INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Netcamera: Ein Dienst zur Visualisierung von Backbone-Topologiedaten

Changyuan Qiu

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering
Betreuer: Dipl.-Inf. Timo Baur

Abgabetermin: 20. April 2006

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Netcamera: Ein Dienst zur Visualisierung von Backbone-Topologiedaten

Changyuan Qiu

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering
Betreuer: Dipl.-Inf. Timo Baur

Abgabetermin: 20. April 2006

Hiermit versichere ich, dass ich die vorliegende Auarbeitung selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 20. April 2006

. . . . . .

(Unterschrift des Kandidaten)

Das in dieser Arbeit behandelte Thema ist die Erstellung eines Dienstes zur visuellen 3D-Darstellung von Backbone-Topologiedaten eines Netzwerkes, die mithilfe des Frameworks Neve (Network Visualization Environment) realisiert wird. Die aktuellen Topologiedaten werden in einem Datenbank-Server namens Nyx gespeichert. Der Datenbank-Server bietet nach aussen eine XML-Anfrage Schnittstelle an. D.h., eine Anfrage wird im XML-Format gestellt, das Ergebnis im XML-Format wird mit einem XML-Parser bearbeitet und die Netztopologieinformationen daraus extrahiert. Als Anfrageparameter dienen die IP-Adresse des zu beobachteten Endsystemes und eine Integerzahl für die Rekursionstiefe. Je grösser die Integerzahl ist, umso grösser ist die ausgegebene Netztopologie. Nach der Bearbeitung durch NetCamera wird die Netztopologie ähnlich wie in einer Kamera angezeigt. Man kann die Topologie von bestimmtem Endsystemen und deren Umgebung dann dreidimensional dargestellt betrachten. Zur Erreichung dieses Ziels müssen folgende Aufgaben durchgeführt werden: Erstens, das Informationsmodell von Neve muss um Attribute aus Nyx erweitert werden. Zweitens, ein Adapter für die Abbildung zwischen Nyx und Neve ist zu entwerfen, um managementrelevante Attribute auf der Managementplattform speichern zu können. Drittens, managementrelevante Informationen müssen aus den Daten aus Nyx extrahiert werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung und Aufgabenstellung

Computernetze sind in unserem heutigen Leben nicht mehr wegzudenken und spielen eine immer wichtigere Rolle. Die Netzstrukturen und damit das Netzmanagement werden auch immer komplexer. Um Managementinformation über Netztopologie, Netzkonfiguration, Netzkomponenten und deren Eigenschaften schnell und einfach zu erhalten, können Visualisierungen nützlich sein. Visuelle 3D-Darstellung von Backbone-Topologiedaten eines Netzwerkes und bequemer Zugriff auf Managementinformation sind sowohl für Netzadministratoren als auch für Lehre und Aussendarstellung sehr interessant und nützlich. Aufgabe dieser Arbeit ist es, einen Dienst mithilfe einer Managementplattform zu realisieren, der Backbone-Topologiedaten visualisiert und Managementinformation über Netzkonfiguration, Netzkomponenten und deren Eigenschaften auf einer benutzerfreundlicher Weise liefert.

2 Ausgangssituation

Als Rahmenwerk und Managementplattform mit Visualisierungsfunktionalitäten steht Neve (Network Visualization Environment) zur Verfügung. Backbone-Topologiedaten stellt ein Datenbank-Server namens Nyx bereit. Im Folgenden sollen die beiden Werkzeuge vorgestellt werden.

2.1 Neve Vorstellung

Neve ist ein Applicationserver mit Diensten zur Visualisierung und Management vom Rechnernetzen. Es ist ein Rahmenwerk, um die beim Management von Rechnernetzen anfallenden Daten dreidimensional und dynamisch zu visualisieren, wobei der Anwender die Möglichkeit hat, durch die erzeugte Darstellung zu navigieren und mit ihr zu interagieren. Neve hat eine Komponentenarchitektur und ist als Client-Server Anwendung realisiert. Die wichtigsten verwendeten Technologien von Neve sind Java 3D und die Java Management Extensions JMX [JMX]. Als Implementierung von JMX wurde die open-source Bibliothek mx4j [mx4j] benutzt. mx4j stellt einen Application Server bereit, indem die funktionalen Einheiten des Programms sowie Managementobjekte, die das Netz beschreiben, gehalten werden. Der Server holt Netzdaten, bereitet diese auf und legt durch eine Visualisierungspipeline die Darstellung aller gespeicherten Managementobjekte fest, wobei die Ausgabe Java3D-Objekte sind. Diese werden dann anhand den Relationen (z.B. *enthält*,*verbunden-mit*) zwischen den Managementobjekten durch ein Layout-Service organisiert und später im Client zu einer Java3D-Szene zusammengefügt. Ein Client hat dann die Aufgabe, die vom Server gelieferten Daten dreidimensional zu repräsentieren. Die Interaktion bzw. Netzwerkverwaltung mit Neve kann man durch ein formularbasiertes Browserinterface erledigen, das von mx4j bereitgestellt wird oder über den Client.

Als offene Managementplattform ist Neve in der Lage, Daten aus verschieden Quellen und in verschiedenen Formaten in das Informationsmodell zu integrieren, z.B aus Logfiles oder Datenbanken. Es kann mit Adaptoren zwischen Neve und der Datenquelle erreicht werden (siehe Abbildung 2.1). Dies nennt man Integration durch einen Proxy (siehe [HAN 99] Seite 223).

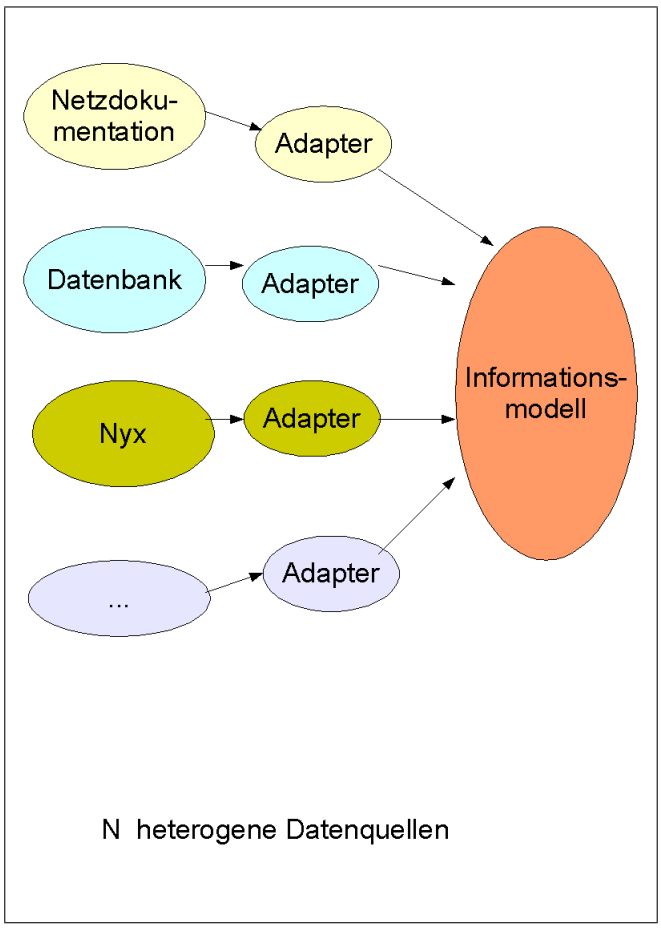

Abbildung 2.1: Informationsmodell: n zu 1 Abbildung

In unserem Fall kommen die Daten aus dem Datenbankserver Nyx (siehe 2.2). Die Daten liegen im XML-Format vor und besitzen mehrere Nyx-spezifische Attribute. Zur Integration von Nyx-Daten muss das Informationsmodell von Neve erweitert werden. Bisher gibt es nur ein Attribut: die IP eines Endsystems. Das Informationsmodell von Neve lässt sich sehr einfach erweitern. Z.b. können auch Attribute aus CIM (Common Informationsmodell) und SNMP Management Information Base hinzugefügt werden.Wenn man einmal die Attributmenge *N* des Neve-Informationsmodells genug erweitert hat, braucht man diese Arbeit bei anderen Anwendungen, die Neve benutzen und deren Attributmenge *A* eine Teilmenge von *N* ($A <= N$) ist, nicht mehr durchzuführen.

Wir betrachten jetzt die Datenquelle Nyx.

2.2 Nyx Kurzvorstellung

Nyx ist die Fopra-Arbeit von Stefan Fischer [Fis05]. Nyx ist ein Datenbank-Server, der sich im Netz befindet und per TCP/IP erreichbar ist. In Nyx werden die aktuellen Topologiedaten gespeichert. Nyx bietet nach aussen eine XML Anfragenschnittstelle an, d.h, eine Anfrage wird im XML-Format gestellt, das Ergebnis wird auch im XML-Format zurückgeliefert. Als Anfrageparameter dienen die IP-Adresse des zu beobachtenden Endsystemes und eine Integerzahl für Rekursionstiefe. Je grösser die Integerzahl ist, umso grösser ist die ausgegebene Netztopologie. Im Folgenden wird die Anfragestruktur an Nyx betrachtet.

3 Analyse der Anfrage-Struktur in der Datenquelle

Um die Netztopologie rekonstruieren zu können, werden Informationen benötigt, die besagen, welche Komponente mit welcher anderen verbunden ist. Information über die Grösse einer Topologie wird über die Rekursionstiefe ausgedrückt.

Nun betrachten wir, wie die Rekursion in der Anfrage ausgedrückt werden kann.

3.1 Beispielanfrage

Eine Anfrage an Nyx für ein Endsystem mit der IP 10.187.160.1 hat bei einer Rekursionstiefe von 3 folgende Struktur:

```
<query>
    <filter id="link">
        <include attr="linkdevice" value="" compare="!eqs"/>
    </filter>
    <filter id="dev">
        <include attr="ip" value="10.187.160.1" compare="eqs"/>
    </filter>
    <device filter="dev">
        <interface filter="link">
            <link>
                <interface filter="link">
                    <link>
                        <interface filter="link">
                            <link>
                            </link>
                        </interface>
                    </link>
                </interface>
            </link>
        </interface>
    </device>
</query>
```

Abbildung 3.1: Anfrage an Nyx bei Rekusionstiefe 3

Durch diese Abfrage wird nach der Information gefragt: welche Endsysteme sind mit dem Endsystem mit der IP 10.187.160.1 direkt und indirekt verbunden.

Um die indirekten Verbindungen feststellen zu können, wird Rekursionstiefe benötigt. Durch Rekursionstiefe wird die Grösse der ausgegebenen Netzumgebung festgelegt.

Eine Rekurson in der Anfrage wird ausgedrückt dadurch, dass das Kindelement *link* des Elements *interface* wieder ein Kindelement *interface* hat.

3.2 XML Ergebnis-Struktur und Elemente

Nun betrachten wir das Anfragenergebnis und dessen Struktur und Elemente (siehe Abbildung 3.2).

```
<result>
    <device >
            <interface >
                <device/>
            </interface >
            <interface >
                <device/>
            </interface >
            <interface >
                <device/>
            </interface >
          .....
    </device >
<result>
```

Abbildung 3.2: Struktur des Anfragenergebnisses

Das Wurzelelement von Anfragenergebnis (XML-Stream) ist *result*, dessen einziges Kindelement ein *device* Element ist. Dieses *device* Element hat (kann) mehrere *interface* Element als Kindelement (haben), wobei ein *interface* Element wieder *device* als Kindelement hat.

Betrachten wir das *device* Element des Anfragenergebnisses im XML-Format (siehe Abbildung 3.3):

```
<device cdp="1" checklayer2="1" checklayer3="0" created="Dec 6, 2005 10:21:30
AM" device="HPJ4865ASG22860979" dns="swg1-0ab.net.lrz-muenchen.de"
faultcount="0" ip="10.187.160.1" lastupdate="Dec 8, 2005 2:19:48 PM" layer2="1"
layer3="0" syscontact="LRZ" sysdescription="HP J4865A ProCurve Switch 4108GL,
revision G.07.70, ROM G.05.02 (/sw/code/build/gamo(m03))" syslocation="TUM
Garching Beschleunigerlabor Bau 5120" sysname="swg1-0ab"
sysobjectid=".1.3.6.1.4.1.11.2.3.7.11.23" sysuptime="46 days, 1 hours, 31 minutes, 22
seconds.">
```

Abbildung 3.3: Device Element

Das Attribut *device* ist ein eindeutiger String, der später im Bearbeitungsalgorithmus als Schlüssel für eine HashMap benutzt wird.

Betrachten wir das *interface* Element des Anfragenergebnisses im XML-Format (siehe Abbildung 3.4):

```
<interface cdp="1" created="Dec 6, 2005 10:21:48 AM"
device="HPJ4865ASG22860979" ifadminstatus="1" ifalias="" ifdescription="A1"
ifoperstatus="1" ifspeed="1,0 Gigabit/s" interface="1" lastupdate="Dec 8, 2005
2:19:59 PM" linkdevice="HPJ4887ASG330AD04A">
```

Abbildung 3.4: Interface Element

Das Attribut *device* besagt, zu welchem Device das *interface* gehört. Das Attribut *linkdevice* besagt, mit welchem Device das *interface* verbunden ist.

Nachdem wir uns mit dem Werkzeug zur Visualisierung und dem Management der Daten und der Datenquelle beschäftigt haben, können wir anfangen, die für uns nützlichen Topologie-und Managementinformationen aus der Datenquelle zu extrahieren und die Netztopologie zu rekonstruieren.

4 Durchführung

4.1 Lösungsansatz

Wie oben gezeigt, sind drei Schritte notwendig :

1. Erweiterung des Informationsmodells von Neve um Attribute aus Nyx.
2. Entwurf eines Adapters für die Abbildung zwischen Nyx und Neve. Dies kann realisiert werden, indem man einen Adapter zwischen Neve und Nyx programmiert. Dieser Adapter greift durch setter-Funktionen auf managementrelevante Attribute zu.
3. Extrahieren der Daten aus Nyx.

4.2 Erweiterung des Informationsmodells

Folgendes Klassendiagramm zeigt die Erweiterung des Informationsmodells. Diese wird durch Vererbung realisiert (siehe Abbildung 4.1).

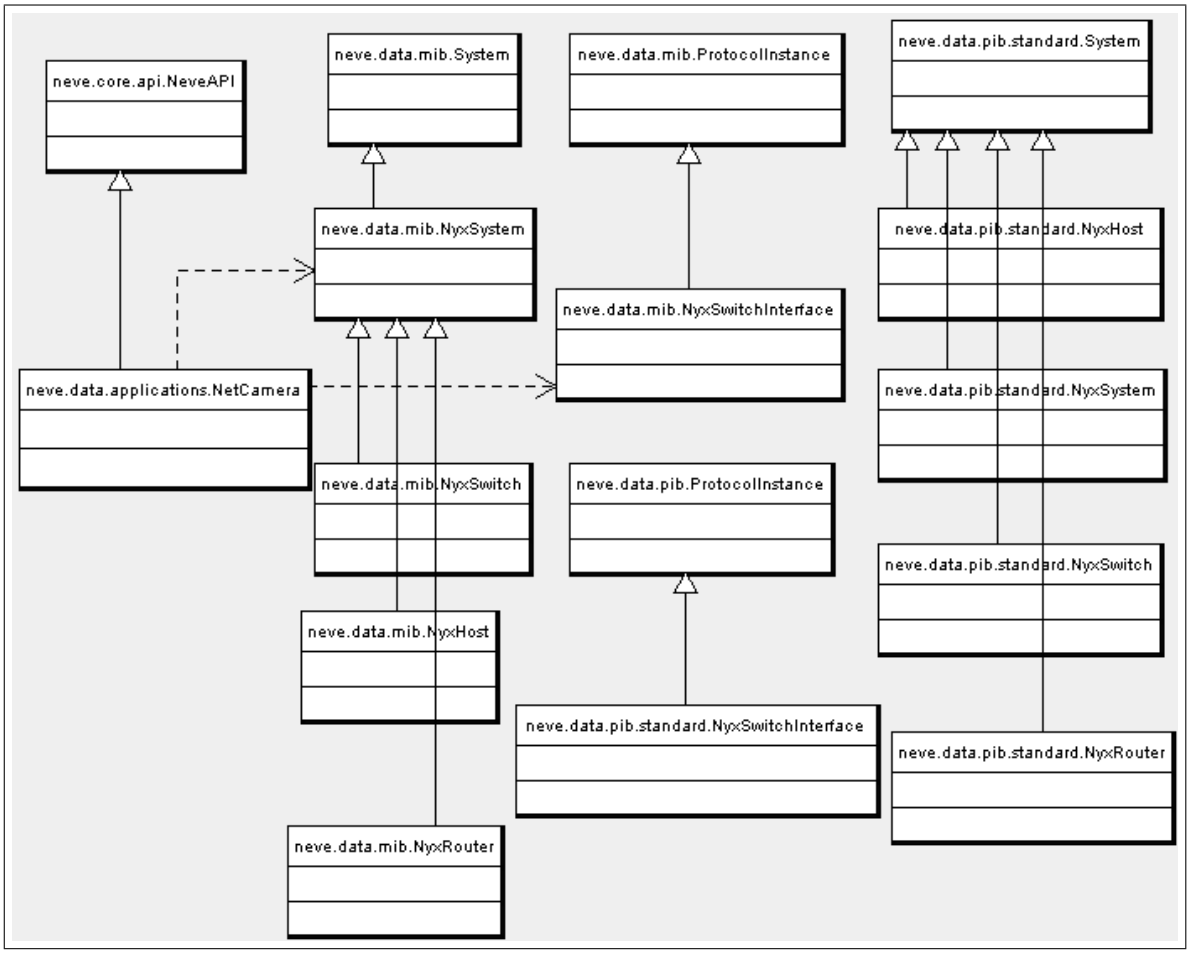

Abbildung 4.1: Erweiterung des Informationsmodells

Das Informationsmodell wird um folgende Attribute aus Nyx erweitert (siehe Tabelle 4.2):

NyxSystem	NyxSwitchInterface
sysdescription	ifadminstatus
dns	ifoperstatus
sysobjectid	ifspeed
created	created
device	device
layer2	interface
layer3	linkdevice
cdp	cdp
checklayer2	ifalias
sysname	ifdescription
checklayer3	rec-marked
lastupdate	lastupdate
syscontact	
faultcount	
syslocation	
sysuptime	

NyxRouter, NyxHost und NyxSwitch sind von NyxSystem abgeleitet und haben die Attribute von NyxSystem automatisch geerbt. Für ein NyxSystem gilt: wenn das Attribut *layer2* auf true gesetzt ist und das Attribut *layer3* auch auf true gesetzt ist, dann wird daraus ein NyxRouter erzeugt. Wenn *layer2* true und *layer3* false ist,dann wird ein NyxSwitch erzeugt, sonst wird ein NyxHost erzeugt.

4.3 Entwurf eines Adapters

Um die in Abbildung 2.1 gezeigte N : 1 Abbildung des Informationsmodells zu realisieren, ist ein Adapter zwischen Neve und Nyx zu programmieren.

4.3.1 Abhängigkeiten des Adapters

Hier eine grobe Skizze (siehe Abbildung 4.2) für den Workflow des Systems, die die Funktion und Position von NetCamera als Adapter zeigt: ganz unten im Bild ist das zu visualisierende Netz. Nyx sammelt und speichert die Rohdaten vom Netz. NetCamera stellt Anfragen in XML-Format an Nyx und erhält Daten im XML-Format zurück. NetCamera extrahiert die Daten und speichert mithilfe der Neve-API managementrelevante Information auf der Managementplattform Neve. Neve führt die Visualisierung durch. Der Client erhält die visualisierte Netztopologie.

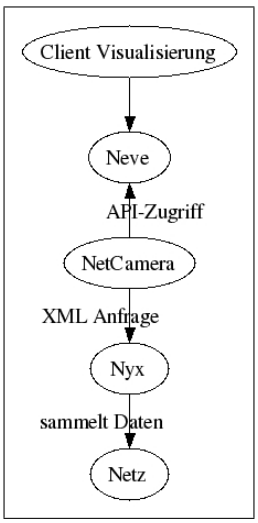

Abbildung 4.2: Ablauf in Kürze

4.3.2 Algorithmus

Um vom Anfragenergebnis (genau gesagt dem XML-Stream) die interesanten Informationen zu extrahieren, wird ein XML-Parser und ein effizienter Algorithmus benutzt. Der Algorithmus ist effizient in dem Sinn, dass jedes XML-Element nur einmal bearbeitet wird.

1. Initialisierung: eine HashMap *alldevices* für alle Devices, ein HashMap *verbindungen* für alle Verbindungen anlegen.
2. Hole Document Root Element.
3. Hole Kindelement von Root, das ist ein *Device* Element (siehe Abbildung 3.2).
4. Attribute wie *ip,dns,faultcont...* u.s.w. in diesem Objekt speichern, dieses Objekt anhand eindeutiges Attribut *device* (.z.B *HPJ4865ASG22860979*) in *alldevices* speichern.
5. Hole alle Kinderelemente von diesem *Device* Element, d.h. die *Interface* Elemente,die dem *Device* angehören (siehe Abbildung 3.2).
6. für jedes *Interface* ein *NyxSwitchinterface* Objekt erzeugen, Attribute *device*, *linkdevice* für dieses Objekt setzten.
7. anhand der eindeutigen Stringdarstellung von Attribut *linkdevice* (z.B. *HPJ4865ASG22860979*) dessen Verbindungen in HashMap *verbindungen* speichern.
8. Relation *enthält* bilden zwischen *NyxSystem* und *NyxSwitchInterface* (*NyxSystem* enthält *NyxSwitchInterface*). Wenn ein *Interface* Element Kinderelement hat, gehe zurück zu Schritt 4.
9. Alle XML Elemente sind nun durchgearbeitet. Die nötige Information ist gesammelt in den HashMaps *alldevices* und *verbindungen*.
10. Mithilfe von Neve-API Relation *verbunden-mit* bilden (durch eine Schleife HashMap *verbindungen* iterieren).

Der Benutzer bekommt von Neve anschlissend die in 3D visualisierte Netztopologie zu sehen. Wenn er mit der Maus auf ein Endsystem zeigt und auf die rechte Maustaste klickt, sieht er Managementinformationen wie z.B. *IP*, *DNS-Name*, *Faultcount* u.s.w.

5 Konkrete Implementierung in Java

Um die Visualisierungsfunktionalität von Neve benutzen zu können, müssen die Rohdaten durch eine Visualisierungspipeline in MPOs (Managementpräsentationsobjekt) transformatiert werden. Als erstes kommt dazu der Anwendungsservice zum Einsatz. Dieser ist für die Transformation der Rohdaten aus der Managementanwendung wie der NetCamera in MIOs (Managementinformationsobjekt) zuständig. Nach der Sammlung der Rohdaten durch eine Managementanwendung wie Nyx müssen die Objekte auf der Managementplattform durch das Informationsmodell beschrieben gespeichert werden. Dies ermöglicht die Neve API, die dazu die Funktion *createMIO()* zur Verfügung stellt. Die NetCamera erhält die Rohdaten im XML-Format, extrahiert sie und erzeugt Managementinformationsobjekte (MIO) durch den Methodenaufruf *createMIO (String type,String domain)*. Diese MIOs werden dann durch einen Mappingservice in MPOs transformatiert und schliesslich visualisiert.

Nun betrachten wir das NetCamera-Modul. Die Hauptklasse des NetCamera-Moduls ist NetCamera. Diese Klasse ist von neve.core.api.NeveAPI abgeleitet. Die Klasse NeveAPI bietet alle Funktionen des Anwendungsservice an und stellt alle Funktionalitäten bereit, die notwendig sind, um auf den Anwendungsservice zugreifen zu können. Somit unterstützt NeveAPI eine Managementanwendung bei der Herstellung der Verbindung mit dem Anwendungsservice.

Klassen, die sich im neve.data.mib Paket befinden und deren Name mit Nyx anfangen, erweitern das Informationsmodell von Neve um Attribute aus Nyx . Klassen, die sich im neve.data.pib Paket befinden und deren Name mit Nyx anfangen, erweitern das Präsentationsmodell von Neve entsprechend (Siehe Abbildung 4.1).

Ausserdem muss in der Klasse neve.server.BaseAgent (Serverklasse) NetCamera noch als MBean registriert werden:

Listing 5.1: Neve als MBean bei Applicationserver registrieren

```
NetCamera nc = new NetCamera ();
ObjectName nc_name = null;
try {
        nc_name = new ObjectName("Neve.Managementanwendung:name=NetCamera");
        System.out.println("\tOBJECT␣NAME␣␣␣␣␣␣␣␣␣␣=␣" + nc_name);
        server.registerMBean(nc, nc_name);
} catch(Exception e) {
        System.out.println("\t!!!␣Could␣not␣create␣the␣NetCamera␣!!!");
        e.printStackTrace();
        return;
}
```

5.1 JDOM als XML-Parser

JDOM [JDOM] ist ein open-source Projekt, bietet eine Schnittstelle für die Arbeit mit XML-Dokumenten an und basiert vollständig auf Java. Hierzu nutzt es spezifische Elemente dieser Programmiersprache wie die Standard-Kollektionen, Methodenüberladung und Reflection. Diese Vorgehensweise macht die Benutzung für Java-Programmierer intuitiv und leicht erlernbar. NetCamera greift zum Parsen auf JDOM zurück.

5.2 Quelltext

Quelltext siehe Anhang 7

6 Ergebnis/Tragfähigkeitsnachweis

Zum Nachweis der Funktion und Tragfähigkeit des implementierten Moduls wurden folgende Tests durchgeführt:

6.1 Der Routerbackbone des LRZ: 10.187.160.1, Rekursionstiefe 2

Visualisiert wird hier der Routerbackbone des LRZ mit IP 10.187.160.1 bei Rekursionstiefe 2 (siehe Abbildung 6.1). Es werden 10 Komponenten visualisiert, zwei davon sind Router (als Zylinder dargestellt), die anderen sind Switch (als Quader dargestellt). Im CIP-Pool hat es knapp eine Minute gedauert.

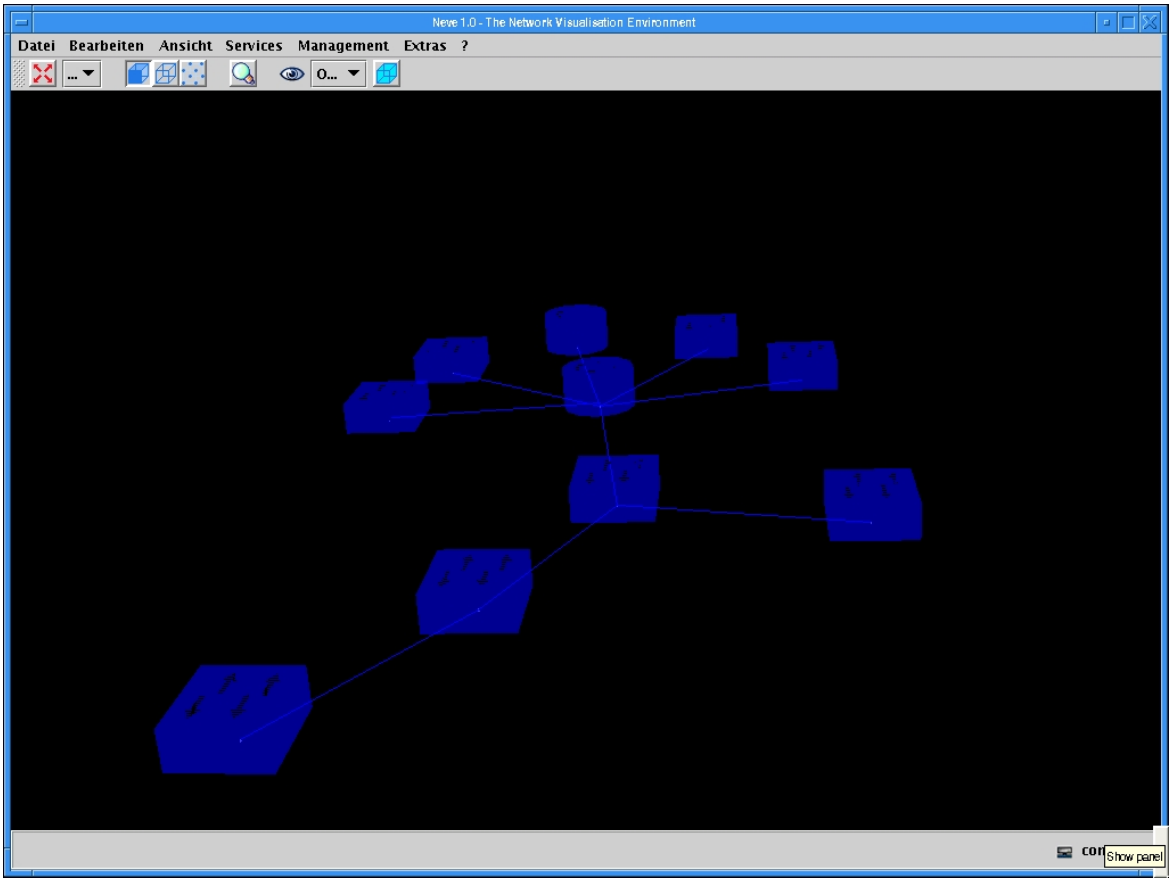

Abbildung 6.1: Der Routerbackbone des LRZ bei Rekursionstiefe 2

6.2 Der Routerbackbone des LRZ: 10.187.160.1, Rekursionstiefe 3

Es wird hier der Routerbackbone des LRZ mit IP 10.187.160.1 bei Rekursionstiefe 3 visualisiert (siehe Abbildung 6.2). Es werden 48 Komponenten visualisiert, zwei davon sind Router (als Zylinder dargestellt), die anderen sind Switch (als Quader dargestellt). Im CIP-Pool hat es ungefähr halbe Stunde gedauert.

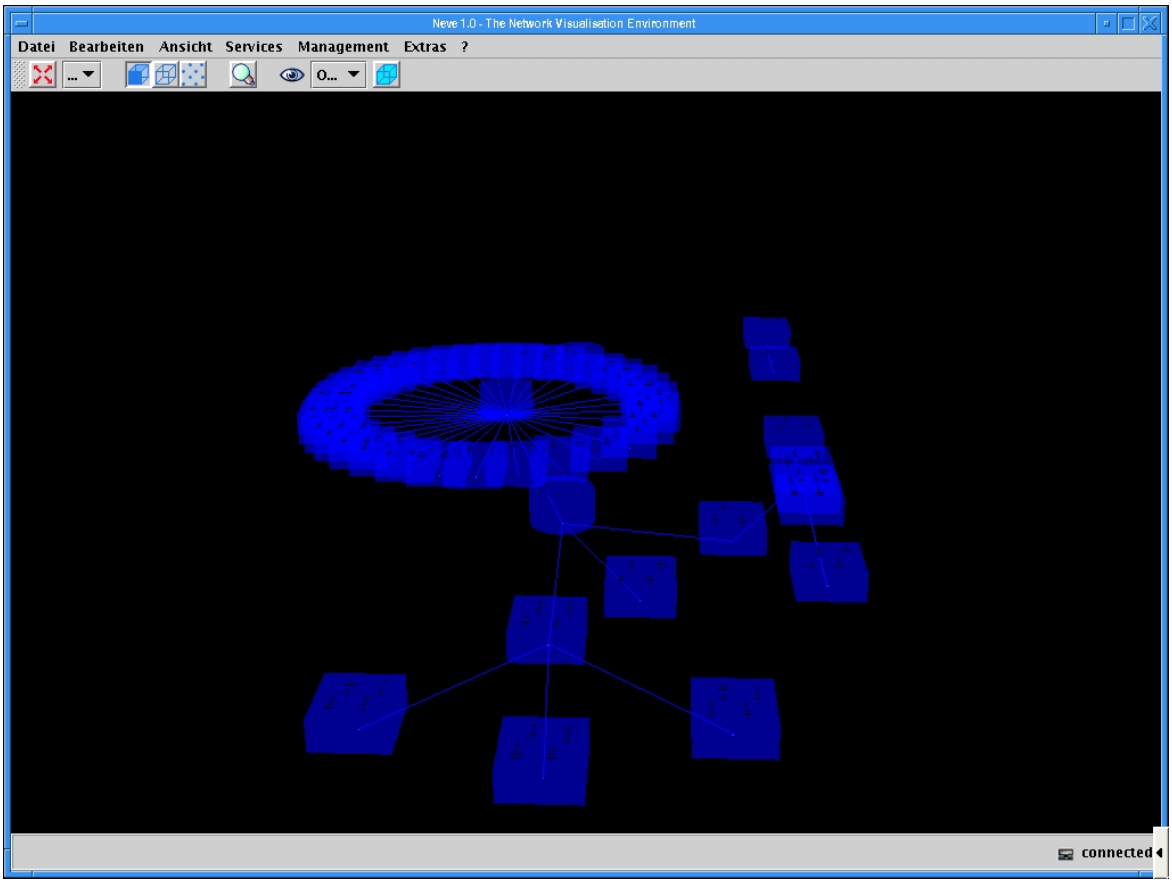

Abbildung 6.2: Der Routerbackbone des LRZ bei Rekursionstiefe 3

Wenn das Anfragenergebnis visualisiert dargestellt ist, hat der Benutzer nun die Möglichkeit, mit der rechten Maustaste auf ein Endsystem zu klicken, und so bekommt er Managementinformationen wie in Abbildung 6.3 zu sehen. So ist es nun möglich, Informationen über die Netzkonfiguration des LRZ , ihre Komponenten, und deren Eigenschaften erhalten zu können.

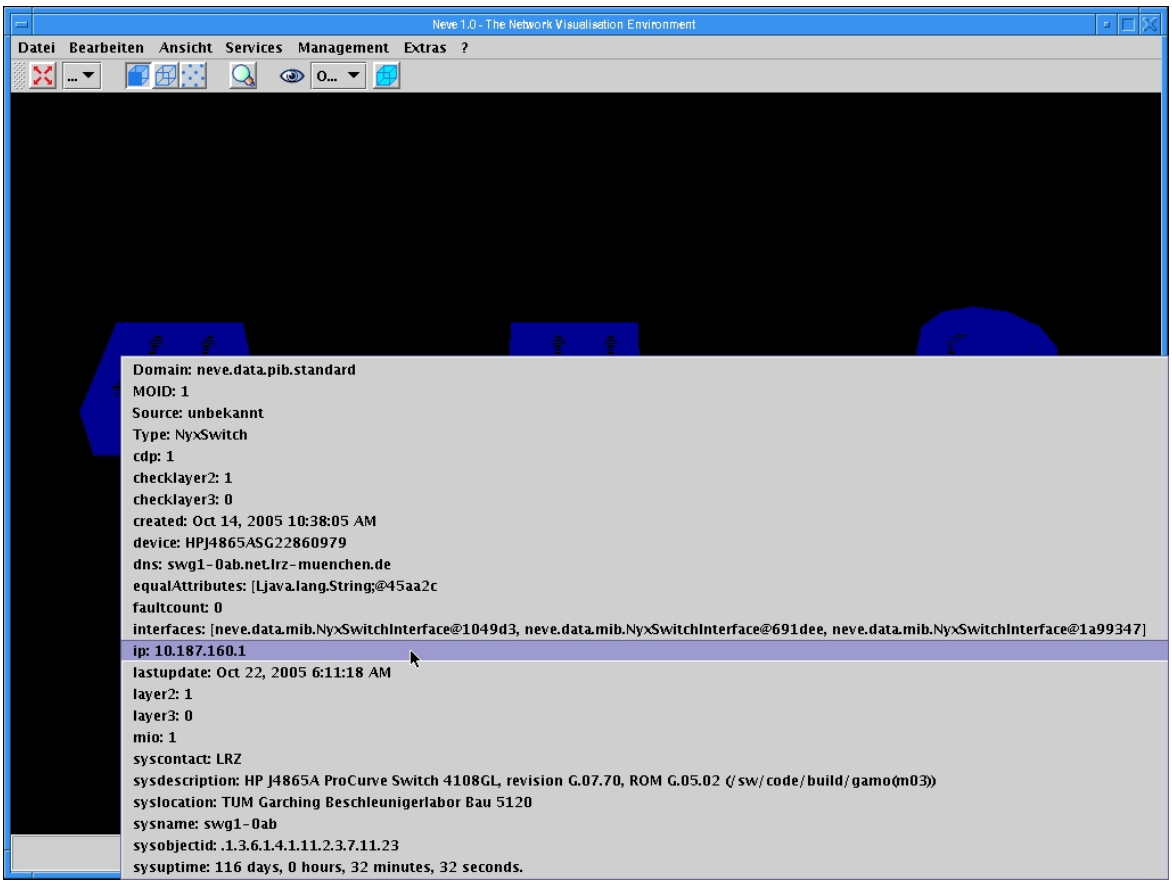

Abbildung 6.3: Managementinformationen bei rechtem Mausklick

Literaturverzeichnis

[Baur02] TIMO, BAUR: *Entwurf einer Architektur zur Integration von Netzplanungs- und managementwerkzeugen in eine VR-Umgebung*. 2002, http://www.nm.ifi.lmu.de/pub/Diplomarbeiten/baur02/PDF-Version/baur02.pdf .

[Fis05] STEFAN, FISCHER: *http://www.nm.ifi.lmu.de/pub/Fopras/#2005*.

[HAN 99] HEGERING, H.-G., S. ABECK und B. NEUMAIR: *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application*. Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999. 651 p.

[java] SUN: *http://java.sun.com/j2se/1.4.2/docs/api/*.

[JDOM] *http://www.jdom.org/*.

[JMX] *http://java.sun.com/products/JavaManagement/reference/docs/index.html*.

[mx4j] *http://mx4j.sourceforge.net/*.

7 Anhang

```
1  //****************************************
2  // NetCamera  Klasse  (Zeile 1 - 375)    *
3  //****************************************
4  package neve.data.applications;
5  import java.lang.reflect.Field;
6  import neve.core.api.*;
7  import javax.management.*;
8  import java.lang.reflect.Method;
9  import java.lang.reflect.*;
10 import java.net.*;
11 import java.io.*;
12 import java.util.*;
13 import java.text.*;
14 import org.jdom.*;
15 import org.jdom.input.*;
16 import org.jdom.output.*;
17 import org.jdom.xpath.*;
18 import neve.data.mib.NyxSwitchInterface;
19 import neve.data.mib.NyxSwitch;
20 import neve.data.mib.NyxSystem;
21 import java.awt.Container;
22 import java.awt.FlowLayout;
23 import java.awt.event.*;
24 import javax.swing.*;
25 import javax.swing.event.*;
26 public class NetCamera extends NeveAPI
27 {
28         HashMap alldevices=new HashMap();
29         HashMap verbindungen=new HashMap();
30         Vector allcable= new Vector();
31         private String ip="129.187.1.250";
32         private String depth="5";
33         private String queryString;
34         public void setQueryString(String ip,int depth)
35         {
36                 String head="";
37                 String tail=" </device> </query> ";
38                 head= "<query> <filter id=\"link\"> <include attr=\"linkdevice\"
value=\"\" compare=\"!eqs\"/> </filter> <filter id=\"dev\"> <include attr=\"ip\"
value=\"";
39                 head=head+ip;
40                 head=head+ "\" compare=\"eqs\"/> </filter> <device
filter=\"dev\">";
41                 for (int i=0;i<depth;i++)
42                 {
43                         head=head+ "   <interface filter=\"link\"> <link>";
44                         tail="  </link> </interface>" +tail;
45                 }
46                 queryString=head + tail;
47         }
48         public void setIp(String s)
49         {
50                 this.ip=s;
51         }
52         public NetCamera()
53         {
54                 super(); // initialise NeveAPI
55                 Authenticator.setDefault (new MyAuthenticator ());
56                 buildDynamicMBeanInfo();
57         }
58         public void createNetCamera()
59         {
60                 getAndProcessInputStream(queryString);
61                 buildGraph( );
```

```
  62          }
  63          public void createNetCamera(String ip,int depth)
  64          {
  65                  setQueryString( ip, depth);
  66                  getAndProcessInputStream(queryString);
  67                  buildGraph( );
  68          }
  69          public void createNetCamera(String ip)
  70          {
  71                  setQueryString( ip, 5);
  72                  getAndProcessInputStream(queryString);
  73                  buildGraph( );
  74          }
  75          public void createNetCamera(int depth)
  76          {
  77                  setQueryString( "129.187.1.250", depth);
  78                  getAndProcessInputStream(queryString);
  79                  buildGraph( );
  80          }
  81          private void buildDynamicMBeanInfo()
  82          {
  83                  //ip,depth
  84                  MBeanParameterInfo parInfo_3[] = {new MBeanParameterInfo(
"ip","java.lang.String","java.lang.String"),new MBeanParameterInfo(
"depth","int","depth of query")};
  85                  addOperation(new MBeanOperationInfo("createNetCamera", "
createNetCamera at ip depth", parInfo_3, "void", 1));
  86                  try {
  87                          Class[] params_3 = {String.class,int.class};
  88                          Method method_3 =
this.getClass().getMethod("createNetCamera", params_3);
  89                          addInvoker("createNetCamera", method_3, params_3);
  90                  } catch (Exception e) { e.printStackTrace();}
  91          }
  92          public void getAndProcessInputStream(String queryString)
  93          {
  94                  System.out.println("xml queryString:");
  95                  System.out.println(queryString);
  96                  neve.data.mib.NyxSystem firstDevice=new neve.data.mib.NyxSystem
();
  97                  try
  98                  {
  99                          String url2=URLEncoder.encode(queryString, "UTF-8");
 100                          URL url = new URL("http://mongwu/nyx/nyxControl");
 101                          URLConnection connection = url.openConnection();
 102                          connection.setDoOutput(true);
 103                          PrintWriter out = new
PrintWriter(connection.getOutputStream());
 104                          out.println("query="+url2);
 105                          out.close();
 106                          InputStream ips=connection.getInputStream();
 107                          SAXBuilder builder = new SAXBuilder();
 108                          Document doc = builder.build(ips);
 109                          XMLOutputter xout = new XMLOutputter("   " ,true);
 110                          System.out.println(" xml Anfrage Ergebnis: begin" );
 111                          xout.output( doc, System.out );
 112                          System.out.println(" xml Anfrage Ergebnis: end" );
 113                          Element root = doc.getRootElement();//we have <result>
element
 114                          Element f = root.getChild("device");
 115                          if(f==null) return;
 116                          readAndSetSystemAttributes( firstDevice,f);
 117
alldevices.put(f.getAttribute("device").getValue().trim(),firstDevice);
```

```
118                             List resultList = f.getChildren("interface");
119                             if(resultList ==null) return;
120                             ListIterator li =resultList.listIterator();
121                             while (li.hasNext())
122                             {
123                                     Object o = li.next();
124                                     Element e = (Element) o;//e ist vom Type
interface
125                                     NyxSwitchInterface inf=new NyxSwitchInterface();
126                                     readAndSetInterfaceAttributes( inf,e);
127
verbindungen.put(inf,e.getAttribute("linkdevice").getValue().trim());
128
relate("enthält",firstDevice.getMio(),inf.getMio());
129                                     firstDevice.getInterfaces().add(inf);
130                                     if(e.hasChildren())
131                                     {
132                                             dealWithInterface(inf,e);
133                                     }
134                             }
135                             ips.close();
136                     }
137                     catch (Exception e)
138                     {
139                             e.printStackTrace();
140                     }
141             }
142             public void dealWithInterface(NyxSwitchInterface inf,Element e)
143             {
144                     readAndSetInterfaceAttributes( inf,e);
145
verbindungen.put(inf,e.getAttribute("linkdevice").getValue().trim());
146                     neve.data.mib.NyxSystem device=null;
147                     Element d = e.getChild("device");
148                     if(d==null) return;
149
if(!alldevices.containsKey(d.getAttribute("device").getValue().trim()) )
150                     {
151                             device=new neve.data.mib.NyxSystem();
152                             readAndSetSystemAttributes( device,d);
153
alldevices.put(d.getAttribute("device").getValue().trim(),device);
154                             if(d.hasChildren())
155                             {
156                                     List resultList = d.getChildren("interface");
157                                     ListIterator li =resultList.listIterator();
158                                     while (li.hasNext())
159                                     {
160                                             Object o = li.next();
161                                             Element ee = (Element) o; //ee vom Typ
interface
162                                             NyxSwitchInterface interf=new
NyxSwitchInterface();
163                                             readAndSetInterfaceAttributes(
interf,ee);
164
verbindungen.put(interf,ee.getAttribute("linkdevice").getValue().trim());
165
relate("enthält",device.getMio(),interf.getMio());
166                                             device.getInterfaces().add(interf);
167                                             if(ee.hasChildren())
168                                             {
169                                                     dealWithInterface( interf,ee);
170                                             }
```

```
171                                     }
172                                 }
173                         }
174                 }
175                 public void readAndSetSystemAttributes(neve.data.mib.NyxSystem
device,Element element)
176                 {
177                         if(device==null || element==null)
178                                 return;
179                         org.jdom.Attribute a= element.getAttribute("checklayer2");
180                         org.jdom.Attribute b= element.getAttribute("checklayer3");
181                         device.setChecklayer2(a.getValue().trim());
182                         device.setChecklayer3(b.getValue().trim());
183                         if(device.getMio()==-1)
184                         {
185                                 device.setMio(createSystemMIO(device));
186                         }
187                         //cdp
188                         org.jdom.Attribute c= element.getAttribute("cdp");
189                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("cdp",c.getValue().trim()));
190                         //checklayer2
191                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("checklayer2",a.getValue().trim()));
192                         //checklayer3
193                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("checklayer3",b.getValue().trim()));
194                         //created
195                         org.jdom.Attribute d= element.getAttribute("created");
196                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("created",d.getValue().trim()));
197                         //device
198                         org.jdom.Attribute e= element.getAttribute("device");
199                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("device",e.getValue().trim()));
200                         //dns
201                         org.jdom.Attribute f= element.getAttribute("dns");
202                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("dns",f.getValue().trim()));
203                         //faultcount
204                         org.jdom.Attribute g= element.getAttribute("faultcount");
205                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("faultcount",g.getValue().trim()));
206                         //ip
207                         org.jdom.Attribute h= element.getAttribute("ip");
208                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("ip",h.getValue().trim()));
209                         //lastupdate
210                         org.jdom.Attribute i= element.getAttribute("lastupdate");
211                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("lastupdate",i.getValue().trim()));
212                         //layer2
213                         org.jdom.Attribute j= element.getAttribute("layer2");
214                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("layer2",j.getValue().trim()));
215                         //layer3
216                         org.jdom.Attribute k= element.getAttribute("layer3");
217                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("layer3",k.getValue().trim()));
218                         //syscontact
219                         org.jdom.Attribute l= element.getAttribute("syscontact");
220                         setMIOAttribute(device.getMio(),new
javax.management.Attribute("syscontact",l.getValue().trim()));
221                         //sysdescription
```

```
222                     org.jdom.Attribute m= element.getAttribute("sysdescription");
223                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("sysdescription",m.getValue().trim()));
224                     //syslocation
225                     org.jdom.Attribute n= element.getAttribute("syslocation");
226                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("syslocation",n.getValue().trim()));
227                     //sysname
228                     org.jdom.Attribute o= element.getAttribute("sysname");
229                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("sysname",o.getValue().trim()));
230                     //sysobjectid
231                     org.jdom.Attribute p= element.getAttribute("sysobjectid");
232                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("sysobjectid",p.getValue().trim()));
233                     //sysuptime
234                     org.jdom.Attribute q= element.getAttribute("sysuptime");
235                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("sysuptime",q.getValue().trim()));
236                     //mio
237                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("mio",new Integer(device.getMio())));
238                     //interfaces
239                     setMIOAttribute(device.getMio(),new
javax.management.Attribute("interfaces",device.getInterfaces()));
240             }
241             public void readAndSetInterfaceAttributes(NyxSwitchInterface intf,Element
element)
242             {
243                     if(intf==null || element==null)
244                             return;
245                     List list = element.getAttributes() ;
246                     if(list==null) return;
247                     Iterator it =list.iterator();
248                     while (it.hasNext())
249                     {
250                             org.jdom.Attribute a=(org.jdom.Attribute)it.next();
251                             intf.setValue(a.getName().trim(), a.getValue().trim());
252                             if(intf.getMio()==-1)
253                             {
254                                     intf.setMio(createInterfaceMIO( intf));
255                             }
256                     }
257                     //device
258                     org.jdom.Attribute d= element.getAttribute("device");
259                     setMIOAttribute(intf.getMio(),new
javax.management.Attribute("device",d.getValue().trim()));
260                     //linkdevice
261                     org.jdom.Attribute e= element.getAttribute("linkdevice");
262                     setMIOAttribute(intf.getMio(),new
javax.management.Attribute("linkdevice",e.getValue().trim()));
263             }
264             public int createSystemMIO(neve.data.mib.NyxSystem device)
265             {
266                     int mio;
267                     int layer2=-1;
268                     int layer3=-1;
269                     try{
270                             layer2=Integer.parseInt(device.getChecklayer2().trim());
271                             layer3=Integer.parseInt(device.getChecklayer3().trim());
272                     }
273                     catch (Exception e){
274                             e.printStackTrace();
275                     }
```

```
276                     if(layer2==1 && layer3==1)
277                     {
278                             mio= createMIO("NyxRouter", "standard");
279                     }
280                     else{
281                             if(layer2==1 && layer3==0)
282                             {
283                                     mio     = createMIO("NyxSwitch", "standard");
284                             }
285                             else
286                             {
287                                     mio     = createMIO("NyxHost", "standard");
288                             }
289                     }
290                     device.setMio(mio);
291                     return mio;
292             }
293             public int createInterfaceMIO(NyxSwitchInterface interface_)
294             {
295                     int inf ;
296                     inf = createMIO("NyxSwitchInterface", "standard");
297                     interface_.setMio(inf);
298                     return inf;
299             }
300             public void buildGraph( )
301             {
302                     Iterator it = verbindungen.entrySet().iterator();
303                     Iterator it_3 = verbindungen.entrySet().iterator();
304                     //print out connections
305                     int n=1;
306                     while (it_3.hasNext())
307                     {
308                             Map.Entry entry_3 = (Map.Entry)it_3.next();
309                             //System.out.println("verbindung:"+n+ " : "+
(String)entry_3.getKey()+" --->   "+((NyxSwitchInterface )
entry_3.getValue()).getDevice());
310                             n++;
311                     }
312                     //print out all devices
313                     Iterator it_2 = alldevices.entrySet().iterator();
314                     int m=1;
315                     while (it_2.hasNext())
316                     {
317                             Map.Entry entry_2 = (Map.Entry)it_2.next();
318                             //System.out.println("Device "+m+ " : "+
(String)entry_2.getKey()+" "+((neve.data.mib.NyxSystem)entry_2.getValue()).getIp());
319                             m++;
320                     }
321                     NyxSwitchInterface nsi=null;
322                     NyxSwitchInterface uplink_1;
323                     String  linkdevice=null;
324                     neve.data.mib.NyxSystem d= null;
325                     int cable_up_1;
326                     //verbindungen iterator
327                     while (it.hasNext())
328                     {
329                             Map.Entry entry = (Map.Entry)it.next();
330                             linkdevice =  (String)entry.getValue();
331                             //nsi ist vom Typ NyxSwitchInterface
332                             nsi= (NyxSwitchInterface )entry.getKey();
333                             //d ist vom Typ NyxSystem
334                             d=(neve.data.mib.NyxSystem)alldevices.get(linkdevice);
335                             if(d!=null&&d.getInterfaces().size()==0)
336                             {
```

```
 337                                     uplink_1=new NyxSwitchInterface();
 338                                     createInterfaceMIO(uplink_1);
 339                                     relate("enthält",d.getMio(),uplink_1.getMio());
 340                                     cable_up_1=createMIO("TwistedPair","standard");
 341                                     allcable.add(new Integer(cable_up_1));
 342                                     relate("verbunden-
mit",uplink_1.getMio(),cable_up_1,nsi.getMio());
 343                                     uplink_1.getConnectedWith().add(nsi);
 344                                     nsi.getConnectedWith().add(uplink_1);
 345                             }
 346                             if(d!=null&&d.getInterfaces().size()>0)
 347                             {
 348                                     //finde uplink
 349                                     for(int i=0;i<d.getInterfaces().size();i++)
 350                                     {
 351                                             NyxSwitchInterface
tmp=(NyxSwitchInterface)(d.getInterfaces().elementAt(i));
 352                                             String ld=tmp.getLinkdevice();
 353                                             if(ld.equalsIgnoreCase(nsi.getDevice( )))
 354                                             {
 355
cable_up_1=createMIO("TwistedPair","standard");
 356                                                     allcable.add(new
Integer(cable_up_1));
 357                                                     relate("verbunden-
mit",nsi.getMio(),cable_up_1,tmp.getMio());
 358                                                     nsi.getConnectedWith().add(tmp);
 359                                                     tmp.getConnectedWith().add(nsi);
 360                                                     break;
 361                                             }
 362                                     }
 363                             }
 364                     }
 365             }
 366             class MyAuthenticator extends Authenticator
 367             {
 368                     protected PasswordAuthentication getPasswordAuthentication()
 369                     {
 370                             char [] pwd ={'e','v','e','n'};
 371                             return new PasswordAuthentication ("neve",pwd);
 372                     }
 373             }
 374     }
 375
 376     //*******************************************************
 377     // Informationsmodell erweitern  (Zeile 375 -1179)     *
 378     //*******************************************************
 379
 380     package neve.data.mib;
 381     import neve.core.objects.*;
 382     import java.lang.reflect.Method;
 383     import java.lang.reflect.Array;
 384     import java.lang.reflect.Constructor;
 385     import java.util.Iterator;
 386     import javax.management.*;
 387     import java.net.*;
 388     import java.util.*;
 389     // The Systemnode
 390     public class NyxSystem extends neve.data.mib.System
 391     {
 392             private int mio=-1;
 393             public int getMio() {
 394                     return mio;
 395             }
```

```
396         public void setMio(int mio) {
397                 this.mio= mio;
398         }
399         private String cdp="";
400         public String getCdp() {
401                 return cdp;
402         }
403         public void setCdp(String cdp) {
404                 this.cdp = cdp;
405         }
406         private String checklayer2="";
407         public String getChecklayer2() {
408                 return checklayer2;
409         }
410         public void setChecklayer2(String checklayer2) {
411                 this.checklayer2 = checklayer2;
412         }
413         private String checklayer3="";
414         public String getChecklayer3() {
415                 return checklayer3;
416         }
417         public void setChecklayer3(String checklayer3) {
418                 this.checklayer3 = checklayer3;
419         }
420         private String created="";
421         public String getCreated() {
422                 return created;
423         }
424         public void setCreated(String created) {
425                 this.created = created;
426         }
427         private String device="";
428         public String getDevice() {
429                 return device;
430         }
431         public void setDevice(String device) {
432                 this.device = device;
433         }
434         private String dns="";
435         public String getDns() {
436                 return dns;
437         }
438         public void setDns(String dns) {
439                 this.dns = dns;
440         }
441         private String faultcount="";
442         public String getFaultcount() {
443                 return faultcount;
444         }
445         public void setFaultcount(String faultcount) {
446                 this.faultcount = faultcount;
447         }
448         private String ip="";
449         public String getIp() {
450                 return ip;
451         }
452         public void setIp(String ip) {
453                 this.ip = ip;
454         }
455         private String lastupdate="";
456         public String getLastupdate() {
457                 return lastupdate;
458         }
459         public void setLastupdate(String lastupdate) {
```

```
460                 this.lastupdate = lastupdate;
461         }
462         private String layer2="";
463         public String getLayer2() {
464                 return layer2;
465         }
466         public void setLayer2(String layer2) {
467                 this.layer2 = layer2;
468         }
469         private String layer3="";
470         public String getLayer3() {
471                 return layer3;
472         }
473         public void setLayer3(String layer3) {
474                 this.layer3 = layer3;
475         }
476         private String syscontact="";
477         public String getSyscontact() {
478                 return syscontact;
479         }
480         public void setSyscontact(String syscontact) {
481                 this.syscontact = syscontact;
482         }
483         private String sysdescription="";
484         public String getSysdescription() {
485                 return sysdescription;
486         }
487         public void setSysdescription(String sysdescription) {
488                 this.sysdescription = sysdescription;
489         }
490         private String syslocation="";
491         public String getSyslocation() {
492                 return syslocation;
493         }
494         public void setSyslocation(String syslocation) {
495                 this.syslocation = syslocation;
496         }
497         private String sysname="";
498         public String getSysname() {
499                 return sysname;
500         }
501         public void setSysname(String sysname) {
502                 this.sysname = sysname;
503         }
504         private String sysobjectid="";
505         public String getSysobjectid() {
506                 return sysobjectid;
507         }
508         public void setSysobjectid(String sysobjectid) {
509                 this.sysobjectid = sysobjectid;
510         }
511         private String sysuptime="";
512         public String getSysuptime() {
513                 return sysuptime;
514         }
515         public void setSysuptime(String sysuptime) {
516                 this.sysuptime = sysuptime;
517         }
518         private Vector interfaces = new Vector();
519         public Vector getInterfaces() {
520                 return interfaces;
521         }
522         public void setInterfaces(Vector interfaces) {
523                 this.interfaces = interfaces;
```

```
524         }
525         public NyxSystem(){
526                 buildDynamicMBeanInfo();
527                 String[] attributs={"Type","device"};
528                 setequalAttributes(attributs);
529         }
530         private void buildDynamicMBeanInfo(){
531                 addAttribute(new MBeanAttributeInfo("mio",
532                                         "java.lang.int",
533                                         "mio:  ",
534                                         true,
535                                         true,
536                                         false));
537                 //mio
538                 try{
539                         Method method=this.getClass().getMethod("getMio",new
Class[0]);
540                         addGetter ("mio",method);
541                 } catch (Exception e) {e.printStackTrace();}
542                 try{
543                         Class[] params=new Class[1];
544                         params[0]=int.class;
545                         Method method=this.getClass().getMethod("setMio",params);
546                         addSetter ("mio",method,params);
547                 } catch (Exception e) {e.printStackTrace();}
548                 //cdp
549                 addAttribute(new MBeanAttributeInfo("cdp",
550                                         "java.lang.int",
551                                         "cdp:  ",
552                                         true,
553                                         true,
554                                         false));
555                 try{
556                         Method method=this.getClass().getMethod("getCdp",new
Class[0]);
557                         addGetter ("cdp",method);
558                 } catch (Exception e) {e.printStackTrace();}
559                 try{
560                         Class[] params=new Class[1];
561                         params[0]=String.class;
562                         Method method=this.getClass().getMethod("setCdp",params);
563                         addSetter ("cdp",method,params);
564                 } catch (Exception e) {e.printStackTrace();}
565                 //Checklayer2
566                 addAttribute(new MBeanAttributeInfo("checklayer2",
567                                         "java.lang.String",
568                                         "checklayer2:  ",
569                                         true,
570                                         true,
571                                         false));
572                 try{
573                         Method
method=this.getClass().getMethod("getChecklayer2",new Class[0]);
574                         addGetter ("checklayer2",method);
575                 } catch (Exception e) {e.printStackTrace();}
576                 try{
577                         Class[] params=new Class[1];
578                         params[0]=String.class;
579                         Method
method=this.getClass().getMethod("setChecklayer2",params);
580                         addSetter ("checklayer2",method,params);
581                 } catch (Exception e) {e.printStackTrace();}
582                 //checklayer3
583                 addAttribute(new MBeanAttributeInfo("checklayer3",
```

```
584                                             "java.lang.String",
585                                             "checklayer3:  ",
586                                             true,
587                                             true,
588                                             false));
589                     try{
590                             Method
method=this.getClass().getMethod("getChecklayer3",new Class[0]);
591                             addGetter ("checklayer3",method);
592                     } catch (Exception e) {e.printStackTrace();}
593                     try{
594                             Class[] params=new Class[1];
595                             params[0]=String.class;
596                             Method
method=this.getClass().getMethod("setChecklayer3",params);
597                             addSetter ("checklayer3",method,params);
598                     } catch (Exception e) {e.printStackTrace();}
599                     //created
600                     addAttribute(new MBeanAttributeInfo("created",
601                                             "java.lang.String",
602                                             "created:  ",
603                                             true,
604                                             true,
605                                             false));
606                     try{
607                             Method method=this.getClass().getMethod("getCreated",new
Class[0]);
608                             addGetter ("created",method);
609                     } catch (Exception e) {e.printStackTrace();}
610                     try{
611                             Class[] params=new Class[1];
612                             params[0]=String.class;
613                             Method
method=this.getClass().getMethod("setCreated",params);
614                             addSetter ("created",method,params);
615                     } catch (Exception e) {e.printStackTrace();}
616                     //device
617                     //
618                     addAttribute(new MBeanAttributeInfo("device",
619                                             "java.lang.String",
620                                             "device:  ",
621                                             true,
622                                             true,
623                                             false));
624                     try{
625                             Method method=this.getClass().getMethod("getDevice",new
Class[0]);
626                             addGetter ("device",method);
627                     } catch (Exception e) {e.printStackTrace();}
628                     try{
629                             Class[] params=new Class[1];
630                             params[0]=String.class;
631                             Method
method=this.getClass().getMethod("setDevice",params);
632                             addSetter ("device",method,params);
633                     } catch (Exception e) {e.printStackTrace();}
634                     //dns
635                     addAttribute(new MBeanAttributeInfo("dns",
636                                             "java.lang.String",
637                                             "dns:  ",
638                                             true,
639                                             true,
640                                             false));
641                     try{
```

```
642                             Method method=this.getClass().getMethod("getDns",new
Class[0]);
643                             addGetter ("dns",method);
644                     } catch (Exception e) {e.printStackTrace();}
645                     try{
646                             Class[] params=new Class[1];
647                             params[0]=String.class;
648                             Method method=this.getClass().getMethod("setDns",params);
649                             addSetter ("dns",method,params);
650                     } catch (Exception e) {e.printStackTrace();}
651                     //faultcount
652                     addAttribute(new MBeanAttributeInfo("faultcount",
653                                             "java.lang.String",
654                                             "faultcount:  ",
655                                             true,
656                                             true,
657                                             false));
658                     try{
659                             Method
method=this.getClass().getMethod("getFaultcount",new Class[0]);
660                             addGetter ("faultcount",method);
661                     } catch (Exception e) {e.printStackTrace();}
662                     try{
663                             Class[] params=new Class[1];
664                             params[0]=String.class;
665                             Method
method=this.getClass().getMethod("setFaultcount",params);
666                             addSetter ("faultcount",method,params);
667                     } catch (Exception e) {e.printStackTrace();}
668                     //ip
669                     addAttribute(new MBeanAttributeInfo("ip",
670                                             "java.lang.String",
671                                             "ip:  ",
672                                             true,
673                                             true,
674                                             false));
675                     try{
676                             Method method=this.getClass().getMethod("getIp",new
Class[0]);
677                             addGetter ("ip",method);
678                     } catch (Exception e) {e.printStackTrace();}
679                     try{
680                             Class[] params=new Class[1];
681                             params[0]=String.class;
682                             Method method=this.getClass().getMethod("setIp",params);
683                             addSetter ("ip",method,params);
684                     } catch (Exception e) {e.printStackTrace();}
685                     //lastupdate
686                     addAttribute(new MBeanAttributeInfo("lastupdate",
687                                             "java.lang.String",
688                                             "lastupdate:  ",
689                                             true,
690                                             true,
691                                             false));
692                     try{
693                             Method
method=this.getClass().getMethod("getLastupdate",new Class[0]);
694                             addGetter ("lastupdate",method);
695                     } catch (Exception e) {e.printStackTrace();}
696                     try{
697                             Class[] params=new Class[1];
698                             params[0]=String.class;
699                             Method
method=this.getClass().getMethod("setLastupdate",params);
```

```
700                         addSetter ("lastupdate",method,params);
701                 } catch (Exception e) {e.printStackTrace();}
702                 //layer2
703                 addAttribute(new MBeanAttributeInfo("layer2",
704                                         "java.lang.String",
705                                         "layer2:  ",
706                                         true,
707                                         true,
708                                         false));
709                 try{
710                         Method method=this.getClass().getMethod("getLayer2",new
Class[0]);
711                         addGetter ("layer2",method);
712                 } catch (Exception e) {e.printStackTrace();}
713                 try{
714                         Class[] params=new Class[1];
715                         params[0]=String.class;
716                         Method
method=this.getClass().getMethod("setLayer2",params);
717                         addSetter ("layer2",method,params);
718                 } catch (Exception e) {e.printStackTrace();}
719                 //layer3
720                 addAttribute(new MBeanAttributeInfo("layer3",
721                                         "java.lang.String",
722                                         "layer3:  ",
723                                         true,
724                                         true,
725                                         false));
726                 try{
727                         Method method=this.getClass().getMethod("getLayer3",new
Class[0]);
728                         addGetter ("layer3",method);
729                 } catch (Exception e) {e.printStackTrace();}
730                 try{
731                         Class[] params=new Class[1];
732                         params[0]=String.class;
733                         Method
method=this.getClass().getMethod("setLayer3",params);
734                         addSetter ("layer3",method,params);
735                 } catch (Exception e) {e.printStackTrace();}
736                 //syscontact
737                 addAttribute(new MBeanAttributeInfo("syscontact",
738                                         "java.lang.String",
739                                         "syscontact:  ",
740                                         true,
741                                         true,
742                                         false));
743                 try{
744                         Method
method=this.getClass().getMethod("getSyscontact",new Class[0]);
745                         addGetter ("syscontact",method);
746                 } catch (Exception e) {e.printStackTrace();}
747                 try{
748                         Class[] params=new Class[1];
749                         params[0]=String.class;
750                         Method
method=this.getClass().getMethod("setSyscontact",params);
751                         addSetter ("syscontact",method,params);
752                 } catch (Exception e) {e.printStackTrace();}
753                 //sysdescription
754                 addAttribute(new MBeanAttributeInfo("sysdescription",
755                                         "java.lang.String",
756                                         "sysdescription:  ",
757                                         true,
```

```
 758                                             true,
 759                                             false));
 760                     try{
 761                             Method
method=this.getClass().getMethod("getSysdescription",new Class[0]);
 762                             addGetter ("sysdescription",method);
 763                     } catch (Exception e) {e.printStackTrace();}
 764                     try{
 765                             Class[] params=new Class[1];
 766                             params[0]=String.class;
 767                             Method
method=this.getClass().getMethod("setSysdescription",params);
 768                             addSetter ("sysdescription",method,params);
 769                     } catch (Exception e) {e.printStackTrace();}
 770                     //
 771                     //syslocation
 772                     addAttribute(new MBeanAttributeInfo("syslocation",
 773                                             "java.lang.String",
 774                                             "syslocation:  ",
 775                                             true,
 776                                             true,
 777                                             false));
 778                     try{
 779                             Method
method=this.getClass().getMethod("getSyslocation",new Class[0]);
 780                             addGetter ("syslocation",method);
 781                     } catch (Exception e) {e.printStackTrace();}
 782                     try{
 783                             Class[] params=new Class[1];
 784                             params[0]=String.class;
 785                             Method
method=this.getClass().getMethod("setSyslocation",params);
 786                             addSetter ("syslocation",method,params);
 787                     } catch (Exception e) {e.printStackTrace();}
 788                     //sysname
 789                     addAttribute(new MBeanAttributeInfo("sysname",
 790                                             "java.lang.String",
 791                                             "sysname:  ",
 792                                             true,
 793                                             true,
 794                                             false));
 795                     try{
 796                             Method method=this.getClass().getMethod("getSysname",new
Class[0]);
 797                             addGetter ("sysname",method);
 798                     } catch (Exception e) {e.printStackTrace();}
 799                     try{
 800                             Class[] params=new Class[1];
 801                             params[0]=String.class;
 802                             Method
method=this.getClass().getMethod("setSysname",params);
 803                             addSetter ("sysname",method,params);
 804                     } catch (Exception e) {e.printStackTrace();}
 805                     //sysobjectid
 806                     addAttribute(new MBeanAttributeInfo("sysobjectid",
 807                                             "java.lang.String",
 808                                             "sysobjectid:  ",
 809                                             true,
 810                                             true,
 811                                             false));
 812                     try{
 813                             Method
method=this.getClass().getMethod("getSysobjectid",new Class[0]);
 814                             addGetter ("sysobjectid",method);
```

```
815                     } catch (Exception e) {e.printStackTrace();}
816                     try{
817                             Class[] params=new Class[1];
818                             params[0]=String.class;
819                             Method
method=this.getClass().getMethod("setSysobjectid",params);
820                             addSetter ("sysobjectid",method,params);
821                     } catch (Exception e) {e.printStackTrace();}
822                     //sysuptime
823                     addAttribute(new MBeanAttributeInfo("sysuptime",
824                                                     "java.lang.String",
825                                                     "sysuptime:  ",
826                                                     true,
827                                                     true,
828                                                     false));
829                     try{
830                             Method
method=this.getClass().getMethod("getSysuptime",new Class[0]);
831                             addGetter ("sysuptime",method);
832                     } catch (Exception e) {e.printStackTrace();}
833                     try{
834                             Class[] params=new Class[1];
835                             params[0]=String.class;
836                             Method
method=this.getClass().getMethod("setSysuptime",params);
837                             addSetter ("sysuptime",method,params);
838                     } catch (Exception e) {e.printStackTrace();}
839                     //interfaces
840                     addAttribute(new MBeanAttributeInfo("interfaces",
841                                                     "java.util.Vector",
842                                                     "interfaces:  ",
843                                                     true,
844                                                     true,
845                                                     false));
846                     try{
847                             Method
method=this.getClass().getMethod("getInterfaces",new Class[0]);
848                             addGetter ("interfaces",method);
849                     } catch (Exception e) {e.printStackTrace();}
850                     try{
851                             Class[] params=new Class[1];
852                             params[0]=Vector.class;
853                             Method
method=this.getClass().getMethod("setInterfaces",params);
854                             addSetter ("interfaces",method,params);
855                     } catch (Exception e) {e.printStackTrace();}
856             }
857             public void setValue(String what,String value) {
858                     if(what.equals("cdp"))
859                             setCdp(value);
860                     if(what.equals("checklayer2"))
861                             setChecklayer2(value);
862                     if(what.equals("checklayer3"))
863                             setChecklayer3(value);
864                     if(what.equals("created"))
865                             setCreated(value);
866                     if(what.equals("device"))
867                             setDevice(value);
868                     if(what.equals("dns"))
869                             setDns(value);
870                     if(what.equals("faultcount"))
871                             setFaultcount(value);
872                     if(what.equals("ip"))
873                             setIp(value);
```

```
874                     if(what.equals("lastupdate"))
875                             setLastupdate(value);
876                     if(what.equals("layer2"))
877                             setLayer2(value);
878                     if(what.equals("layer3"))
879                             setLayer3(value);
880                     if(what.equals("syscontact"))
881                             setSyscontact(value);
882                     if(what.equals("sysdescription"))
883                             setSysdescription(value);
884                     if(what.equals("syslocation"))
885                             setSyslocation(value);
886                     if(what.equals("sysname"))
887                             setSysname(value);
888                     if(what.equals("sysobjectid"))
889                             setSysobjectid(value);
890                     if(what.equals("sysuptime"))
891                             setSysuptime(value);
892             }
893     }
894     package neve.data.mib;
895     import neve.core.objects.*;
896     import java.lang.reflect.Method;
897     import java.lang.reflect.Array;
898     import java.lang.reflect.Constructor;
899     import java.util.Iterator;
900     import javax.management.*;
901     import java.net.*;
902     import java.util.*;
903     // The Hostnode
904     public class NyxHost extends NyxSystem {
905             private String _MAC;
906             public NyxHost(){
907                     buildDynamicMBeanInfo();
908                     String[] attribs={"Type","MAC"};
909                     setequalAttributes(attribs);
910             }
911             private void buildDynamicMBeanInfo() { addAttribute(new
MBeanAttributeInfo("MAC",
912                                     "java.lang.String",
913                                     "MAC: The MAC adress of the object.",
914                                     true,
915                                     true,
916                                     false));
917             try{
918                     Method method=this.getClass().getMethod("getMAC",new Class[0]);
919                     addGetter ("MAC",method);
920             } catch (Exception e) {e.printStackTrace();}
921             try{
922                     Class[] params=new Class[1];
923                     int i;
924                     params[0]=String.class;
925                     Method method=this.getClass().getMethod("setMAC",params);
926                     addSetter ("MAC",method,params);
927             } catch (Exception e) {e.printStackTrace();
928             }
929             }
930             public String getMAC(){
931                     return _MAC;
932             }
933             public void setMAC(String MAC){
934                     if (MAC!=""){
935                             _MAC=MAC;
936                     }
```

```
        }
}
package neve.data.mib;
import neve.core.objects.*;
import java.lang.reflect.Method;
import java.lang.reflect.Array;
import java.lang.reflect.Constructor;
import java.util.Iterator;
import javax.management.*;
import java.net.*;
import java.util.*;
// The Hostnode
public class NyxRouter extends NyxSystem {
        public NyxRouter(){
        }
}
package neve.data.mib;
import neve.core.objects.*;
import java.lang.reflect.Method;
import java.lang.reflect.Array;
import java.lang.reflect.Constructor;
import java.util.Iterator;
import javax.management.*;
import java.net.*;
import java.util.*;
// The Hostnode
public class NyxSwitch extends NyxSystem {
        public NyxSwitch(){
        }
}
package neve.data.mib;
import java.lang.reflect.Method;
import javax.management.*;
import java.util.Vector;
public class NyxSwitchInterface extends ProtocolInstance {
        private int VLAN=0;
        public int getVLAN(){
                return VLAN;
        }
        public void setVLAN(int vlan){
                this.VLAN=vlan;
        }
        private int mio=-1;
        public int getMio() {
                return mio;
        }
        public void setMio(int mio) {
                this.mio= mio;
        }
        private String linkdevice="";
        public String getLinkdevice() {
                return linkdevice;
        }
        public void setLinkdevice(String linkdevice) {
                this.linkdevice = linkdevice;
        }
        private String device="";
        public String getDevice() {
                return device;
        }
        public void setDevice(String device) {
                this.device = device;
        }
        private Vector  connectedWith=new Vector();
```

```
1001          public Vector getConnectedWith()
1002          {
1003                  return connectedWith;
1004          }
1005          public void setConnectedWith(Vector other)
1006          {
1007                  this.connectedWith=other;
1008          }
1009          public NyxSwitchInterface (){
1010                  buildDynamicMBeanInfo();
1011          }
1012          private void buildDynamicMBeanInfo(){
1013                  addAttribute(new MBeanAttributeInfo("VLAN",
1014                                          "java.lang.int",
1015                                          "VLAN: VLAN Id-Number",
1016                                          true,
1017                                          true,
1018                                          false));
1019                  try{
1020                          Method method=this.getClass().getMethod("getVLAN",new
Class[0]);
1021                          addGetter ("VLAN",method);
1022                  } catch (Exception e) {e.printStackTrace();}
1023                  try{
1024                          Class[] params=new Class[1];
1025                          params[0]=int.class;
1026                          Method
method=this.getClass().getMethod("setVLAN",params);
1027                          addSetter ("VLAN",method,params);
1028                  } catch (Exception e) {e.printStackTrace();}
1029                  //Mio
1030                  addAttribute(new MBeanAttributeInfo("mio",
1031                                          "java.lang.int",
1032                                          "mio: mio Id-Number",
1033                                          true,
1034                                          true,
1035                                          false));
1036                  try{
1037                          Method method=this.getClass().getMethod("getMio",new
Class[0]);
1038                          addGetter ("mio",method);
1039                  } catch (Exception e) {e.printStackTrace();}
1040                  try{
1041                          Class[] params=new Class[1];
1042                          params[0]=int.class;
1043                          Method method=this.getClass().getMethod("setMio",params);
1044                          addSetter ("mio",method,params);
1045                  } catch (Exception e) {e.printStackTrace();}
1046                  //linkdevice
1047                  addAttribute(new MBeanAttributeInfo("linkdevice",
1048                                          "java.lang.String",
1049                                          "linkdevice: ",
1050                                          true,
1051                                          true,
1052                                          false));
1053                  try{
1054                          Method
method=this.getClass().getMethod("getLinkdevice",new Class[0]);
1055                          addGetter ("linkdevice",method);
1056                  } catch (Exception e) {e.printStackTrace();}
1057                  try{
1058                          Class[] params=new Class[1];
1059                          params[0]=String.class;
1060                          Method
```

```
method=this.getClass().getMethod("setLinkdevice",params);
                        addSetter ("linkdevice",method,params);
                } catch (Exception e) {e.printStackTrace();}
                //device
                addAttribute(new MBeanAttributeInfo("device",
                                        "java.lang.String",
                                        "device: ",
                                        true,
                                        true,
                                        false));
                try{
                        Method method=this.getClass().getMethod("getDevice",new
Class[0]);
                        addGetter ("device",method);
                } catch (Exception e) {e.printStackTrace();}
                try{
                        Class[] params=new Class[1];
                        params[0]=String.class;
                        Method
method=this.getClass().getMethod("setDevice",params);
                        addSetter ("device",method,params);
                } catch (Exception e) {e.printStackTrace();}
        }
        private String cdp="";
        private String created="";
        private String ifadminstatus="";
        private String ifalias="";
        private String ifdescription="";
        private String ifoperstatus="";
        private String ifspeed="";
        private String interface_="";
        private String lastupdate="";
        private String rec_marked="";
        public String getCdp() {
                return cdp;
        }
        public void setCdp(String cdp) {
                this.cdp = cdp;
        }
        public String getCreated() {
                return created;
        }
        public void setCreated(String created) {
                this.created = created;
        }
        public String getIfadminstatus() {
                return ifadminstatus;
        }
        public void setIfadminstatus(String ifadminstatus) {
                this.ifadminstatus = ifadminstatus;
        }
        public String getIfalias() {
                return ifalias;
        }
        public void setIfalias(String ifalias) {
                this.ifalias = ifalias;
        }
        public String getIfdescription() {
                return ifdescription;
        }
        public void setIfdescription(String ifdescription) {
                this.ifdescription = ifdescription;
        }
        public String getIfoperstatus() {
```

```
1122                    return ifoperstatus;
1123            }
1124            public void setIfoperstatus(String ifoperstatus) {
1125                    this.ifoperstatus = ifoperstatus;
1126            }
1127            public String getIfspeed() {
1128                    return ifspeed;
1129            }
1130            public void setIfspeed(String ifspeed) {
1131                    this.ifspeed = ifspeed;
1132            }
1133            public String getInterface_() {
1134                    return interface_;
1135            }
1136            public void setInterface_(String interface_) {
1137                    this.interface_ = interface_;
1138            }
1139            public String getLastupdate() {
1140                    return lastupdate;
1141            }
1142            public void setLastupdate(String lastupdate) {
1143                    this.lastupdate = lastupdate;
1144            }
1145            public String getRec_marked() {
1146                    return rec_marked;
1147            }
1148            public void setRec_marked(String rec_marked) {
1149                    this.rec_marked = rec_marked;
1150            }
1151            public void setValue(String what,String value) {
1152                    if(what.equals("cdp"))
1153                            setCdp(value);
1154                    if(what.equals("created"))
1155                            setCreated(value);
1156                    if(what.equals("device"))
1157                            setDevice(value);
1158                    if(what.equals("ifadminstatus"))
1159                            setIfadminstatus(value);
1160                    if(what.equals("ifalias"))
1161                            setIfalias(value);
1162                    if(what.equals("ifdescription"))
1163                            setIfdescription(value);
1164                    if(what.equals("ifoperstatus"))
1165                            setIfoperstatus(value);
1166                    if(what.equals("ifspeed"))
1167                            setIfspeed(value);
1168                    if(what.equals("interface"))
1169                            setInterface_(value);
1170                    if(what.equals("lastupdate"))
1171                            setLastupdate(value);
1172                    if(what.equals("linkdevice"))
1173                            setLinkdevice(value);
1174                    if(what.equals("rec_marked"))
1175                            setRec_marked(value);
1176            }
1177    }
1178
1179
1180    //*******************************************************
1181    // Präsentationsmodell erweitern (Zeile 1180 - 1334)   *
1182    //*******************************************************
1183
1184    package neve.data.pib.standard;
1185    import neve.core.objects.*;
```

```
1186  public class NyxSystem extends neve.data.pib.standard.System {
1187  }
1188  package neve.data.pib.standard;
1189  import neve.core.objects.*;
1190  import java.lang.reflect.Method;
1191  import java.lang.reflect.Array;
1192  import java.lang.reflect.Constructor;
1193  import java.util.Iterator;
1194  import javax.management.*;
1195  import java.net.*;
1196  import java.util.*;
1197  import com.sun.j3d.utils.geometry.*;
1198  import com.sun.j3d.utils.universe.*;
1199  import javax.media.j3d.*;
1200  import javax.vecmath.*;
1201  import com.sun.j3d.loaders.*;
1202  import com.glyphein.j3d.loaders.milkshape.MS3DLoader;
1203  public class NyxHost extends System {
1204          private Loader loader;
1205          private BranchGroup group;
1206          private String modelResourceName="neve/data/pib/standard/Host.ms3d";
1207          public NyxHost(){
1208                  loader = new MS3DLoader(MS3DLoader.LOAD_ALL);
1209                  java.io.File file=null;
1210                  try{
1211                          URL modelURL =
this.getClass().getClassLoader().getResource( modelResourceName );
1212                          Scene scene = loader.load(modelURL);
1213                          group = scene.getSceneGroup();
1214                  }catch(Exception e){java.lang.System.out.println("File not found:
"+file);e.printStackTrace();group=new BranchGroup(); }
1215          }
1216          public void POAttributeChange(String attrib){
1217                  setShape(group);
1218                  super.POAttributeChange(attrib);
1219          }
1220          public void MIOAttributeChange(String attrib){
1221                  setShape(group);
1222                  super.MIOAttributeChange(attrib);
1223          }
1224  }
1225  package neve.data.pib.standard;
1226  import neve.core.objects.*;
1227  import java.lang.reflect.Method;
1228  import java.lang.reflect.Array;
1229  import java.lang.reflect.Constructor;
1230  import java.util.Iterator;
1231  import javax.management.*;
1232  import java.net.*;
1233  import java.util.*;
1234  import com.sun.j3d.utils.geometry.*;
1235  import com.sun.j3d.utils.universe.*;
1236  import javax.media.j3d.*;
1237  import javax.vecmath.*;
1238  import com.sun.j3d.loaders.*;
1239  import com.glyphein.j3d.loaders.milkshape.MS3DLoader;
1240  public class NyxRouter extends System {
1241          private Loader loader;
1242          private BranchGroup group;
1243          private String modelResourceName="neve/data/pib/standard/Router.ms3d";
1244          public NyxRouter(){
1245                  loader = new MS3DLoader(MS3DLoader.LOAD_ALL);
1246                  java.io.File file=null;
1247                  try{
```

```
1248                            URL modelURL =
this.getClass().getClassLoader().getResource( modelResourceName );
1249                            Scene scene = loader.load(modelURL);
1250                            group = scene.getSceneGroup();
1251                    }catch(Exception e){java.lang.System.out.println("File not found:
"+file);e.printStackTrace();group=new BranchGroup(); }
1252            }
1253            public void POAttributeChange(String attrib){
1254                    setShape(group);
1255                    super.POAttributeChange(attrib);
1256            }
1257            public void MIOAttributeChange(String attrib){
1258                    setShape(group);
1259                    super.MIOAttributeChange(attrib);
1260            }
1261    }
1262    package neve.data.pib.standard;
1263    import neve.core.objects.*;
1264    import java.lang.reflect.Method;
1265    import java.lang.reflect.Array;
1266    import java.lang.reflect.Constructor;
1267    import java.util.Iterator;
1268    import javax.management.*;
1269    import java.net.*;
1270    import java.util.*;
1271    import com.sun.j3d.utils.geometry.*;
1272    import com.sun.j3d.utils.universe.*;
1273    import javax.media.j3d.*;
1274    import javax.vecmath.*;
1275    import com.sun.j3d.loaders.*;
1276    import com.glyphein.j3d.loaders.milkshape.MS3DLoader;
1277    public class NyxSwitch extends System {
1278            private Loader loader;
1279            private BranchGroup group;
1280            private String modelResourceName="neve/data/pib/standard/Switch.ms3d";
1281            public NyxSwitch(){
1282                    loader = new MS3DLoader(MS3DLoader.LOAD_ALL);
1283                    java.io.File file=null;
1284                    try{
1285                            URL modelURL =
this.getClass().getClassLoader().getResource( modelResourceName );
1286                            Scene scene = loader.load(modelURL);
1287                            group = scene.getSceneGroup();
1288                    }catch(Exception e){java.lang.System.out.println("File not found:
"+file);e.printStackTrace();group=new BranchGroup(); }
1289            }
1290            public void POAttributeChange(String attrib){
1291                    setShape(group);
1292                    super.POAttributeChange(attrib);
1293            }
1294            public void MIOAttributeChange(String attrib){
1295                    setShape(group);
1296                    super.MIOAttributeChange(attrib);
1297            }
1298    }
1299    package neve.data.pib.standard;
1300    import com.sun.j3d.utils.geometry.*;
1301    import com.sun.j3d.utils.universe.*;
1302    import javax.media.j3d.*;
1303    import javax.vecmath.*;
1304    import javax.management.*;
1305    public class NyxSwitchInterface extends ProtocolInstance {
1306            Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
1307            Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
```

```
1308          Color3f red = new Color3f(0.7f, .15f, .15f);
1309          Color3f blue = new Color3f(0.15f, .7f, .15f);
1310          public void MIOAttributeChange(String attribute){
1311                  BranchGroup bg= new BranchGroup();
1312                  float VLAN=(float)getVLAN();
1313                  Color3f col = new Color3f(0.5f, 0.5f, .1f/(VLAN+.01f));
1314                  ColoringAttributes ca=new
ColoringAttributes(col,ColoringAttributes.NICEST);
1315                  Appearance ap=new Appearance();
1316                  ap.setColoringAttributes(ca);
1317                  Cone co = new Cone(0.15f,0.3f,ap);
1318                  bg.addChild(co);
1319                  setShape(bg);
1320                  super.MIOAttributeChange(attribute); // Important
1321          }
1322          public int getVLAN(){
1323                  ObjectName MIO=null;
1324                  try{
1325                          MIO=new ObjectName(getMIOID().toString());
1326                  }catch(Exception e){e.printStackTrace();}
1327                  int vlan=0;
1328                  try{
1329
vlan=(int)((Integer)getServer().getAttribute(MIO,"VLAN")).intValue();
1330                  }catch(Exception e){e.printStackTrace();}
1331                  return vlan;
1332          }
1333  }
1334
1335  //*********************************************************
1336  // BaseAgent Serverklasse  (Zeile 1335 - 1596)             *
1337  // NetCamera als MBean bei Applicationserver registrieren  *
1338  //*********************************************************
1339  package neve.server;
1340  import neve.data.applications.*;
1341  //import neve.data.applications.CNM.*;
1342  import neve.core.services.*;
1343  import neve.core.objects.*;
1344  import neve.data.functions.*;
1345  //
1346  import java.util.*;
1347  import java.io.*;
1348  import java.net.*;
1349  // imports
1350  //
1351  import java.lang.reflect.Constructor;
1352  import java.util.Iterator;
1353  import javax.management.*;
1354  import javax.management.ObjectName;
1355  import javax.management.MBeanServer;
1356  import javax.management.MBeanServerFactory;
1357  import javax.management.MalformedObjectNameException;
1358  import mx4j.adaptor.rmi.jrmp.*;
1359  import mx4j.util.*;
1360  import mx4j.adaptor.http.HttpAdaptor;
1361  import mx4j.log.Log;
1362  import mx4j.log.Logger;
1363  import com.sun.jdmk.comm.HtmlAdaptorServer;
1364  public class BaseAgent {
1365          /*
1366           * ----------------------------------------
1367           *  CONSTRUCTORS
1368           * ----------------------------------------
1369           */
```

```
1370
1371            public BaseAgent() {
1372
1373            }
1374            /*
1375             * -------------------------------------
1376             *  PUBLIC METHODS
1377             * -------------------------------------
1378             */
1379
1380            public static void main(String[] args) {
1381
1382
1383                    // Toggle Logger
1384
1385                    Log.setDefaultPriority(Logger.WARN);
1386                    // CREATE the MBeanServer
1387                    //
1388                    System.out.println("\n\tCREATE the MBeanServer.");
1389                    MBeanServer server =
MBeanServerFactory.createMBeanServer("Neve");
1390                    //System.out.println(server.getMBeanServerId());
1391                    // CREATE and START a new HTML adaptor
1392                    //
1393                    System.out.println("\n\tCREATE, REGISTER and START the HTML
Adaptors and the Agents:");
1394                    HttpAdaptor http = new HttpAdaptor();
1395                    ObjectName http_name = null;
1396                    try {
1397                            http_name = new
ObjectName("Adaptor:name=http,port=9091");
1398                            System.out.println("\tOBJECT NAME           = " +
http_name);
1399                            server.registerMBean(http, http_name);
1400                            http.setPort(9091);
1401                            http.setHost("0.0.0.0");
1402                            http.start();
1403                    } catch(Exception e) {
1404                            System.out.println("\t!!! Could not create the HTTP
adaptor !!!");
1405                            e.printStackTrace();
1406                            return;
1407                    }
1408                    HtmlAdaptorServer html = new HtmlAdaptorServer(9090);
1409                    ObjectName html_name = null;
1410                    try {
1411                            html_name = new
ObjectName("Adaptor:name=html,port=9090");
1412                            System.out.println("\tOBJECT NAME           = " +
html_name);
1413                            server.registerMBean(html, html_name);
1414                    } catch(Exception e) {
1415                            System.out.println("\t!!! Could not create the HTML
adaptor !!!");
1416                            e.printStackTrace();
1417                            return;
1418                    }
1419                    html.start();
1420                    // Erzeuge AnwendungsService
1421
1422                    AnwendungsService API = new AnwendungsService();
1423
1424                    ObjectName API_name = null;
1425                    try {
```

```
1426                            API_name = new
ObjectName(NeveService.AnwendungsServName);
1427                            System.out.println("\tOBJECT NAME           = " +
API_name);
1428                            server.registerMBean(API, API_name);
1429                    } catch(Exception e) {
1430                            System.out.println("\t!!! Der AnwendungsService konnte
nicht erzeugt werden !");
1431                            e.printStackTrace();
1432                            return;
1433                    }
1434                    // Erzeuge MappingService
1435
1436                    MappingService mapper = new MappingService();
1437                    ObjectName mapper_name = null;
1438                    try {
1439                            mapper_name = new
ObjectName(NeveService.MappingServName);
1440                            System.out.println("\tOBJECT NAME           = " +
mapper_name);
1441                            server.registerMBean(mapper, mapper_name);
1442                    } catch(Exception e) {
1443                            System.out.println("\t!!! Could not create the
MappingService !!!");
1444                            e.printStackTrace();
1445                            return;
1446                    }
1447
1448                    LayoutService layout = new LayoutService();
1449                    ObjectName layout_name = null;
1450                    try {
1451                            layout_name = new
ObjectName("Neve.Base:name=LayoutService");
1452                            System.out.println("\tOBJECT NAME           = " +
layout_name);
1453                            server.registerMBean(layout, layout_name);
1454                    } catch(Exception e) {
1455                            System.out.println("\t!!! Could not create the
LayoutService !!!");
1456                            e.printStackTrace();
1457                            return;
1458                    }
1459                    // Create GUI-Connector
1460
1461                    try{
1462                            // Create and start the naming service
1463                            ObjectName naming = new
ObjectName("Naming:type=rmiregistry");
1464                            server.createMBean("mx4j.tools.naming.NamingService",
naming, null);
1465                            server.invoke(naming, "start", null, null);
1466
1467                    } catch(Exception e) {
1468                            System.out.println("\t!!! Could not create the
NamingService !!!");
1469                            e.printStackTrace();
1470                            return;
1471                    }
1472
1473                    System.out.println("\tOBJECT NAME           = NamingService");
1474
1475                    try {
1476                            // Create the JRMP adaptor
1477                            ObjectName adaptor = new
```

```
ObjectName("Adaptor:protocol=JRMP");
1478                            server.createMBean("mx4j.adaptor.rmi.jrmp.JRMPAdaptor",
adaptor, null);
1479                            JRMPAdaptorMBean mbean =
(JRMPAdaptorMBean)StandardMBeanProxy.create(JRMPAdaptorMBean.class, server, adaptor);
// Set the JNDI name with which will be registered
1480                            String jndiName = "jrmp";
1481                            mbean.setJNDIName(jndiName);
1482                            // Register the JRMP adaptor in JNDI and start it
1483                            System.out.println("\tOBJECT NAME           = JRMP
Adaptor");
1484                            mbean.start();
1485                    } catch(Exception e) {
1486                            System.out.println("\t!!! Could not create the
Connector !!!");
1487                            e.printStackTrace();
1488                            return;
1489                    }
1490                    // Create Visualiser
1491                    try {
1492                            VisualiserObject vis = new VisualiserObject();
1493                            ObjectName vis_name = new
ObjectName("Neve.Base:name=Visualiser");
1494                            System.out.println("\tOBJECT NAME           = " +
vis_name);
1495                            server.registerMBean(vis, vis_name);
1496                    } catch(Exception e) {
1497                            System.out.println("\t!!! Could not create the Visualiser
!!!");
1498                            e.printStackTrace();
1499                            return;
1500                    }
1501                    // Create BeanCreator
1502
1503                    ExampleNetCreator creator = new ExampleNetCreator();
1504                    ObjectName creator_name = null;
1505                    try {
1506                            creator_name = new
ObjectName("Neve.Managementanwendung:name=ExampleNetCreator");
1507                            System.out.println("\tOBJECT NAME           = " +
creator_name);
1508                            server.registerMBean(creator, creator_name);
1509                    } catch(Exception e) {
1510                            System.out.println("\t!!! Could not create the
ExampleNetCreator !!!");
1511                            e.printStackTrace();
1512                            return;
1513                    }
1514                    // Create VLANCreator
1515
1516                    ExampleVLANCreator vlancreator = new ExampleVLANCreator();
1517                    ObjectName vlancreator_name = null;
1518                    try {
1519                            vlancreator_name = new
ObjectName("Neve.Managementanwendung:name=ExampleVLANCreator");
1520                            System.out.println("\tOBJECT NAME           = " +
vlancreator_name);
1521                            server.registerMBean(vlancreator, vlancreator_name);
1522                    } catch(Exception e) {
1523                            System.out.println("\t!!! Could not create the
ExampleVLANCreator !!!");
1524                            e.printStackTrace();
1525                            return;
1526                    }
```

```
1527
1528                    // Create VLANHighlighter
1529
1530                    VLANHighlighter vlanhl = new VLANHighlighter();
1531                    ObjectName vlanhl_name = null;
1532                    try {
1533                            vlanhl_name = new
ObjectName("Neve.Visualisierungsfunktionen:name=VLANHighlighter");
1534                            System.out.println("\tOBJECT NAME           = " +
vlanhl_name);
1535                            server.registerMBean(vlanhl, vlanhl_name);
1536                    } catch(Exception e) {
1537                            System.out.println("\t!!! Could not create the
VLANHighlighter !!!");
1538                            e.printStackTrace();
1539                            return;
1540                    }
1541
1542                    // Create HTTPCreator
1543
1544                    ExampleHTTPCreator httpcreator = new ExampleHTTPCreator();
1545                    ObjectName httpcreator_name = null;
1546                    try {
1547                            httpcreator_name = new
ObjectName("Neve.Managementanwendung:name=ExampleHTTPCreator");
1548                            System.out.println("\tOBJECT NAME           = " +
httpcreator_name);
1549                            server.registerMBean(httpcreator, httpcreator_name);
1550                    } catch(Exception e) {
1551                            System.out.println("\t!!! Could not create the
ExampleHTTPCreator !!!");
1552                            e.printStackTrace();
1553                            return;
1554                    }
1555                    // Create NetCamera
1556
1557                    NetCamera nc = new NetCamera ();
1558                    ObjectName nc_name = null;
1559                    try {
1560                            nc_name = new
ObjectName("Neve.Managementanwendung:name=NetCamera");
1561                            System.out.println("\tOBJECT NAME           = " +
nc_name);
1562                            server.registerMBean(nc, nc_name);
1563                    } catch(Exception e) {
1564                            System.out.println("\t!!! Could not create the
NetCamera !!!");
1565                            e.printStackTrace();
1566                            return;
1567                    }
1568                    // Create Sniffer
1569                    PCap pcap=new PCap();
1570                    ObjectName pcap_name=null;
1571                    try{
1572                            pcap_name=new
ObjectName("Neve.Managementanwendung:name=PaketSniffer");
1573                            System.out.println("\tOBJECT Name                = " +
pcap_name);
1574                            server.registerMBean(pcap, pcap_name);
1575                    } catch(Exception e){
1576                            System.out.println("\t!!! Could not create the SnifferApp
!!!!");
1577                            e.printStackTrace();
1578                            return;
```

```
1579                    }
1580                    // Traceroute Function
1581                    Traceroute tracerouter = new Traceroute();
1582                    ObjectName oname = null;
1583                    try {
1584                            oname = new
ObjectName("Neve.Managementanwendung:name=Tracerouter");
1585                            System.out.println("\tOBJECT NAME           = " + oname);
1586                            server.registerMBean(tracerouter, oname);
1587                    } catch(Exception e) {
1588                            System.out.println("\t!!! Could not create the
Tracerouter !!!");
1589                            e.printStackTrace();
1590                            return;
1591                    }
1592
1593                    System.out.println("Server "+server+" ready !");
1594
1595            }
1596    }
1597    //**************************************
1598    // JDOM benutzen  (Zeile 1597 - 1627)  *
1599    //**************************************
1600
1601    /*Für NetCamera wird jdom-1.0 benötigt.
1602
1603    http://www.jdom.org/
1604
1605    http://www.jdom.org/dist/binary/jdom-1.0.zip
1606
1607    Nach dem Unpacken folgende vier jar Datei in lib Verzeichnis
1608    von Neve kopieren.
1609
1610    jaxen-core.jar, jaxen-jdom.jar, jdom.jar, saxpath.jar
1611
1612    ls -la neve/lib
1613
1614    160K    lib/jaxen-core.jar
1615    8.0K    lib/jaxen-jdom.jar
1616    276K    lib/jdom.jar
1617    24K     lib/saxpath.jar
1618
1619    Nach dem Kompilieren folgendes in startserver.sh hinzufuegen:
1620
1621     CLASSPATH=$CLASSPATH:../../lib/jaxen-jdom.jar
1622     CLASSPATH=$CLASSPATH:../../lib/jdom.jar
1623     CLASSPATH=$CLASSPATH:../../lib/saxpath.jar
1624     CLASSPATH=$CLASSPATH:../../lib/jaxen-core.jar:.
1625
1626    */
1627
```