

Novamente, após análise de todos os objectos e respectivos atributos, verificou-se que todos eles são relevantes e têm utilidade se partilhados e foram por isso mapeados no serviço de directório LDAP.

O resultado do mapeamento da BD de Tendências no serviço de directório LDAP é o que se pode observar na Figura 5.

[XXI] MIB – Management Information Base

**Figura 5 – Estrutura da BD de Tendências mapeada em LDAP**



# 4 VANTAGENS DE UM SUPORTE NORMALIZADO

Um suporte normalizado faz todo o sentido na gestão de redes actual. Neste momento, diversas aplicações tratam de áreas de relevo na gestão de redes e cada uma delas guarda a sua informação num formato proprietário e que muitas vezes não é legível, ou então a informação é guardada simplesmente em ficheiros de texto ou mesmo em memória e descartada no final da execução do programa.

Existindo uma mesma fonte para a informação, fonte essa capaz de conter todos os dados necessários às várias aplicações utilizadas para gerir uma rede, que fosse fiável, robusta e que seja ela própria já conhecida e tivesse formas de ser acedida fáceis, porque não utilizá-la?

Uma única fonte de informação que cumprisse os pressupostos apresentados reduzia o trabalho de administração de uma rede, evitava a duplicação de informação ou a sua perda e facilitava a partilha da informação e automatização de processos, podendo mesmo levar a um maior desenvolvimento de outras aplicações satélites que necessitassem da informação que estaria agora disponível para todos num formato conhecido. Seria ainda possível disponibilizar a informação obtida ou parte dela a outras entidades que necessitassem dela para monitorizar ou gerir serviços que dependem de várias redes para serem cumpridos.

Um exemplo seria a partilha de informação relativa aos mapas de descoberta de diversos ISP's. Se um número considerável de ISP's disponibilizasse desta forma a informação, haveria um conhecimento mais preciso e global da Internet e poderiam ser tomadas decisões de acordo com a informação existente de forma a garantir níveis de Qualidade de Serviço, detectar problemas de encaminhamento ou mesmo falhas de segurança e ataques.

Como já foi indicado anteriormente, a escolha para esta única fonte de informação recaiu sobre o LDAP. Em seguida vão ser apresentados de forma mais extensa dois casos em que a existência de um suporte normalizado faz todo o sentido.

## *4.1 NAGIOS*

O HPOV NNM é bastante eficaz na descoberta de nós e topologias de rede, bem como a administração desses mesmos nós (máquinas) descobertos.

No entanto, faz falta uma monitorização a nível dos serviços disponibilizados por essas mesmas máquinas e em relação ao seu estado.

Para cumprir essas funções existe um *software* de código livre dedicado exclusivamente a monitorizar serviços e o estado das máquinas (recursos das mesmas) que os suportam. Esse *software* é o Nagios.

Da mesma forma que o HPOV NNM monitoriza o estado da máquina em termos de conectividade e gera alarmes, registos e avisos diversos, o Nagios faz o mesmo mas em relação aos serviços e estado da máquina hospedeira.

Desta forma é possível ter um mapa de serviços com o estado actual de cada um deles, definir e tomar acções em caso de falha, gerar relatórios e alertas diversos.

O Nagios vai complementar desta forma o trabalho realizado pelo HPOV NNM, permitindo a quem realize a tarefa de gestão ter uma visão global da rede e nós a gerir.

Há no entanto um aspecto que dificulta a configuração e colocação em gestão dos serviços e recursos das máquinas a gerir, que é a incapacidade de realização da descoberta das máquinas e serviços e a colocação em monitorização das mesmas.

É aqui que a existência de um suporte normalizado como o LDAP se revela importante. Caso ele exista, basta ao Nagios ir buscar a informação armazenada no serviço de directório LDAP e de uma forma rápida e eficiente fica configurado.

Além destas vantagens há ainda outra que é ter a informação sobre as máquinas e serviços a gerir actualizada, evitando falsos alarmes devidos a serviços ou recursos que já não estão a ser geridos ou não existem.

## 4.2 AGENTES

Os agentes são entidades autónomas que são criados para recolhem informação da máquina onde residem ou do meio onde estão inseridos e enviá-la para uma entidade central que recolhe as informações de um ou mais agentes que esteja a gerir. Pode também enviar informação que não seja directamente solicitada, se para isso for configurado [34].

Para que um agente seja realmente autónomo, é necessário que ele possua toda a informação necessária à realização das suas tarefas e que ela seja actual.

Tomando como exemplo uma agente capaz de recolher informação sobre a configuração de equipamentos de rede [35], sendo eles de diversos fabricantes e com formas diferentes de obter essa configuração, como seria realizada essa tarefa pelo agente?

Para que fosse eficaz ele teria que ter antes da realização da sua tarefa uma lista com todos os equipamentos de rede que estivessem activos e aptos para a recolha da informação pelo agente e de que tipo de equipamento se trata, até mesmo o procedimento a adoptar para a recolha da informação.

Mas o estado dos equipamentos de rede pode variar, o seu endereço IP pode ser alterado, equipamentos diferentes podem ser trocados e manter um mesmo endereço IP ou o procedimento para obter a informação pode variar.

Se o agente for capaz de comunicar com um serviço LDAP, pode pesquisar no seu directório e obter toda a informação que necessita para realizar a sua tarefa, tornando-se mais autónomo.

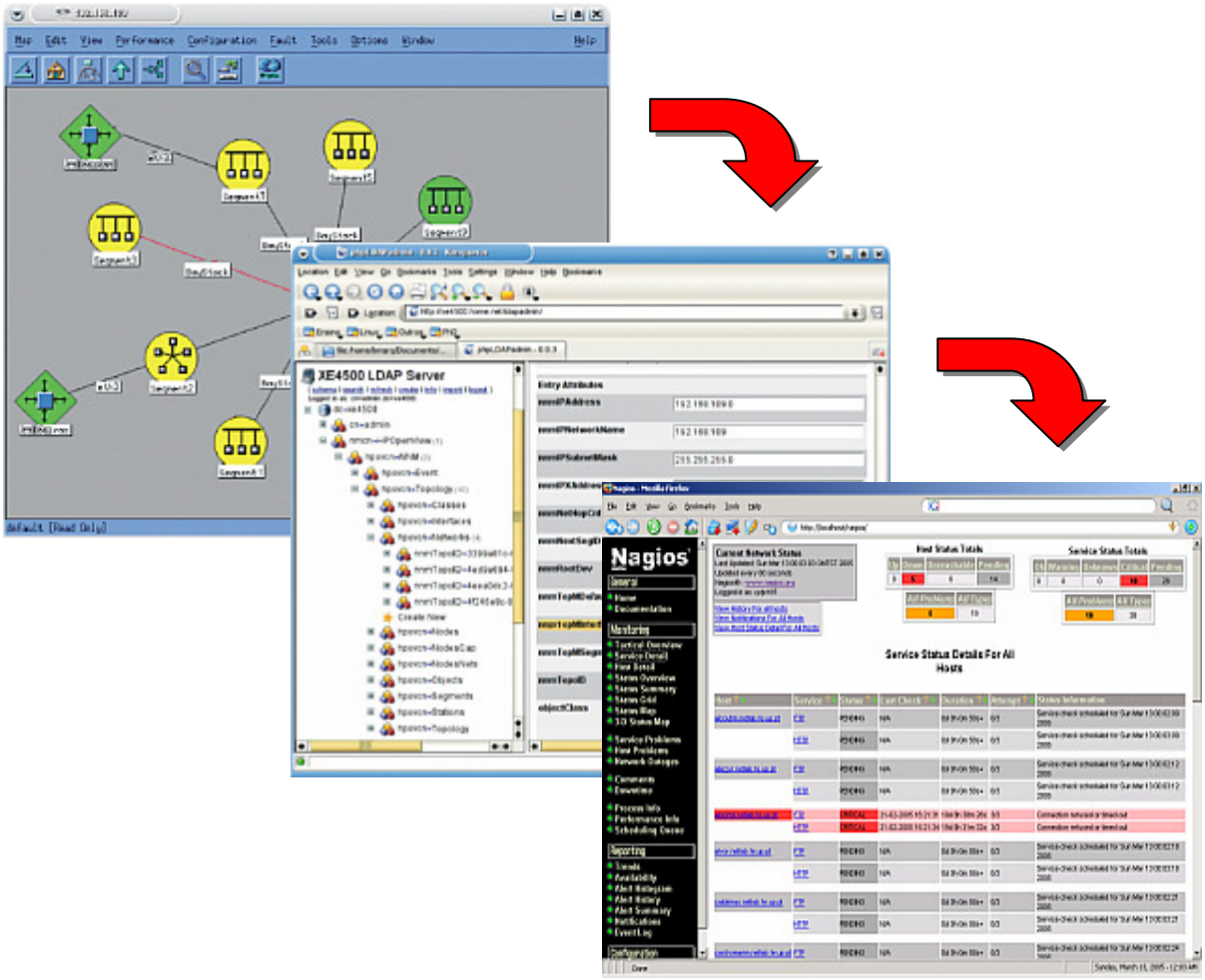
Um agente capaz de obter informação de um serviço de directório LDAP poderá mesmo mudar de tarefas ou o seu comportamento e funções consoante o que lhe seja indicado, tornando-se mais flexível e útil.

# 5 PROTÓTIPO

Como prova de conceito da escolha do LDAP como suporte normalizado, foi montado um pequeno ambiente de testes no laboratório do DEEC[XXII] da FEUP[XXIII]. Este ambiente consistia da rede do laboratório com as suas diversas máquinas, equipamentos de rede e endereçamentos de rede vários, um servidor com o HPOV NNM instalado, outro com um serviço de directório LDAP [36] e outro com uma aplicação capaz de realizar monitorização de serviços.

O objectivo de teste foi mostrar a possibilidade de utilização de um suporte normalizado para os mapas de descoberta obtidos pelo HPOV NNM, sendo esse suporte o LDAP, e, numa segunda fase, a partilha da informação dos mapas de descoberta agora num serviço de directório LDAP com outras aplicações.

**Figura 6 – Sequência de realização das diferentes fases de implementação do protótipo**




XXII DEEC – Departamento de Engenharia Electrotécnica e de Computadores
XXIII FEUP – Faculdade de Engenharia da Universidade do Porto

Neste caso, as outras aplicações foram representadas pelo Nagios [37], que é a aplicação escolhida para realizar a monitorização de serviços. Este objectivo e as fases necessárias para a sua implementação estão patentes na Figura 6.

No final foi ainda testada uma solução para dar a conhecer de forma simples qual o servidor de LDAP (o seu endereço IP) que terá a informação a disponibilizar. Isto foi conseguido recorrendo a um servidor de DNS e a um registo especial do serviço de DNS. Desta forma qualquer utilizador pode saber facilmente qual a máquina onde pode ir procurar a informação sobre os mapas de descoberta.

Para que isto fosse possível foi realizada numa primeira fase a descoberta das máquinas e equipamentos de rede do laboratório do DEEC, bem como as suas diferentes redes IP. Esta descoberta foi levada a cabo pelo HPOV NNM de uma forma automatizada.

Após descoberta de toda a topologia e nós do laboratório, era necessário que a informação guardada num formato proprietário do HOPV NNM fosse "traduzida" para poder ser colocada no serviço de directório LDAP. Para isso toda a informação contida na BD foi enviada para diversos ficheiros de texto, realizando *dumps* de todos os objectos contidos na mesma.

A BD estava agora repartida por vários ficheiros em formato de texto, com múltiplos campos. Cada ficheiro correspondia a um objecto da BD (Nós, Redes, Segmentos, Interfaces, …) que deveria ser mapeado e ter correspondência a um objecto no serviço de directório LDAP.

Este mapeamento para ser possível requeria uma estrutura do serviço de directório LDAP que correspondesse à estrutura delineada pelo HPOV NNM e depois, para inserir os dados no serviço de directório de uma forma simples e rápida, era necessário que a informação contida nos ficheiros dos objectos fosse formatada de forma a ser aceite pelo serviço de directório. Esta foi a segunda fase da implementação do protótipo.

A estrutura do serviço de directório foi definida de acordo com os objectos existentes no HPOV NNM, obtidos nos manuais do respectivo *software*. Com esse conhecimento foi criado um ficheiro *schema*, que deu corpo e definiu toda a estrutura do serviço de directório. Para definir os diversos atributos dos objectos a mapear foi criado um outro ficheiro *schema*. Ambos os ficheiros *schema* podem ser encontrado no Anexo A.

Depois de toda a estrutura do serviço de directório LDAP estar criada e os seus objectos e respectivos atributos definidos, foi então elaborada uma forma simples de criar os ficheiros LDIF necessários para colocar todos os objectos retirados do HPOV NNM.

A automatização e simplificação foi conseguida graças a três *scripts* feitos em Perl, que tomam como dados de entrada os ficheiros criados pelo *dump* do HOPV NNM e têm como dados de saída os ficheiros LDIF necessários para "povoar" o serviço de directório. Cada *script* trata dos dados obtidos de uma das BD do HOPV NNM (Eventos, Tendências e Topologia) e devolve o respectivo ficheiro LDIF. O código dos *scripts* para tratar os dados das BD's de Eventos, Tendências e Topologia encontram-se no Anexo B.

Com os ficheiros criados foi fácil importar toda a informação para o serviço de directório LDAP e ter mapeados todos os objectos e atributos da três BD's do HPOV NNM.

O passo seguinte e terceira fase do protótipo foi conseguir que o Nagios, que é uma aplicação que não tem possibilidades de realizar o *discovery*, tivesse acesso à informação colocada no serviço de directório LDAP e a utilizasse para realizar a monitorização de serviços disponibilizados por essas máquinas.

Para obter a informação necessária do serviço de directório LDAP, foi feita uma pesquisa nele com uma aplicação cliente de LDAP e a informação devolvida foi guardada num ficheiro. Como alternativa, o *script* poderia questionar o serviço de directório LDAP directamente e obter toda a informação necessária.

Esse ficheiro foi depois utilizado por um outro *script* (Anexo C) que criou o ficheiro necessário ao Nagios para que ele possa saber quais as máquinas com serviços a monitorizar.

Os serviços a monitorizar foram o tempo de resposta, o FTP[XXIV] e o HTTP[XXV], sendo esses serviços definidos para todas as máquinas por defeito. Caso o HPOV NNM descobrisse informação referente a serviços disponíveis nas máquinas que descobriu, ou em opção algum outro programa com capacidade de o fazer, colocasse a informação sobre os serviços disponíveis no serviço de directório LDAP, seria possível utilizar essa informação em vez de definir por defeito os mesmos serviços a monitorizar para todas as máquinas.

Com as três fases anteriores implementadas foi possível ter uma aplicação de gestão de serviços a monitorizar máquinas da rede de uma forma fácil e com a informação necessária à gestão dessas mesmas máquinas actual, sem que essa mesma aplicação tivesse que realizar a descoberta dessas máquinas e manter essa informação actual. Por sua vez, a informação necessária para realizar a gestão está por sua vez colocada num formato e local acessível a outras aplicações e sem necessidade de ser replicada (está no serviço de directório LDAP).

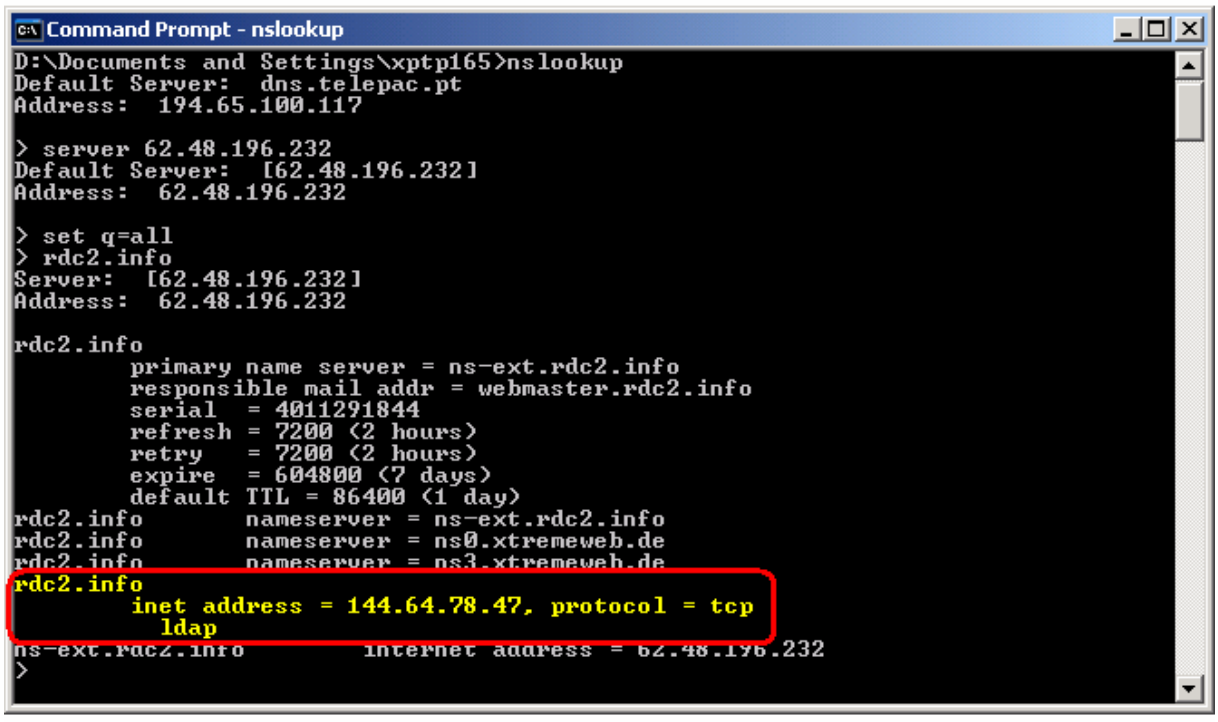
.....
[XXIV] FTP – File Transfer Protocol
[XXV] HTTP – Hypertext Transfer Protocol

Na fase seguinte foi testada a solução para se conseguir divulgar de forma fácil a informação sobre a localização do serviço de directório LDAP, i. e., o seu endereço IP. Foi para isso configurado um servidor de DNS onde um registo especial (*Record Well Known Services*) contém o IP, o protocolo utilizado e o porto de protocolo da máquina onde foi instalado o serviço de directório LDAP.

Após configuração do servidor de DNS, o endereço IP da máquina onde está instalado o serviço de directório LDAP para o domínio local passou a ser conhecido e qualquer aplicação ou utilizador podia agora saber qual era. Desta forma os mapas de descoberta que tinham sido transportados para o serviço de directório LDAP estavam agora acessíveis e eram fáceis de encontrar.

Um resultado de uma pesquisa para se verificar qual o endereço da máquina que disponibiliza o serviço de directório LDAP para um determinado domínio pode ser observado na Figura 7.

**Figura 7 – Pesquisa da máquina que suporta o serviço de directório LDAP de um domínio**




Como se pode verificar na Figura 7, a pesquisa foi feita a partir da Internet. O servidor onde foram realizados os testes está conectado à Internet e permite que lhe sejam feitos pedidos relacionados com o serviço de resolução de nomes. Desta forma conseguiu-se testar e demonstrar a possibilidade de divulgar de uma forma fácil e a todos os diferentes domínios existentes na Internet o endereço IP da máquina onde está o serviço de directório LDAP.

# 6 CONCLUSÕES

Cada vez mais as aplicações de gestão fazem parte integrante do conjunto de ferramentas essenciais para quem deseja gerir de forma eficiente máquinas e redes de uma organização. Essa necessidade é tanto maior quanto maior for também o parque de máquinas a gerir.

Para que os gestores de rede se possam concentrar apenas na gestão, as aplicações oferecem facilidades capazes de minimizar o esforço e tempo dispendidos na actualização da informação sobre máquinas e redes a gerir. Entre essas facilidades está a capacidade de descobrir e criar mapas de descoberta dos elementos presentes numa rede.

Com o número de aplicações necessárias ao processo de gestão de redes a aumentar, dá-se também um aumento do grau de dificuldade do processo de actualização de informação, o aumento de informação replicada e também o aumento de tráfego referente a processos de descoberta, verificação ou actualização de informações sobre os elementos de rede.

Urge assim a criação de um meio de armazenamento de toda a informação que é retirada da rede a gerir e colocada em diversos mapas de descoberta. Essa informação é depois guardada em formatos proprietários e torna-se dessa maneira inacessível a outras aplicações.

Esse meio de suporte à informação contida em mapas de descoberta deve ser capaz não só de servir de suporte à informação já obtida pelas aplicações na sua fase de descoberta mas também capaz de fornecer essa informação a outras aplicações incapazes de a obter.

O serviço de directório LDAP pode ser uma solução para estes problemas e foi a solução escolhida e abordada neste trabalho. A sua flexibilidade, escalabilidade e facilidade de administração aliadas ao facto de este ser já um standard e existir a possibilidade que divulgação de forma fácil e por toda a Internet do endereço IP da máquina onde o serviço está alojado, fazem dele uma ferramenta ideal para responder aos problemas já apontados e a novos problemas e desafios que possam surgir relacionados com a gestão de redes e a partilha da informação existente sobre a rede e os seus elementos.

# ANEXO A

Neste anexo podem ser encontrados os dois ficheiros que foram necessários para criar e definir a estrutura do serviço de directório LDAP.

### FICHEIRO MANIA_OBJECTS.SCHEMA

O código seguinte representa o ficheiro *schema* necessário para definir os objectos pertencentes ao serviço de directório LDAP.

```
# mania_objects.schema
#
# Schema for MANIA's Network Management Framework
# OIDs are owned by the MANIA Research Group
# MANIA IANA's OID Number: 19248
#
# Author: Bruno Filipe Marques <bmarq@elect.estv.ipv.pt>
# Created: 17 Jan, 2004
# Updated: 20 Fev, 2004
# Version: 0.0.1
#
# MANIA SINTAXES:
# 1.3.6.1.4.1.19248.x MANIA

#objectIdentifier IANAEnterprises 1.3.6.1.4.1
#objectIdentifier MANIA IANAEnterprises:19248

#ObjectIdentifier SNMPElements MANIA:1
#ObjectIdentifier LDAPElements MANIA:2

ObjectIdentifier LDAPElementsObjectClasses LDAPElements:2


#-------------------------------------------------------------------------------
# ------------ HP OpenView OBJECT CLASS DEFINITION

objectclass ( LDAPElementsObjectClasses:1
       NAME 'hpov'
       DESC 'HP OpenView'
       SUP top STRUCTURAL
       MUST ( hpovcn )
       MAY ( hpovDesc ) )


#-------------------------------------------------------------------------------
# ------------ HP OpenView Topology Data OBJECT CLASS DEFINITION
# HP OpenView NNM nnm_xxxx Topology Collected Data

objectclass ( LDAPElementsObjectClasses:2
       NAME 'nnm'
       DESC 'HP OpenView Network Node Manager Data Infrastructure'
       SUP top STRUCTURAL
       MUST ( hpovcn ) )

objectclass ( LDAPElementsObjectClasses:3
       NAME 'nnmObjects'
       DESC 'Contains the attributes common to all topology tables '
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmOVWID $ nnmRemoteID $ nnmStationID $ nnmClassID $
              nnmCTime $ nnmMTime $ nnmSymTime $ nnmStatTime $
              nnmLabelTime $ nnmFlagsTime $ nnmIPStatus $ nnmRemStatus $
```

```
              nnmFlags $ nnmExtra0 $ nnmExtra1 $ nnmCTimestamp $
              nnmMTimestamp $ nnmSymTimestamp $ nnmStatTimestamp $
              nnmLabelTimestamp $ nnmFlagsTimestamp ) )

objectclass ( LDAPElementsObjectClasses:4
       NAME 'nnmTopology'
       DESC 'Contains the global topology information'
       SUP top  STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmExpireDate $ nnmGwCnt $ nnmTopMNetworkCount
              $ nnmTopMSegmentCount $ nnmTopMNodeCount $ nnmTopMInterfaceCnt $
              nnmEventNum $ nnmStationCount $ nnmLicenseCount $
              nnmManageCount $ nnmExpireDatestamp) )

objectclass ( LDAPElementsObjectClasses:5
       NAME 'nnmNetworks'
       DESC 'Contains an entry for each network and subnet contained in the topology'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmIPNetworkName $ nnmIPAddress $
              nnmIPSubNetMask $ nnmTopMInterfaceCnt $ nnmTopMSegmentCount $
              nnmNextSegID $ nnmTopMDefaultSegID $ nnmRootDev $
              nnmIPXAddress $ nnmNetHopCnt ) )

objectclass ( LDAPElementsObjectClasses:6
       NAME 'nnmSegments'
       DESC 'Contains an  entry for each segment contained in the topology'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmSegmentName $  nnmNetID $ nnmRemNetID $
              nnmRemIFID $  nnmSegNetID $  nnmSegType $
              nnmTopMInterfaceCnt $  nnmHubIFID ) )

objectclass ( LDAPElementsObjectClasses:7
       NAME 'nnmNodes'
       DESC 'Contains an  entry for each node contained in the topology'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmIPHostname $ nnmTopMInterfaceCnt $
              nnmIPForwarding $ nnmSNMPState $ nnmAtCycleTime $
              nnmRootPort $ nnmVendor $ nnmAgent $ nnmSNMPSysDescr $
              nnmSNMPSysLocation $ nnmSNMPSysContact $ nnmSNMPSysObjID $
              nnmSNMPAddr $ nnmIPXAddress $ nnmIPXServerName $ nnmSysName $
              jpegPhoto $ labeledURI $ userid ) )

objectclass ( LDAPElementsObjectClasses:8
       NAME 'nnmInterfaces'
       DESC 'Includes all the interfaces contained in the topology'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmSegChangeTime $ nnmIPAddress $ nnmIFNumber $
              nnmNetID $ nnmSegID $ nnmNodeID $ nnmRemNetID $ nnmRemSegID $
              nnmRemNodeID $ nnmSegType $ nnmSNMPIFType $ nnmSegChangeTimestamp $
nnmSNMPIFName $ nnmIPSubnetMask $ nnmSNMPIFPhysAddr $ nnmSNMPIFDescr $ nnmSNMPState $
nnmLlaFrom $ nnmMaskFrom $ nnmPortClass $
              nnmIPXAddress $ nnmSNMPIFAlias ) )

objectclass ( LDAPElementsObjectClasses:9
       NAME 'nnmStations'
       DESC 'Contains an entry for each station contained in the topology'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmStationName $ nnmDbCreateTime $ nnmEventNum $
              nnmStationType $ nnmStationVersion $ nnmLicensedNodes $ nnmManagedNodes
$
              nnmAccessMode $ nnmOverlapMode $ nnmCheckInterval $ nnmLastUpdate $
nnmFilterChecksum $ nnmStationDesc $ nnmExpireDate $ nnmSNMPCommunity $
              nnmSNMPAuth1 $ nnmSNMPAuth2 $ nnmFilter ) )

objectclass ( LDAPElementsObjectClasses:10
       NAME 'nnmClasses'
       DESC 'Maps a class_name to  a class_id for use in referencing other objects
and determining the versions supported tables'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmClassID $ nnmClassName $ nnmVersion ) )
```

```
objectclass ( LDAPElementsObjectClasses:11
       NAME 'nnmNodesCap'
       DESC 'Contains characteristics about the node itself'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmVendorName $ nnmAgentName $ nnmIsComputer $ nnmIsPC $
              nnmIsWorkstation $ nnmIsMini $ nnmIsMainFrame $ nnmIsConnector $
              nnmIsBridge $ nnmIsRouter $ nnmIsIPRouter $ nnmIsIPXRouter $
              nnmIsRepeater $ nnmIsHub $ nnmIsSwitch $ nnmIsDevice $
              nnmIsPrinter $ nnmIsAnalyzer $ nnmIsIP $ nnmIsIPX $
              nnmIsSNMPSupported $ nnmIsSNMPProxied $ nnmIsNetwareServer $
              nnmIsMcClusterMember $ nnmIsCollectionStationNode $ nnmIsRDMISupported
$
              nnmIsHTTPSupported $ nnmIsHTTPManaged $ nnmIsRMON $ nnmIsRMON2 $
              nnmIsDS1 $ nnmIsDS3 $ nnmIsFrameRelay $ nnmIsSONET $
              nnmIsATM $ nnmIsCDP ) )

objectclass ( LDAPElementsObjectClasses:12
       NAME 'nnmNodesNets'
       DESC 'Provides a way to see all nodes on a network without going through
nnmObjects and nnmInterfaces'
       SUP top  STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmNodeName $ nnmNetIPAddress $ nnmShownEntries ) )


#-------------------------------------------------------------------------------
# ------------ HP OpenView Trend Data OBJECT CLASS DEFINITION
# HP OpenView NNM SNMP Collected Data

objectclass ( LDAPElementsObjectClasses:13
       NAME 'nnmTrendVersion'
       DESC 'Provides the version numbers to confirm that the schema did not change
version since the last time the tools were used'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTableName $ nnmVersion $ nnmLastTimestamp ) )

objectclass ( LDAPElementsObjectClasses:14
       NAME 'nnmTrendDesc'
       DESC 'Provides a description of the monitored MIB variables and MIB
expressions'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmDescriptID $ nnmMIBOID $ nnmUnits $ nnmTitle $
       nnmDescript $ nnmType $ nnmMIBExpr ) )

objectclass ( LDAPElementsObjectClasses:15
       NAME 'nnmTrendDataset'
       DESC 'Defines a trend dataset by identifying the target host, MIB variable or
expression, and instance'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTopoID $ nnmDescriptID $ nnmInstance $ nnmDatasetID $
              nnmHostName $ nnmServerName $ nnmTLastRawSec $
              nnmTLastRedRec $ nnmTitle $ nnmExportSet ) )

objectclass ( LDAPElementsObjectClasses:16
       NAME 'nnmRawTrend'
       DESC 'Strores raw data collected from the SNMP environment'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmDatasetID $ nnmSampleTime $ nnmSampleTimestamp $
              nnmPeriod $ nnmValue ) )

objectclass ( LDAPElementsObjectClasses:17
       NAME 'nnmReducedTrend'
       DESC 'Stores reduce data derived from raw SNMP data'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmDatasetID $ nnmSampleTime $ nnmSampleTimestamp $
              nnmPeriod $ nnmMinValue $ nnmMaxValue $
              nnmDataCount $ nnmDataSum $ nnmSumSqrs ) )

objectclass ( LDAPElementsObjectClasses:18
       NAME 'nnmTrendAggregation'
```

```
       DESC 'Stores aggregated daily/weekly/monthly trend data derived from raw SNMP
data'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmDatasetID $ nnmDescriptID $ nnmHostName $ nnmMIBOID $
              nnmInstance $ nnmSampleTime $ nnmWeek $ nnmMonth $
              nnmYear $ nnmMinValue $ nnmMaxValue $ nnmDataCount $
              nnmDataSum $ nnmDatapointCount $ nnmSumSqrs ) )

objectclass ( LDAPElementsObjectClasses:19
       NAME 'nnmTextCollections'
       DESC 'Provides text information without numerical analysis'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmDatasetID $ nnmSampleTime $ nnmSampleTimestamp $
              nnmPeriod $ nnmValue) )


#-------------------------------------------------------------------------
# ------------ HP OpenView Event Data OBJECT CLASS DEFINITION
# HP OpenView NNM Event Collected Data

objectclass ( LDAPElementsObjectClasses:20
       NAME 'nnmEventCat'
       DESC 'Pairs the category number with a text description of the event type'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmCategory $ nnmText ) )

objectclass ( LDAPElementsObjectClasses:21
       NAME 'nnmEventSev'
       DESC 'Pairs the severity code with a text description of the event severity'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmSeverity $ nnmText ) )

objectclass ( LDAPElementsObjectClasses:22
       NAME 'nnmEventDetail'
       DESC 'Contains the raw metadata for the event'
       SUP top STRUCTURAL
       MUST ( nnmEntryID)
       MAY ( nnmEventUUID $ nnmEventTimestamp $ nnmCategory $ nnmNodename $
nnmApplicationID $
              nnmMessage $ nnmSeverity $ nnmEventOID $ nnmOVObjectID $
              nnmProtocol $ nnmEventType $ nnmIPAddress $ nnmTrapsource $
              nnmTrapName $ nnmPID $ nnmForwardAddress $
              nnmEventSource $ nnmEventTime $ nnmNumberVarbinds ) )

objectclass ( LDAPElementsObjectClasses:23
       NAME 'nnmEventVarbinds'
       DESC 'Contains additional data associated with the event, as defined in the
trap.conf file for the specified event OID'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmEventUUID $ nnmVarbindSequence $ nnmEventOID $ nnmValLen $ nnmType $
              nnmTypeString $ nnmTypeInteger $ nnmObjID $ nnmUnsigned32 $
nnmUnsigned64 ) )

objectclass ( LDAPElementsObjectClasses:24
       NAME 'nnmEventThresh'
       DESC 'Contains the varbinds for the threshold event'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmEventUUID $ nnmEventOID $ nnmEventTimestamp $ nnmEventTime $
nnmApplicationID $
              nnmNodename $ nnmOVObjectID $ nnmMIBOID $ nnmCollectionName $
              nnmInstance $ nnmSeverity $ nnmThreshold $ nnmSample $
              nnmHightSample $ nnmHightTime $ nnmLowSample $ nnmLowTime $
              nnmThresholdOp $ nnmThresholdCount ) )

objectclass ( LDAPElementsObjectClasses:25
       NAME 'nnmEventAggregation'
       DESC 'Contains aggregated data for a specified period of time (daily, weekly,
monthly, and yearly). Thes four event aggregation objects contain the same fields'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
```

```
       MAY ( nnmNodename $ nnmEventOID $ nnmEventCnt $ nnmEventTimestamp $
nnmEventTime $
              nnmDay $ nnmWeek $ nnmMonth $ nnmYear ) )

objectclass ( LDAPElementsObjectClasses:26
       NAME 'nnmEventLimits'
       DESC 'Contains data about filters to be used to filter event types when
reducing the event data. Use ovdweventflt to add filters to this database'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmNodename $ nnmEventOID $ nnmAlwaysNever $ nnmLimit ) )

objectclass ( LDAPElementsObjectClasses:27
       NAME 'nnmDwData'
       DESC 'Olds operational data, such as timestamps for the control of data export
operations. During the previous event export operation, the timestamp for the last
event to be exportaded is saved in this object. The next event export operation
continues the export after this saved timestamp'
       SUP top STRUCTURAL
       MUST ( nnmEntryID )
       MAY ( nnmTableName $ nnmLastTimestamp $ nnmVersion ) )


#-------------------------------------------------------------------------------
# ------------ HP OpenView Event MetaData OBJECT CLASS DEFINITION
# HP OpenView NNM Event Metadata (for Alarm browsing)

objectclass ( LDAPElementsObjectClasses:28
       NAME 'nnmAlarm'
       DESC 'Olds meta-information for network events'
       SUP top STRUCTURAL
       MUST ( nnmAlarmID )
       MAY ( nnmAlarmCat $ nnmAlarmIMAPServer $ nnmAlarmIMAPAccount $ labeledURI $
jpegPhoto ) )
#      MAY ( nnmAlarmCat $ nnmAlarmIMAPServer $ nnmAlarmIMAPAccount $
nnmAlarmXMLschemaURL ) )


# eof mania_object.schema
```

### FICHEIRO MANIA_ATTRIBUTE.SCHEMA

O código seguinte representa o ficheiro *schema* necessário para definir os atributos dos objectos pertencentes ao serviço de directório LDAP.

```
# mania_attribute.schema
#
# Schema for MANIA's Network Management Framework
# OIDs are owned by the MANIA Research Group
# MANIA IANA's OID Number: 19248
#
# Author: Bruno Filipe Marques <bmarq@elect.estv.ipv.pt>
# Created: 17 Jan, 2004
# Updated: 16 Mar, 2004
# Version: 0.0.2
#
# MANIA SINTAXES:
# 1.3.6.1.4.1.19248.x MANIA

objectIdentifier IANAEnterprises 1.3.6.1.4.1
objectIdentifier MANIA IANAEnterprises:19248

ObjectIdentifier SNMPElements MANIA:1
ObjectIdentifier LDAPElements MANIA:2

ObjectIdentifier LDAPElementsAttributeTypes LDAPElements:1


#---------------------------------------------------------------------------
# ------------ MANIA ATTRIBUTE TYPE DEFINITION

#attributetype ( LDAPElementsAttributeTypes:1
#       NAME 'nmcn'
#       DESC 'Network Management Common Name'
#       EQUALITY caseIgnoreMatch
#       SUBSTR caseIgnoreSubstringsMatch
#       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{56}
#       SINGLE-VALUE )



#---------------------------------------------------------------------------
# ------------ HPOV ATTRIBUTE TYPES DEFINITION

attributetype ( LDAPElementsAttributeTypes:1
       NAME 'hpov'
       DESC 'HP OpenView Infrastructure'
       EQUALITY caseIgnoreMatch
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:2
       NAME 'hpovDesc'
       DESC 'HP OpenView Infrastructure Description'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:3
       NAME 'nnm'
       DESC 'HP OpenView Network Node Manager Infrastructure'
       EQUALITY caseIgnoreMatch
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:4
       NAME 'hpovcn'
       DESC 'HP OpenView NNM Common Name'
```

```
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:5
        NAME 'nnmTopoID'
        DESC 'Universal Unique Identifier (UUID) uniquely identifies each entry. Every
object has a unique UUID that can be used to match the entries in different tables'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:6
        NAME 'nnmOVWID'
        DESC 'OVW object ID  for the object. Multiple versions of the same type of
object may have the same OVW ID when a n object is monitored in a distributed
environment by multiple management stations'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:7
        NAME 'nnmRemoteID'
        DESC 'OVW object ID  for the object, as reported by the remote station
monitoring this version of the object. For locally-monitored objects, the nnmremoteid
is the same as the nnmovwid'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:8
        NAME 'nnmStationID'
        DESC 'The local OVW Object D of the station monitoring this version of the
object, or zero (0) for locally-managed objects'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:9
        NAME 'nnmClassID'
        DESC 'Identifies whether the object is a station, network, segment, node,
interface, or the global internet object. Eache type of object has a class entry in
the nnmclasses table'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:10
        NAME 'nnmCTime'
        DESC 'The creation time of the object (the first time the object was placed in
the database)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:11
        NAME 'nnmMTime'
        DESC 'The last modification time od the object'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:12
        NAME 'nnmSymTime'
        DESC 'The last time the object was modified in a way that would affect the
symbol for the object on a map'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
```

```
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:13
        NAME 'nnmStatTime'
        DESC 'The last time the status of the object changed'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:14
        NAME 'nnmLabelTime'
        DESC 'The last time the object was modified in a way that would affect the
label for the object on a map'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:15
        NAME 'nnmFlagsTime'
        DESC 'The last time the internal flag for the object changed'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:16
        NAME 'nnmIPStatus'
        DESC 'The IP-derived status of the object. The valid status values are the
same as those accepted by ovw (0: No status; 1: Unknown; 2: Normal; 3: Minor; 4:
Critical; 5:Unmanage; 6: Warning; 7: Major; 8: Restricted; 9: Testing; 10: Disabled)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{2}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:17
        NAME 'nnmRemStatus'
        DESC 'The IP status of the object, as reported by the station monitoring the
object (0: No status; 1: Unknown; 2: Normal; 3: Minor; 4: Critical; 5:Unmanage; 6:
Warning; 7: Major; 8: Restricted; 9: Testing; 10: Disabled)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{2}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:18
        NAME 'nnmFlags'
        DESC 'A field used to store internal-only, Boolean-valued flags'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:19
        NAME 'nnmExtra0'
        DESC '(Reserved for future use)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:20
        NAME 'nnmExtra1'
        DESC '(Reserved for future use)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:21
        NAME 'nnmCTimestamp'
        DESC 'The first time the object was stored in the topology database expressed
in Epoch time.'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:22
        NAME 'nnmMTimestamp'
        DESC 'The last time the object was stored in the topology database expressed
in Epoch time'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:23
        NAME 'nnmSymTimestamp'
        DESC 'The time of the last change affecting the objects status expressed in
Epoch time'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:24
        NAME 'nnmStatTimestamp'
        DESC 'The time of the last change affecting the objects status expressed in
Epoch time'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:25
        NAME 'nnmLabelTimestamp'
        DESC 'The time of the last change affecting the objects label expressed in
Epoch time'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:26
        NAME 'nnmFlagsTimestamp'
        DESC 'The time of the last change affecting the objects flag expressed in
Epoch time'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_topology Object Table Attribute Types

attributetype ( LDAPElementsAttributeTypes:27
        NAME 'nnmGwCnt'
        DESC 'The number of gateways in the topology'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:28
        NAME 'nnmTopMNetworkCount'
        DESC 'The number of networks in the topology'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:29
        NAME 'nnmTopMNodeCount'
        DESC 'The number of nodes in the topology'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:30
```

```
        NAME 'nnmStationCount'
        DESC 'The number of stations in the topology'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:31
        NAME 'nnmLicenseCount'
        DESC 'The number of nodes licensed to be monitored by this management station.
A value -1 represents an unlimited node license'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:32
        NAME 'nnmManageCount'
        DESC 'The number of nodes currently monitored by this management station'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:33
        NAME 'nnmExpireDatestamp'
        DESC 'Expiration date of the license'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_networks Topology Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:34
        NAME 'nnmIPNetworkName'
        DESC 'Either the IP Address of the network or the name of the network from
/etc/networks (or equivalent) configuration file'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:35
        NAME 'nnmIPAddress'
        DESC 'The network IP address'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:36
        NAME 'nnmIPSubnetMask'
        DESC 'The network IP subnet mask'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:37
        NAME 'nnmTopMInterfaceCnt'
        DESC 'The  number of interfaces on this network'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:38
        NAME 'nnmTopMSegmentCount'
        DESC 'The number of segments in this network'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )
```

```
attributetype ( LDAPElementsAttributeTypes:39
       NAME 'nnmNextSegID'
       DESC 'An internal counter used to generate unique segment names for segments
in this network'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:40
       NAME 'nnmTopMDefaultSegID'
       DESC 'The OVW Object ID of the default segment for this network. This value is
the segment in which newly-discovered nodes are placed when they are discovered by
netmon'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:41
       NAME 'nnmRootDev'
       DESC 'The IP address of the root device used in the IP discovery and layout
processes'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:42
       NAME 'nnmIPXAddress'
       DESC 'The IPX Address'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:43
       NAME 'nnmNetHopCnt'
       DESC 'The network hop count'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_segments Topology Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:44
       NAME 'nnmSegmentName'
       DESC 'The name of the segment'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:45
       NAME 'nnmRemIFID'
       DESC 'For star-type segments, the OVW Object ID of the main hub of this
segment, as reported by a remote collection station. For locally-monitored segments,
the hub_if_id should match the rem_if_id'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:46
       NAME 'nnmSegNetID'
       DESC 'The Internal ID of this segment within this network, as based on the
next_seg_id field in the network at the time the segment is created'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:47
       NAME 'nnmHubIFID'
```

```
        DESC 'For star-type segments, the OVW Object ID of the main hub of this
segment'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_nodes Topology Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:48
        NAME 'nnmIPHostName'
        DESC 'The Name of the node: either the IP address of one of the interfaces on
the node or the name of the node as determined vi IP hostname resolution of one of
the interfaces'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:49
        NAME 'nnmSNMPSysDescr'
        DESC 'The SNMP sysDescr field. netmon obtains this information from the agent
on the node, if possible'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:50
        NAME 'nnmSNMPSysLocation'
        DESC 'The SNMP sysLocation field. netmon obtains this information from the
agent on the node, if possible'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:51
        NAME 'nnmSNMPSysContact'
        DESC 'The SNMP sysContact field. netmon obtains this information from the
agent on the node, if possible'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:52
        NAME 'nnmSNMPSysObjID'
        DESC 'The SNMP sysObjectId field. netmon obtains this information from the
agent on the node, if possible. The sysObjectId may be used to determine the vendor
and snmpAgent fields, various topology attributes, and the symbol for the node. See
the oid_to_type(4) and oidto_sym(4) manpages or reference pages'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:53
        NAME 'nnmIPForwarding'
        DESC 'The SNMP forwarding field.  netmon obtains this information from the
agent on the node, if possible'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:54
        NAME 'nnmSNMPState'
        DESC 'Used to store internal-only, Boolean-valued flags'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:55
```

```
        NAME 'nnmSNMPAddr'
        DESC 'The SNMP IP address used for sending SNMP requests to the node. This
field may change based on th status of various inerfaces on the node'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:56
        NAME 'nnmAtCycleTime'
        DESC 'The time between discovery polls for this node. This time internal is
dynamically-adjusted by netmon, based on the number of successful discoveries in the
last discovery poll'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:57
        NAME 'nnmRootPort'
        DESC 'Used with the IP discovery and layout processes to determine the bridge
or hub topology'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:58
        NAME 'nnmVendor'
        DESC 'The vendor of the node based on the mapping established via oid_to_type
file. This enumeration matches the enumerated values in ovwdb for the vendor field'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:59
        NAME 'nnmAgent'
        DESC 'The SNMP agent of the node based on the mapping established via
oid_to_type file. This enumeration matches the enumerated values in ovwdb for the
SNMPAgent field'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:60
        NAME 'nnmIPXServerName'
        DESC 'The IPX server name'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:61
        NAME 'nnmSysName'
        DESC 'The SNMP system name'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HPOV NNM Interfaces Table Attribute Types

attributetype ( LDAPElementsAttributeTypes:62
        NAME 'nnmSNMPIFName'
        DESC 'The SNMP ifname (interface name)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:63
        NAME 'nnmSegChangeTime'
        DESC 'The last time the interface changed segments'
```

```
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:64
        NAME 'nnmIFNumber'
        DESC 'The SNMP ifindex. If the device cannot be contacte vi SNMP, the value is
0.'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:65
        NAME 'nnmNetID'
        DESC 'The OVW Object ID of the network containing this interface'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:66
        NAME 'nnmSegID'
        DESC 'The OVW Object ID of the segment containing this interface. May be null
or 0, to indicate an unconnected interface'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:67
        NAME 'nnmNodeID'
        DESC 'The OVW Object ID of the node containing this interface'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:68
        NAME 'nnmRemNetID'
        DESC 'The OVW Object ID of the network containing this interface, as reported
by the station monitoring this interface. For locally-monitored interfaces, the
rem_net_id matchs the net_id field'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:69
        NAME 'nnmRemSegID'
        DESC 'The OVW Object ID of the segment containing this interface, as reported
by the station monitoring this interface. For locally-monitored interfaces, the
rem_seg_id matchs the seg_id field'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:70
        NAME 'nnmRemNodeID'
        DESC 'The OVW Object ID of the node containing this interface, as reported by
the station monitoring this interface. For locally-monitored interfaces, the
rem_node_id matchs the node_id field'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:71
        NAME 'nnmSegType'
        DESC 'An enumerated integer that indicates the segment type expected for this
interface (as determined by the IP discovery process). Possible values are: 0=Unset
(indicates no preferred type); 1=Bus Segment; 2=Star Segment; 3=Token Ring; 4 = FDDI
Ring; 5=Serial'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{2}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:72
        NAME 'nnmSNMPIFType'
        DESC 'The SNMP iftype (interface type) according to the IANA definitions'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:73
        NAME 'nnmSNMPIFPhysAddr'
        DESC 'The SNMP iPhysaddr, if the node supports SNMP; otherwise this value is
the physical address as teremined by the discovery process'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:74
        NAME 'nnmSNMPIFDescr'
        DESC 'The SNMP ifDescr (interface description)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:75
        NAME 'nnmLlaFrom'
        DESC 'The IP Address of the node that snmp_ifphysaddr was discovered from, or
0 (if the node itself)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:76
        NAME 'nnmMaskFrom'
        DESC 'An enumerated value indicating the source of the IP subnet mask. The
possible values are: 0=Unknown; 1=SNMP IP Address table ; 2=A guess based on
containing network ; 3=A guess based on network class ; 4=ICMP Mask request; 5=A
guess based on user input'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{2}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:77
        NAME 'nnmPortClass'
        DESC 'For hub and bridge interfaces, the group for this instance'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:78
        NAME 'nnmIPXAdress'
        DESC 'The IPX Address'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:79
        NAME 'nnmSNMPIFAlias'
        DESC 'The value of the objects interface. The value set for the device by the
administrator (for example, an alias name that is mor meaningful to the
administrator). If no value is set, the default is ifnumber. Use this to join to the
instance column of the SNMP tables'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:80
        NAME 'nnmSegChangeTimestamp'
```

```
        DESC 'The last time the segment associated with the inerface changed expressed
in Epoch time'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_stations Topology Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:81
        NAME 'nnmStationName'
        DESC 'The IP hostname of the station, determined in the same manner as the
ip_hostname field for nodes'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:82
        NAME 'nnmDBCreateTime'
        DESC 'The time of the topology database creation on the remote station. Used
by ovrepld to determine when a synchronization may need to occur'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:83
        NAME 'nnmEventNum'
        DESC 'The event sequence number of the last event reported by the remote
station. The event sequence number is used by ovrepld to detect lost events from a
remote collection station'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:84
        NAME 'nnmStationType'
        DESC 'The type of the collection station: 0=Unknown or unset; 1=NNM/UX;
2=OpenView for Windows; 3=NNM/NT'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:85
        NAME 'nnmStationVersion'
        DESC 'The version of the topology data exported by the remote collection
station'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:86
        NAME 'nnmStationDesc'
        DESC 'A string description of the remote station'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:87
        NAME 'nnmExpireDate'
        DESC 'The date the license expires for the remote station'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:88
        NAME 'nnmLicensedNodes'
        DESC 'The number of nodes licensed that are monitoring by the remote station'
        EQUALITY caseIgnoreIA5Match
```

```
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:89
        NAME 'nnmManagedNodes'
        DESC 'The number of nodes currently managed by the remote station'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:90
        NAME 'nnmAccessMode'
        DESC 'The type of acess granted to the remote station: 1=read only;
2=read/write. Currently, all remote stations have a value of read only, and only the
local station has a value of read/write'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:91
        NAME 'nnmOverlapMode'
        DESC 'The mode determining how to handle overlaps between this station and
other station. Possible modes are: 0=Allow overlap; 1=Delete; 2=Unmanage. The
overlap_mode only applies to the local management station.'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:92
        NAME 'nnmCheckInterval'
        DESC 'How often ovrepld should verify the status of a managed remote
collection station (in seconds)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:93
        NAME 'nnmSNMPCommunity'
        DESC '(Reserved for future use)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:94
        NAME 'nnmSNMPAuth1'
        DESC '(Reserved for future use)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:95
        NAME 'nnmSNMPAuth2'
        DESC '(Reserved for future use)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:96
        NAME 'nnmLastUpdate'
        DESC 'The time of the last event or object modification on the remote
collection station (according to the remote)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:97
        NAME 'nnmFilter'
        DESC 'The failover filter'
```

```
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:98
        NAME 'nnmFilterChecksum'
        DESC '(Reserved for future use)'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HPOV NNM Classes Table Attribute Types

attributetype ( LDAPElementsAttributeTypes:99
        NAME 'nnmClassName'
        DESC 'the name of the class'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:100
        NAME 'nnmVersion'
        DESC 'the version of the table representing the class'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_nodes_cap Topology Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:101
        NAME 'nnmVendorName'
        DESC 'Name of the manufacturer of the node device'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:102
        NAME 'nnmAgentName'
        DESC 'SNMP agent tyoe. Can represent the operating system'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:103
        NAME 'nnmIsComputer'
        DESC 'The device is a computer'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:104
        NAME 'nnmIsPC'
        DESC 'The device is a PC'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:105
        NAME 'nnmIsWorkstation'
        DESC 'The device is a Workstation'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )
```

```
attributetype ( LDAPElementsAttributeTypes:106
       NAME 'nnmIsMini'
       DESC 'The device is a mini computer'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:107
       NAME 'nnmIsMainFrame'
       DESC 'The device is a mainframe computer'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:108
       NAME 'nnmIsConnector'
       DESC 'The device is a connectorr'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:109
       NAME 'nnmIsBridge'
       DESC 'The device is a bridge'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:110
       NAME 'nnmIsRouter'
       DESC 'The device is a router'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:111
       NAME 'nnmIsIPRouter'
       DESC 'The device is a IP router'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:112
       NAME 'nnmIsIPXRouter'
       DESC 'The device is a IPX router'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:113
       NAME 'nnmIsRepeater'
       DESC 'The device is a repeater'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:114
       NAME 'nnmIsHub'
       DESC 'The device is a hub'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:115
       NAME 'nnmIsSwitch'
       DESC 'The device is a Swhitch'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
```

```
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:116
       NAME 'nnmIsDevice'
       DESC 'The device is a device'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:117
       NAME 'nnmIsPrinter'
       DESC 'The device is a Printer'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:118
       NAME 'nnmIsAnalyzer'
       DESC 'The device is a an analyzer'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

       attributetype ( LDAPElementsAttributeTypes:119
       NAME 'nnmIsIP'
       DESC 'The device is a node that supports Internet Protocol'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:120
       NAME 'nnmIsIPX'
       DESC 'The device is a node that supports IPX'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:121
       NAME 'nnmIsSNMPSupported'
       DESC 'The device supports SNMP'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:122
       NAME 'nnmIsSNMPProxied'
       DESC 'The device proxies SNMP'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:123
       NAME 'nnmIsNetWareServer'
       DESC 'The device is a netware server'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:124
       NAME 'nnmIsMcClusterMember'
       DESC 'The device is a node that netmon determines to be a member of an HP-UX
Mc Cluster (for example, a Service Guard cluster)'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:125
```

```
        NAME 'nnmIsCollectionStationNode'
        DESC 'The device is an NNM station that provides information to another NNM
station about the object that it is managing'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:126
        NAME 'nnmIsRDMISupported'
        DESC 'The device supports remote DMI'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:127
        NAME 'nnmIsHTTPSupported'
        DESC 'The device supports HTTP'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:128
        NAME 'nnmIsHTTPManaged'
        DESC 'The device manages HTTP'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:129
        NAME 'nnmIsRMON'
        DESC 'The device implements the RMON MIB'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:130
        NAME 'nnmIsRMON2'
        DESC 'The device implements RMON2 MIB'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:131
        NAME 'nnmIsDS1'
        DESC 'The device implements the DS1 MIB'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:132
        NAME 'nnmIsDS3'
        DESC 'The device implements DS3 MIB '
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:133
        NAME 'nnmIsFrameRelay'
        DESC 'The device implements the FrameRelay MIB'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:134
        NAME 'nnmIsSONET'
        DESC 'The device implements the SONET MIB'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:135
        NAME 'nnmIsATM'
        DESC 'The device implements the ATM MIB'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:136
        NAME 'nnmIsCDP'
        DESC 'The device implements CDP MIB '
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM nnm_nodes_nets Topology Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:137
        NAME 'nnmNodeName'
        DESC 'The name of the node'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:138
        NAME 'nnmNetIPAddress'
        DESC 'The network IP address'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
#------------- HP OpenView Trend Data Tables LDAP Mapping ----------------------
#-------------------------------------------------------------------------------
# HP OpenView SNMP Trend Version Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:139
        NAME 'nnmTableName'
        DESC 'The name of the data table (HPOVCN) that was previously exported'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:140
        NAME 'nnmLastTimestamp'
        DESC 'Time of the last exported detailed record'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView SNMP Trend Description Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:141
        NAME 'nnmDescriptID'
        DESC 'An unique identifier for this variable or expression'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:142
        NAME 'nnmMIBOID'
```

```
        DESC 'A MIB variable object identifier in either dot notation with no spaces
between digits and dots (with a leading dot) or a MIB expression label as defined in
the mibExpr.conf file'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:143
        NAME 'nnmUnits'
        DESC 'The units of the data. For example: units, units/sec, percent,
errors/sec, or numJobs'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:144
        NAME 'nnmTitle'
        DESC 'A label for this MIB variable or MIB expression (for example,
ifInOctets'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:145
        NAME 'nnmDescript'
        DESC 'A description of this MIB variable or expression, as derived from the
DESCRIPTION field of the MIB definition or the mibExpr.conf file. Description more
than 512 characters are truncated'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{512}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:146
        NAME 'nnmType'
        DESC 'Type of the data. Either ans SNMP data type or EXPRESSION. Possible
values are: COUNTER, GAUGE, INTEGER, TIMETICKS, EXPRESSION or STRING'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:147
        NAME 'nnmMIBExpr'
        DESC 'If a MIB expression, the actual expression'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView SNMP Trend Dataset Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:148
        NAME 'nnmInstance'
        DESC 'The instance being monitored'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:149
        NAME 'nnmDatasetID'
        DESC 'The instance being monitored'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:150
        NAME 'nnmHostName'
        DESC 'Official hostname of the host being monitored'
        EQUALITY caseIgnoreIA5Match
```

```
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:151
        NAME 'nnmServerName'
        DESC 'The node name of the NNM station that exported the data'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:152
        NAME 'nnmTLastRawSec'
        DESC 'The timestamp of the last raw data record exported for the dataset'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:153
        NAME 'nnmTLastRedRec'
        DESC 'The timestampo of the last reduced data record exported for the dataset'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:154
        NAME 'nnmExportSet'
        DESC 'Name of the export set to which this dataset belongs. The value is
determined by the EXPORT_SET field in the SNMP data collectors meta-data files. Non-
reporting datasets have NULL export_set value by default'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView SNMP RAW Trend Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:155
        NAME 'nnmSampleTime'
        DESC 'The time at which this trend period ended (in a human readable format'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:156
        NAME 'nnmSampleTimestamp'
        DESC 'The time in Epoch seconds which this trend period ended'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:157
        NAME 'nnmPeriod'
        DESC 'The length of the period, in seconds'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:158
        NAME 'nnmValue'
        DESC 'The sampled data value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView SNMP REDUCED TREND Object (Table) Class
```

```
attributetype ( LDAPElementsAttributeTypes:159
       NAME 'nnmMinValue'
       DESC 'The minimum value of the data values in the reduced set'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:160
       NAME 'nnmMaxValue'
       DESC 'The maximum value of the data values in the reduced set'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:161
       NAME 'nnmDataCount'
       DESC 'The number of raw data values used to make up the reduced data'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:162
       NAME 'nnmDataSum'
       DESC 'The summation of the data values in the reduced set'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:163
       NAME 'nnmSumSqrs'
       DESC 'The summation of the squares of the data values in the reduced set'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:164
       NAME 'nnmDay'
       DESC 'The day of the summarized data set/aggregated event data'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:165
       NAME 'nnmWeek'
       DESC 'The week of the summarized data set/aggregated event data'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:166
       NAME 'nnmMonth'
       DESC 'The month of the summarized data set/aggregated event data'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:167
       NAME 'nnmYear'
       DESC 'The year of the summarized data set/aggregated event data'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:168
       NAME 'nnmDatapointCount'
       DESC 'The total number of reduced datapoints. (Use hpovsnnmpDataCount for the
total number of raw datapoints before reducing the data)'
```

```
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
#------------- HP OpenView Event Data Tables LDAP Mapping ----------------------
#-------------------------------------------------------------------------------
# HP OpenView NNM EVENT CATEGORY Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:169
        NAME 'nnmCategory'
        DESC 'The category number for the event'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:170
        NAME 'nnmText'
        DESC 'The default categories for events/severity'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM EVENT SEVERITY Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:171
        NAME 'nnmSeverity'
        DESC 'The severity number for the event'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM EVENT DETAILS Object (Table) Class

attributetype ( LDAPElementsAttributeTypes:172
        NAME 'nnmEventUUID'
        DESC 'The identifier for the event'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:173
        NAME 'nnmEventTimestamp'
        DESC 'The time in Epoch seconds when this event occurred'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:174
        NAME 'nnmApplicationID'
        DESC 'The software source of the event that indicates whitch NNM application
or process generated the event'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:175
        NAME 'nnmMessage'
        DESC 'The formatted string describing the event as described in trapd.conf
file'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )
```

```
attributetype ( LDAPElementsAttributeTypes:176
       NAME 'nnmEventOID'
       DESC 'The object identifier of the event'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:177
       NAME 'nnmOVObjectID'
       DESC 'The HP OpenView Object ID, or 0 if not available'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:178
       NAME 'nnmProtocol'
       DESC 'The SNMP protocol type'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:179
       NAME 'nnmEventType'
       DESC 'The enterprise-specific event object identifier'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:180
       NAME 'nnmTrapsource'
       DESC 'The nodename for the node sending the trap'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:181
       NAME 'nnmTrapName'
       DESC 'Name of the SNMP trap that caused the event (for example, OV_Node_Down,
OV_Node_Up)'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:182
       NAME 'nnmPID'
       DESC 'Process ID for the process sending the event'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:183
       NAME 'nnmForwardAddress'
       DESC 'IP address of another NNM pmd to forward event to'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:184
       NAME 'nnmEventSource'
       DESC 'The hostname for the source of the event'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:185
       NAME 'nnmEventTime'
       DESC 'The time the event was received in a readble format'
       EQUALITY caseIgnoreIA5Match
```

```
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:186
        NAME 'nnmNumberVarbinds'
        DESC 'The number of records associated with the event in the nnmEventVarbinds
object class'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:187
        NAME 'nnmVarbindSequence'
        DESC 'Sequence of varbinds of the event as defined in the trap.conf file'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:188
        NAME 'nnmValLen'
        DESC 'The length of the string value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:189
        NAME 'nnmTypeString'
        DESC 'The varbind string value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:190
        NAME 'nnmTypeInteger'
        DESC 'The varbind integer value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:191
        NAME 'nnmObjID'
        DESC 'The varbind OID value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:192
        NAME 'nnmUnsigned32'
        DESC 'The varbind 32-bit integer value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:193
        NAME 'nnmUnsigned64'
        DESC 'The varbind 64-bit integer value'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:194
        NAME 'nnmCollectionName'
        DESC 'The name (label) of the data collector collection whose threshold
violation triggered the threshold event'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )
```

```
attributetype ( LDAPElementsAttributeTypes:195
       NAME 'nnmThreshold'
       DESC 'The numeric value that you set as the standard for comparing collection
samples to identify threshold violations'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:196
       NAME 'nnmSample'
       DESC 'Numeric value collected for the MIB or MIB expression'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:197
       NAME 'nnmHightSample'
       DESC 'Highest sample (peak) value collected for node and instance'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:198
       NAME 'nnmHightTime'
       DESC 'The time that HightSample was collected'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:199
       NAME 'nnmLowSample'
       DESC 'Lowest sample value collected for node and instance'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:200
       NAME 'nnmLowTime'
       DESC 'The time that LowSample was collected'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:201
       NAME 'nnmThresholdOp'
       DESC 'Operator ( >,<,==,>=,<=,!= ) used to compare the sample with the
threshold'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:202
       NAME 'nnmThresholdCount'
       DESC 'The number of consecutive times the reset value was exceeded before
threshold rearm'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:203
       NAME 'nnmEventCnt'
       DESC 'The number of times the associated event occurred for the nodename or
event_oid within time period (of 1 day)'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )
```

```
attributetype ( LDAPElementsAttributeTypes:204
       NAME 'nnmAlwaysNever'
       DESC 'Filtering indicator for whether to always or never filter. Y=always
filter, do not export; N=never filter, export '
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{2}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:205
       NAME 'nnmLimit'
       DESC 'The maximum number of rows for this node or MIB OID to be exported on a
daily basis. This is used only when the always_never attribute is NULL'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:206
       NAME 'nnmShownEntries'
       DESC 'Number of entries with the same name (applies to nnmNodesNets having the
same nnmNodeName). This attribute doesn t exist in OpenView nnm_nodes_nets table !'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{9}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:207
       NAME 'nnmEntryID'
       DESC 'Universal Unique Identifier (UUID) uniquely identifies each entry'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )


#-------------------------------------------------------------------------------
# HP OpenView NNM Alarm Browser Class - for Metadata

attributetype ( LDAPElementsAttributeTypes:208
       NAME 'nnmAlarmID'
       DESC 'Identifier for each alarm entry'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:209
       NAME 'nnmAlarmCat'
       DESC 'Alarm Category for the alarm'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:210
       NAME 'nnmAlarmIMAPServer'
       DESC 'IMAP Server for the alarm information'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:211
       NAME 'nnmAlarmIMAPAccount'
       DESC 'IMAP Account for the alarm information'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
       SINGLE-VALUE )

attributetype ( LDAPElementsAttributeTypes:212
       NAME 'nnmAlarmXMLschemaURL'
       DESC 'URL for the XML DTD/XSL schema for the alarm information'
       EQUALITY caseIgnoreIA5Match
       SUBSTR caseIgnoreSubstringsMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
```

```
        SINGLE-VALUE )


#--------------------------------------------------------------------------------
attributetype ( LDAPElementsAttributeTypes:1000
        NAME 'nmUID'
        DESC 'Unique Identifier (UID): uniquely identifies each nm entry'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{56}
        SINGLE-VALUE )


# eof mania_attribute.schema
```

# ANEXO B

Neste anexo podem ser encontrados os três *scripts* que foram criados para simplificar, facilitar e automatizar o processo de inserção dos objectos definidos no HPOV NNM no serviço de directório LDAP.

### SCRIPT PARA CRIAR LDIF DOS EVENTOS

*Script* utilizado para criar o ficheiro LDIF com os eventos a partir dos ficheiros de *dump* da BD dos Eventos do HPOV NNM.

```
#!/usr/bin/perl
#================================================================
# File parser for the files obtained from dumpping
# the NNM Event Tables of HP OpenView
#================================================================

$/ = "\n";    # Input record separator is ENTER
$\ = "\n";    # Output record separator is ENTER
#$, = " ";    # Output field separator for function prints is SPACE
#$_ = <>;     # Needed only if the reading routine is executed in MAIN


#================================================================
# Main function
#================================================================

{
$COUNTER1 = 1;
$COUNTER2 = 100;
$MULTIPLE = 20;
my @DURATION = times;
my $OUTPUT_FILE = "hpov_event_parsed.ldif";
$PHRASE = "$0: invalid option\nUsage: $0 [-o output_file] [-m positive_integer]";

##---------------------------------------------------------------
# Routine to check the options (switches) of the script
if ($ARGV[0]) {
       $FIRST = $ARGV[0];
       $SECOND = $ARGV[2];
       if (!$ARGV[1]) {
              print $PHRASE;       # No valid First Attribute
              exit;
       }
       if ($FIRST ne "-o") {
              if ($FIRST ne "-m") {
                     print $PHRASE;
                     exit;
              }
              # First ARG is equal to -m
              unless ($ARGV[1] =~ /\d/) {
                     print $PHRASE;       # No valid First Attribute
                     exit;
              }
              # First ARG is equal to -m and with valid integer multiplier
              $MULTIPLE = $ARGV[1];
              if ($ARGV[2]) {
                     $SECOND = $ARGV[2];
                     if (!$ARGV[3]) {
                            print $PHRASE;       # No valid Second Attribute
                            exit;
```

```
                        }
                        if ($SECOND ne "-o") {
                                print $PHRASE;       # Not a valid Second ARG
                                exit;
                        }
                        # Second ARG is equal to -o and as a file name
                        $OUTPUT_FILE = $ARGV[3];
                        goto DONE_ARGS;
                }
                # Just has First ARG -m, no Second ARG
                goto DONE_ARGS;
        }
        # First ARG is equal to -o and as a file name
        $OUTPUT_FILE = $ARGV[1];
        if ($ARGV[2]) {
                $SECOND = $ARGV[2];
                if ($SECOND ne "-m") {
                        print $PHRASE;       # Not a valid Second ARG
                        exit;
                        }
                        # Second ARG is equal to -m
                if (!$ARGV[3] =~ /([0-9])*([1-9])+/) {
                        print $PHRASE;       # No valid Second Attribute
                        exit;
                }
                # Second ARG is equal to -m and with valid integer multiplier
                $MULTIPLE = $ARGV[3];
        }
}

DONE_ARGS:
print "Using output file $OUTPUT_FILE..." if ($OUTPUT_FILE ne
"hpov_event_parsed.ldif");
print "Using default output file $OUTPUT_FILE..." if ($OUTPUT_FILE eq
"hpov_event_parsed.ldif");
print "Using output counter multiplier $MULTIPLE","x...\n" if ($MULTIPLE != 20);
print "Using default output counter multiplier $MULTIPLE","x...\n" if ($MULTIPLE ==
20);
#-----------------------------------------------------------------

open (FHO, ">$OUTPUT_FILE") or die ("Impossible to open file $OUTPUT_FILE for
output!");

&FILE_HEADER();
&NNM_EVENT_CAT();
&NNM_EVENT_SEV();
&NNM_EVENT_DETAIL();
&NNM_EVENT_VARBINDS();
&NMM_EVENT_THRESH();
&NNM_EVENT_DAILY();
&NNM_EVENT_WEEKLY();
&NNM_EVENT_MONTHLY();
&NNM_EVENT_YEARLY();
&NNM_EVENT_LIMITS();

close (FHO);

print "\n\nLDIF file created.\n$COUNTER1 Objects parsed.";

}


#=================================================================
# sub FILE_HEADER()
#
# Creates header of LDIF file.
#=================================================================

sub FILE_HEADER {

my $SUM_DATE_HOUR;
my @DATE_HOUR;

$SUM_DATE_HOUR = time;      # Time in seconds since January 1, 1970
@DATE_HOUR = gmtime $SUM_DATE_HOUR;      # Transforms time in second in GMT time
if ($DATE_HOUR[2] < 10) { $DATE_HOUR[2] = "0$DATE_HOUR[2]";} # Adds a zero to the
hour if < 10
```

```
if ($DATE_HOUR[1] < 10) { $DATE_HOUR[1] = "0$DATE_HOUR[1]";} # Adds a zero to the
minutes if < 10
if ($DATE_HOUR[0] < 10) { $DATE_HOUR[0] = "0$DATE_HOUR[0]";} # Adds a zero to the
seconds if < 10
if ($DATE_HOUR[3] < 10) { $DATE_HOUR[3] = "0$DATE_HOUR[3]";} # Adds a zero to the day
if < 10
$DATE_HOUR[4] = $DATE_HOUR[4]+1;  # Adjusts month from (1..11) to (1..12)

##-----------------------------------------------------------------
# Transforms numeric month in name month
if ($DATE_HOUR[4] = 1) { $DATE_HOUR[4] = "January";}
if ($DATE_HOUR[4] = 2) { $DATE_HOUR[4] = "February";}
if ($DATE_HOUR[4] = 3) { $DATE_HOUR[4] = "March";}
if ($DATE_HOUR[4] = 4) { $DATE_HOUR[4] = "April";}
if ($DATE_HOUR[4] = 5) { $DATE_HOUR[4] = "May";}
if ($DATE_HOUR[4] = 6) { $DATE_HOUR[4] = "June";}
if ($DATE_HOUR[4] = 7) { $DATE_HOUR[4] = "July";}
if ($DATE_HOUR[4] = 8) { $DATE_HOUR[4] = "August";}
if ($DATE_HOUR[4] = 9) { $DATE_HOUR[4] = "September";}
if ($DATE_HOUR[4] = 10) { $DATE_HOUR[4] = "October";}
if ($DATE_HOUR[4] = 11) { $DATE_HOUR[4] = "November";}
if ($DATE_HOUR[4] = 12) { $DATE_HOUR[4] = "December";}
#-----------------------------------------------------------------

$DATE_HOUR[5] = $DATE_HOUR[5]+1900;
$USER = getlogin;

print FHO "# LDIF file generated by Perl script from HPOV dump files.";
print FHO "# Script ran by user $USER.";
print FHO "# Time of creation: $DATE_HOUR[4] $DATE_HOUR[3], $DATE_HOUR[5]
$DATE_HOUR[2]:$DATE_HOUR[1]:$DATE_HOUR[0]\n";

}


#=================================================================
# sub NNM_EVENT_CAT
#
# Creates nnm_event_cat entries of LDIF file.
#=================================================================

sub NNM_EVENT_CAT {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventCat.dump";
my $CLASS_NAME = "nnmEventCat";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_CAT = ( nnmCategory, nnmText);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Category, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Category,hpovcn=Event,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500"
;
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
```

```
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);
}


#===================================================================
# sub NNM_EVENT_SEV();
#
# Creates nnm_event_sev entries of LDIF file.
#===================================================================

sub NNM_EVENT_SEV {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventSev.dump";
my $CLASS_NAME = "nnmEventSev";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_SEV = ( nnmSeverity , nnmText);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Severity, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Severity,hpovcn=Event,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500"
;
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_SEV[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}


#===================================================================
# sub NNM_EVENT_DETAIL();
#
# Creates nnm_event_details entries of LDIF file.
#===================================================================

sub NNM_EVENT_DETAIL {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventDetail.dump";
my $CLASS_NAME = "nnmEventDetail";
my $MUST = "nnmEntryID";
```

```
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_DETAIL = ( nnmEventUUID , nnmApplicationID , nnmCategory ,
nnmEventOID , nnmEventSource , nnmEventTime , nnmEventTimestamp , nnmEventType ,
nnmForwardAddress , nnmIPAddress , nnmMessage , nnmNodename , nnmNumberVarbinds ,
nnmOVObjectID , nnmPID , nnmProtocol , nnmSeverity , nnmTrapName , nnmTrapsource );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Details,hpovcn=Event,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_DETAIL[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}

#================================================================
# sub NNM_EVENT_VARBINDS();
#
# Creates nnm_event_varbinds entries of LDIF file.
#================================================================

sub NNM_EVENT_VARBINDS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventVarbinds.dump";
my $CLASS_NAME = "nnmEventVarbinds";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_VARBINDS = ( nnmEventUUID , nnmVarbindSequence , nnmEventOID ,
nnmValLen , nnmType , nnmTypeString , nnmTypeInteger , nnmObjID , nnmUnsigned32 ,
nnmUnsigned64 );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Varbinds,hpovcn=Event,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500"
;
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
```

```
	$OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
	$OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
	s/(<)+//ig;
	s/(>)+//ig;
	print FHO "$DESC_NNM_EVENT_VARBINDS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
	$COUNTER3++;
	}
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}

#================================================================
# sub NMM_EVENT_THRESH();
#
# Creates nnm_event_tresh entries of LDIF file.
#================================================================

sub NMM_EVENT_THRESH {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventThresh.dump";
my $CLASS_NAME = "nnmEventThresh";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NMM_EVENT_THRESH = ( nnmEventUUID , nnmEventOID , nnmEventTimestamp ,
nnmEventTime , nnmApplicationID , nnmNodename , nnmOVObjectID , nnmMIBOID ,
nnmCollectionName , nnmInstance , nnmSeverity , nnmThreshold , nnmSample ,
nnmHightSample , nnmHightTime , nnmLowSample , nnmLowTime , nnmThresholdOp ,
nnmThresholdCount );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event,...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Threshold,hpovcn=Event,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500
";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
	$OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
	$OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
	s/(<)+//ig;
	s/(>)+//ig;
	print FHO "$DESC_NMM_EVENT_THRESH[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
	$COUNTER3++;
	}
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}
```

```
#=============================================================
# sub NNM_EVENT_DAILY();
#
# Creates nnm_event_daily entries of LDIF file.
#=============================================================

sub NNM_EVENT_DAILY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventDaily.dump";
my $CLASS_NAME = "nnmEventAggregation";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_DAILY = ( nnmNodename, nnmEventOID , nnmEventCnt ,
nnmEventTimestamp , nnmEventTime , nnmDay , nnmWeek , nnmMonth , nnmYear );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";

print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Daily,hpovcn=Aggregation,hpovcn=Event,hpovcn=NNM,nmcn=HPOpe
nView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_DAILY[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}

#=============================================================
# sub NNM_EVENT_WEEKLY();
#
# Creates nnm_event_weekly entries of LDIF file.
#=============================================================

sub NNM_EVENT_WEEKLY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventWeekly.dump";
my $CLASS_NAME = "nnmEventAggregation";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_WEEKLY = ( nnmNodename , nnmEventOID , nnmEventCnt ,
nnmEventTimestamp , nnmEventTime , nnmDay , nnmWeek , nnmMonth , nnmYear );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
```

```
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";

print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Weekly,hpovcn=Aggregation,hpovcn=Event,hpovcn=NNM,nmcn=HPOp
enView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_WEEKLY[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}

#================================================================
# sub NNM_EVENT_MONTHLY();
#
# Creates nnm_event_monthly entries of LDIF file.
#================================================================

sub NNM_EVENT_MONTHLY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventMonthly.dump";
my $CLASS_NAME = "nnmEventAggregation";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_MONTHLY = ( nnmNodename , nnmEventOID , nnmEventCnt ,
nnmEventTimestamp , nnmEventTime , nnmDay , nnmWeek , nnmMonth , nnmYear );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";

print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Monthly,hpovcn=Aggregation,hpovcn=Event,hpovcn=NNM,nmcn=HPO
penView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_MONTHLY[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
```

```
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}

#=============================================================
# sub NNM_EVENT_YEARLY();
#
# Creates nnm_event_yearly entries of LDIF file.
#=============================================================

sub NNM_EVENT_YEARLY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventYearly.dump";
my $CLASS_NAME = "nnmEventAggregation";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_YEARLY = ( nnmNodename , nnmEventOID , nnmEventCnt ,
nnmEventTimestamp , nnmEventTime , nnmDay , nnmWeek , nnmMonth , nnmYear );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";

print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Yearly,hpovcn=Aggregation,hpovcn=Event,hpovcn=NNM,nmcn=HPOp
enView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_YEARLY[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}

#=============================================================
# sub NNM_EVENT_LIMITS();
#
# Creates nnm_event_limits entries of LDIF file.
#=============================================================

sub NNM_EVENT_LIMITS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmEventLimits.dump";
```

```
my $CLASS_NAME = "nnmEventLimits";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_NNM_EVENT_LIMITS = ( nnmNodename , nnmEventOID , nnmAlwaysNever , nnmLimit
);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";

print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Details, hpovcn=Event, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Limits,hpovcn=Event,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*? ~<*? none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_EVENT_LIMITS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}
```

## SCRIPT PARA CRIAR LDIF DAS TENDÊNCIAS

*Script* utilizado para criar o ficheiro LDIF com as tendências a partir dos ficheiros de *dump* da BD das Tendências do HPOV NNM.

```
#!/usr/bin/perl
#================================================================
# File parser for the files obtained from dumpping
# the NNM Trend Tables of HP OpenView
#================================================================
$/ = "\n";    # Input record separator is ENTER
$\ = "\n";    # Output record separator is ENTER
#$, = " ";    # Output field separator for function prints is SPACE
#$_ = <>;     # Needed only if the reading routine is executed in MAIN

#================================================================
# Main function
#================================================================
{
$COUNTER1 = 1;
$COUNTER2 = 100;
$MULTIPLE = 20;
my @DURATION = times;
my $OUTPUT_FILE = "hpov_trend_parsed.ldif";
$PHRASE = "$0: invalid option\nUsage: $0 [-o output_file] [-m positive_integer]";

##----------------------------------------------------------------
# Routine to check the options (switches) of the script
if ($ARGV[0]) {
       $FIRST = $ARGV[0];
       $SECOND = $ARGV[2];
       if (!$ARGV[1]) {
              print $PHRASE;       # No valid First Attribute
              exit;
       }
       if ($FIRST ne "-o") {
              if ($FIRST ne "-m") {
                     print $PHRASE;
                     exit;
              }
              # First ARG is equal to -m
              unless ($ARGV[1] =~ /\d/) {
                     print $PHRASE;       # No valid First Attribute
                     exit;
              }
              # First ARG is equal to -m and with valid integer multiplier
              $MULTIPLE = $ARGV[1];
              if ($ARGV[2]) {
                     $SECOND = $ARGV[2];
                     if (!$ARGV[3]) {
                            print $PHRASE;       # No valid Second Attribute
                            exit;
                     }
                     if ($SECOND ne "-o") {
                            print $PHRASE;       # Not a valid Second ARG
                            exit;
                     }
                     # Second ARG is equal to -o and as a file name
                     $OUTPUT_FILE = $ARGV[3];
                     goto DONE_ARGS;
              }
              # Just has First ARG -m, no Second ARG
              goto DONE_ARGS;
       }
       # First ARG is equal to -o and as a file name
       $OUTPUT_FILE = $ARGV[1];
       if ($ARGV[2]) {
              $SECOND = $ARGV[2];
              if ($SECOND ne "-m") {
                     print $PHRASE;       # Not a valid Second ARG
                     exit;
```

```
                }
                # Second ARG is equal to -m
            if (!$ARGV[3] =~ /([0-9])*([1-9])+/) {
                print $PHRASE;      # No valid Second Attribute
                exit;
            }
            # Second ARG is equal to -m and with valid integer multiplier
            $MULTIPLE = $ARGV[3];
        }
}
DONE_ARGS:
print "Using output file $OUTPUT_FILE..." if ($OUTPUT_FILE ne
"hpov_trend_parsed.ldif");
print "Using default output file $OUTPUT_FILE..." if ($OUTPUT_FILE eq
"hpov_trend_parsed.ldif");
print "Using output counter multiplier $MULTIPLE","x...\n" if ($MULTIPLE != 20);
print "Using default output counter multiplier $MULTIPLE","x...\n" if ($MULTIPLE ==
20);
#-------------------------------------------------------------
open (FHO, ">$OUTPUT_FILE") or die ("Impossible to open file $OUTPUT_FILE for
output!");

&FILE_HEADER();
&SNMP_TREND_VERSION();
&SNMP_TREND_DESC();
&SNMP_TREND_DATASET();
&SNMP_RAW_TREND();
&SNMP_REDUCED_TREND();
&SNMP_AGGREGATION_DAILY();
&SNMP_AGGREGATION_WEEKLY();
&SNMP_AGGREGATION_MONTHLY();
&SNMP_TEXT_COLLECTIONS();

close (FHO);

print "\n\nLDIF file created.\n$COUNTER1 Objects parsed.";
}


#=============================================================
# sub FILE_HEADER()
#
# Creates header of LDIF file.
#=============================================================
sub FILE_HEADER {
my $SUM_DATE_HOUR;
my @DATE_HOUR;

$SUM_DATE_HOUR = time;      # Time in seconds since January 1, 1970
@DATE_HOUR = gmtime $SUM_DATE_HOUR;      # Transforms time in second in GMT time
if ($DATE_HOUR[2] < 10) { $DATE_HOUR[2] = "0$DATE_HOUR[2]";} # Adds a zero to the
hour if < 10
if ($DATE_HOUR[1] < 10) { $DATE_HOUR[1] = "0$DATE_HOUR[1]";} # Adds a zero to the
minutes if < 10
if ($DATE_HOUR[0] < 10) { $DATE_HOUR[0] = "0$DATE_HOUR[0]";} # Adds a zero to the
seconds if < 10
if ($DATE_HOUR[3] < 10) { $DATE_HOUR[3] = "0$DATE_HOUR[3]";} # Adds a zero to the day
if < 10
$DATE_HOUR[4] = $DATE_HOUR[4]+1;  # Adjusts month from (1..11) to (1..12)

##-------------------------------------------------------------
# Transforms numeric month in name month
if ($DATE_HOUR[4] == 1) { $DATE_HOUR[4] = "January";}
if ($DATE_HOUR[4] == 2) { $DATE_HOUR[4] = "February";}
if ($DATE_HOUR[4] == 3) { $DATE_HOUR[4] = "March";}
if ($DATE_HOUR[4] == 4) { $DATE_HOUR[4] = "April";}
if ($DATE_HOUR[4] == 5) { $DATE_HOUR[4] = "May";}
if ($DATE_HOUR[4] == 6) { $DATE_HOUR[4] = "June";}
if ($DATE_HOUR[4] == 7) { $DATE_HOUR[4] = "July";}
if ($DATE_HOUR[4] == 8) { $DATE_HOUR[4] = "August";}
if ($DATE_HOUR[4] == 9) { $DATE_HOUR[4] = "September";}
if ($DATE_HOUR[4] == 10) { $DATE_HOUR[4] = "October";}
if ($DATE_HOUR[4] == 11) { $DATE_HOUR[4] = "November";}
if ($DATE_HOUR[4] == 12) { $DATE_HOUR[4] = "December";}
#-------------------------------------------------------------
$DATE_HOUR[5] = $DATE_HOUR[5]+1900;
$USER = getlogin;
```

```
print FHO "# LDIF file generated by Perl script from HPOV dump files.";
print FHO "# Script ran by user $USER.";
print FHO "# Time of creation: $DATE_HOUR[4] $DATE_HOUR[3], $DATE_HOUR[5]
$DATE_HOUR[2]:$DATE_HOUR[1]:$DATE_HOUR[0]\n";
}


#================================================================
# sub SNMP_TREND_VERSION
#
# Creates snmp_trend_cat entries of LDIF file.
#================================================================
sub SNMP_TREND_VERSION {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpTrendVersion.dump";
my $CLASS_NAME = "nnmTrendVersion";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmTableName , nnmVersion , nnmLastTimestamp);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Version, hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Version,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#================================================================
# sub SNMP_TREND_DESC
#
# Creates snmp_trend_cat entries of LDIF file.
#================================================================
sub SNMP_TREND_DESC {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpTrendDesc.dump";
my $CLASS_NAME = "nnmTrendDesc";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmDescriptID , nnmMIBOID , nnmUnits , nnmTitle ,
nnmDescript , nnmType , nnmMIBExpr );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));
```

```
while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Description, hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Description,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpenView,dc=xe45
00";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#================================================================
# sub SNMP_TREND_DATASET
#
# Creates snmp_trend_cat entries of LDIF file.
#================================================================
sub SNMP_TREND_DATASET {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpTrendDataset.dump";
my $CLASS_NAME = "nnmTrendDataset";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmTopoID , nnmDescriptID , nnmInstance , nnmDatasetID ,
nnmHostName , nnmServerName , nnmTLastRawSec , nnmTLastRedRec , nnmTitle ,
nnmExportSet );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Dataset, hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Dataset,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;
```

```
print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#===============================================================
# sub SNMP_RAW_TREND
#
# Creates snmp_trend_cat entries of LDIF file.
#===============================================================
sub SNMP_RAW_TREND {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpRawTrend.dump";
my $CLASS_NAME = "nnmRawTrend";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmDatasetID , nnmSampleTime , nnmSampleTimestamp ,
nnmPeriod , nnmValue );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Raw, hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Raw,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#===============================================================
# sub SNMP_REDUCED_TREND
#
# Creates snmp_trend_cat entries of LDIF file.
#===============================================================
sub SNMP_REDUCED_TREND {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpReducedTrend.dump";
my $CLASS_NAME = "nnmReducedTrend";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmDatasetID , nnmSampleTime , nnmSampleTimestamp ,
nnmPeriod , nnmMinValue , nnmMaxValue , nnmDataCount , nnmDataSum , nnmSumSqrs );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));
```

```
while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Raw, hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Raw,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}

#================================================================
# sub SNMP_AGGREGATION_DAILY
#
# Creates snmp_trend_cat entries of LDIF file.
#================================================================
sub SNMP_AGG
b40
REGATION_DAILY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpDailyTrend.dump";
my $CLASS_NAME = "nnmTrendAggregation";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmDatasetID , nnmDescriptID , nnmHostName , nnmMIBOID ,
nnmInstance , nnmSampleTime , nnmWeek , nnmMonth , nnmYear , nnmMinValue ,
nnmMaxValue , nnmDataCount , nnmDataSum , nnmDatapointCount , nnmSumSqrs );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Daily, hpovcn=Aggregation,
hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Daily,hpovcn=Aggregation,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpe
nView,
 dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
```

```
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#===============================================================
# sub SNMP_AGGREGATION_WEEKLY
#
# Creates snmp_trend_cat entries of LDIF file.
#===============================================================
sub SNMP_AGGREGATION_WEEKLY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpWeeklyTrend.dump";
my $CLASS_NAME = "nnmTrendAggregation";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmDatasetID , nnmDescriptID , nnmHostName , nnmMIBOID ,
nnmInstance , nnmSampleTime , nnmWeek , nnmMonth , nnmYear , nnmMinValue ,
nnmMaxValue , nnmDataCount , nnmDataSum , nnmDatapointCount , nnmSumSqrs );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Weekly, hpovcn=Aggregation,
hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Weekly,hpovcn=Aggregation,hpovcn=Trend,hpovcn=NNM,nmcn=HPOp
enView,
 dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#===============================================================
# sub SNMP_AGGREGATION_MONTHLY
#
# Creates snmp_trend_cat entries of LDIF file.
#===============================================================
sub SNMP_AGGREGATION_MONTHLY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpMonthlyTrend.dump";
my $CLASS_NAME = "nnmTrendAggregation";
```

```
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nnmDatasetID , nnmDescriptID , nnmHostName , nnmMIBOID ,
nnmInstance , nnmSampleTime , nnmWeek , nnmMonth , nnmYear , nnmMinValue ,
nnmMaxValue , nnmDataCount , nnmDataSum , nnmDatapointCount , nnmSumSqrs );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Monthly, hpovcn=Aggregation,
hpovcn=Trend, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Monthly,hpovcn=Aggregation,hpovcn=Trend,hpovcn=NNM,nmcn=HPO
penView,
 dc=xe4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}
close (FHI);
}


#================================================================
# sub SNMP_TEXT_COLLECTIONS
#
# Creates snmp_trend_cat entries of LDIF file.
#================================================================
sub SNMP_TEXT_COLLECTIONS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "snmpTextCollections.dump";
my $CLASS_NAME = "nnmTextCollections";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary object
my @DESC_SNMP_TREND_CAT = ( nmDatasetID , nnmSampleTime , nnmSampleTimestamp ,
nnmPeriod , nnmValue );

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=TextCollections, hpovcn=Trend,
...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=TextCollections,hpovcn=Trend,hpovcn=NNM,nmcn=HPOpenView,dc=
xe4500";
print FHO "$MUST:$COUNTER";
```

```
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_SNMP_TREND_CAT[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}
```

## SCRIPT PARA CRIAR LDIF DA TOPOLOGIA

*Script* utilizado para criar o ficheiro LDIF com a topologia a partir dos ficheiros de *dump* da BD de Topologia do HPOV NNM.

```
#!/usr/bin/perl
#================================================================
# File parser for the files obtained from dumpping
# the NNM Topology Tables of HP OpenView
#================================================================

$/ = "\n";    # Input record separator is ENTER
$\ = "\n";    # Output record separator is ENTER
#$, = " ";    # Output field separator for function prints is SPACE
#$_ = <>;     # Needed only if the reading routine is executed in MAIN

#================================================================
# Main function
#================================================================
{

$COUNTER1 = 1;
$COUNTER2 = 100;
$MULTIPLE = 20;
my @DURATION = times;
my $OUTPUT_FILE = "hpov_topo_parsed.ldif";
$PHRASE = "$0: invalid option\nUsage: $0 [-o output_file] [-m positive_integer]";

##----------------------------------------------------------------
# Routine to check the options (switches) of the script
if ($ARGV[0]) {
       $FIRST = $ARGV[0];
       $SECOND = $ARGV[2];
       if (!$ARGV[1]) {
              print $PHRASE;       # No valid First Attribute
              exit;
       }
       if ($FIRST ne "-o") {
              if ($FIRST ne "-m") {
                     print $PHRASE;
                     exit;
              }
              # First ARG is equal to -m
              unless ($ARGV[1] =~ /\d/) {
                     print $PHRASE;       # No valid First Attribute
                     exit;
              }
              # First ARG is equal to -m and with valid integer multiplier
              $MULTIPLE = $ARGV[1];
              if ($ARGV[2]) {
                     $SECOND = $ARGV[2];
                     if (!$ARGV[3]) {
                            print $PHRASE;       # No valid Second Attribute
                            exit;
                     }
                     if ($SECOND ne "-o") {
                            print $PHRASE;       # Not a valid Second ARG
                            exit;
                     }
                     # Second ARG is equal to -o and as a file name
                     $OUTPUT_FILE = $ARGV[3];
                     goto DONE_ARGS;
              }
              # Just has First ARG -m, no Second ARG
              goto DONE_ARGS;
       }
       # First ARG is equal to -o and as a file name
       $OUTPUT_FILE = $ARGV[1];
       if ($ARGV[2]) {
              $SECOND = $ARGV[2];
              if ($SECOND ne "-m") {
```

```
                        print $PHRASE;       # Not a valid Second ARG
                        exit;
                        }
                        # Second ARG is equal to -m
                if (!$ARGV[3] =~ /([0-9])*([1-9])+/) {
                        print $PHRASE;       # No valid Second Attribute
                        exit;
                }
                # Second ARG is equal to -m and with valid integer multiplier
                $MULTIPLE = $ARGV[3];
        }
}

DONE_ARGS:
print "Using output file $OUTPUT_FILE..." if ($OUTPUT_FILE ne
"hpov_topo_parsed.ldif");
print "Using default output file $OUTPUT_FILE..." if ($OUTPUT_FILE eq
"hpov_topo_parsed.ldif");
print "Using output counter multiplier $MULTIPLE","x...\n" if ($MULTIPLE != 20);
print "Using default output counter multiplier $MULTIPLE","x...\n" if ($MULTIPLE ==
20);
#-----------------------------------------------------------------

open (FHO, ">$OUTPUT_FILE") or die ("Impossible to open file $OUTPUT_FILE for
output!");

&FILE_HEADER();
&NNM_OBJECTS();
&NNM_TOPOLOGY();
&NNM_NETWORKS();
&NNM_SEGMENTS();
&NNM_NODES();
&NNM_INTERFACES();
&NNM_STATIONS();
&NNM_CLASSES();
&NNM_NODESCAP();
&NNM_NODESNETS();

close (FHO);

print "\n\nLDIF file created.\n$COUNTER1 Objects parsed in $DURATION[0] seconds.";

}



#=================================================================
# sub FILE_HEADER()
#
# Creates header of LDIF file.
#=================================================================

sub FILE_HEADER {

my $SUM_DATE_HOUR;
my @DATE_HOUR;

$SUM_DATE_HOUR = time;      # Time in seconds since January 1, 1970
@DATE_HOUR = gmtime $SUM_DATE_HOUR;       # Transforms time in second in GMT time
if ($DATE_HOUR[2] < 10) { $DATE_HOUR[2] = "0$DATE_HOUR[2]";} # Adds a zero to the
hour if < 10
if ($DATE_HOUR[1] < 10) { $DATE_HOUR[1] = "0$DATE_HOUR[1]";} # Adds a zero to the
minutes if < 10
if ($DATE_HOUR[0] < 10) { $DATE_HOUR[0] = "0$DATE_HOUR[0]";} # Adds a zero to the
seconds if < 10
if ($DATE_HOUR[3] < 10) { $DATE_HOUR[3] = "0$DATE_HOUR[3]";} # Adds a zero to the day
if < 10
$DATE_HOUR[4] = $DATE_HOUR[4]+1;  # Adjusts month from (1..11) to (1..12)

##---------------------------------------------------------------
# Transforms numeric month in name month
if ($DATE_HOUR[4] == 1) { $DATE_HOUR[4] = "January";}
if ($DATE_HOUR[4] == 2) { $DATE_HOUR[4] = "February";}
if ($DATE_HOUR[4] == 3) { $DATE_HOUR[4] = "March";}
if ($DATE_HOUR[4] == 4) { $DATE_HOUR[4] = "April";}
if ($DATE_HOUR[4] == 5) { $DATE_HOUR[4] = "May";}
if ($DATE_HOUR[4] == 6) { $DATE_HOUR[4] = "June";}
```

```
if ($DATE_HOUR[4] == 7) { $DATE_HOUR[4] = "July";}
if ($DATE_HOUR[4] == 8) { $DATE_HOUR[4] = "August";}
if ($DATE_HOUR[4] == 9) { $DATE_HOUR[4] = "September";}
if ($DATE_HOUR[4] == 10) { $DATE_HOUR[4] = "October";}
if ($DATE_HOUR[4] == 11) { $DATE_HOUR[4] = "November";}
if ($DATE_HOUR[4] == 12) { $DATE_HOUR[4] = "December";}
#-----------------------------------------------------------------

$DATE_HOUR[5] = $DATE_HOUR[5]+1900;
$USER = getlogin;

print FHO "# LDIF file generated by Perl script from HPOV dump files.";
print FHO "# Script ran by user $USER.";
print FHO "# Time of creation: $DATE_HOUR[4] $DATE_HOUR[3], $DATE_HOUR[5]
$DATE_HOUR[2]:$DATE_HOUR[1]:$DATE_HOUR[0]\n";

}


#================================================================
# sub NNM_OBJECTS()
#
# Creates nnm_object entrys of LDIF file.
#================================================================

sub NNM_OBJECTS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmObjects.dump";
my $CLASS_NAME = "nnmObjects";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_object object
my @DESC_NNM_OBJECTS = ( nnmTopoID , nnmOVWID , nnmRemoteID , nnmStationID ,
nnmClassID , nnmCTime , nnmMTime , nnmSymTime , nnmStatTime , nnmLabelTime ,
nnmFlagsTime , nnmIPStatus , nnmRemStatus , nnmFlags , nnmExtra0 , nnmExtra1 ,
nnmCTimestamp , nnmMTimestamp , nnmSymTimestamp , nnmStatTimestamp ,
nnmLabelTimestamp , nnmFlagsTimestamp);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Objects, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Objects,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe450
0";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_OBJECTS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}
```

```
#================================================================
# sub NNM_TOPOLOGY()
#
# Creates nnm_object entrys of LDIF file.
#================================================================

sub NNM_TOPOLOGY {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmTopology.dump";
my $CLASS_NAME ="nnmTopology";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_topoloy object
my @DESC_NNM_TOPOLOGY = ( nnmTopoID , nnmExpireDate , nnmGwCnt , nnmTopMNetworkCount
, nnmTopMSegmentCount , nnmTopMNodeCount , nnmTopMInterfaceCnt , nnmEventNum ,
nnmStationCount , nnmLicenseCount , nnmManageCount , nnmExpireDatestamp);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Topology, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Topology,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe45
00";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_TOPOLOGY[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";

}

close (FHI);

}


#================================================================
# sub NNM_NETWORKS()
#
# Creates nnm_networks entrys of LDIF file.
#================================================================

sub NNM_NETWORKS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmNetworks.dump";
my $CLASS_NAME = "nnmNetworks";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_networks object
my @DESC_NNM_NETWORKS = ( nnmTopoID , nnmIPNetworkName , nnmIPAddress ,
nnmIPSubNetMask , nnmTopMInterfaceCnt , nnmTopMSegmentCount , nnmNextSegID ,
nnmTopMDefaultSegID , nnmRootDev , nnmIPXAddress , nnmNetHopCnt);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));
```

```
while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Networks, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Networks,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe45
00";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_NETWORKS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}


#================================================================
# sub NNM_SEGMENTS()
#
# Creates nnm_segments entrys of LDIF file.
#================================================================

sub NNM_SEGMENTS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmSegments.dump";
my $CLASS_NAME = "nnmSegments";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_segments object
my @DESC_NNM_SEGMENTS = ( nnmTopoID , nnmSegmentName , nnmNetID , nnmRemNetID ,
nnmRemIFID , nnmSegNetID , nnmSegType , nnmTopMInterfaceCnt , nnmHubIFID);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Segments, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Segments,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe45
00";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_SEGMENTS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
```

```
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}


#================================================================
# sub NNM_NODES()
#
# Creates nnm_nodes entrys of LDIF file.
#================================================================

sub NNM_NODES {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmNodes.dump";
my $CLASS_NAME = "nnmNodes";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_nodes object
my @DESC_NNM_NODES = ( nnmTopoID , nnmIPHostname , nnmSNMPSysDescr ,
nnmSNMPSysLocation , nnmSNMPSysContact , nnmTopMInterfaceCnt , nnmSNMPSysObjID ,
nnmIPForwarding , nnmSNMPState ,nnmSNMPAddr , nnmAtCycleTime , nnmRootPort ,
nnmVendor , nnmAgent, nnmIPXAddress , nnmIPXServerName , nnmSysName);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Nodes, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Nodes,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe4500"
;
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_NODES[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}


#================================================================
# sub NNM_INTERFACES()
#
# Creates nnm_interfaces entrys of LDIF file.
#================================================================
```

```
sub NNM_INTERFACES {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmInterfaces.dump";
my $CLASS_NAME = "nnmInterfaces";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_interfaces object
my @DESC_NNM_INTERFACES = ( nnmTopoID , nnmSNMPIFName , nnmSegChangeTime ,
nnmIPAddress , nnmIPSubnetMask, nnmIFNumber , nnmNetID , nnmSegID , nnmNodeID ,
nnmRemNetID , nnmRemSegID , nnmRemNodeID , nnmSegType , nnmSNMPIFType ,
nnmSNMPIFPhysAddr , nnmSNMPIFDescr , nnmSNMPState, nnmLlaFrom, nnmMaskFrom ,
nnmPortClass , nnmIPXAddress , nnmSNMPIFAlias , nnmSegChangeTimestamp);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Interfaces, hpovcn=Topology,
...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Interfaces,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe
4500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_INTERFACES[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";

}

close (FHI);

}


#================================================================
# sub NNM_STATIONS()
#
# Creates nnm_stations entrys of LDIF file.
#================================================================

sub NNM_STATIONS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmStations.dump";
my $CLASS_NAME = "nnmStations";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_stations object
my @DESC_NNM_STATIONS = ( nnmTopoID , nnmStationName , nnmDbCreateTime , nnmEventNum
, nnmStationType , nnmStationVersion , nnmLicensedNodes , nnmManagedNodes ,
nnmAccessMode , nnmOverlapMode , nnmCheckInterval , nnmLastUpdate , nnmFilterChecksum
, nnmStationDesc , nnmExpireDate , nnmSNMPCommunity , nnmSNMPAuth1 , nnmSNMPAuth2 ,
nnmFilter);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));
```

```
while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Stations, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Stations,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe45
00";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_STATIONS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}


#================================================================
# sub NNM_CLASSES()
#
# Creates nnm_classes entrys of LDIF file.
#================================================================

sub NNM_CLASSES {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmClasses.dump";
my $CLASS_NAME = "nnmClasses";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_classes object
my @DESC_NNM_CLASSES = ( nnmClassID , nnmClassName , nnmVersion);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;             # Deletes the \n that's on the end of each array element

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=Classes, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=Classes,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe450
0";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_CLASSES[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
```

```
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}


#===============================================================
# sub NNM_NODESCAP()
#
# Creates nnm_nodes_cap entrys of LDIF file.
#===============================================================

sub NNM_NODESCAP {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmNodesCap.dump";
my $CLASS_NAME = "nnmNodesCap";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_nodes_cap object
my @DESC_NNM_NODESCAP = ( nnmTopoID , nnmVendorName , nnmAgentName , nnmIsComputer ,
nnmIsPC , nnmIsWorkstation , nnmIsMini , nnmIsMainFrame , nnmIsConnector ,
nnmIsBridge , nnmIsRouter , nnmIsIPRouter , nnmIsIPXRouter , nnmIsRepeater , nnmIsHub
, nnmIsSwitch , nnmIsDevice , nnmIsPrinter , nnmIsAnalyzer , nnmIsIP , nnmIsIPX ,
nnmIsSNMPSupported , nnmIsSNMPProxied , nnmIsNetwareServer , nnmIsMcClusterMember ,
nnmIsCollectionStationNode , nnmIsRDMISupported , nnmIsHTTPSupported ,
nnmIsHTTPManaged , nnmIsRMON , nnmIsRMON2 , nnmIsDS1 , nnmIsDS3 , nnmIsFrameRelay ,
nnmIsSONET , nnmIsATM , nnmIsCDP);

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=NodesCap, hpovcn=Topology, ...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=NodesCap,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe45
00";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_NODESCAP[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);
}


#===============================================================
# sub NNM_NODESNETS()
#
# Creates nnm_nodes_net entrys of LDIF file.
#===============================================================
```

```
sub NNM_NODESNETS {
my $COUNTER= 1;
my $LAST_VAL;
my $FILE_NAME = "nnmNodesNets.dump";
my $CLASS_NAME = "nnmNodesNets";
my $MUST = "nnmEntryID";
my $COUNTER3 = 0;    # Counter for the Index of the array @OBJECT
my @OBJECT;          # Array with a temporary nnm_nodes_net object
my @DESC_NNM_NODESNETS = ( nnmNodeName , nnmNetIPAddress);
my %COUNT_ENTRIES;   # Hash with the number of accurences of each nnmNodeName
my %SHOWN_ENTRIES;   # Copy of %COUNT_ENTRIES because %COUNT_ENTRIES will be
decremented

open (FHI, "<$FILE_NAME") or ((close (FHO)) and (die "Impossible to open the file
$FILE_NAME"));

while (<FHI>) {
@OBJECT = split (/\t/,);
chomp @OBJECT;

$\ = "";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
$\ = "\n";
print "!" if (($COUNTER1 % $MULTIPLE) == 0);
print FHO "# Entry $COUNTER2: $MUST=$COUNTER, hpovcn=NodesNets, hpovcn=Topology,
...";
print FHO
"dn:$MUST=$COUNTER,hpovcn=NodesNets,hpovcn=Topology,hpovcn=NNM,nmcn=HPOpenView,dc=xe4
500";
print FHO "$MUST:$COUNTER";
foreach (@OBJECT) {
       $OBJECT[$COUNTER3] = "" if ($OBJECT[$COUNTER3] =~ /(\s)*?none(\s)*?/ig);
       $OBJECT[$COUNTER3] = "" unless ($OBJECT[$COUNTER3] =~ /(\S)+/ig);
       s/(<)+//ig;
       s/(>)+//ig;
       print FHO "$DESC_NNM_NODESNETS[$COUNTER3]: $OBJECT[$COUNTER3]" unless
($OBJECT[$COUNTER3] eq "");
       $COUNTER3++;
       }
$COUNTER++;
$COUNTER1++;
$COUNTER2++;
$COUNTER3 = 0;

print FHO "objectClass: $CLASS_NAME";
print FHO "objectClass: top\n";
}

close (FHI);

}
```

# ANEXO C

Neste anexo pode ser encontrado o *script* que foi criado para retirar a informação necessária proveniente do ficheiro obtido pela procura da informação referente à topologia já contida no serviço de directório LDAP. O resultado da execução do *script* vai ser o ficheiro de *hosts* utilizado pelo Nagios.

### SCRIPT PARA CRIAR FICHEIRO DE HOSTS DO NAGIOS

```
#!/usr/bin/perl
#=====================================================================
# Description:
#
# File parser for the file obtained from searching
# the LDAP tree for nodes.
#=====================================================================

$/ = "\n";    # Input record separator is Carriage Return
$\ = "\n";    # Output record separator is Carriage Return


#=====================================================================
# Main function
#
# Has the minimum configuration, all the necessary subroutines
# are called from here.
#=====================================================================
{

       $COUNTER1 = 0;       # Holds the number of nodes (hosts) parsed
       $OUTPUT_FILE = "nagioshosts.cfg";        # The default output filename
       $INPUT_FILE = "ldapsearch.dump";         # The default intput filename
       @DATE_HOUR;   # Array containning the elements of current date and time

       my @DURATION = times;       # Holds the time spent running the script

       &CHECK_OPTIONS();

       open (FHO, ">$OUTPUT_FILE") or die ("Impossible to open file $OUTPUT_FILE for
output!");

       &FILE_HEADER();
       &FILE_BODY();

       close (FHO);

       print "\n\nNagios Hosts file created.\n$COUNTER1 hosts parsed in $DURATION[0]
seconds.";
}


#=====================================================================
# sub CHECK_OPTIONS()
#
# Subroutine to check the options (switches) passed as
# arguments on the command line.
#=====================================================================

sub CHECK_OPTIONS {

       my $PHRASE = "$0: invalid option\nUsage: $0 [-i input_file] [-o output_file]";
       # Phrase to print when incorrect options are used
```

```
	my $PHRASE2 = "$0: Usage: $0 [-i input_file] [-o output_file]";
			# Phrase to print to inform about right options

	if ( $ARGV[0] ) {
		my $FIRST = $ARGV[0];
		my $SECOND = $ARGV[2];
		if ( !$ARGV[1] ) {
			print $PHRASE;		# No valid First Att.
			exit;
		}
		if ( $FIRST ne "-o" ) {
			if ( $FIRST ne "-i" ) {
				print $PHRASE;
				exit;
			}
			if ( $ARGV[1] ) {
				# First ARG is equal to -i and has valid file name
				$INPUT_FILE = $ARGV[1];
			}
			if ( $ARGV[2] ) {
				$SECOND = $ARGV[2];
				if ( !$ARGV[3] ) {
					print $PHRASE;		# No valid Second Att.
					exit;
				}
				if ( $SECOND ne "-o" ) {
					print $PHRASE;		# Not a valid Second ARG
					exit;
				}
				# Second ARG is equal to -o and has valid file name
				$OUTPUT_FILE = $ARGV[3];
				goto DONE_ARGS;
			}
		# Just has First ARG -i, no Second ARG
		goto DONE_ARGS;
		}
		# First ARG is equal to -o and has valid file name
		$OUTPUT_FILE = $ARGV[1];
		if ( $ARGV[2] ) {
			$SECOND = $ARGV[2];
			if ( $SECOND ne "-i" ) {
				print $PHRASE;		# Not a valid Second ARG
				exit;
				}
				# Second ARG is equal to -i
			if ( $ARGV[3] ) {
				# Second ARG is equal to -i and has valid file name
				$INPUT_FILE = $ARGV[3];
			}
		}
	}

	DONE_ARGS:
	print "Using output file $OUTPUT_FILE..." if ( $OUTPUT_FILE ne
"nagioshosts.cfg" );
	print "Using default output file $OUTPUT_FILE..." if ( $OUTPUT_FILE eq
"nagioshosts.cfg" );
	print "Using input file $INPUT_FILE..." if ( $INPUT_FILE ne "ldapsearch.dump"
);
	print "Using default input file $INPUT_FILE..." if ( $INPUT_FILE eq
"ldapsearch.dump" );
}


#====================================================================
# sub FILE_HEADER()
#
# Creates the header and common parts of the nagios hosts file.
#====================================================================

sub FILE_HEADER {

&MY_TIME();
my $USER = getlogin;

print FHO "#####################################################################";
print FHO "# Hosts definition file generated by Perl script from LDAP tree search.";
```

```
print FHO "# Script ran by user $USER.";
print FHO "# Time of creation: $DATE_HOUR[4] $DATE_HOUR[3], $DATE_HOUR[5]
$DATE_HOUR[2]:$DATE_HOUR[1]:$DATE_HOUR[0]";
print FHO "##############################################################################";
print FHO "";
print FHO "# Generic host definition template";
print FHO "define host{";
print FHO "\tname\t\t\t\tgeneric-host\t;\tThe name of this host template - referenced
in other host definitions, used for template recursion/resolution";
print FHO "\tnotifications_enabled\t\t1\t;\tHost notifications are enabled";
print FHO "\tevent_handler_enabled\t\t1\t;\tHost event handler is enabled";
print FHO "\tflap_detection_enabled\t\t1\t;\tFlap detection is enabled";
print FHO "\tprocess_perf_data\t\t\t1\t;\tProcess performance data";
print FHO "\tretain_status_information\t1\t;\tRetain status information across
program restarts";
print FHO "\tretain_nonstatus_information\t1\t;\tRetain non-status information across
program restarts";
print FHO "\tcheck_command\t\tcheck-host-alive\t; Used to check if a host is UP if no
services are UP";
print FHO "\tmax_check_attempts\t10";
print FHO "\tnotification_interval\t120";
print FHO "\tnotification_period\t24x7";
print FHO "\tnotification_options\td,u,r";
print FHO "";
print FHO "\tregister\t\t\t0\t; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST,
JUST A TEMPLATE!";
print FHO "\t}";
print FHO "";
}


#==========================================================================
# sub MY_TIME()
#
# Gets date and time in a standard format.
#==========================================================================

sub MY_TIME {

my $SUM_DATE_HOUR;

$SUM_DATE_HOUR = time;      # Time in seconds since January 1, 1970
@DATE_HOUR = gmtime $SUM_DATE_HOUR;      # Transforms time in seconds in GMT time
if ( $DATE_HOUR[2] < 10 ) { $DATE_HOUR[2] = "0$DATE_HOUR[2]"; }     # Adds a zero to
the hour if < 10
if ( $DATE_HOUR[1] < 10 ) { $DATE_HOUR[1] = "0$DATE_HOUR[1]"; }     # Adds a zero to
the minutes if < 10
if ( $DATE_HOUR[0] < 10 ) { $DATE_HOUR[0] = "0$DATE_HOUR[0]"; }     # Adds a zero to
the seconds if < 10
if ( $DATE_HOUR[3] < 10 ) { $DATE_HOUR[3] = "0$DATE_HOUR[3]"; }     # Adds a zero to
the day if < 10
$DATE_HOUR[4] = $DATE_HOUR[4]+1;  # Adjusts month from (1..11) to (1..12)

##-----------------------------------------------------------------------
# Transforms numeric month in name month
if ( $DATE_HOUR[4] == 1 ) { $DATE_HOUR[4] = "January"; }
if ( $DATE_HOUR[4] == 2 ) { $DATE_HOUR[4] = "February"; }
if ( $DATE_HOUR[4] == 3 ) { $DATE_HOUR[4] = "March"; }
if ( $DATE_HOUR[4] == 4 ) { $DATE_HOUR[4] = "April"; }
if ( $DATE_HOUR[4] == 5 ) { $DATE_HOUR[4] = "May"; }
if ( $DATE_HOUR[4] == 6 ) { $DATE_HOUR[4] = "June"; }
if ( $DATE_HOUR[4] == 7 ) { $DATE_HOUR[4] = "July"; }
if ( $DATE_HOUR[4] == 8 ) { $DATE_HOUR[4] = "August"; }
if ( $DATE_HOUR[4] == 9 ) { $DATE_HOUR[4] = "September"; }
if ( $DATE_HOUR[4] == 10 ) { $DATE_HOUR[4] = "October"; }
if ( $DATE_HOUR[4] == 11 ) { $DATE_HOUR[4] = "November"; }
if ( $DATE_HOUR[4] == 12 ) { $DATE_HOUR[4] = "December"; }
#-----------------------------------------------------------------------

$DATE_HOUR[5] = $DATE_HOUR[5]+1900;

}


#==========================================================================
# sub FILE_BODY()
#
```

```
# Creates the body of the nagios hosts file.
#=====================================================================

sub FILE_BODY {

my $HOSTNAME;        # Will hold the hostname of the node being parsed
my $ALIAS;           # Will hold the alias of the node being parsed
my $ADDRESS;         # Will hold the IP address of the node being parsed
my $DATA_HOST = 0;   # Keeps the state of needed info for each host definition
my $DATA_ALIAS = 0;  # Keeps the state of needed info for each alias definition
my $DATA_ADDRESS = 0;       # Keeps the state of needed info for each address
definition

open (FHI, "<$INPUT_FILE") or ( (close (FHO)) and ((die "Impossible to open the file
$INPUT_FILE")) );

while (<FHI>) {
       unless ( /^#/ ) {
              if ( /(nnmIPHostname:)([\s\t]*)(.+)/ig ) {
                     $HOSTNAME = $3;
                     $DATA_HOST = 1;
                     }
              if ( /(nnmSNMPSysDescr:)([\s\t]*)(.+)/ig and ($3 ne "0.0.0.0") ) {
                     $ALIAS = $3;
                     $DATA_ALIAS = 1;
                     }
              if ( /(nnmSNMPAddr:)([\s\t]*)(.+)/ig ) {
                     $ADDRESS = $3;
                     if ($ADDRESS eq "0.0.0.0") {
                            $DATA_HOST = 0;
                            $DATA_ALIAS = 0;
                            }
                     else {
                            $DATA_ADDRESS = 1;
                            }
                     }
              if ( $DATA_HOST == 1 and $DATA_ADDRESS == 1 ) {
                     print FHO "# '$HOSTNAME' host definition";
                     print FHO "define host{";
                     print FHO "\tuse\t\t\t\tgeneric-host\t\t; Name of host template
to use";
                     print FHO "";
                     print FHO "\thost_name\t\t$HOSTNAME";
                     if ( $DATA_ALIAS ==1 ) {
                            print FHO "\talias\t\t\t$ALIAS";
                            }
                     else {
                            $HOSTNAME =~ /([\w-]+)/i;
                            $ALIAS = $1;
                            print FHO "\talias\t\t\t$ALIAS";
                            }
                     print FHO "\taddress\t\t\t$ADDRESS";
                     print FHO "\t}";
                     print FHO "";
                     print FHO "";

                     $DATA_HOST = 0;
                     $DATA_ALIAS = 0;
                     $DATA_ADDRESS = 0;
                     $COUNTER1++;
                     }
              }
       }

close (FHI);
}
```

# BIBLIOGRAFIA

[1] M. Wahl, T. Howes, S. Kille. "Lightweight Directory Access Protocol", Internet RFC 2251. *IETF*, December 1997.

[2] G. Good. "The LDAP Data Interchange Format (LDIF) - Technical Specification", Internet RFC 2849. *IETF*, June 2000.

[3] Information Sciences Institute of University of Southern California. "Internet Protocol", Internet RFC 791. *IETF*, September 1981.

[4] Yuri Breitbart, Minos Garofalakis, Cliff Martin, Rajeev Rastogi, S. Seshadri, Avi Silberschatz. "Topology Discovery in Heterogeneous IP Networks". *IEEE Press*, June 2004.

[5] Giovanni Vigna, Fredrik Valeur, Jingyu Zhou, Richard A. Kemmerer. "Composable Tools for Network Discovery and Security Analysis". *Proceedings of the 18th Annual Computer Security Applications Conference, IEEE Computer Society*, December 2002

[6] R. Siamwalla, R. Sharma, S. Keshav. "Discovering Internet Topology". *Submitted to Infocom'99*, 1999.

[7] G. Kessler, S. Shepard. "A Primer On Internet and TCP/IP Tools and Utilities", RFC 2151. *IETF*, June 1997.

[8] J. Postel. "Internet Control Message Protocol", RFC 792. *IETF*, September 1981.

[9] CCITT. "Data Communication Network - Management Framework for Open Systems Interconnection (OSI) for CCITT applications", Recommendation X.700. *ITU-T*, September 1992.

[10] CCITT. "Data Communication Network - Management Framework for Open Systems Interconnection (OSI) for CCITT applications", Recommendation X.701. *ITU-T*, January 1992.

[11] Y. Yemini. "The OSI Network Management Model". *IEEE Communications Magazine*, 1993.

[12] J. Case, M. Fedor, M. Schoffstall, J. Davin. "A Simple Network Management Protocol (SNMP)", RFC 1157. *IETF*, May 1990.

[13] P. Mockapetris. "Domain Names – Concepts and Facilities", RFC1034. *IETF*, November 1987.

[14] R. Droms. “Dynamic Host Configuration Protocol”, RFC 2131. *IETF*, March 1997.

[15] Wikipedia Open Source definition. http://en.wikipedia.org/wiki/Open_source. March 2005.

[16] Hewlett-Packard. “HP OpenView Network Node Manager: Managing Your Network with HP OpenView Network Node Manager”. *Hewlett-Packard*, September 2004.

[17] Hewlett-Packard Network Node Manager homepage. http://www.managementsoftware.hp.com/products/nnm/index.html. 2005.

[18] Tcl/tk Wiki homepage. http://wiki.tcl.tk/691. November 2004.

[19] Scotty homepage. http://wwwhome.cs.utwente.nl/~schoenw/scotty/. February 2000.

[20] Scotty documents homepage. http://wwwhome.cs.utwente.nl/~schoenw/scotty/docs/getstart.html. July 1996.

[21] Solarwinds homepage. http://www.solarwinds.net/Tools/Engineer/index.htm. 2005.

[22] WhatsUp homepage. http://www.ipswitch.com/Products/WhatsUp/professional/index.html. 2004.

[23] SNMPc homepage. http://www.castlerock.com/products/snmpc/default.php. 2004.

[24] LAN Surveyor homepage. http://www.neonsoftware.com/LSwin.shtml. 2004.

[25] ManageEngine OpManager homepage. http://manageengine.adventnet.com/products/opmanager/index.html. 2005.

[26] OpenNMS homepage. http://wiki.opennms.org/tiki-index.php. February 2005.

[27] Cheops-ng homepage. http://cheops-ng.sourceforge.net/. May 2003.

[28] Information Sciences Institute of University of Southern California. RFC 793: Transmission Control Protocol. *IETF*, September 1981.

[29] Hewlett-Packard. "HP OpenView Network Node Manager: Performance and Configuration Guide". *Hewlett-Packard*, February 2004.

[30] Hewlett-Packard. HP OpenView Network Node Manager: Reporting and Data Analysis with Network Node Manager. *Hewlett-Packard*, September 2003.

[31] Wikipedia Perl definition. `http://en.wikipedia.org/wiki/Perl`. March 2005.

[32] The Perl Directory. `http://www.perl.org/`. 2005.

[33] Wikipedia MIB definition. `http://en.wikipedia.org/wiki/Management_information_base`. January 2005.

[34] M. Wooldridge, N.R. Jennings. Intelligent Agent: Theory and Practice. *The Knowledge Engineering Review*, 1995.

[35] Raul Filipe Teixeira de Oliveira. Managing Awareness in Networks Through Software Agents. *PhD thesis, Ecole Nationale Supérieure des Télécomunications*, 1998.

[36] OpenLDAP homepage. `http://www.openldap.org/`. January 2005.

[37] Nagios homepage. `http://www.nagios.org/`. March 2005.