JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

SNMP Wrapperlibrary

EasySnmp

Diplomarbeit zur Erlangung des akademischen Grades

Diplom-Ingenieur

in der Studienrichtung *Informatik*

Angefertigt am Institut für

Informationsverarbeitung und Mikroprozessortechnik (FIM)

Betreuung:

o. Univ.-Prof. Dr. Jörg R. Mühlbacher

Dipl. Ing. Rudolf Hörmaseder

Von:

Daniel Fallmann

Beurteilung:

o. Univ.-Prof. Dr. Jörg R. Mühlbacher

Linz, 30.Oktober 2004.

Widmung

meiner Freundin

Danksagung

Mein Dank gilt zu allererst meinen Eltern, die mir diese Ausbildung ermöglicht haben. Während der gesamten Studiendauer haben sie mich sowohl finanziell als auch moralisch unterstützt, meinen Weg zu gehen.

Besonderer Dank gilt meinem sehr engagierten Betreuer o. Prof. Dr. Mühlbacher Jörg, dem Vorstand des Instituts für Informationsverarbeitung und Mikroprozessortechnik (FIM), der mir so viel Vertrauen entgegenbrachte und mir ermöglichte dieses Projekt im Rahmen meiner Diplomarbeit zu bearbeiten.

Persönlich sehr geholfen hat mir die Unterstützung von Dipl.-Ing. Hörmanseder Rudolf, der mir laufend über ein Jahr hinweg gezeigt hat, wie man mit einem Projekt dieser Größe zurande kommt und vor allem dabei das Ziel, niemals aus den Augen verliert. Besten Dank auch, für die langen Nächte, die wir gemeinsam an meinem Projekt getüftelt haben.

Dank gilt auch Andreas Dangl, Andreas Pramöck und Bernhard Scholze der Firma Fabasoft, die mich im Zuge dieses Projektes richtungsweisend unterstützt haben.

Ich möchte an dieser Stelle vier weiteren, für mich ganz besonderen Menschen, meinen tiefsten Dank aussprechen.

- Meinem Bruder Helmut, der diese Diplomarbeit erst ermöglicht hat. Er hat mich immer wieder mit guten Ratschlägen unterstützt und mir geholfen, in schwierigen Zeiten zu versuchen die eingeschlagene Richtung beizubehalten.
- Meinem Bruder Dietmar, der mich privat, neben dem Studium und meiner Berufstätigkeit versucht hat mich durch Arbeit am Bauernhof (er betreibt nebenberuflich eine Landwirtschaft) abzulenken.
- Meiner Schwester Petra, die mir immer liebevoll ihre beiden Kinder Nicole und Lukas (meine Patenkinder) anvertraut hat. Die beiden haben es stets geschafft, mich aufzuheitern.
- Last but not least, möchte ich meiner Freundin Iris danken. Sie ist für mich ein ständiger Lichtblick. Sie hat mir im letzten Jahr sehr viel Geduld und Vertrauen entgegengebracht und mich ständig motiviert und bestärkt meinen Weg zu gehen. Ich hab´ Dich lieb.

Danke!

Kurzfassung / Abstract

Kurzfassung

Netzwerkmanagement ist für den Betrieb und die Überwachung mittlerer bis großer Computernetze unverzichtbar. Das Simple Network Management Protocol (SNMP), ist das de facto Standardprotokoll für Netzwerkmanagement, insbesondere in heterogenen Umgebungen. Gängige SNMP Agenten sind dabei oft monolithisch und stellen Informationen bereit, die so genannten Managed Objects. Das Konzept der Subagenten erlaubt nun die Bereitstellung von Managed Objects auf mehrere solche Subagenten zu verteilen, die alle von einem gemeinsamen Masteragenten verwaltet werden.

Diese Arbeit untersucht, inwieweit SNMP und das Subagentenkonzept für das Management von Komponenten eines großen Software Systems (konkret Fabasoft eGov Suite) geeignet sind.

Die im Zuge dieser Arbeit entstandenen Wrapper-Bibliothek EasySnmp bietet eine einfache Möglichkeit zur Erstellung von Standard Managed Objects anzubieten.

Die Subagenten sind dabei direkt in den jeweiligen zu verwaltenden Komponenten implementiert und kommunizieren mit ihrem zuständigen Masteragenten. Auf Basis EasySnmp, wird anhand von zwei Beispielszenarien aufgezeigt, wie die, in dieser Arbeit entstandene Bibliothek verwendet werden kann. EasySnmp baut auf die Open Source Bibliothek Net-SNMP auf, bietet jedoch eine einfacherere Schnittstelle. Zusätzlich kann bei Bedarf die Standardfunktionalität durch Net-SNMP erweitert werden.

Abstract

Network management is essential for the operation and supervision of medium to large computer networks. The Simple Network Management Protocol (SNMP) is the de facto standard protocol for network management. Many available implementations of SNMP agents are monolithic and implement all information, the so called managed objects, in a single program. The concept of subagents makes it possible to delegate the implementation of Managed Objects to several subagents and they all are managed by a so called master agent.

This thesis examines how SNMP and the concept of subagents can be used for the management of a large software system, like the Fabasoft eGov Suite. With the implemented wrapper-library we will show an easy way how to implement standard managed objects. We assume that all subagents are implemented by the managed components and communicate themselves with the central master agent. Furthermore we show that the functionality of EasySnmp can be extended by the usage of the open source library Net-SNMP.

Inhaltsangabe

Leseanleitung

Die Struktur dieser Arbeit versucht, möglichst speziell für den Informationsbedarf einzelnener Benutzergruppen spezielle Kapitel anzubieten:

Für **interessierte Leserinnen und Leser** mit Interesse an SNMP gibt es eine ausführliche Einleitung, die hoffentlich Lust auf mehr macht. Hier gibt es die Möglichkeit zu schmökern und erste Gehversuche, wie am Beispiel „helloWorld SNMP“ eingeführt, zu machen.

Für **Administratoren** sind vor allem die Themenbereiche Installation und Konfiguration von Interesse. Aus unserer Sicht sind die Basiskapitel ... von Vorteil.

Für **Softwareentwickler** sind im speziellen der Bereich Agentensysteme, deren Architektur und die Bibliothek EasySnmp geeignet. Um einen Einblick in EasySnmp und die umfassende Grundbibliothek Net-SNMP zu erhalten ist das Kapitel ... zu empfehlen. Natürlich ist in diesem Zusammenhang auch das Kapitel ... über die Konfiguration von Net-SNMP und AgentX essentiell wichtig.

VORSICHT: SNMP kann süchtig machen.

Viel Spaß beim Lesen!

1 Allgemeine Einführung

Even goldfish can be taught to play the piano,
if you use enough voltage.
— Jeffrey D. Case

1.1 Motivation für Systemmanagement

Heutige IT Landschaften unterscheiden sich von denen früherer Zeiten deutlich. Die Anzahl der in einem Netzwerk eingebundenen Komponenten, sowohl Hardware- als auch Softwarekomponenten steigt stetig weiter an. Diese Tatsache ist vor allem in der umfangreichen Nutzung der elektronischen Daten und der dadurch nötigen Datenhaltung in den verschiedensten Lebensbereichen begründet. Die Rechenzentren müssen folglich sehr hohen Anforderungen genügen. Wesentliche Herausforderungen sind die hohe Verfügbarkeit, vor allem durch Online-Backup und Online-Recovery von kritischen Daten als auch eine immer höhere effektive Rechenleistung, die es vor allem auch ermöglicht, große Datenbestände, vernünftig zu verwalten und aufzubereiten.

Diesen Ansprüchen und der als Folge explodierenden Anzahl von zu wartenden Systemen, wird versucht, mittels Netzwerkmanagementsystemen Herr zu werden.

Das de facto Standardprotokoll für das Management von Hardware- und Softwarekomponenten ist das Simple Network Management Protocol (SNMP). Dieses Protokoll wurde 1988 eingeführt, um Systeme per Internet Protocol (IP) zu managen. Über die Jahre hat sich SNMP immer mehr zu dem Universalprotokoll im Netzwerkmanagementbereich entwickelt.

Die Kernfunktionalität von SNMP hat sich aber über die Jahre nicht wesentlich verändert, was die Qualität des ersten Ansatzes unterstreicht. Das SNMP Protokoll besteht aus einer kleinen Menge von essentiellen Funktionen (GET, SET, Trap). Mit GET-Operationen kann der Anwender den aktuellen Status eines Systems abfragen. Mit einer SET-Funktion werden Werte gesetzt. Und Traps werden von Systemen benutzt, um aktiv auf Ereignisse aufmerksam zu machen.

Im Gegensatz zu seinem Vorgängersystem SGMP (Simple Gateway Management Protocol), welches nur für den Einsatz in Hardware, im Speziellen in Router, bestimmt war, ist SNMP nun sowohl für Hardware (Router, Switches, Modems, etc.) als auch für alle Softwareapplikationen (Windows, Unix, Linux, Datenbanken, Serversysteme, Webserver, Applicationserver, Cluster,

etc.) im Einsatz. Diese Systeme liefern eine Fülle an Information, daher ist es wichtig vor allem im Systemmanagementbereich ganz genau zu überlegen, welche Werte wirklich aussagekräftig sind und benötigt werden. Der Benutzer soll durch die Präsentation von signifikanten Werten in die Lage versetzt werden Probleme deutlich früher zu erkennen und proaktiv auf diese Situationen zu reagieren.
Vor einigen Jahren warb die Firma SUN Microsystems mit folgendem Werbeslogan: „The Network is the Computer“. Obwohl dieser Ausspruch als Werbung für die Programmiersprache Java gedacht war, trifft diese Aussage im Grunde auch für heutige Netzwerke in vielen Bereichen zu. Da Dienste mittlerweile nicht nur lokal genutzt, sondern auch verteilt im Netzwerk bereitgestellt werden, ist Hochverfügbarkeit des Netzwerkes ein entscheidender Faktor. Jedoch nützt das beste Netzwerk nichts, wenn über die Dienste keine Statusinformation vorliegt, die Auskunft darüber gibt, wie es um den jeweiligen Dienst steht. Typische Beispiele für solche Statusinformationen sind Auslastung, Antwortzeiten, Speicherbedarf,...
Im Wesentlichen kann man für diese Arbeit drei Bereiche unterscheiden, die ein Managementsystem genauer klassifizieren:

- Netzwerkmanagement
- Systemmanagement
- Anwendungsmanagement

Im Folgenden soll ein kurzer Überblick über die einzelnen Bereiche gegeben werden.

Netzwerkmanagement: Unter Netzwerkmanagement versteht man das Management von Kommunikationsdiensten und Netzwerkkomponenten. Es ist zuständig für die Fehlersuche und den reibungslosen Betrieb von Netzwerkkomponenten und deren Kopplungselementen.
Netzwerkmanagement teilt sich wieder in fünf Funktionsbereiche:

- Netzsteuerung,
- Fehlermanagement,
- Konfigurationsmanagement,
- Performance-Management und
- Benutzerverwaltung.

Später wird auf die so genannten OSI Specific Management Functional Areas eingegangen [Lei93], die im Wesentlichen den Bereich Netzwerkmanagement vollständig abdecken.

Systemmanagement: Systemmanagement befasst sich mit der Abbildung der Ressourcen von Endsystemen (Hosts, Drucker, etc.) und generell auch mit komplexeren Systemen, die aus mehreren Komponenten aufgebaut sind. Wesentliche Funktionsbereiche von Systemmanagement sind die Fehlererkennung, Leistungsanalyse, Installation/Update von Software auf verteilten Systemen, Sicherheit und Überwachung von Endsystemen und die Überwachung der aktuell im Netzwerk befindlichen Komponenten mit Hilfe einer Inventarliste. Komponenten wie CPU, Speicher, laufende Prozesse, etc. werden durch diese Managementkategorie adressiert.

Anwendungsmanagement: Anwendungsmanagement kümmert sich um die Konfiguration, Überwachung, Aktivierung und Deaktivierung von verteilten Anwendungen (bzw. Prozessen). Das Gebiet des Anwendungsmanagements auf Basis eines standardisierten Managementprotokolls ist noch im Anfangsstadium und bedarf noch weiterer Forschung. Die Diplomarbeit untersucht diese Problematik in Hinblick auf die Möglichkeiten von SNMP im Anwendungsmanagement, konkret an Hand der Fabasoft Components Bereiche MMCStore, COOStore, RPCService, Kernel, etc..

Alle drei Managementkernbereiche decken einen spezifischen Teilbereich im System ab und helfen in Kombination dabei, eine komplexe IT Infrastruktur gezielt zu überwachen und zu konfigurieren.

(weitere Informationen kann [Tan03], [Per97], [Lei93] entnommen werden)

1.2 Geschichte von SNMP

Der Ursprung des Simple Network Management Protocol (SNMP) liegt schon einige Jahre zurück. Mitte der 80er Jahre stieg die Anzahl, der mit dem Internet verbundenen Netzwerke deutlich an. Zu diesem Zeitpunkt gab es keinen offenen Standard im Bereich des Netzwerkmanagements und so begannen die Hersteller unabhängig voneinander Netzwerkmanagementprodukte zu entwickeln. Da aber nicht alle Netzwerkkomponenten von einem einzigen Hersteller waren, wurde der Aufwand für die Überwachung der verschiedensten Geräte sehr groß. 1987 traten die Entwickler verschiedenster Firmen zusammen und begannen, ein gemeinsames Protokoll zu erarbeiten, das einen offenen Standard darstellte. Mit dem Simple Gateway Monitoring Protocol (SGMP) konnte man einen ersten gemeinsamen Erfolg verzeichnen. Da SGMP nur für das Management von Gateways geeignet war, wurde davon ausgehend ein neues Protokoll namens SNMP (Simple Network Management Protocol) entwickelt und 1988 als RFC standardisiert. Mit SNMP hatte man nun ein allgemeines Übertragungsprotokoll zur Verfügung, das man für Managementzwecke von verschiedensten Netzwerkknoten einsetzen konnte.

1988 SNMPv0: In einer ersten experimentellen Version, SNMPv0, wurden die Fähigkeiten des Protokolls getestet.

1988 SNMPv1: Im selben Jahr entstand die erste kommerziell eingesetzte Version, SNMPv1. Zu dieser Zeit wurde parallel ein weiteres Managementprotokoll namens CMIP[1] entwickelt, welches aber von der Wirtschaft nicht im gleichen Maße wie SNMP angenommen wurde. SNMP hat ab 1988 kontinuierlich an Beliebtheit gewonnen und diese bis heute nicht verloren. (RFC ... einfügen)

1993 SNMPv2: Da die SNMP Version 1 eine Reihe von Problemen beinhaltete, wurde 1993 die neue Version, SNMPv2, vorgestellt. Mit dieser Version wurde das Problem der Übertragung von großen Datenmengen gelöst. Auch wurde versucht die mangelhafte Sicherheit zu verbessern und notwendige Manager-Agenten-Hierarchien einzubauen. Bei der ersten SNMPv2 Version (man

[1] In dieser Arbeit wird nicht näher auf das Protokoll CMIP (Common Managment Information Protocol) und CMOT (CMIP over TCP/IP) eingegangen, jedoch wird kurz zur Vollständigkeit erklärt, was es damit auf sich hat. Parallel zu SNMP wurde Ende der 80er Jahre ein weiteres Protokoll entwickelt, das versuchte auf Basis des OSI-Modells Netzwerke zu verwalten. Obwohl anfänglich die Organisation IAB (Internet Activities Board) versuchte, die technische Entwicklung und den Einsatz von CMOT zu forcieren und nur kurzfristig SNMP einzusetzen, entschied sich die Wirtschaft gegen CMOT. Heute gibt es weltweit ganz wenige CMOT Implementierungen.

nennt diese Version heute **SNMPv2p**) wurde der Sicherheitsaspekt zwar ausreichend abgedeckt, jedoch war der verwendete Algorithmus und das System als Ganzes so schwierig zu implementieren, dass nur ein großer Hersteller (IBM) dies auch tatsächlich getan hat. Aus diesem Grund nahm man den Sicherheitsaspekt wieder aus dem Protokoll und hatte 1996 mit SNMPv2c ein neues Derivat. SNMPv2c wurde nun tatsächlich wieder von sehr vielen Anbietern umgesetzt. 1997 wurde auf Basis eines neuen Sicherheitmodells versucht, Sicherheitsaspekte in SNMPv2 (als SNMPv2*) einzubauen. Jedoch wenig erfolgreich. **SNMPv2*** wird heute als schwer verständlich eingestuft und wurde niemals als RFC dokumentiert. Im selbem Jahr wurde parallel dazu mit SNMPv2u an einer Alternative gearbeitet, die richtungweisend für die weitere Entwicklung sein sollte. Mit dem in dieser Version bereits angesprochenen „User-based Security Model (USM)“, wurde ein wesentlicher Schritt in Richtung Authentifizierung von Benutzern getan. **SNMPv2u** ließ wesentliche Fragen, wie die konkret zu verwendenden Standard-Verschlüsselungsalgorithmen offen. Leider behob man diesen Mißstand erst ein Jahr später.

1998 SNMPv3: Da SNMPv2c (das 'c' steht für Community, da hier wieder Communitystrings als Passwörter verwendet werden) fand zwar großen Anklang bei den Herstellern, war aber nach wie vor nicht sicher. Als logische Konsequenz wurde eine neue Version des Protokolls (SNMPv3) standardisiert. Mit SNMPv3 wurde das bereits in SNMPv2p vorgestellte USM Modell teilweise übernommen, verbessert und konkretisiert. Die Ergänzung durch das VACM (View-Based Access Control Model) führte nun auch einen vielseitigen Autorisierungsmechanismus in SNMP ein. In diesem Modell wird beschrieben, wie Benutzergruppen und Rechte verwaltet werden. Später wird im Subkapitel SNMPv3 näher auf die Funktionalität eingegangen. Im Zuge dieser Arbeit musste leider festgestellt werden, dass noch nicht viele Hersteller den SNMPv3 Standard unterstützen. Am weitesten verbreitet ist zurzeit noch SNMPv2c.

(Literatur entnommen aus [Mil96], [Mar95], [Sta03])

1.3 Gründe für den Einsatz von Managementsystemen

Die Gründe für den Einsatz von Managementsystemen sind vielfältig. Der Einsatzbereich ist durch Erkenntnisse aus Forschung und Wirtschaft bereits sehr gut und fundiert ausgearbeitet. In einschlägiger Literatur findet man hier immer wieder einen Verweis auf die so genannten OSI Specific Management Functional Areas (SMFAs). Unter den SMFAs versteht man die definierte Aufteilung von komplexen Managementaufgaben in die Bereiche Performance, Accounting, Security, Fault und Configuration.

1.3.1 OSI SMFA

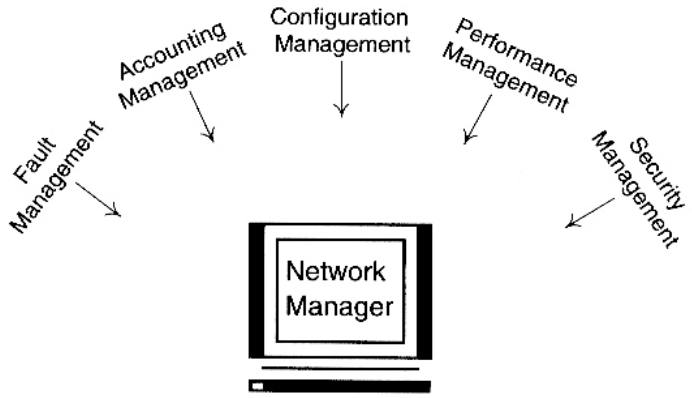

Abbildung 1: Specific Management Functional Areas, Quelle [Mil96, S. 21]

Fault Management

Der Funktionsbereich des Fault Management befasst sich mit der Fehlererkennung und Fehlerbehebung. In diesen Bereich fallen sowohl die Problemerkennung, die Durchführung und Auswertung von Netzwerk- und Komponententests, die Problembehebung als auch die Überwachung des Netzwerks und seiner Komponenten.

Accounting Management

In der OSI Umgebung versteht man unter dem Bereich des Accounting Management das Arbeiten mit benutzerzentrierten Daten und Netzwerkkonfigurationen. Darunter versteht man, die von einem Benutzer hervorgerufenen Auslastung von Netzwerkkomponenten, das limitieren dieser Auslastung durch Quotas, etc.

Das Accounting Management wird nicht direkt von heute verfügbarer Software abgedeckt. Anstelle dessen kümmert sich ein Administrator darum, dass die Netzwerkstruktur den

Anforderungen der einzelnen Benutzer genügt. Dafür kann ein Administrator jedoch wiederum ein Managementwerkzeug einsetzen, um beispielsweise Performanceengpässe im System aufzudecken.

Configuration Management

Netzwerke und Applikationen sind äußerst dynamische Elemente. Die Anzahl der Komponenten, sowohl in Hardware als auch in Software und deren Funktionsvielfalt steigt stetig. Das „Configuration Management“ befasst sich mit der Konfiguration dieser Komponenten von zentraler Stelle aus. Es wird zunehmend möglich ohne markenspezifisches Detailwissen der jeweiligen Komponente verschiedene heterogene Systeme auf einer Ebene zu verwalten und zu konfigurieren.

Performance Management

Hierunter versteht man die Analyse und Konfiguration der Leistung eines Rechners oder eines Netzwerks. Im Detail geht es um die Messung, Aufbereitung und Analyse von Performancedaten zur Erkennung von so genannten „bottlenecks“ (Engpässen, Engstellen) in Netzwerken oder Applikationen. Auch das Erstellen von Vorhersagen bzw. Trends der Performanceveränderung sind Teil des Performacemanagements. Darunter fallen auch Vorher-/Nachheranalysen bei der Umkonfiguration von Netzen oder Systemen. Alle diese Daten sind Basis zum Tuning des Netzwerkes.

Security Management

Ein leider oftmals vernachlässigter Bereich ist das Security Management. Mit Hilfe des Sicherheitsmanagements versucht man die sicherheitsrelevanten Kriterien aufzuspüren und diese zu konfigurieren.

Dazu gehören die Verwaltung der Zugangskontrolle zu bestimmten Diensten, die Authentifikation (Ist ein Benutzer der, der er vorgibt zu sein?), die Autorisierung (Rechtezuweisung an bestimmte Benutzer) und die Verwaltung von Passwörtern und Regelwerken zur Erstellung von sicheren Passwörtern zu einem vollständigen Security Management.

(Für vertiefende Information [Mil96], [Gol00])

1.4 Ziel der Arbeit

Die Ziele dieser Diplomarbeit stehen in engem Zusammenhang mit den im vorigen Kapitel angeführten generellen Zielen von Systemmanagement. Vorrangiges Ziel war, eine flexible und leicht zur Laufzeit erweiterbare Architektur zu schaffen, die es ermöglicht, ein Agentengefüge sehr schnell mittels einer einfach gehaltenen Schnittstelle zu erweitern.

Die Erfordernisse und der geplante Funktionsumfang wurden in enger Kooperation mit dem Auftraggeber ausgearbeitet. Dabei sollte möglichst wenig „neu erfunden“ werden. Vielmehr ging es darum nach aktiven Projekten im Open Source Bereich zu suchen und diese in Hinblick auf Stabilität und Kompatibilität zu den bestehenden Anforderungen im Systemmanagementbereich zu evaluieren. Für die Auwahl entscheidend ist die Plattformunabhängigkeit, die Kompatibilität zu bestehenden Standards und die zu erwartende Weiterentwicklung des zur Auswahl stehenden Open Source Projektes.

Auf Basis von SNMP sollte eine eigene WrapperLibrary (die in dieser Arbeit den Namen EasySnmp bekam) geschaffen werden. Wesentliches Ziel dabei war, eine genau an die evaluierten Anforderungen definierte Schnittstelle zur Verfügung zu stellen, damit Systeme als sogenannte Managed Systems nach außen per SNMP abgebildet werden können.
Ein Schwerpunkt lag darauf, das System so zu gestalten, dass es zur Laufzeit erweiterbar ist und durch diese Erweiterungen der durchgänige Bertieb des Systems nicht beeinflußt wird („no restart“).

Zusätzlich sollten die für das Systemmanagement nötigen Funktionalitäten für die Fabasoft eGov Suite herausgearbeitet und Basis-MIBs (Management Information Bases) erstellt werden.

Anmerkung: Die hier verwendete Begriffswelt wird in den nächsten Kapiteln vertiefend erläutert.

2 Einführung in SNMP

The sun rises, the sun sets, the Sun crashes.
[. . .] It is the way of things.
— Steve Conley

Wie bereits erwähnt liegt der Ursprung von SNMP („jedenfalls aus IT Sicht“) schon sehr weit zurück. Zu dieser Zeit gab es Objektorientierung, Komponenten und Agenten in der heutigen Form und Vielfalt noch nicht. In diesem Kapitel wird ein technischer Überblick über die einzelnen SNMP Versionen gegeben und die wesentlichen Unterschiede und Evolutionsstufen erläutert. SNMP wurde nicht für heutige Systeme konzipiert, kann aber durch gezielte Anwendung durchaus auch heutigen Maßstäben gerecht werden, da es zwar nur eine minimale, jedoch ausreichende Menge von Befehlen versteht (SET, GET, Trap).
Dieses Kapitel ist speziell für technisch interessierte Leser geschrieben, die Hintergrundinformationen erhalten möchten, um technische Entscheidungen zu treffen.

Die Motivation bei der Konzeption und für den Einsatz von Systemmanagementprodukten zielt immer auf die selben vorherrschenden Ziele ab, die bei der komplexer werdenden Hard- und Softwarearchitektur heutiger heterogener Systeme schwierig zu erreichen sind:

- schnelle Fehlererkennung
- raschest mögliche Fehlerbehebung
- Ursachenanalyse
- Systemüberwachung (Wie geht es meinem System?)
- Systeminformationen für Benutzer aufbereiten
- Konfiguration von eingeschränkten Systembereichen möglich
- viele Systeme von einer einzigen Konsole (einem Programm) aus überwachen
- hohe Sicherheitsanforderungen (insbesondere auch für Systemmanagement)

Vor allem Version 3 von SNMP war für die Anforderungen dieser Arbeit (im Lichte der oben angeführten Punkte) sehr gut geeignet. Im Zuge dieser Diplomarbeit wurde auch eine praktische Erprobung durchgeführt. Man muss sich klar darüber sein, dass SNMP alleine kein Allheilmittel ist, aber sich damit sehr große Infrastrukturen relativ einfach überwachen und managen lassen.

2.1 Grundgedanke zu SNMP

Anhand der sehr einfach zu verstehenden SNMP Version 1 soll nun das Basiswissen aufgebaut werden, das man benötigt, um mit SNMP tatsächlich auf technischer Ebene arbeiten zu können.

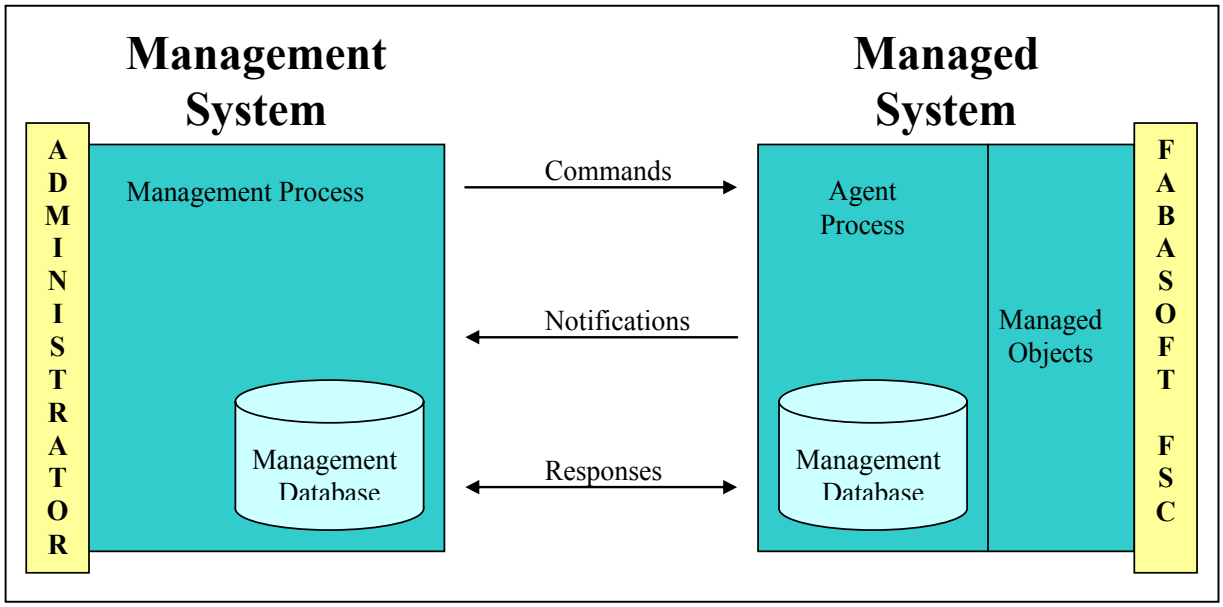

Abbildung 2: Komponenten einer System Management Architektur, adaptiert von Quelle [Mil96, S. 11]

In Abbildung 2 ist auf der linken Seite das Management System zu sehen (in der Literatur oftmals auch Network Node Manager genannt). Es gibt sehr viele ausgereifte Produkte am Markt. Als besonders etablierte Systeme sollen hier HP Openview, IBM Tivoli und Unicenter TNG genannt werden. Es sei aber erwähnt, dass es auch kleinere Systeme gibt, die durchaus ihren Einsatzzweck erfüllen. Diese Systeme sind dazu da, die verschiedenen Managed Systems von zentraler Stelle zu steuern. Abgesehen vom tatsächlich bestehenden Standard kann man überlegen, was es für ein Managementsystem grundsätzlich unbedingt braucht. Nach näherer Betrachtung erkennt man drei wesentliche Funktionalitäten (siehe dazu auch Abbildung 2):

- **Commands (Befehle):** Mit Hilfe von Befehlen kann man das Managed System steuern. So können mittels SET-Operationen Konfigurationsänderungen vorgenommen werden und mittels GET-Operationen Statusinformationen des Managed Systems gelesen werden. Das System antwortet auf diese Commands mit
- **Responses (Antworten).** Unter Antworten versteht man in diesem Zusammenhang die aktive Reaktion eines der beiden Beteiligten Systeme (Management System, Managed System) auf eine Anfrage des jeweils anderen Partners. Ganz besonders ist darunter die

Antwort auf eine Statusanfrage, die Bestätigung der Änderung von Daten und auch die Bestätigung des Empfangs von Daten zu verstehen.

- **Notifications (Benachrichtigungen):** Wenn das Managed System einen Fehler im lokalen System bemerkt, muss es möglich sein, dies einem oder mehreren Management Systemen mitzuteilen (ohne dass diese bzw. bevor diese im Rahmen eines normalen Pollingzyklus danach fragen).

Dieser einfache Standardaufbau zieht sich durch alle SNMP Versionen. Alles was auf Basis dieses Konzeptes realisierbar ist, kann mit SNMP tatsächlich verwirklicht werden. Alles Weitere wird nicht adressiert.
Um die Abbildung 2 vollständig zu verstehen, fehlt eigentlich nur mehr das Verständnis der Management Database. Leider wird in der Literatur oft, wenn man von so genannten Management Information Bases (MIBs) spricht, eine Verbindung zu Datenbanken hergestellt, was oftmals zur Verwirrung beiträgt.
Für dieses gedankliche Schema fehlt noch eine Vereinbarung zwischen der linken (Mangement System) und der rechten Seite (Managed System), in der beide Seiten übereinkommen, welche Daten angeboten werden und wie diese zu verstehen/interpretieren sind. Konkret bietet das Managed System Daten dem System Manager an. Die angebotenen Daten (Managed Objects) werden in einer MIB definiert. Man darf eine MIB jedoch nicht als Datenbank sehen, sondern vielmehr als einen Vertrag der beiden Vertragspartner Management System und Managed System. Die folgende Grafik soll dies verdeutlichen.

(vertiefende Literatur findet man unter [Per97], [Mil96], [Sch03])

MIB = Management Information Base

Aufgabe u.a. Zuordnung OID zu symbolischer Name,

Datentyp, Beschreibung, Zugriffsmöglichkeit

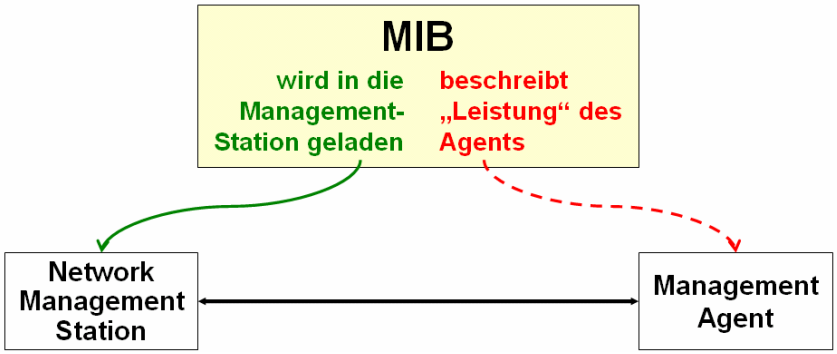

Abbildung 3: MIB, Quelle [aus LVA: Netzwerkadministration, FIM, 2002]

Eine Management Information Base (MIB) beschreibt die Leistung eines Managed Systems (auch Management Agent genannt) mit Hilfe einer Textdatei. Mit einer MIB wird eine Baumstruktur beschrieben. Ein Knoten hat jeweils einen speziellen Typ und kann dann per SNMP geschrieben oder gelesen werden. Eine Position im Baum wird durch einen eindeutigen OID (Object Identifier) spezifiziert. Eine OID ist eine durch Punkte getrennte Zahlenfolge, zum Beispiel **1.8.5.1.0**, wobei jede Zahl die n-te Abzweigung im Baum beschreibt. Daten werden nicht von allen Baumknoten bereitgestellt, sondern nur von den Blättern.

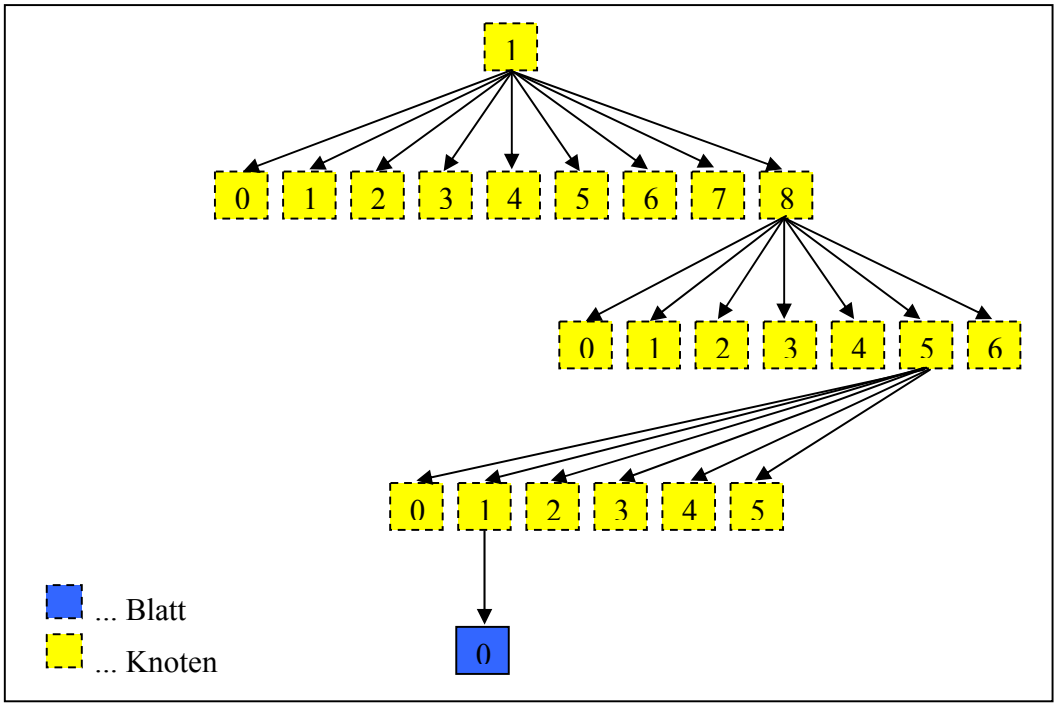

Abbildung 4: OID Erklärung

In einer MIB müssen zusammenfassend für jeden Eintrag einige wichtige Felder beschrieben werden:

- ein eindeutiger Bezeichner für jeden Knoten im Baum (OID)
- der Datentyp der abzubildenden Daten
- die Zugriffsart (read-only, write-only, read-write, not-accessible)
- der Status (current, deprecated, obsolete)
- die Beschreibung was dieser konkrete Knoten beschreibt

```
numinstalledcooservices OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Number of Server COO-service instances."
  ::= { cooService 1 }
```

Am Beispiel ist zu sehen, wie ein einzelnes Managed Object textuell in einer MIB beschrieben wird. „numinstalledcooservices" ist der Name des Managed Objects (siehe dazu Abbildung 2). Mit dem Schlüsselwort SYNTAX wird der Dateityp angegeben. Mit MAX-ACCESS wird festgelegt, welche Zugriffsarten auf das Objekt möglich sind. Mit STATUS wird der Status des Objekts näher spezifiziert und mittels DESCRIPTION kann man zur besseren Interpretation eines Wertes eine textuelle Beschreibung angeben. Die letzte Zeile im Beispiel legt die Position des Objektes im Baum fest. In diesem konkreten Fall wird das „numinstalledcooservices" Objekt unter „cooServices" als erste (1) Abzweigung eingehängt.

Die einzelnen Ausprägungen der Schlüsselwörter und die Werte, die diesen zugewiesen werden dürfen, sind in einem eigenen RFC festgelegt. Man spricht in diesem Kontext auch von der Structure of Management Information (SMI). Mittlerweile gibt es zwei Versionen von SMI, die sich vor allem in den unterstützten Datentypen unterscheiden. Tabelle 1 zeigt eine Gegenüberstellung der möglichen Datentypen in der jeweiligen Version.

SMIv1	**SMIv2**
INTEGER	INTEGER
OCTET STRING	OCTET STRING

OBJECT IDENTIFIER	OBJECT IDENTIFIER
INTEGER	Integer32
-	Unsigned32
Gauge	Gauge32
Counter	Counter32
-	Counter64
TimeTicks	TimeTicks
IpAddress	IpAddress
Opaque	Opaque
-	BITS
NetworkAddress	-

Tabelle 1: SNMP Datentypen

Diese Arbeit konzentriert sich auf die neuere Version 2, da diese sehr nützliche Datentypen, wie beispielsweise Counter64, unterstützt.

2.1.1.1 Datentypen

Es gibt eine Vielzahl an Datentypen, die mit SNMP abgebildet werden können. Ein kurzer Abriß über die SMIv2 Typen soll Überblick verschaffen. Die Typen in Großbuchstaben sind jene, die auch in ASN.1 (Abstract Syntax Notation) [RFC 1024] verfügbar sind. Alle weiteren sind SNMP-spezifische Typen.

- **INTEGER:** Integer wurde in SMIv2 durch Integer32 und Integer64 in seiner eigentlichen Funktion ersetzt und dient unter SMIv2 nur mehr als Aufzähltyp. Man kann dadurch zum Beispiel verschiedene Zustände realisieren, wie 0 für IDLE, 1 für STOP, 2 für START, ...
- **OCTET STRING:** Bei einem OCTET STRING handelt es sich um einen Bytestring mit maximalen Länge von 65635 Zeichen. Dieser Typ wird in SNMP sehr häufig für den Transport von (für Menschen lesbare) Zeichenketten verwendet. Als aussagekräftigere Abwandlung dieses Typs gibt es beispielsweise den abgeleiteten Typ DisplayString.

- **OBJECT IDENTIFIER:** Es gibt diesen Typ um auch die bereits erwähnten OIDs als solche über SNMP versenden zu können. Somit kann man Positionen im MIB Baum exakt versenden bzw. zwischen Managed System austauschen.
- **Integer32:** Hierbei handelt es sich um einen, vorzeichenbehafteten, 32 Bit großen Ganzzahltyp.
- **Unsigned32:** Analog zu Integer32 gibt es auch einen vorzeichenlosen 32 Bit Ganzzahltypen.
- **Gauge32:** Dieser Typ ist ebenso 32 Bit groß und vorzeichenlos. Gauge32 wird verwendet, um beispielsweise Temperaturanzeigen abzubilden.
- **Counter32:** Bei Counter32 handelt es sich um einen vorzeichenlosen 32 Bit Typ, der als Zähler verwendet wird.
- **Counter64:** Dieser Datentyp ist ein vorzeichenloser 64 Bit Typ für große Zahlenbereiche. (zum Beispiel zur Darstellung der Übertragenen Bytes bei sehr schnellen Datenleitungen)
- **Timeticks:** Variablen von diesem Typ definieren, ein 100-stel Sekunden genau, eine Zeitangabe. Dieser Typ wird normalerweise zum Messen von Zeitintervallen benutzt. Das bekannteste Beispiel im SNMP Bereich ist sicherlich die sysUpTime, welche angibt, wie lange ein System bereits läuft.
- **IpAddress:** Wird vor allem verwendet, um IPv4 Adressen von Interfaces zu konfigurieren und anzuzeigen. In SMIv2 gibt es keinen Datentyp für IPv6. Man kann diese zwar mit Hilfe des Typs OCTET STRING definieren, jedoch wird in einer Folgeversion von SMI ein weiterer Typ hinzugefügt werden. Der Nachfolger von SMIng (next generation) ist bereits als RFC 3780 vorgestellt worden und beinhaltet auch einen Datentyp für IPv6 Adressen.
- **Opaque:** Mit Opaque kann man eigene Typen realisieren, indem man diese auf Byteebene definiert. (Das Ergebnis ist intern wieder ein OCTET STRING.)
- **BITS:** Bei diesem Typ wird jedem Bit ein Name zugewiesen, was den Vorteil hat, dass man die einzelnen Bits direkt ansprechen kann. Die Länge ergibt sich aufgrund der belegten und verwendeten Bits, wobei man immer auf ganze Bytes aufrundet.

2.2 SNMP Version 1

SNMP verfügt über wenige, jedoch ausreichend viele Befehle, die es erlauben, sowohl den Status eines Managed System (auch Agent genannt) abzufragen (GET-Operation) als auch zu setzen (SET-Operation). SNMPv1 unterstützt das Aussenden von Traps. Unter einem Trap versteht man die Möglichkeit der Benachrichtigung einer Managementstation durch einen Agenten. Sinn eines Traps ist es, auftretende Ereignisse bekannt zu geben, ohne dass die Managementstation zuvor genau diesen Wert abfragen musste. Es steht damit also auch die Möglichkeit der aktiven Benachrichtigung zur Verfügung. Man muss sich aber bewußt sein, dass Traps wie auch alle anderen SNMP Pakete per Standard über das User Datagram Protocol (UDP) versandt werden und so die tatsächliche Ankunft von Pakten beim Empfänger nicht garantiert werden kann. Besonderes Augenmerk gilt dabei den Traps, da hier nicht bemerkt wird, wenn ein Trap durch Übertragungsfehler, etc. verloren geht. Eine weiterer Befehl ermöglicht das Zurückgeben, des jeweils nächstgrößeren Wertes für eine Angegebene OID, die in der Literatur oft als vierte SNMP Grundoperation angegeben wird, GET-NEXT.

2.2.1 GET-Operation

Abbildung 5: GET-Operation, adaptiert von Quelle [Per97, S. 6]

Mit Hilfe der GET-Operation kann ein Manager Informationen vom Agenten abfragen. Eine Standardanwendung ist zum Beispiel das Auslesen der Systembeschreibung eines Netzelements. Bei Verwendung des SNMP Protokolls wird immer von so genannten „Variable Bindings“ gesprochen. Darunter versteht man einen an eine Variable gekoppelten Wert. Bei SNMP wird der Wert konkret an eine OID gebunden. Eine typische GET-Operation würde also wie folgt aussehen. Die Managementstation sendet einen Get-Befehl wie zum Beispiel eine GET-PDU `get(sysDescr.0, <>)` an den Agenten und bekommt als Antwort einen an sysDescr.0 gebundenen Wert zurück, zum Beispiel per `get-response(sysDescr.0, <„RedHat Linux 2.6-192smp“>)`, als GET-RESPONSE-PDU.

Wird ein angefordertes Variable Binding nicht gefunden, weil beispielsweise die OID nicht exisitert, liefert der Agent einen „noSuchName“ Fehler zurück.
Damit ein Wert lesbar für das Managementsystem ist, muss dieser zumindest den Zugriffsmodus „read-only“ aufweisen und zusätzlich muss der im Paket angegebene Community String mit dem im Agentensystem gespeicherten Read-Community String übereinstimmen.

2.2.2 GET-NEXT-Operation

In SNMPv1 gibt es die Möglichkeit durch eine bestehende MIB durchzuwandern. Der Befehl der dies ermöglicht ist GET-NEXT. Vom Manager wird eine OID per GET-NEXT-PDU an den Agenten gesandt. Zürckgegeben wird aber das Variable Binding mit der nächsthöheren OID. Dadurch ist es möglich mittels „SNMPWALK“ die Datenbasis sequentiell zu „durchwandern“.

2.2.3 SET-Operation

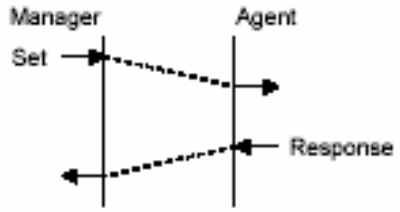

Abbildung 6: SET-Operation, adaptiert von Quelle [Per97, S. 6]

Mit der SET-Operation kann ein Manager definierten MIB-Variablen, Werte zuweisen. Wesentlich ist dabei, dass man dadurch nicht nur Werte ändert, sondern oftmals komplexe Aktionen beim Agenten anstößt, die wiederum eingesetzt werden um Systeme zu konfigurieren. Die Steuerung von Systemen kann sehr weit getrieben werden, beispielsweise kann ein System per SNMP ausgeschaltet werden. Leider ist es aufgrund der mangelnden Sicherheit durch die verwendeten Community Strings nicht empfehlenswert, Steuerungsfunktionen mit SNMPv1 zuzulassen. Als Beispiel kann man sich, um wieder auf den Wert sysDescr.0 zurückzukommen vorstellen, dass man den String der ein System beschreibt, ändern möchte. Der Manager sendet dazu den Set-Befehl mit dem jeweiligen Variable Binding aus, beispielsweise `set(sysDescr.0, <„RedHat Linux 2.6-208smp“>)`, als SET-PDU. Als Antwort bekommt der Manager vom Agenten eine SET-RESOPNSE-PDU zurück `set-resopnse(sysDescr.0, <„RedHat Linux 2.6-208smp“>).`

SET-Operationen sind atomar, was sich beim Bau eines Agenten auswirkt, da beispielsweise zu lange Werte nicht gesetz werden. SNMP definiert im Standard einige Errorcodes, die dazu verwendet werden, zu erkennen, ob ein Wert vorhanden ist und ob dieser auch tatsächlich geschrieben werden kann.
Es gibt zwei Bedingungen, die erfüllt sein müssen, um einen Wert setzen zu können:

1. Die Set-Community muss übereinstimmen.
2. Der Wert muss den Zugriffsmodus „write-only“ bzw. „read-write“ unterstützen.

2.2.4 TRAP-Operation

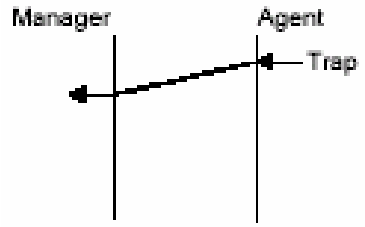

Abbildung 7: Trap-Operation, adaptiert von Quelle [Per97, S. 6]

Da SNMP Systeme überwacht, muss eine Möglichkeit bestehen, wie diese Systeme die Managementstation über Ereignisse informieren können. Dies soll asynchron geschehen, also ohne, dass man den Agenten pausenlos nach den enstprechenden Werten pollen muss. Dies ist bis zu einem gewissen Grad mit einem Interrupt vergleichbar. Traps bieten genau diese Funktionalität. Falls am Agenten ein solch wichtiges Ereignis auftritt, so wird ein zugehöriger Trap an zuvor festgelegte IP-Adressen der Managementstationen gesandt. Die Managmentstationen ihrereseits lauschen (standardmäßig auf Port 162) auf das Eintreffen von solchen Benachrichtigungen. In einem Trap wird sinnvollerweise gleich mitgegeben, welche Werte dieses Ereignis verursacht haben. In diesem Zusammenhang spricht man bei SNMP von so genannten „VarBindLists“. Die Managementstation muss nun wiederum für die Darstellung bzw. Weiterverarbeitung des Traps sorgen.
Vorsicht ist geboten, da Traps üblicherweise wichtige Ereignisse anzeigen, dafür jedoch kein Acknowledge und keine Wiederholungen von verlorengegagenen Daten vorgesehen sind. (Bei SNMPv2 hat man versucht dieses Problem in den Griff zu bekommen.)
Man unterscheidet bei Traps zwischen „generic traps“ und „specific traps“. Ein „generic trap“ definiert eines von wenigen außerordentlichen Ereignissen. Im Speziellen exisiteren die „generic

traps“: coldstart, warmstart, linkdown, linkup, authenticationFailure, egpNeighborLoss, enterpriseSpecific.
Falls der „generic trap“ den Wert enterpriseSpecific beinhaltet, wird im Trapfeld „specific trap“ die genaue OID des Traps angegeben. Weiteres besitzt ein Trap einen Zeitstempel, welcher der SysUpTime des Agenten entnommen wird und angibt, wann der Trap aufgetreten ist. Abschließend wird noch die Liste von beteiligten Variablen als VarBindList angehängt.

2.2.5 Probleme von SNMPv1

Aufgrund der raschen Verbreitung von SNMP kamen auch die Probleme rasch zum Vorschein. Da normalerweise die Daten durch Polling gesammelt werden müssen, steigt der Netzverkehr bei großen Systemen rasch an. Durch das zentrale Managementkonzept muss die zentrale Managementstation viele Werte aller Geräte jeweils einzeln abfragen. Auf Sicherheitsmaßnahmen wurde, da SNMP anfänglich nur als Übergangslösung gedacht war, fast vollständig verzichtet (Communit Strings können bei besonders netter Auslegung als Passwörter gesehen werden). Speziell lauern folgende Gefahren auf den Benutzer bei Einsatz von SNMPv1:

- **Maskerade:** Dabei handelt es sich um das Vortäuschen einer falschen Identität (IP-Adresse), um dann Managementfunktionen auszuführen (SET-Operationen!!!). Durch UDP lassen sich auch IP-Absender leicht fälschen.
- **Veränderung der Reihenfolge der Pakete:** SNMP-Daten werden über UDP transportiert. Es können daher ungewünschte Veränderungen der Reihenfolge von Paketen auftreten.
- **Lauschen am Netzwerk:** Durch passives Lauschen (Sniffing) am Netzwerk lässt sich bei SNMPv1 sehr viel über die SNMP Komponenten herausfinden.
- **Community Name:** Die Community Strings, die als Passwörter Verwendung finden, werden in Klartext in jedem Paket mitübertragen. Die Passworte herauszufinden ist also für einen Angreifer eine einfache Aufgabenstellung.

(zusätzliche Details unter [Sta03], [Sch03])

2.3 SNMP Version 2

Um die Probleme von SNMPv1 in den Griff zu bekommen, entwickelte man die Folgeversion SNMPv2. Wie bereits erwähnt, konnten sich die Sicherheitskonzepte von Version 2 nicht durchgesetzen. Das von der Wirtschaft akzeptierte Ergebnis SNMPv2c ist quasi der kleinste gemeinsame Nenner im Sicherheitsbereich, nämlich die „Unsicherheit“ von Version 1 beizubehalten. SNMPv2c bietet trotzdem einige Verbesserungen, die im Folgenden kurz angeführt werden.

2.3.1 Inform-Operation

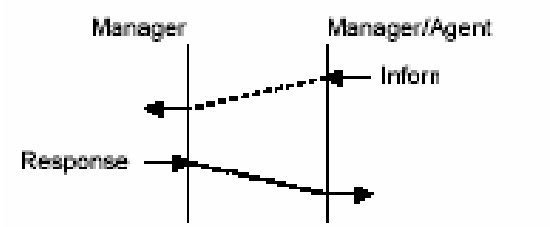

Abbildung 8: Inform-Operation, Anleihen von [Per97, S. 6]

Ursprünglich waren Informs nicht für die Kommunikation zwischen Manager und Agent sondern als Nachrichten zwischen mehreren Managementstationen gedacht (hierarchisches Managementkonzept). Heute werden Informs oftmals an stelle von Traps verwendet. Der Agent sendet einfach anstatt eines Traps eine Inform-PDU. Diese muss vom Manager mit einer Response-PDU bestätigt werden. Falls dies nicht geschieht wird die Informnachricht nochmals vom Agenten gesandt. Informs verstehen sich hier als verbesserte Traps, da durch das Acknowledge und durch die Wiederholungen Informs nicht verloren gehen können.

2.3.2 GET-BULK-Operation

Die GET-BULK-Operation wird verwendet, um mehrere Variable Bindings (also mehrere SNMP Werte) mit einem einzigen Befehl anzufordern. Der Vorteil ist, dass vor allem das Netzwerk viel weniger belastet wird. Der Agent muss nicht so viele Pakete versenden, da er mehrere Werte in einem Paket und somit in einer Transaktion zurücksenden kann. Zusätzlich entfällt die Round-Trip-Time die sonst bei jedem GET/GET-RESOPNSE auftritt.

2.3.3 SNMPv2 Trap

Traps haben sich in ihrer Funktionalität zu den SNMPv1-Vorgängern nicht verändert, jedoch wurde das Kodierungsschema der anderen SNMP-PDU übernommen. Dadurch lassen sich SNMPv2 Traps leichter aufbauen.

Konvention: Jeder Trap besteht aus drei Teilen. Erstens der sysUpTime.0 Variable, die angibt wie lange ein System bereits läuft. Zweitens aus der snmpTrapOID.0 Variable, welche die OID des Traps angibt und drittens aus einer VarBindList, die mehrere betroffene Variablenwerte mitsendet.

2.4 SNMP Version 3

Das Hauptaugenmerk dieser Arbeit liegt auf dem sicheren SNMPv3 Protokoll. Als Einstieg in die Thematik dienen sehr gut lesbare und umfassende RFCs und Bücher:

- RFC 2271: An Architecture for Describing SNMP Management Framwork
- RFC 2272: Message Processing and Dispatching for the SNMP Protocol
- RFC 2273: SNMPv3 Applications
- RFC 2274: User-based Security Model (USM) for version 3 of the SNMP Protocol
- RFC 2275: View-based Access Control Model (VACM) for the SNMP Protocol
- mitp – Trainingsbuch SUSE Liunx Sicherheit [Sch04]
- Understanding SNMP MIBs [Per97]

SNMPv3 ist kein neuer Standard für SNMP. Die SNMP PDUs wurden direkt von SNMPv2c übernommen. Aus diesem Grund hat sich auch an der grundsätzlichen Funktionalität nichts geändert. SNMPv3 hat erfolgreich einen weiteren Aspekt in die Architektur integriert: **Sicherheit** (Authentifikation, Autorisation, Verschlüsselung).

2.4.1 Das SNMPv3 Modell

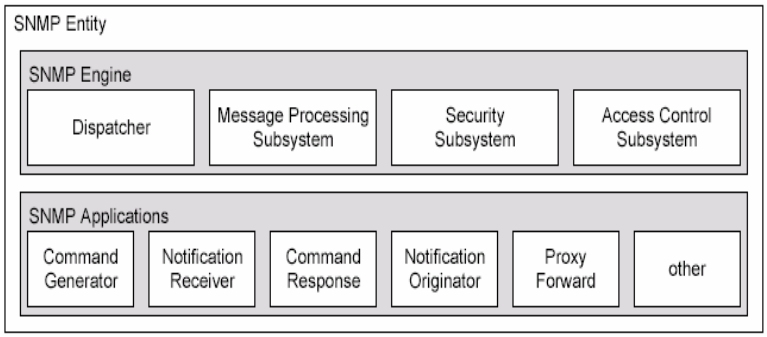

Abbildung 9: SNMP Einheit, Quelle [Sch03, S. 176]

Das zentrale Element von SNMPv3 ist seine Architektur, speziell die SNMP Einheit (SNMP Entity). Oberstes Ziel war, die Elemente, insbesondere die Operationen, PDUs, etc. von SNMPv2c vollständig ohne Änderung zu integrieren und so für Kompatibilität zu sorgen. Es sollte vermieden werden, neue Paketformate, wie schon zuvor bei der Transition von SNMPv1 auf SNMPv2, festzulegen. Natürlich werden die SNMPv3 Pakete mit einem zusätzlichen Header versehen und verschlüsselt übertragen. Der Klartextinhalt entspricht genau den SNMPv2 PDUs. Damit lässt sich in SNMPv3 die SNMPv2 Engine (siehe Abbildung 9), insbesondere Dispatcher

und Message Processing Subsystem weiterzuverwenden. Die anderen beiden Subsysteme (Security und Access Control Subsystem) sind für SNMPv3 zu erweitern.
Jede SNMP Einheit stellt verschiedene, durch Subsysteme getrennte Verarbeitungseinheiten zur Verfügung. Die angeboteten Dienste, die von allen SNMP basierten Anwendungen verwendet werden, werden zu einer SNMP Engine zusammengefasst. Wesentlich für die spätere Programmierung von Agenten ist die Information, dass jede SNMPv3 Engine eine eigene im Agentsystem eindeutige ID besitzt. Über diese sogenannte EngineID können in Folge die Anwendungen mit der Engine kommunizieren. Eine SNMP Engine deckt die folgenden Bereiche ab:

- **Verteilung von Nachrichten** (Dispatcher):
  Vom Verteiler werden die jeweiligen Anfragen und Antworten an die zuständige Anwendung weitergeleitet. Nach der Verteilung erfolgt die Nachrichtenverarbeitung.
- **Verarbeitung von Nachrichten** (Message Processing Subsystem):
  Dieses Teilsystem erstellt SNMP Nachrichten für das jeweilige Modell (aktuell sind SNMPv1, SNMPv2c und SNMPv3 verfügbar) und liest aus SNMP Nachrichten die Informationen zur Weiterverarbeitung heraus.
- **Sicherheitssystem** (Security Subsystem):
  Dem Sicherheitssystem wird ein SNMP Paket (also ein SNMPv2c konformes Paket) übergeben. Das Subsystem übernimmt die Verschlüsselung dieser Nachricht. Weiters wird auch die Authentifizierung und Entschlüsselung von diesem Subsystem erledigt. Die EngineID ist wichtig für das Sicherheitsystem, da die kryptographischen Prüfsummen (Hashing) unter anderem auf der EngineID basieren.
- **Zugriffskontrolle** (Access Control Subsystem):
  Dieses diskrete Subsystem realisiert die Zugriffsrechte, also ob ein Benutzer autorisiert wird, eine gewünschte SNMP-Aktion auszuführen. Die Einstellung der möglichen Parameter wird im Kapitel „SNMP aus Administratorsicht“ praktisch erarbeitet und vorgezeigt.

Abbildung 9 zeigt die Anwendungssubsysteme der SNMP Engine. Alle Anwender/Applikationen werden zur Gruppe der SNMP Applications zusammengefasst. Bisher wurden laut Standard die folgenden Anwendungen definiert:

- **Kommandoerzeuger** (Command Generator):
  Der Kommandoerzeuger initiiert Get-, Get-Next-, Get-Bulk- und Set-Operationen. Weiters werden hier auch die Antworten verarbeitet.
- **Meldungsempfänger** (Notification Receiver):
  Meldungsempfänger sind für alle Benachrichtigungen zuständig, also für den Empfang von allen asynchronen Meldungen.
- **Kommandoempfänger** (Command Response):
  Ein Kommandoempfänger verarbeitet alle Get-, Get-Next-, Get-Bulk- und Set-Operationen und generiert dazu entsprechende Antworten.
- **Meldungserzeuger** (Notification Originator):
  Der Meldungserzeuger generiert, falls ein spezifisches Event im System auftritt, eine Meldung in Form eines Traps (v1,v2) oder Informs. Mittlerweile gibt es auch die ersten Softwarebibliotheken (z.B. Net-SNMP) die SNMPv3 Traps senden können. Später wird dies noch genauer erläutert.
- **Proxy Weiterleitung** (Proxy Forward):
  Die Weiterleitungseinheit leitet Nachrichten, in für den Programmierer transparenter Weise an die SNMP Einheiten weiter.

2.4.1.1 Beispiel einer GET-Operation

An einem Beispiel kann man am besten das soeben erklärte vertiefen. Eine Managementstation stellt über SNMPv3 eine Anfrage (GET-Operation) an das Agentensystem. Die verschlüsselte SNMPv3 GET-PDU kommt bei der Gegenstelle (der SNMP Entity) an. Das Paket wird von dem „Dispatcher“ übernommen. Diese leitet die PDU an das „Message Processing Subsystem“ weiter. Da die Daten der PDU noch verschlüsselt sind, ist der nächste logische Schritt die Entschlüsselung des Pakets. Dazu wird es von dem „Message Processing Subsystem“ an das „Security Subsystem“ weitergeleitet. Die Nachricht wird entschlüsselt und die Daten werden als Klartextpaket wieder an den Dispatcher über das „Message Processing Subsystem“ zurückgesandt. Nun sendet der „Dispatcher“ die vom „Message Processing Subsystem“ aufbereiteten Daten an die zuständige Applikation (Application) „Command Response“. Es wird über das „Access Control Subsystem“ geprüft, ob die Anfrage von einem autorisierten Benutzer gestellt wurde und ein Antwortpaket generiert. Das Antwortpaket geht wieder an den „Dispatcher“, welcher für die Weiterleitung der Daten an das „Message Processing Subsystem“ sorgt. Das Paket wird mit den Credentials des Benutzers verschlüsselt und danach wieder an den „Dispatcher“ retourniert. Das Paket kann nun wieder an den Absender zurückgesendet werden.

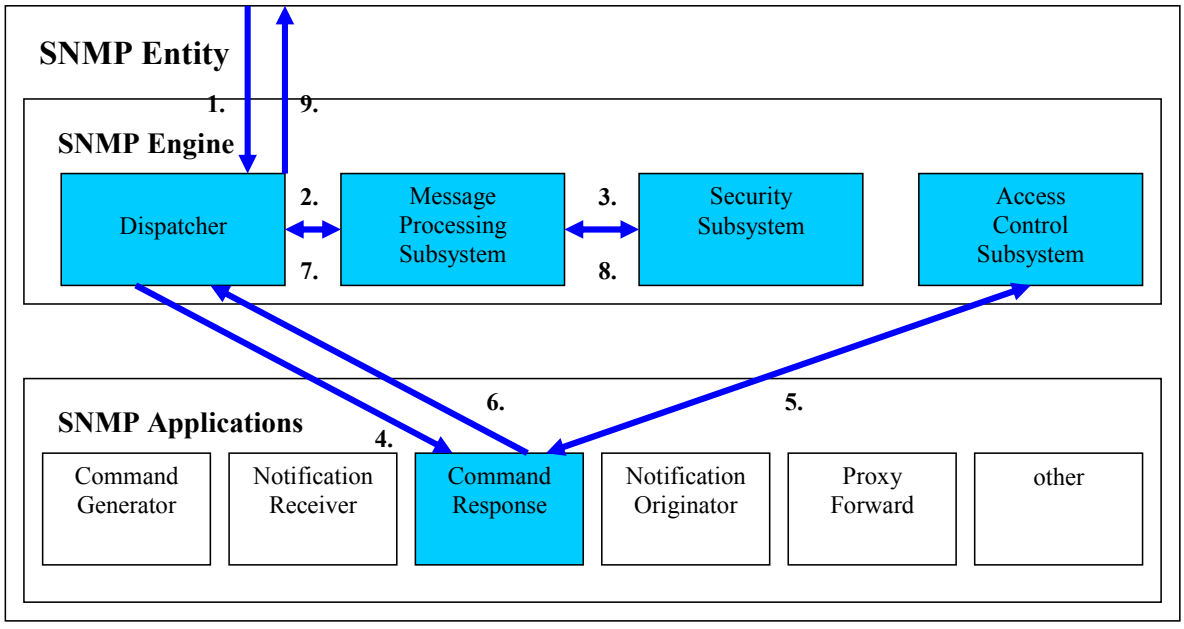

Abbildung 10: GET-Operation einer Managementstation

2.4.1.2 Sicherheitskonzept

SNMPv3 stützt sich auf zwei Säulen:

Einerseits auf das **User-Based Security Model (USM)**, welches die Authentifikation der Benutzer übernimmt und andererseits

auf das **View Access Control Model (VACM)**, welches für die Zugriffskontrolle (Autorisation) zuständig

2.4.1.3 User-based Security Model (USM)

In RFC 2274 werden die wesentlichen Ziele definiert. Allen voran:

- **Aktualität:** SNMPv3 versucht die Probleme in den Griff zu bekommen, die UDP-basierten Nachrichten zugrunde liegen. Es versucht verzögerte, umgeordnete und mehrmals wiedergegebene Nachrichten zu erkennen.
- **Schlüsselverwaltung:** Authentifizierung und Verschlüsselung setzen voraus, dass beide SNMP-Engines die selben Schlüssel verwenden bzw. kennen. Um dieses Ziel zu erreichen müssen geheime Schlüssel sicher über das Netzwerk übertragen und diese dann lokal zu gespeichert werden können.
- **Authentifizierung:** Die Authentifizierung hat zwei wichtige Bedeutungen für SNMPv3. Erstens wird damit über die Benutzerkennung mittels z.B. HMAC-MD5 sichergestellt, dass die sendende SNMP-Engine tatsächlich die ist, für die sie sich ausgibt. Zweitens kann man über die Hashsumme (in wleche die Engine-ID mit-eingeht, siehe oben) sicher-stellen. dass eine Nachricht nicht verändert wurde.
- **Verschlüsselung:** Zusätzlich zur gerade besprochenen Authentifizierung ist es nun möglich, das Paket vor dem Versenden über das Netzwerk zu verschlüsseln. Dazu müssen beide Systeme den Schlüssel („shared secret“) kennen.

2.4.1.4 View Access Control Model (VACM)

In SNMPv3 wird zusätzlich zur Authentifikation ein Zugriffskontrollmodell zur Autorisierung festgelegt. Die Zugriffskontrolle wird immer durchlaufen wenn ein Lese– oder Schreibzugriff auf ein Managed Object erfolgt. Auch die Berechtigung zum Senden von SNMPv3-Traps wird überprüft.

Die Daten, wer auf welche Mangaged Objects wie zugreifen darf, steht in einer eigenen Tabelle der VACM-MIB. Dies Tabelle läßt sich ihrererseits per SNMP administrieren und damit

verändern. So kann man (sofern man auf die in dieser MIB definierten Managed Objects Zugriff hat!!!!) zur Laufzeit neue Benutzer anlegen oder die Rechte verwalten.

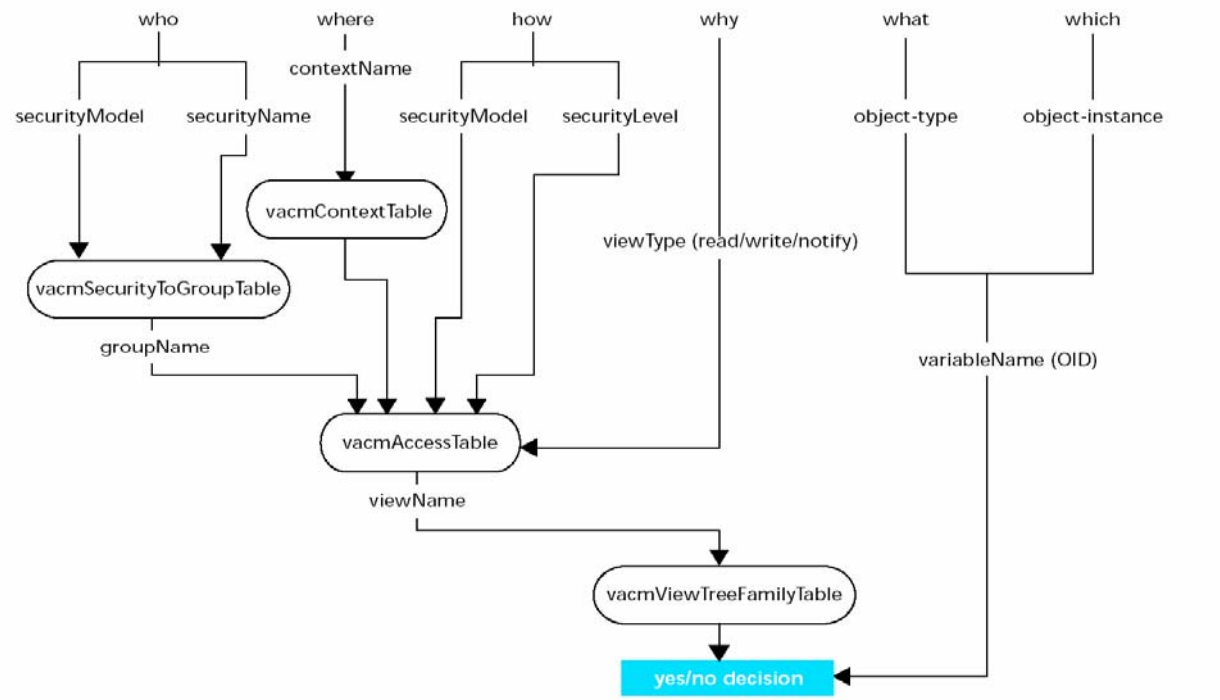

Abbildung 11: VACM Zugriffskontrolle [Com98]

Grundsätzlich ist das VACM Modell recht einfach gehalten. Wesentlich sind die sechs Fragewörter die WHO, WHERE, HOW, WHY, WHAT, WHICH.

- **WHO:** WHO spezifiziert in diesem Modell eindeutig den Benutzer, der versucht zuzugreifen.

  Das „**securityModel**“ gibt an, welches Sicherheitmodell verwendet wird, also Community-basiert (also nur SNMPv1- / SNMPv2c-Sicherheit) oder USM basiert.

  Der „**securityName**“ gibt den Namen des Anwenders an, für den der Zugriff überprüft werden soll.

- **WHERE:** Die Zugriffskontrolle prüft, ob der „**contextName**“ in der „**vacmContextTable**“ enthalten ist. Falls dies nicht der Fall ist, so wird der Fehler noSuchContext zurückgegeben. („contextName“ spezifiert die Instant einer gesamten MIB. Also ist dieser „contextName“ insb. wichtig, wenn mehrere gleiche MIBs von einem Agent instanziert werden.) Bei Net-SNMP kann in der Konfigurationsdatei gewählt werden, ob der Context „exact“, also 1:1 mit dem mitgegebenen übereinstimmen muss, oder nur der Anfangsteil übereinstimmen muss. So wird eine

Schachtelung von „Context“ erlaubt. (Net-SNMP steuert dies durch Angabe des Schlüsselwortes „exact“.)

- **HOW:** „HOW“ legt fest, wie ein Benutzer authentifiziert werden muss und ob die Verschlüsselung ein oder ausgeschaltet ist. Hier wird geprüft, ob die „**vacmAccessTable**“ Tabelle einen Eintrag für den „groupName“, „contextName“, „securityModel“ und „**securityLevel**“ (auth = nur mit Authentikation des Benutzers, XXXX) enthält. Falls kein Eintrag gefunden wurde so wird ein noAccessEntry Fehler zurückgegeben.
- **WHY:** „WHY“ legt fest, welche Rechte der User „WHO“ hat.
  Der hier definierte viewType (siehe Konfigurationsdatei snmpd.conf bei Net-SNMP) wird in dem Eintrag, der in der viewAccessTable definiert wurde, gesucht. Wenn dieser gefunden wird dann resultiert daraus die Rückgabe des „**viewName**“, andernfalls wird der Fehler noSuchView zurückgegeben.
  „viewName“ zeigt also das Subset der MIB an, auf welche der User „WHO“ mit Konext „WHERE“ und Authentifikation/Verschlüsselung „HOW“ mit Zugriffsmodus „WHY“ zugreifen darf.
- **WHAT + WHICH:** Diese beiden Werte konkateniert geben die OID „**variableName (OID)**“ an, auf deren Wert zugegriffen werden soll. Die Teilung in „**object-type**“ (OID-Präfix, z.B. sysDescr) und Objektinstanz „**object-instance**“ (für „normale“ Skalare immer 0) wird gemacht, um ganze Unterzweige spezifizieren zu können (vergleiche auch Get-Bulk-Operationen).

Nun muss nur noch geprüft werden, ob die OID „variableName (OID)“ in der Liste der erlaubten OIDs, die der Table „**vacmViewTreeFamilyTable**“ generiert, enthalten ist. Wenn ja, so ist der Zugriff erlaubt.

Anmerkung: Die einzelnen Daten müssen am Agentensystem (Managed System) abgespeichert werden und am Managementsystem konfiguriert werden. Weitere Informationen zur praktischen Anwendung dieses Konzeptes findet man unter dem Kapitel zur Administration.

3 Installation

Inter oves locum praesta, *Give me a favoured place among the sheep,*
et ab hoedis me sequestra. *and separate me from the goats.*
- Guiseppe Verdi, Requiem Mass, Part II (Dies Irae)

Dieses Kapitel setzt voraus, dass in der Umgebung des Lesers die Basisfunktionalität des Net-SNMP Masteragenten zum Einsatz kommt. Dieser Agent bietet die Möglichkeit Systemdaten, Interfacekonfigurationen, und vieles nach außen abzubilden. Für die bloße Installation des Subagenten würde es ausreichen, einen minimal konfigurierten Masteragenten zu verwenden, der im Grunde nicht mehr machen muss, als die Verwaltung der Subagenten (die Indexallokierung, das Verteilen der Pakete) und die Konvertierung der SNMP-PDUs (je nach Version) in die äquivalenten AgentX-PDUs. Der Masteragent muss natürlich auch vice versa die Antworten der AgentX-Subagenten in SNMP konforme Antwortpakete transformieren können.
Im Folgenden wird die benötigte Software und die Installation aus Administratorsicht beschrieben. In dieser Beschreibung wird bestehendes Linux Basiswissen, vor allem der Umgang mit einer Shell, vorausgesetzt. Das Auflösen von Softwareabhängigkeiten, die je nach System auftreten können, wird nicht genauer behandelt. Die OpenSSL Installation wir auf Basis des „RedHat Package Managers (rpm)“ erläutert. Dieser erlaubt sehr einfaches Installieren von Paketen..

3.1 Benötigte Software

Für die Inbetriebnahme und volle Funktionalität der EasySnmp Bibliothek benötigt man eine kleine Anzahl an wichtigen Komponenten die heute mit Standard – Linux - Distributionen, wie SUSE oder RedHat bereits mit ausgeliefert werden.
Die benötigten Komponenten sind:

1. OpenSSL (in aktueller Version)
2. net-snmp (in aktueller Version)
3. EasySnmp (Installation der „easysnmp.so“ Datei im Linuxsystem)

Man kann unter Linux sehr einfach feststellen, ob ein Paket bereits installiert ist oder nicht. Bei Verwendung des RedHat Packet Managements funktioniert dies auf der Kommandozeile wie folgt:

```
daniel@fab001:/> rpm qa | grep openssl
daniel@fab001:/> rpm qa | grep netsnmp
```

Als Ergebnis der beiden Kommandos sollte eine Liste der jeweils installierten Pakete zurückgegeben werden. Beispielsweise für OpenSSL folgendes:

```
daniel@fab001:/> rpm -qa | grep openssl
openssl-0.9.7d-15.13
openssl-devel-0.9.7d-15
openssl-doc-0.9.7d-15
```

Für Net-SNMP wird auf jeden Fall die Installation der aktuellsten stabilen Version empfohlen. Die aktuelle Version findet man auf der Net-SNMP Webseite. Hier sollte man sich einen aktuellen Tarball (so nennt man *.tar.gz Dateien, da die wie ein Ball alle Dateien in einer gepackten Datei enthalten) herunterladen und diesen in z.B. das Verzeichnis /usr/local/src/ entpacken.
Die gleiche Vorgehensweise kann man nun auch für das Net-SNMP Paket wählen. Da sich vor allem bei diesem Open Source Projekt jeden Tag sehr viel am Quellcode ändert, ist es nicht sinnvoll ein Standardpaket zu installieren, sondern jeweils die aktuelle vom Quellcode Version direkt zu kompilieren. Die Vorgehensweise wird im folgenden Teil noch genauer erläutert.

.

3.2 Installation der einzelnen Softwarekomponenten

Da der hier vorliegende Agent höchstmögliche Sicherheitsansprüche erfüllen soll, ist es sinnvoll, die Authentifizierung mittels MD5 und die Verschlüsselung mit DES zu lösen. Diese Art der Authentifizierung und Verschlüsselung setzt das OpenSSL Paket voraus. Nebenbei sei erwähnt, dass es aktuell keinen Sinn macht, stärkere Verschlüsselungsalgorithmen einzusetzen, da die Performance sehr stark darunter leidet. Dies lässt sich auf die Tatsache zurückführen, dass der ausgewählte Algorithmus für jedes einzelne Paket angewandt werden muss. Dabei sei nochmals festgehalten, dass bei SNMPv3 die einzelnen Pakete verschlüsselt werden, jedoch kein sicherer Verbindungskanal (z.B. im Sinne des IPSec-VPN-Tunnels) aufgebaut wird.
Für OpenSSL ist es vernünftig, ein bestehendes natives binär RPM-Paket zu installieren. Man könnte dazu bei gängigen Distributionen so vorgehen:

```
fab001:/tmp # rpm -Uhv openssl-32bit-9.1-200408050331.i586.rpm
Preparing...                ########################################### [100%]
1:openssl-32bit             ########################################### [100%]
fab001:/tmp #
```

Das aktuelle OpenSSL Paket befindet sich nun auf dem Rechner.

Nun beginnt die Installation von Net-SNMP. Um die Binaries und die zugehörigen Header Dateien zu installieren, muss zuvor überlegt werden, welche Daten das Masteragent nach außen liefern soll. Zum Betreiben von gezieltem Systemmanagement sind diese Überlegungen sehr wichtig, da es gilt, alle interessanten, jedoch nur die wirklich wertvollen Standard MIBs, des Systems, nach außen abzubilden..

Wie oben erläutert, wird an dieser Stelle davon ausgegangen, dass der Tarball bereits heruntergeladen wurde und dieser nun in das Verzeichnis /usr/local/src/net-snmp-x.y.z/ entpackt wird. Die Platzhalter 'x', 'y', 'z' stehen dabei für die jeweilige Version des Net-SNMP Quellcodes.

```
# entpacken
fab001:/usr/local/src # tar -xvzf /tmp/net-snmp/net-snmp-5.2.rc3.tar.gz

# Vorkonfiguration des zu installierenden Packetes mittels Optionen
fab001:/usr/local/src # ./configure --with-mib-modules=\
                                      „host disman/event-mib“\
                                    --prefix=/usr/local
...
# Konfiguration durch Beantwortung von Fragen
...
Default version of SNMP to use (3): 3
...
System Contact Information (root@): root@fabalabs.org setting System Contact
...
System Location (Unknown): Linz
...
Location to write logfile (/var/log/snmpd.log): /var/log/snmpd.log
...
Location to write persistent information (/var/net-snmp): /var/net-snmp
...
...
-----------------------------------------------------
            Net-SNMP configuration summary:
-----------------------------------------------------

  SNMP Versions Supported:    1 2c 3
  Net-SNMP Version:           5.2.rc3
  Building for:               linux
  Network transport support:  Callback Unix TCP UDP
  SNMPv3 Security Modules:    usm
  Agent MIB code:             mibII ucd_snmp snmpv3mibs notification target
                              agent_mibs agentx utilities host
                              disman/event-mib
  SNMP Perl modules:          disabled
  Embedded perl support:      disabled
  Authentication support:     MD5 SHA1
  Encryption support:         DES AES

-----------------------------------------------------
```

Wie man obigem Snippet entnehmen kann, ist es möglich durch weitere Aspekte die Konfiguration zu beeinflussen. Alle Möglichkeiten kann man sich mittels `./configure --help` ausgeben lassen. Für die Standardüberwachung sollten jedoch, die hier angeführten Eingaben ausreichend sein.

Am Ende werden von `configure` noch einige Werte abgefragt:

Default version of SNMP: Legt die SNMP-Version fest, die für die enthaltenen Kommandos (snmpget, snmpset, snmpwalk, ...) verwendet werden soll, wenn nicht explizit eine Version angegeben wird. Vorgabe ist Version 3.

System Contatct: Definiert die E-Mail Adresse des Administrators.

System Location: Gibt den Standort des Systems (Raum, Gebäude, etc.) an.

Logfile location: Position der Log-Datei. Die Standardvorgabe `/var/log/snmpd.log` hält sich an Linuxkonventionen und sollte nach Möglichkeit beibehalten werden.

snmpd persistent information: Hier speichert der Agent die Daten, die zur Laufzeit verändert wurden und auch den nächsten Neustart des Deamons überdauern sollen.

Es ist auch möglich diese Daten direkt als Parameter beim Aufruf von `configure` mit anzugeben. Das Skript fragt dann nur nach den Parametern die nicht bereits beim Aufruf mit übergeben wurden. Dies ist wesentlich, wenn man ohne zutun automatisiert Net-SNMP auf einem System einspielen muss. Die Optionen für die Aufrufparameter lauten wie folgt:

- **System Contact:** --with-sys-contact=„root@fabalabs.org“
- **System Location:** --with-sys-location=„linz“
- **Logfile location:** --with-logfile=„/var/log/snmpd.log“
- **snmpd persistent information:** --with-persistent-directory=„/var/net-snmp“

Net-SNMP ist jetzt richtig konfiguriert und muss genau den angegebenen Kriterien entsprechend kompiliert und installiert werden. Dafür werden zwei weitere Befehle benötigt:

```
fab001:/usr/local/src # make
fab001:/usr/local/src # make install
```

Die letzte Einstellung betrifft alle symbolischen Namen, die in den MIBS definiert werden. Um diese auch mit ihrem Namen benützen zu können, kann man entweder der Umgebungsvariable MIBS den Wert ALL zuweisen oder aber in der Konfigurationsdatei snmp.conf den entsprechenden Eintrag (mibs) vornehmen.

Auf dem System existieren danach einige ausführbare Dateien und ein paar so genannte Shared Objects. Diese Shared Objects bindet EasySnmp dynamisch ein, damit die bestehende SNMP Funktionalität weiterverwendet werden kann.
Von den ausführbaren Dateien ist für diese Arbeit „snmpd“ die wichtigste. Diese ist unter /usr/local/sbin/ zu finden. „snmpd“ repräsentiert den SNMP Master Deamon. Dieser ist die zentrale Stelle, an die alle Anfragen der SNMP Managementsysteme auf Port 161 gerichtet werden. Der Subagent verbindet sich zu seinem Masteragenten und unterstützt diesen bei der Arbeit.
Nachdem nun sowohl OpenSSL als auch Net-SNMP installiert wurde, fehlt noch EasySnmp. Hierbei handelt es sich um die entstandene Bibliothek EasySnmp die auf Net-SNMP basiert. Für die Verwendung der Bibliothek muss das zugehörige Shared Object (libeasysnmp.so.1.0) im Verzeichnis /usr/local/lib/ abgelegt werden und die Header Dateien in das Verzeichnis /usr/local/include/EasySnmp kopiert werden.
Danach wird mittels „ldconfig“ der Registrierungsvorgang für das neue Shared Object gestartet. „ldconfig“ kümmert sich um die Erstellung einer Referenz, als auch um die Aktualisierung der /etc/ld.so.conf Datei.
Im Folgenden sollen die Befehle gezeigt werden, die nötig sind, um das Shared Object zu installieren.

```
fab001:/usr/local/src # cp /tmp/libeasysnmp.so.1.0 .
fab001:/usr/local/src # chmod 755 libeasysnmp.so.1.0
fab001:/usr/local/lib # ls
.
..
libeasysnmp.so.1.0
libnetsnmpagent.so
libnetsnmpagent.so.5
libnetsnmpagent.so.5.1.2
...
fab001:/usr/local/lib # ldconfig -v -n .
.:
        libeasysnmp.so.1 -> libeasysnmp.so.1.0 (changed)
        libsvncpp.so.0 -> libsvncpp.so.0.0.0
        libnetsnmphelpers.so.5 -> libnetsnmphelpers.so.5.2.0
        libnetsnmptrapd.so.5 -> libnetsnmptrapd.so.5.2.0
        libnetsnmpagent.so.5 -> libnetsnmpagent.so.5.2.0
        libnetsnmp.so.5 -> libnetsnmp.so.5.2.0
        libnetsnmpmibs.so.5 -> libnetsnmpmibs.so.5.2.0

fab001:/usr/local/lib # ls
.
..
libeasysnmp.so.1
libeasysnmp.so.1.0
libnetsnmpagent.so
```

```
libnetsnmpagent.so.5
libnetsnmpagent.so.5.2.0
...
fab001:/usr/local/src # ln -sf libeasysnmp.so.1 libeasysnmp.so
```

Nach diesen Schritten, kann das „libeasysnmp.so“ Shared Object nun verwendet werden.

Als abschließenden Schritt werden noch die Header Dateien kopiert

- SnmpOid.h
- ScalarManager.h
- NotificationManager.h
- TableLayout.h
- TableManager.h
- CalledBackObject.h

Die Dateien werden in das Verzeichnis /usr/local/include/EasySnmp abgelegt.

4 SNMP aus Administrator Sicht

Three can keep a secret if two are dead.
— Ben Franklin, Poor Richard's Almanac

In diesem Kapitel werden drei Beispielkonfigurationen gezeigt. Da EasySnmp, und der darauf basierende Beispielagent, vollständig unter Linux entwickelt wurde, ist es sinnvoll zu zeigen, dass es auch problemlos mit Microsoft Windows basierten System Managern möglich ist, auf den Beispiel-Agenten zuzugreifen. Dazu kann man ein Produkt der Firma MG-SOFT verwenden, da die Managementsoftware „MIB Browser Professional“ speziell für SNMP geschrieben wurde und sogar Version 3 des Protokolls unterstützt.
Der eingesetzte Masteragent unterstützt die Version 1, 2c und 3. An zwei Beispielkonfigurationen wird im Folgenden erläutert wie die SNMP Version 2c mit Community-Strings und die Version 3 mittels USM und VACM konfiguriert wird.

4.1 Konfiguration der Clientseite

4.1.1 Konfigurationsbeispiel für SNMPv2c

Die Konfiguration eines SNMPv2c Benutzers auf Cient (System Manager) Seite ist trivial. Ein SNMPv2c Benutzer hat wie ein SNMPv1 Benutzer Zugriff auf den MIB Tree durch die Community beschränkt. Ein Benutzer muss, damit er lesenden Zugriff bekommt, die „Read community“ kennen und damit dieser schreibenden Zugriff bekommt die „Write community“. Der Vorteil zu SNMPv1 ist aus MG-Soft Sicht nur, dass nun auch die Option „Get-Bulk“ verfügbar ist. Get-Bulk, wird, wie bereits erwähnt, für die Übertragung von Anfrageblöcken und Antwortblöcken verwendet. Damit lässt sich die Last am Netz deutlich reduzieren.
Nun zur praktischen Konfiguration. Der folgende Screenshot zeigt den MIB Browser Professional. Durch Auswahl des Menüeintrags „SNMP Protocol Preferences“ im Menü „View“ kommt man auf ein weiteres Eingabeformular.

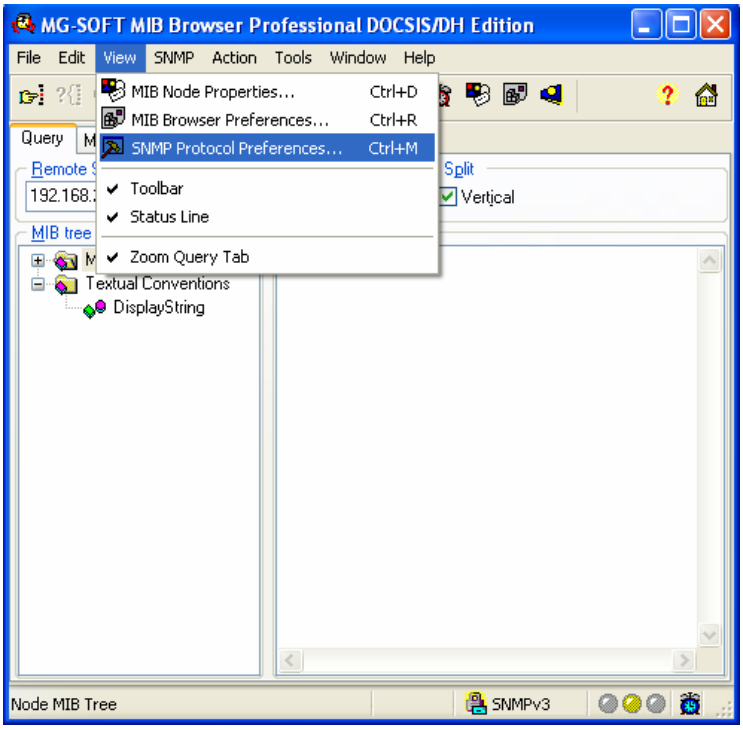

Abbildung 12: MG-SOFT MIB Browser Professional, SNMPv2

Im Eingabeformular „SNMP Protocol Preferences“ wird die gewünschte SNMP Protokoll Version ausgewählt, in dieser Konfiguration also SNMPv2c.

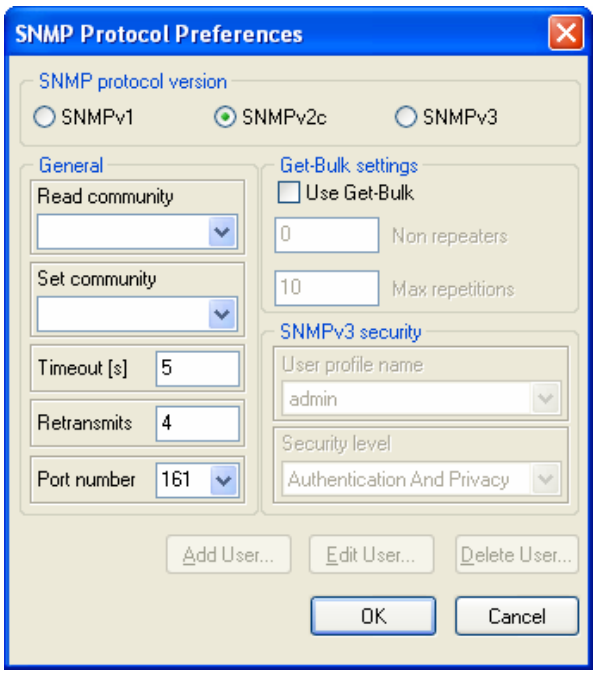

Abbildung 13: SNMP Protokoll Einstellungen – SNMPv2 „Community“

Um die Konfiguration abzuschließen, muss man noch eine „Read community“ als auch eine „Set community“ eintragen. Diese kann nun nicht beliebig gewählt werden, sondern muss für den jeweiligen SNMP Agenten bereits konfiguriert sein. Normalerweise wird hier als Read community „public“ und als Set community „private“ gesetzt. Man sollte aber zumindest die Set community umbenennen. Tatsächliche Sicherheit kann aber mit SNMPv2c ohnehin nicht erreicht werden, dies wurde bereits diskutiert. Zusätzlich aktiviert man die Option „Use Get-Bulk“, damit es möglich ist mehrere SNMP Werte in einem einzigen Paket anzufordern und zu empfangen. Der Wert „Max repetitions“ gibt dabei an wie viele Werte pro Nachricht standardmäßig bei einem SNMP Walk angefordert werden.

Abbildung 14: SNMP Protokoll Einstellungen - SNMPv2 „Use Get-Bulk“

Die Porteinstellung ist bereits korrekt und so kann durch Bestätigung mit OK die neue Zugriffsdefinition auf den Agenten verwendet werden.

4.1.2 Konfigurationsbeispiel für SNMPv3

Die Einstellungen für SNMPv3 sind aus Clientsicht nicht wesentlich komplexer. Es werden konkrete Benutzer mit Profilen angelegt, die sich dann zum jeweiligen Agenten verbinden und dort autentifizieren und autorisieren. Es ist möglich zwei verschiedene Passwörter zu benutzen, eines für den Authentifikations- und eines für den Autoristaionsteil.

Das Eingabeformular wird durch erneutes klicken auf den Menüeintrag „SNMP Protocol Preferences“ aufgerufen.

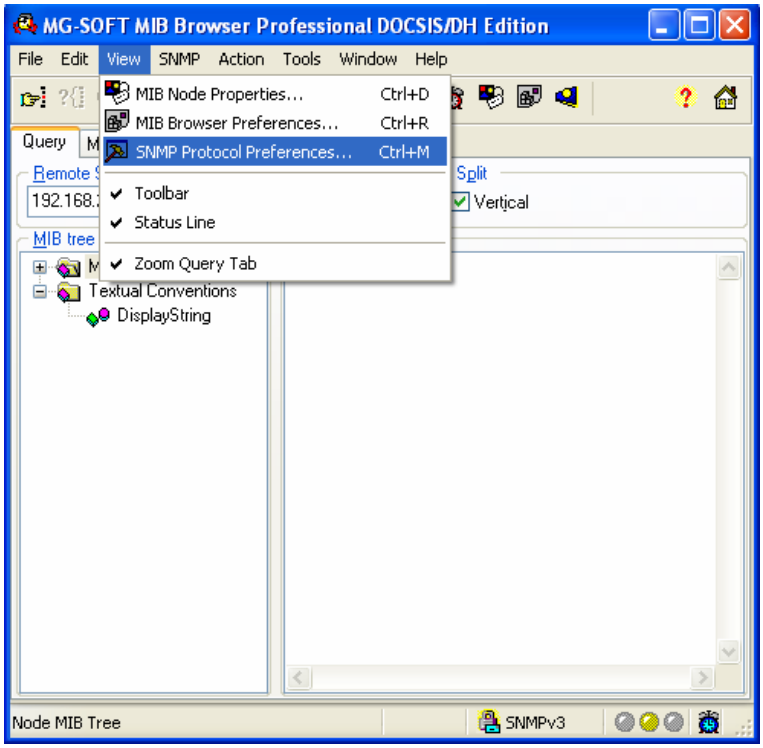

Abbildung 15: MIB Browser Professional, SNMPv3

Nun wählt man im Unterschied zu vorher SNMPv3 aus.

Abbildung 16: SNMPv3, „Edit User...“

Durch klick auf „Edit User...“ kann man einen neuen Benutzer anlegen. In diesem Beispiel wird ein Benutzer angelegt der den „AuthPriv“ Mechanismus unterstützt, also sowohl die Möglichkeit der Autentifikation als auch der Autorisierung nutzt.

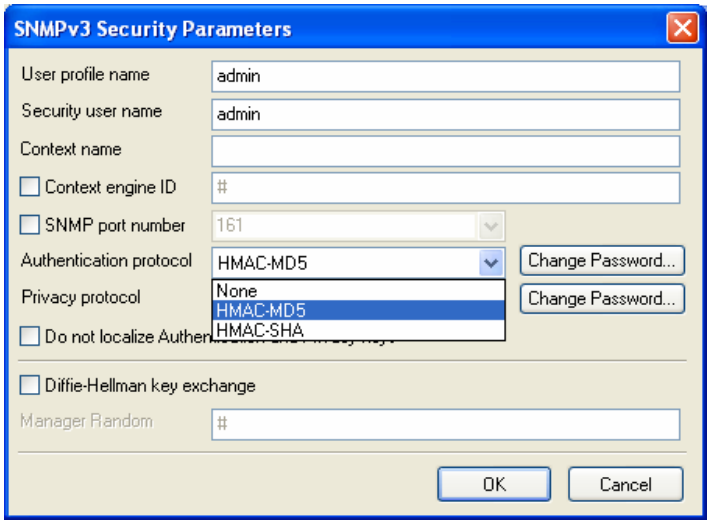

Abbildung 17: SNMPv3, Sicherheitsparameter für USM

Man muss nun einen Namen für das Benutzerprofil (User Profile name) angeben. Dieser wird aber nur von MG-Soft intern verwendet und ist für SNMPv3 irrelevant. Der SNMP

Benutzername wird mit „Security user name“ angegeben. Da in der gesamten Arbeit ohne mehrere Instanzen von MIBs am selben Agenten gearbeitet wird, benötigt man keinen „Context name“. Der vorkonfigurierte Port 161 stimmt mit dem im Standard definierten überein und kann somit beibehalten werden. Um die Konfiguration abzuschließen muss noch eingestellt werden, welche Autentifikationsmethode verwendet wird. Hier wählt man zwischen HMAC-MD5 und HMAC-SHA aus. Beide Einstellungen unterstützt auch die SNMP Gegenseite. Grundsätzlich kann man hier auch keine Verschlüsselungsmethode wählen, was aber nicht zu empfehlen ist, da dann die Daten Klartext über das Netzwerk gesandt werden. Aufgrund von Empfehlungen im Standard wählt man am besten HMAC-MD5 aus.

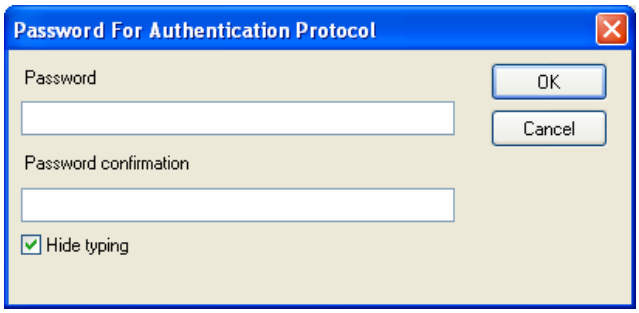

Abbildung 18: SNMPv3, USM-Passworteingabe

Nun klickt man auf „Change password...“ und gibt das Passwort, für die spätere Autentifikation des Benutzers gegen den SNMP Agent, ein. Nach zweimaliger Eingabe und Bestätigung durch OK ist das Passwort gespeichert im MG-Soft System abgelegt.
Selbige Vorgehensweise gilt nun auch für die Konfiguration des Verschlüsselungsalgorithmus, der die einzelnen SNMP Pakete in Folge verschlüsselt.

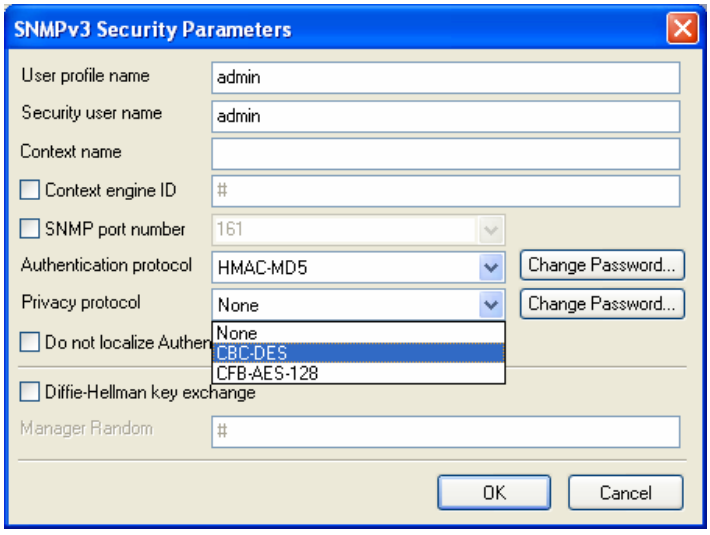

Abbildung 19: SNMPv3, Paketverschlüsselung

Sofern es die Gegenseite gleichermaßen unterstützt, kann hier zwischen CBC-DES und CFB-AS-128 gewählt werden. Nicht empfehlenswert ist es, die Verschlüsselung nicht zu verwenden. Man muss nun wieder, wie schon zuvor, ein Passwort setzen. Die Einstellungen sind nun komplett. Das SNMPv3 Protokoll kann nun verwendet werden.

4.2 Konfiguration der Serverseite

4.2.1 Allgemeines Verständnis

Unter Serverseite versteht man grundsätzlich den SNMP Masteragenten. Da bei der Libraryprogrammierung der bereits beschriebene AgentX Ansatz gewählt wurde und die Subagenten daher keine sicherheitsrelavanten Daten verarbeiten, wurde der Teil der Benutzerverwaltung, Passwortverwaltung als auch die Veraltung der Engine ID beim Masteragenten belassen. Man hat somit den vollen Funktionsumfang der Net-SNMP Bibliothek zur Verfügung, der sich laufend durch Untersützung von Firmen (Cisco, HP, ...) und engagierten Programmierern erweitert.

Der Agent kann, eine bestimmte Konfiguration vorausgesetzt, seine Daten und die der Subagenten nach außen in den drei gänigen SNMP Versionen 1, 2c und 3 zur Verfügung stellen. Diese Tatsache hat aus unserer Sicht den Zweck, dass viele am Markt befindlichen Clientsysteme (Managementsysteme) leider nach wie vor SNMPv3 seten unterstützen und somit kann hier idealerweise SNMPv2c zu Kompatibilitätszwecken zur Anwendung kommen.

4.2.2 Steuerung des Masteragenten

Die Benutzerverwaltung obliegt dem Masteragenten. Der Vorteil dieser Architektur besteht darin, dass an einer zentralen Stelle alle Benutzer, als auch deren Rechte im SNMP System konfiguriert werden können. Es besteht die Möglichkeit, mit Hilfe der Konfigurationsdatei `snmpd.conf` sehr fein granulierte Einstellungen vorzunehmen. Diese Datei befindet sich im Verzeichnis `/etc`. Da in dieser Datei wesentliche Sicherheitsaspekte, wie auch die Passwörter stehen können, muss man darauf achten, dass nur der Benutzer `root` das Recht hat, diese Datei zu lesen und zu schreiben. Man kann dies durch den Befehl `chmod 600 /etc/snmpd.conf` erreichen. In Folge soll kurz auf die hauptsächlich verwendeten Schlüsselwörter eingegangen werden.

view: Wird verwendet um eine MIB-View zu definieren. Dabei können Zugriffe auf Teilbäume der MIB erlaubt (`included`) oder verboten sein (`excluded`). Es ist auch erlaubt, Teile von zuvor erlaubten Bereichen wieder zu verbieten. Man kann also aus einem erlaubten Teilbaum wiederum Subbäume auszuwählen, die man verbietet. Durch diese Vorgehensweise wird eine sehr fein granulierte Rechtsvergabe bis auf Blattebene erreicht.

rocommunity: Mit rocommunity definiert man einen SNMPv1 oder v2c Benutzer und legt gleichzeitig fest, mit welcher Community und von welcher IP aus dieser lesenden Zugriff auf den MIB-Tree erhält. Man sollte sich aber bei Verwendung jedoch bewusst sein, dass ja wie schon erwähnt SNMPv1 als auch v2c sehr einfach abzuhören sind. Zusätzlich bekommt man mit der auf diese Art definierten Community, lesende Zugriff auf den ganzen Baum bzw. einen definierten Subzweig. Eine feinere Granulierung ist hier nicht möglich. Weiters kann man als IP sowohl einen einzelnen Host nennen, z.B. 192.168.200.51, als auch ein ganzes Subnetz definieren, z.B. mit 192.168.200.0/24. Es ist auch möglich, sofern das Netzwerk über einen DNS verfügt, einen Hostnamen anzugeben. Diese werden beim Start des Masteragenten aufgelöst und wieder als statische IPs gebunden. Dieser Tatsache muss man sich bewusst sein, da Änderungen in DNS erst nach einem Neustart des Agenten berücksichtigt werden.

rwcommunity: Dieses Schlüsselwort wird analog zum Schlüsselwort rocommunity verwendet und definiert. Für eine gewisse IP bzw. ein Subnetz legt es jedoch den schreibenden Zugriff für eine Community fest.

com2sec: Erzeugt einen SNMPv1 bzw. SNMPv2c Benutzer. Dabei wird wiederum definiert von welchem IP Netz zugegriffen werden darf, und welche Community benutzt werden muss. Der Vorteil und gleichzeitig auch der Nachteil von com2sec ist, dass es viel flexibler ist. Man kann hier sehr komplexe Zugriffsstrukturen definieren, indem man die Schlüsselwörter view, access und group verwendet. Nichtsdestotrotz hat man nach wie vor die Nachteile der beiden SNMP Versionen.

createUser: Ermöglicht die Erzeugung eines SNMPv3 Benutzers nach dem USM Modell. Dieses Schlüsselwort sollte nur eine sehr kurze Zeitdauer in der `snmpd.conf` Datei verweilen. Es ist dafür gedacht einen Benutzer anzulegen. Damit dies funktioniert muss man folgende Daten vorab wissen:

- Benutzername
- Art der Authentifizierung (MD5/SHA)
- Art der Paketverschlüsselung (DES, optionale Verfahren werden gerade von der IETF überlegt)
- Passwort für die Authentifizierung, als auch für die Paketverschlüsselung: Diese beiden Schlüssel können zur Erhöhung der Sicherheit ungleich sein.

Wenn man diesen Eintrag getätigt hat, wird beim nächsten Neustart des Masteragenten der Benutzer angelegt und das Passwort verschlüsselt in den persistenten Daten des Agenten abgelegt. Nun ist der Benutzer persistent angelegt und man kann diesen Eintrag wieder aus der Datei löschen.
Vorsicht ist allerdings geboten, wenn man den Masteragenten beim Aufruf zwingt, nur eine einzige Konfigurationsdatei einzulesen (durch: `-C -c /etc/snmpd.conf`). In diesem Fall wird das Passwort nicht verschlüsselt in den persistenten Daten abgelegt und nachdem man den Eintrag `createUser` aus der Datei löscht, ist somit auch der Benutzer verschwunden.

group: Dieses Schlüsselwort erlaubt die Erzeugung von Gruppen, denen man mehrere Benutzer zuweisen kann. Bei der Zuweisung wird sowohl der Gruppenname, die SNMP Version und der Benutzer angegeben. Möchte man einem Benutzer Zugriff auf verschiedene Versionen erlauben, so muss man dies explizit durch zusätzliche Einträge festlegen.

access: Definiert die Rechte. Mit diesem Schlüsselwort wird einer Gruppe für eine Protokollversion und einer Zugriffsart, Zugriff auf einen, durch die MIB vView definierten MIB

Bereich erlaubt. Es können dabei bis zu insgesamt drei verschiedene MIB Views festgelegt werden, für die drei verschiedenen Modi (Lesezugriff, Schreibzugriff und Benachrichtigungen). Falls man bei einem der drei Modi keine MIB View definieren möchte so kann man dies mit `none` (der leeren MIB View) ausdrücken.
Als weiteres Argument wird noch der Context angegeben. Grundsätzlich verwendet man hier den leeren Context „„. Der Parameter `exact` gibt an, dass der Context immer exakt mit dem von der Anfrage mitgegebenen Context übereinstimmen muss. Der Wert prefix hingegen bedeutet, dass der Anfang des Context ident sein muss.

trapsink: Mit trapsink spezifiziert man an welche IP ein SNMPv1 Trap gesendet wird. Leider gibt es in Net-SNMP standardmäßig noch keine Möglichkeit gewisse SNMP Traps an bestimmte IPs zu senden und andere an eine andere Gruppe von Managementstationen. Die Filterung, ob ein Trap für eine Managementstation interessant ist oder nicht, muss also bei der jeweiligen Managementstation passieren.
trap2sink: trap2sink sendet an die angegebene IP einen SNMPv2 Trap. Will man an mehrere Stationen einen Trap senden, so muss dies genau wie bei trapsink jeweils einzeln angegeben werden.
trapsess: Dieses Schlüsselwort ermöglicht es einen SNMPv3 Trap an eine Station, für einen bestimmten Benutzer zu senden. Leider ist diese Art der Festlegung statisch und gilt nur für die jeweils angegebene Engine ID.
Im Folgenden soll eine Beispielkonfiguration angegeben werden, die die Verwendung der wesentlichsten Konfigurationselemente anschaulich darstellt.

Die Konfigurationsmöglichkeiten für AgentX Unterstützung werden unter „Zusätzliche Konfiguration für EasySnmp“ erläutert.

```
##############################################################################
# snmpd.conf
##############################################################################
# SECTION: Access Control Setup
#   This section defines who is allowed to talk to your running
#   snmp agent.
##############################################################################
# rwuser: a SNMPv3 read-write user
# rouser: a SNMPv3 read-only user
#   arguments:  user [noauth|auth|priv] [restriction oid]
# rwuser admin priv
# rouser dau priv

createUser admin MD5 „authpasswd“ DES „privpasswd“
createUser dau MD5 „ authpasswd“ DES „privpasswd“
```

```
createUser daniel MD5 „ authpasswd“ DES „privpasswd“

# Definition der „administrator“ Gruppe (Mitglieder: admin, daniel)
# group groupname sec.model user
group administrator usm admin
group administrator usm daniel

# Definition der MIB View „all“
view all included .1

# Definition der Zugriffserlaubnis
# access groupname context sec.model sec.level prefix read write notify
access administrator „„ usm priv exact all all all

# Erstellen der „daugroup“ mit Member „dau“
group daugroup usm dau
view dauview included .1.3.6.1.4.1.17100.2
access daugroup „„ usm priv exact dauview none none

# rocommunity: a SNMPv1/SNMPv2c read-only access community name
#   arguments:  community [default|hostname|network/bits] [oid]
rocommunity public localhost
rocommunity public 194.96.148.9
rocommunity public 192.168.200.0/24

# rwcommunity: a SNMPv1/SNMPv2c read-write access community name
#   arguments:  community [default|hostname|network/bits] [oid]
rwcommunity private localhost
rwcommunity private fab001.fabalabs.org
rwcommunity private fab007.fabalabs.org

# SNMPv1 trap
trapsink localhost public
trapsink 192.168.200.51 public
trapsink 192.168.200.57 public

# SNMPv2 trap
trap2sink localhost public
trap2sink 192.168.200.51 public
trap2sink 192.168.200.57 public

#SNMPv3 trap
trapsess -v 3 -l authPriv -u admin -A danielfallmann -e 0x0102030405
192.168.200.58:162

# send traps on authentication failure
authtrapenable 1
```

4.2.3 Benutzerverwaltung

Es wurden, wie in dem angeführten Snippet beschrieben, bereits drei Benutzer angelegt. Man kann Benutzer auch auf andere Arten anlegen und verwalten. Die Benutzerverwaltung von Net-SNMP verwaltet alle Benutzer in den persistenten Datenbeständen, die die VACM-MIB repräsentieren. Damit man nicht manuell in diese Konfiguration eingreifen muss, gibt es zwei Befehle die man vor allem in großen Umgebungen des Öfteren braucht, `snmpusm` und `snmpvacm`. In diesem Zusammenhang wurde festgestellt, dass es sinnvoller ist, bestehende Benutzerprofile zu klonen und dann deren Passwort zu setzen, als für jeden einzelnen Benutzer die gesamte Prozedur der Erstellung zu durchlaufen.

snmpusm: Dieser Befehl legt einen SNMPv3 (usmUser) an.
snmpvacm: Mittels snmpvacm werden Benutzer einer Gruppe zugeordnet bzw. bestehende Gruppenzugehörigkeiten gelöscht und neue erstellt. Man kann mit diesem Befehl auch das Sicherheitsmodell umstellen und neue MIB Views anlegen.

Im Folgenden wird an Hand von Beispielen, die Verwendung von `snmpusm` und `snmpvacm` erläutert.
Man klont zuallererst aus dem Benutzer `admin` einen neuen Benutzer `daniel`, der demselben Profil entspricht wie der Benutzer `admin`. Die angegebenen Passwörter (authpasswd, privpasswd) sind vom Benutzer admin. Nach der Erstellung hat nun auch der Benutzer daniel dieselben Passwörter. Die Gruppe administrator, der admin und daniel angehören, hat volle Rechte auf alle MIB-Bereiche.

```
snmpusm -v3 -u admin -l authPriv -a MD5 -A authpasswd \
        -x DES -X privpasswd localhost create daniel admin
```

Als logischen nächsten Schritt wird das Passwort des Benutzers daniel geändert. Dazu benötigt man insgesamt zwei Aufrufe. Das Passwort für die Authentifikation des Benutzers und das Passwort für die Paketver- /Paketentschlüsselung müssen getrennt gesetzt werden.

```
snmpusm -Ca -v3 -u daniel -l authPriv -A authpasswd \
        -X privpasswd localhost passwd authpasswd newauthpasswd
snmpusm -Cx -v3 -u daniel -l authPriv -A newauthpasswd \
        -X privpasswd localhost passwd privpasswd newprivpasswd
```

Mit der Option -Ca ändert man das Authentifikationspasswort des Benutzers daniel von authpasswd auf newauthpasswd.
Mit der Option –Cx wird anschließend das Autorisationspasswort von privpasswd auf newprivpasswd geändert. Man muss an dieser Stelle aber darauf achten, dass im zweiten Fall bereits dass neue Authentifikationspasswort von daniel benutzt werden muss, um überhaupt Zugriff auf die VACM-MIB Daten zu bekommen und diese ändern zu können.

In dieser Arbeit wurde auch untersucht, wie man Benutzer in anderen Gruppen mit Hilfe eines Shellzugriffs zuweisen kann. Als Beispiel dazu soll gezeigt werden, wie der Benutzer daniel aus der Gruppe administrator genommen werden kann und er der Gruppe daugroup hinzugefügt werden kann. Leider ist es nicht möglich, innerhalb desselben Sicherheitsmodells (z.B. usm) einen Benutzer zwei unterschiedlichen Gruppen zuzuordnen. Das Tripel (Benutzer, Gruppe, SecurityModel) muss eindeutig sein.
Zunächst wird der Benutzer daniel aus der Gruppe administrator entfernt:

```
snmpvacm -v3 -u admin -l authPriv -A authpasswd -X privpasswd localhost \
         deleteSec2Group 3 daniel

Sec2group successfully deleted.
```

Nun kann man den Benutzer der Gruppe daugroup hinzufügen:

```
snmpvacm -v3 -u admin -l authPriv -A authpasswd -X privpasswd localhost \
         createSec2Group 3 daniel daugroup

Sec2group successfully created.
```

Der Benutzer daniel gehört nun der Gruppe daugroup an und hat daher eingeschränkte Rechte auf den verfügbaren MIB-Baum.

4.2.4 Zusätzliche Konfiguration für EasySnmp

Für die Verwendung von EasySnmp muss auf Masteragentenseite nur die AgentX Schnittstelle aktiviert sein. Die AgentX Unterstützung kann man in der Konfigurationsdatei `/etc/snmpd.conf` aktivieren. Diese zentrale Möglichkeit der Steuerung wird in dieser Arbeit bevorzugt. Nichtsdestotrotz ist es auch möglich beim Start des Agenten die Unterstützung mit der Option -x

Port zu aktivieren. In der `snmpd.conf` kann man eine Vielzahl an wichtigen Optionen für AgentX angeben und konfigurieren. Aufgrund der Bedeutung von AgentX für unsere Arbeit sollen hier kurz die Einstellungsmöglichkeiten erläutert werden.
Um den AgentX Support einzuschalten muss man den Eintrag „**master agentx**“ in die snmpd.conf Datei schreiben.

AgentXSocket addr: Als Parameter addr kann man nun den Port, auf dem der Masteragent Subagentenpakete entgegennimmt, setzen. Standardmäßig sind Unix Domain Sockets eingestellt, also /var/agentx/master. Damit der Masteragent aber über Domain Sockets mit den Subagenten kommunizieren kann, muss dieser dieselbe userid wie die Subagenten haben.

AgentXTimeout timeout: timeout gibt die Anzahl der Sekunden an, bevor der Masteragent, auf eine Anfrage an den Subagenten, ein Timeout erhält.

AgentXRetries retries: Der Parameter retries legt fest, wie oft der Masteragent versucht eine AgentX Anfrage zu stellen, wenn er zuvor auf keine Anfrage eine Antwort des Subagenten zurückerhalten hat. Die Standardeinstellung sieht fünf Versuche vor.

agentPingInterval NUM: Diese Zeile kann in der `subagent.conf` Datei angegeben werden. Jeder Subagent hat seine eigene Konfigurationsdatei. Den Namen erhält die Datei durch den Aufruf API Call `init_snmp(„subagent“)` im jeweiligen Subagentenprogramm. Der String, in unserem Fall „subagent“, gibt der Datei dann seinen Namen.
Der Parameter NUM gibt die Anzahl der Sekunden an, die verstreichen bis der Subagent erneut versucht, den Masteragenten zu erreichen. Dies geschieht mit dem eigens dafür vorgesehenen AgentXPing.

Weiterführende Informationen zum Thema AgentX entnehmen Sie bitte dem Kapitel Implementierung der EasySnmp Bibliothek.

4.2.5 Agenten Verwaltung (EngineID)

Der Benutzer muss sich nicht um die gezielte Verwaltung der einzelnen Subagenten kümmern. Der Masteragent übernimmt diese Aufgaben und kümmert sich ebenso um die EngineIDs. Der

Benutzer muss sich bei SNMPv3 Traps jedoch um die statische Festlegung der EngineID des jeweiligen Benutzers im System kümmern.

5 Erweiterungsmöglichkeiten des Agenten

If you think you are too small to make a difference,
try sleeping in a closed room with a mosquito.
— African proverb

Die Erweiterungsmöglichkeit von SNMP Agentensystemen ist schon in frühen Versionen der SNMP Agenten ein wesentliches Thema gewesen. Die entscheidenden Fragestellungen an dieser Stelle sind:
Kann man einen bestehenden SNMP Agenten in seiner Funktionalität, also in den nach außen abzubildenden Daten erweitern?
Kann man zur Laufzeit eines bestehenden Agentensystems (bestehend aus einem Masteragent und mehreren Subagenten) neue Daten nach außen zur Verfügung stellen, ohne das laufende System zu beeinflussen?
Die Antwort auf diese Fragen ist, ja, man kann. Es gibt eine ganze Reihe von standardisierten Ansätzen, die über die Jahre entstanden sind, um Agentensysteme zu erweitern. Diese Ansätze unterscheiden sich in ihrem Einsatzbereich aber doch ganz beträchtlich. Um herauszufinden welche Methode die für diese Aufgabenstellung passende darstellt, mussten die drei grundsätzlichen Architekturen evaluiert werden. Es besteht die Möglichkeit den Masteragenten direkt zu erweitern, was mittels Shared Objects ermöglicht wird, die dann direkt in den Masteragenten eingebunden werden. Diese Architektur nennt man eine monolithische Architektur. Weiters gibt es die Möglichkeit einen Proxy Agenten einzusetzen, der Anfragen direkt an den jeweiligen Agent weiterleitet, der diese beantworten kann. Zu guter letzt gibt es auch noch die Möglichkeit, bestehende Masteragenten mittels so genannten Subagenten zu erweitern. Für diesen Ansatz muss jedoch der Masteragent diese Fähigkeit nach außen zur Verfügung stellen.

5.1 Auswahl

5.1.1 Monolithische Architektur

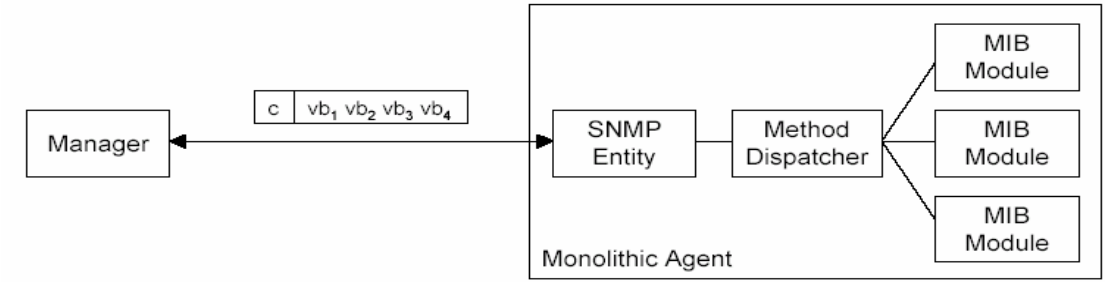

Abbildung 20: Monolithische Agentenarchitektur, Quelle [SCH03, S. 207]

Der Ansatz der monolithischen Agentenarchitektur geht davon aus, dass der Agent über ein Modulsystem erweitert werden kann. Hierzu wird der Quellcode des zusätzlichen MIB Moduls übersetzt. Das fertig übersetzte Modul kann dann in den Agenten geladen werden und das neu erstellte Modul wird dann über eine vordefinierte Schnittstelle aufgerufen. Dieser Ansatz ist aber nur dann sinnvoll, wenn die Funktionalität der neuen Module nicht transparent vom Masteragenten gehalten werden sollen. Bei Anwendung dieser Erweiterungsmethode muss dem Modulentwickler jedoch klar sein, dass seine Module auch die Stabilität des Agenten gefährden können und im schlimmsten Fall sogar dessen Ausfall zur Folge haben können. Weiters müssen Module dem Agenten bekannt gemacht werden, was bei heute gängigen SNMP Agentensystemen auch oft den Neustart des Agenten zur Folge hat. Leider gibt es hier auch keinen Standard für Modulschnittstellen, sodass man beim Bau der Erweiterungsmodule bereits die Zielinfrastruktur genauestens kennen muss.

5.1.2 Proxyagent

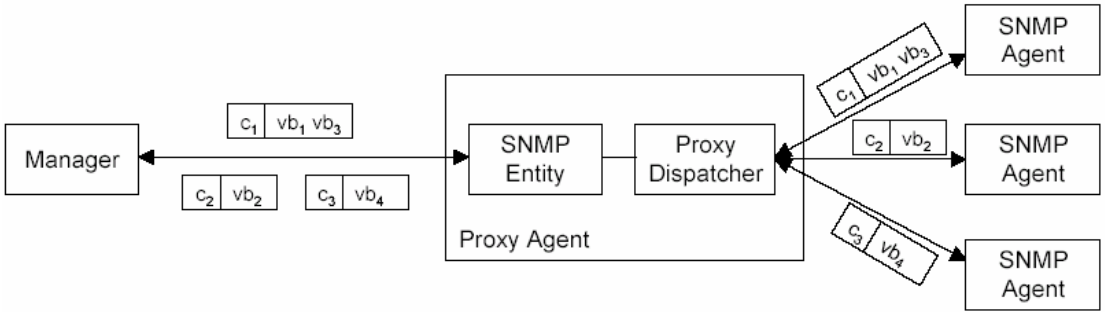

Abbildung 21: Abbildung 19: Proxy Agentenarchitektur, Quelle [SCH03, S. 208]

Das Konzept der Proxyagenten existiert im Wesentlichen in zwei Ausführungen. Einerseits gibt es Agenten die nur als Weiterleitungszentrale für SNMP Anfragen dienen, andererseits gibt es aber auch Agenten die sich nicht nur um eine Weiterleitung kümmern, sondern auch um eine Transformierung der Daten vom SNMP Protokoll in ein weiteres Protokoll, oft auch proprietäres Protokoll. Nebenbei sei hier auch noch erwähnt, dass diese so genannten „Translating Proxies“ oft von großen Softwareherstellern eingesetzt werden, um SNMP kompatible Requests Systemintern auf proprietäre Protokolle abzubilden.
Das Konzept der Proxyagenten, wie es in dieser Evaluierung berücksichtigt wurde, basiert auf dem Weiterleiten von SNMP Anfragen, von einem Managementsystem, an den jeweils für diese Anfrage zuständigen SNMP Agenten. Damit die Weiterleitung erfolgreich sein kann, muss mit jeder Anfrage auch eine Agentenkennung mitgesendet werden, die den zuständigen Agenten identifiziert.
Der Vorteil dieses Ansatzes besteht darin, dass man nur die Adresse des Proxyagenten wissen muss und zusätzlich die Information benötigt, welches Endsystem (z.B. mittels einer AgentID) die Information bereitstellt. Es können nun erfolgreich Anfragen gestellt werden.
Bei Systemen, wo Sicherheit ein ganz wesentlicher Aspekt ist, kann man diese Proxyagenten auch dazu verwenden, um eine zentrale Authentifikationsstelle am Proxyserver einzurichten. Dies ist bei großen Infrastrukturen von essentieller Wichtigkeit. Aufgrund der Systemarchitektur wird hier die Erweiterung im laufenden Betrieb gewährleistet.
Mit einem Proxy wird es auch möglich Konvertierungen zwischen den unterschiedlichen SNMP Versionen vorzunehmen. Ein Proxy ermöglicht aber auch die Konfiguration von Firewalls zwischen den Management-Stationen und den SNMP Agenten.

5.1.3 Subagenten-fähiger Masteragent

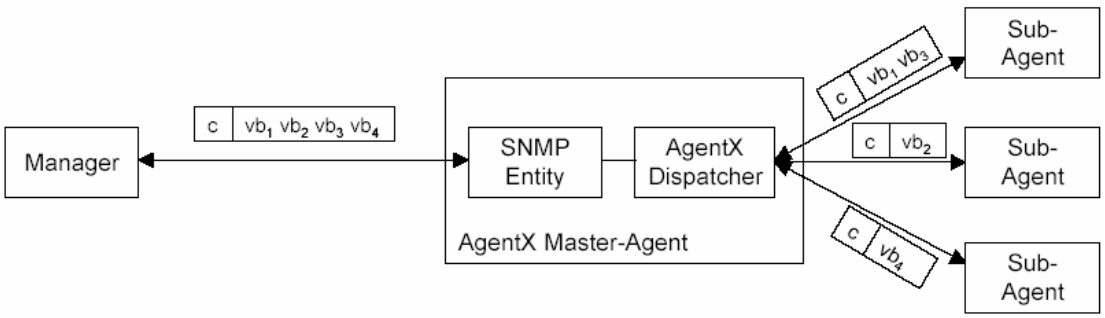

Abbildung 22: Abbildung 19: AgentX Subagentenarchitektur, Quelle [SCH03, S. 209]

Eine weitere Architektur ist die der Master- und Subagenten. Sie nimmt im Agentenbau mittlerweile eine zentrale Stellung ein. Hier ist nun eine Trennung in einen Masteragenten und einen Subagenten gegeben. Diese beiden Teilsysteme, die ein gemeinsames Agentensystem nach außen abbilden, kommunizieren miteinander. Die Aufgabe des Masteragenten ist es, per SNMP Protokoll Daten nach außen, Richtung einer Management-Station zur Verfügung zu stellen. Da der Masteragent aber nur einen Teil der Daten lokal zur Verfügung hat, ist dieser auf Subagenten angewiesen, die ihm weitere Daten liefern, die nach außen abgebildet werden müssen. Die Verarbeitung und das Sammeln der Informationen geschieht aber völlig transparent für die Management-Station. Aus Sicht der Management-Station gibt es eine zentrale Anlaufstelle, an die Anfragen gestellt werden, den Masteragenten. Für diesen Masteragenten sind aber über ein eigenes Protokoll mehrere Subagenten verfügbar. Diese Subagenten helfen ihm, die Anfragen des Managers zu beantworten und abzuarbeiten. Es können unterschiedlichste Protokolle für die Inter Process Communication (IPC) eingesetzt werden. Gängige Protokolle sind AgentX, DPI, SMUX, EMANATE. Diese Protokolle haben, trotz der gleichen zu bewältigenden Aufgabenstellung, ganz spezifische Vor- und Nachteile.

SMUX wurde 1990 spezifiziert und später mit dem RFC 1227 standardisiert. Bei diesem System gibt es genau einen SNMP Agenten, der auf alle SNMP Anfragen hört. Der Agent an sich betreibt aber seinerseits Mulitplexing, indem er die Anfragen auf eine untergeordnete Menge von so genannten SMUX Peers weiterleitet und sich selbst nicht unmittelbar um die Beantwortung kümmern muss, obwohl der SNMP Agent die Antwort des Subagenten an den jeweiligen Manager weiterleitet. SMUX hatte aber eine ganze Reihe von ungelösten Problemen: Eines von ihnen war das „sysUpTime-Problem“. Die sysUpTime gibt an, wie lange ein System schon läuft bzw. eigentlich wie lange das Agentensystem schon läuft. Da aber die SMUX Peers zu verschiedenen Zeiten gestartet werden können, würde jeder dieser Peers eine andere sysUpTime liefern. Ein weiteres Problem wird durch die streng sequentielle Abarbeitungsreihenfolge der Pakete verursacht. Ein SMUX Agent darf erst dann die nächste Anfrage verarbeiten, wenn der zuvor beauftragte Peer bereits eine Antwort geliefert hat. Das hat aber zur Folge, dass sich das System in einem endlos wartenden Zustand befindet, wenn während der Verarbeitung einer Anfrage der Peerprozess stirbt ohne eine Antwort zu liefern.

Ein weiters Problem ist, dass auch zu den einzelnen Peers SNMP gesprochen wird. Daher verdoppelt sich im besten Fall der Aufwand für die Decodierung der SNMP Pakete, was eine schlechte Performance zur Folge hat.

Aus diesem Grund trat ein weiteres Protokoll (DPIv1, RFC 1228), das 1991 veröffentlicht wurde an, um diese Probleme in den Griff zu bekommen. DPI ist, wie schon zuvor SMUX, ein mittels

Subagenten erweiterbares System. Der wesentliche Unterschied zwischen SMUX und DPI, der auch für spätere Systeme kennzeichnend sein sollte, war, dass DPI ein eigenes „lightweight“ Protokoll zwischen dem Masteragenten und Subagenten definierte. SMUX hingegen gab die SNMP Pakete unverändert weiter. Drei Jahre nach der ersten Einführung von DPIv1 wurde vom IBM Research die Version 2 von DPI (DPIv2, RFC 1592) veröffentlicht. Diese Version war wesentlich ausgereifter und umfangreicher als Version 1.
Zusammenfassend bekam DPI zwar das Peformanceproblem von SMUX in den Griff, aber nach wie vor bestand das sysUpTime-Problem und zusätzlich hatte man nun bei DPI auch kein Two-Phase-Commit für SET Operationen.
Aufgrund der großen Nachfrage an erweiterbaren Agentensystemen (so genannten „extensible agents“) kamen 1995 die ersten kommerziellen Systeme (z.B. Envoy von Epilogue Technologies, EMANATE von SNMP Research) auf den Markt, die mit gesteigerter Performance und Arbeitsteilung bis auf SNMP Instanzebene zu punkten versuchten. Ohne Zweifel hat dies zur Verbesserung der Systeme beigetragen. Es wurde schnell klar, dass ein neuer, besserer Standard nötig war, um dem aktuellen Stand der Technik, der kommerziellen Produkte zu entsprechen und hier wieder für Interoperabilität zu sorgen. Die Antwort auf diese Bemühungen heißt AgentX.

5.2 AgentX

5.2.1 Konzept

Bei AgentX handelt es sich, wie zuvor erwähnt, um ein so genanntes eXtensible Agent Protokoll. Dieses Protokoll ist an sich noch sehr jung, jedoch hat es sich sehr rasch zum de facto Standard für heutige Agentensysteme entwickelt. AgentX wurde im Jänner 2000 als RFC 2741 publiziert.
Das Protokoll verteilt die Rollen im SNMP Agenten um. Heute wird zwischen folgenden Agenten unterschieden:

- Masteragenten: Diese Regeln die SNMP Kommunikation, haben aber zunehmend was die Informationsbeschaffung und somit die Beantwortung von Anfragen angeht eine passive Rolle.
- Subagenten: Die Menge der Subagenten sollen möglichst disjunkte Mengen von Management Information abbilden. Ein Subagent ist SNMP omniszent und spricht nur über AgentX mit seinem zugeordneten Masteragent.

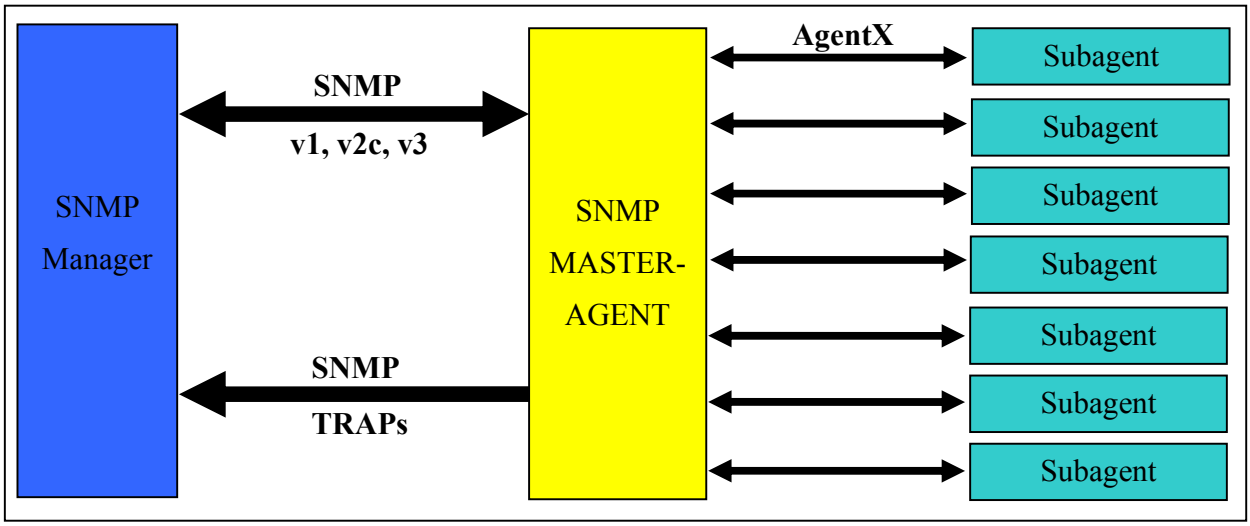

Abbildung 23: AgentX Design

Wie in Abbildung 22 ersichtlich, übernimmt der SNMP Masteragent nur mehr Multiplexingtätigkeiten. Dies hat vor allem entscheidenden Einfluss auf die Kompatibilität zu den unterschiedlichen SNMP Versionen. Das Agentensystem kann nun durch Umstellung des Masteragenten sehr einfach SNMP v1, v2c und v3 sprechen. An den Subagenten muss dafür keine Änderung vorgenommen werden, diese haben eine definierte Schnittstelle, AgentX. Eine Anforderung an den Masteragenten ist, dass er die gängigen SNMP Protokolle unterstützt. Heute verfügbare Masteragenten unterstützen üblicherweise alle drei SNMP Versionen.

5.2.2 Protokolldetails

AgentX kann unter Unix/Linux direkt über Domain Sockets kommunizieren. Außerdem ist es auch möglich die Kommunikation mit Transmission Control Protocol (Standard: Port 705) zu regeln. Eine AgentX Verbindung besteht aus mehreren Komponenten. Zuallererst kann eine Verbindung (Connection) in mehrere sogenannte Sessions zerlegt werden. Jede Session kann ihrereseits wiederum in mehrere Transaktionen zerlegt werden. Über eine solche Transaktion läuft dann schlußendlich die tatsächliche AgentX PDU bzw. weist eine PDU dann eine Session ID, Tranaction ID und eine Packet ID auf die diese eindeutig identifiziert.

AgentX ist ein relativ kleines Protokoll. Es werden insgesamt nur 18 verschiedene PDUs angeboten, die für die Kommunikation ausreichen. Davon sind **9 administrative Pakete** (AgentX Open PDU, AgentX Close PDU, AgentX Register PDU, AgentX Unregister PDU, AgentX Ping PDU, Agentx Indexallocate PDU, Agentx AddAgentCaps PDU, AgentX Remove Caps PDU), **8 Datenpakete** (AgentX Get PDU, AgentX GetNext PDU, AgentX GetBulk PDU, AgentX TestSet PDU, AgentX CommitSet PDU, AgentX UndoSet PDU, AgentX Notify PDU) und **ein generisches Antwortpaket** (AgentX Response PDU).

Ein Großteil der Arbeit nimmt Net-SNMP bereits vorweg, nichtsdestotrotz ist es wichtig ein paar wenige Details zu verstehen, damit man selbst Programme erfolgreich schreiben kann, die AgentX korrekt einsetzen.

5.2.2.1 Verbindungsaufbau

Jedesmal wenn ein AgentX Subagent gestartet wird, versucht dieser sich mit dem Masteragenten zu verbinden. Dazu sendet der Subagent ein AgentX Open PDU Paket zum Masteragenten. Der Masteragent weist den Subagenten nun eine neue Session ID zu und sendet ein Antwortpaket. Falls es keine Verbindungsprobleme gab, sollte nach der Antwort die Session für Datentransaktionen benutzt werden können. Damit der Masteragent weiß, welche MIB Bereiche welchen Subagenten zugeordnet sind, muss jeder Subagent die von ihm übernommenen Bereiche (im Fall von EasySnmp wird aus Performance Gründen jede einzelne OID registriert) beim Masteragenten registrieren. Für die Registrierung gibt es die AgentX Register PDU, die vom Subagenten an den Masteragenten gesandt wird und vom Masteragenten bei erfolgreichem Abschluß mit einer AgentX Response PDU beantwortet (acknowledge) wird.

5.2.2.2 Datenabfrage an AgentX Subagent

Wenn ein Masteragent von einem Manager eine Anfrage empfängt kümmert sich dieser darum, diese zu beantworten. Der Masteragent sucht nun nach dem passenden Subagenten. Nachdem er diesen gefunden hat, wird das SNMP Paket transformiert und ein AgentX Paket erzeugt, welches an den Subagenten gesandt wird. Der Subagent beantwortet den AgentX Get, GetNext und GetBulk Request mittels eines AgentX Response. Die Antwort des Subagenten wird nun vom Masteragenten wieder transformiert und als SNMP Response an den Manager geschickt.

5.2.2.3 Verändern von neuen Werten

Eine Set-Operation ist mit dem Subagentenkonzept ein bisschen aufwändiger durchzuführen. Da eine Anfrage eines Managers an den Masteragenten mehrere Werte ändern kann, muss sichergestellt sein, dass alle Set-Operationen durchgeführt werden oder keine. AgentX stellt drei Phasen zur Verfügung, die eine Absicherung gegen das eben beschriebene Problem darstellen, eine Testphase und eine Zweiphasen-Commit Funktionalität. Somit kann die SNMP Eigenschaft des atomaren Setzens trotz AgentX Subagenten gewahrt werden, auch wenn mehrere Subagenten an der Set-Operation beteiligt sind. Bevor der Masteragent eine Set-Anfrage an die beteiligten Subagenten weiterleitet wird mittels AgentX TestSet PDU sichergestellt, dass die Subagenten und die Managed Objects tatsächlich erreichbar sind. Wenn alle AgentX TestSet Anfragen an die Subagenten keinen Fehler zurückliefern, so wird mit dem AgentX CommitSet fortgefahren. Angenommen die CommitSet-Operation ist erfolgreich so wird an alle beteiligten Subagenten eine CleanupSet PDU gesandt, welche die Änderungen tatsächlich wirksam macht. Falls eine CommitSet PDU einen Fehler zurückliefert so wird an alle Subagenten eine UndoSet SPU geschickt. Dieses UndoSet Paket stößt ein Zurückrollen der Operation bei den Subagenten an. Das Limit dieser Architektur liegt dabei leider bei jeweils einer Set Operation pro Session, Multithreading wird auf Sessionebene nicht unterstützt.

5.2.2.4 Senden von Benachrichtigungen

Benachrichtigungen (Notifications) oder Traps sind aktive Reaktionen des Agenten auf seinen überwachten Bereich. Auch im Fall von AgentX ist es eine Aufgabe des Subagenten diese zu versenden. Der Masteragent transformiert die AgentX Nachricht des Subagenten und leitet diese als SNMP Trap an den jeweiligen definierten Master (trapsink) weiter. Falls ein Fehler auftritt bei der internen Bearbeitung so wird dies dem Subagenten von Masteragenten mitgeteilt. Leider gibt es keine Möglichkeit dem Subagenten mitzuteilen falls ein Trap nicht beim Empfänger ankommt. Selbst bei so genannten Informs ist es nur möglich dass der Masteragent benachrichtigt wird. Eine direkte Benachrichtigung des Subagenten ist im Standard nicht vorgesehen. Um dieses

Problem zu lösen, gibt es mittlerweile Zusatzimplementierungen durch verschiedene Masteragenten, aber leider keinen einheitlichen standardisierten Lösungsweg.

6 Die EasySnmp Library

Each pattern describes a problem which occures over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

— Christopher Alexander et al., A Pattern Language

6.1 Verwendete Mechanismen von Net-SNMP

Die verwendete Grundbibliothek Net-SNMP verfügt über einen sehr großen Funktionsumfang. Anhand der Abbildung 23 kann man sich einfach einen Überblick über Net-SNMP verschaffen. Der Masteragent bietet sehr viele Funktionen, die man aufgrund des generischen Aufbaus auch großteils für den Subagenten verwenden kann.

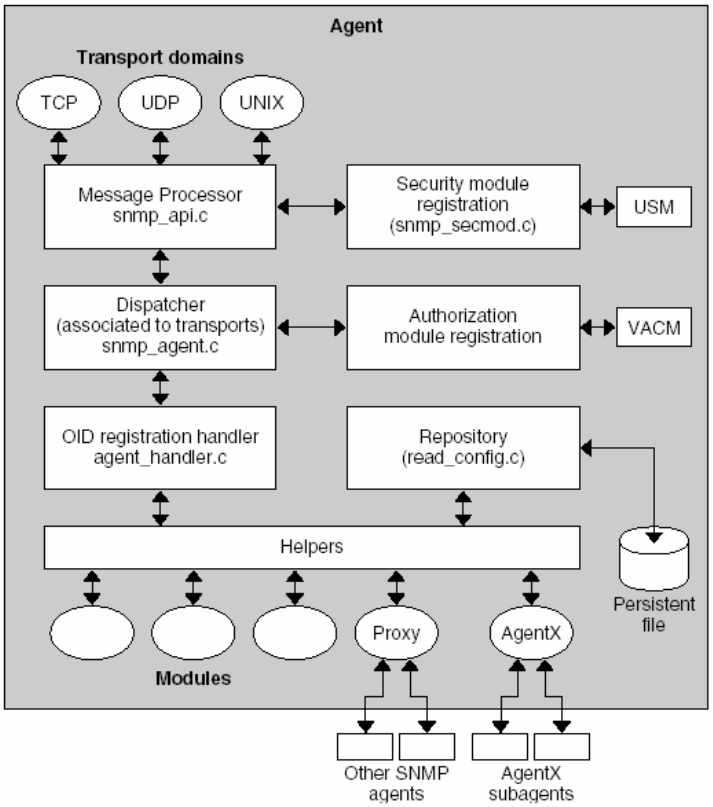

Abbildung 24: Net-SNMP Agentenarchitektur, Quelle: [Sdg04, S. 19]

Der Net-SNMP Masteragent unterstützt drei Transportmechanismen (Protokolle): TCP, UDP und Unix Domain Sockets. Unter Linux hat man alle drei vollständig zur Auswahl. Unter Windows gibt es keine Unix Domain Sockets, weshalb dort diese Variante des Betriebes wegfällt. Die empfangenen SNMP Daten werden an den Message Processor (MP) weitergegeben.

Message Processor

Der Message Processor ist im Wesentlichen in der Datei snmp_api.c codiert. Er empfängt aber nicht nur die SNMP Daten über den jeweiligen Transportmechanismus, sondern kann auch auf jedem Daten senden. Der MP decodiert nun die Daten und baut daraus die internen `net_snmp_pdu` Strukturen auf. Der MP baut auch aus den C-Strukturen SNMP Pakete auf, die nach den Basic Encoding Rules (RFC 3416, RFC1157) codiert werden. An dieser Stelle werden auch die Sicherheitsdaten gefiltert und weitergeleitet, also die USM Daten werden an das USM Modul weitergeleitet und dort verarbeitet.

USM Modul (RFC 3414)

Dieses Modul kümmert sich um die Verarbeitung von USM Daten. Der Message Processor kümmert sich um die Verwendung des korrekten Modules, also auch um das USM Modul. Das Sicherheitsmodul wird bei der Registrierung angegeben. Das USM Modul entschlüsselt hereinkommende Nachrichten. Nun sind die Authentifikationsdaten zugänglich und es können die jeweiligen Strukturen intern aufgebaut werden. Für den Aufbau von neuen Paketen fügt dieses Modul Authentifikationsdaten hinzu und gibt die PDU an den Message Processor weiter. Die Schlüssel für die Ver- und Entschlüsselung werden persistent im Agenten gespeichert.

VACM Modul (RC 3415)

Das VACM Modul löst anhand der im System abgelegten Daten der VACM-MIB, die Zugriffsrechte auf. Das VACM Modul ist aber nicht nur für das USM Modell interessant, sondern wird auch für die Modelle der v1 und v2 Communities verwendet. Es werden somit nicht nur Benutzerdaten verarbeitet, sondern auch Zugriffsrechte für Communities.

Dispatcher

Ein Dispatcher leitet Nachrichten zu den jeweiligen Empfängern weiter. Der Dispatcher setzt direkt nach Prüfung der Authentisierung an und leitet die Nachrichten an die VACM Einheit weiter, die dann ihrerseits die Autorisierung prüft. Nun wird im „Agent Repositiry“ nach dem

zuständigen Subagenten (oder Modul) gesucht, der für diese Anfrage zuständig ist. Über sogenannte Helpers (so nennt man die gemeinsame Schnittstelle zu den jeweiligen Erweiterungen) werden die benötigten Module geladen. Die Antworten der jeweiligen Module werden im folgenden wieder zu vollständigen SNMP Antwortpaketen aufgebaut und an den Client zurückgesandt.

Für die einzelnen Erweiterungsmöglichkeiten gibt es jeweils ein eigenes Modul, das **Proxy Modul**, das **AgentX Modul** und interne **Shared Objects des Masteragenten**.

Die Shared Objects libnetsnmp, libnetsnmpagent, libnetsnmphelpers und libnetsnmpmibs stellen eine bereits sehr gute Funktionsvielfalt zur Verfügung. Leider bietet das Net-SNMP API keine objektorientierte Sichtweise auf die wesentliche Funktionalitäten an. Ein Grund dafür ist, dass der sehr viel ältere Quellcode der „University of California at Davis“ nach wie vor sehr stark von vielen anderen Stellen im Net-SNMP Quellcode referenziert wird. Die EasySnmp Bibliothek greift auf diese Shared Objects zu, um bestehende Funktionalität zu verwenden und neue Versionen von Net-SNMP, vor allem im Bereich Ausfallsicherheit und Performanceoptimierungen, nutzen zu können.

6.1.1 Die Helperfunktion „Watcher“

Ein für diese Arbeit wesentlicher Mechanismus, der von Net-SNMP angeboten wird und den EasySnmp verwendet, ist der des Watchers. Ein Watcher gehört zur Gruppe der Helperfunktionen und wird benutzt um Werte auf SNMP Ebene auf ihre Länge zu überprüfen. Man hat damit die Möglichkeit, eingehende Daten bevor diese tatsächlich zum verantwortlichen Subagenten gelangen, auf Ihre Größe zu prüfen und gezielte Buffer Overflows durch beispielsweise absichtliches Setzen von zu großen Werten (wie zu langen OCTET STRINGs) zu verhindern.

6.2 Architekturüberlegungen

6.2.1 Architektur-Überblick

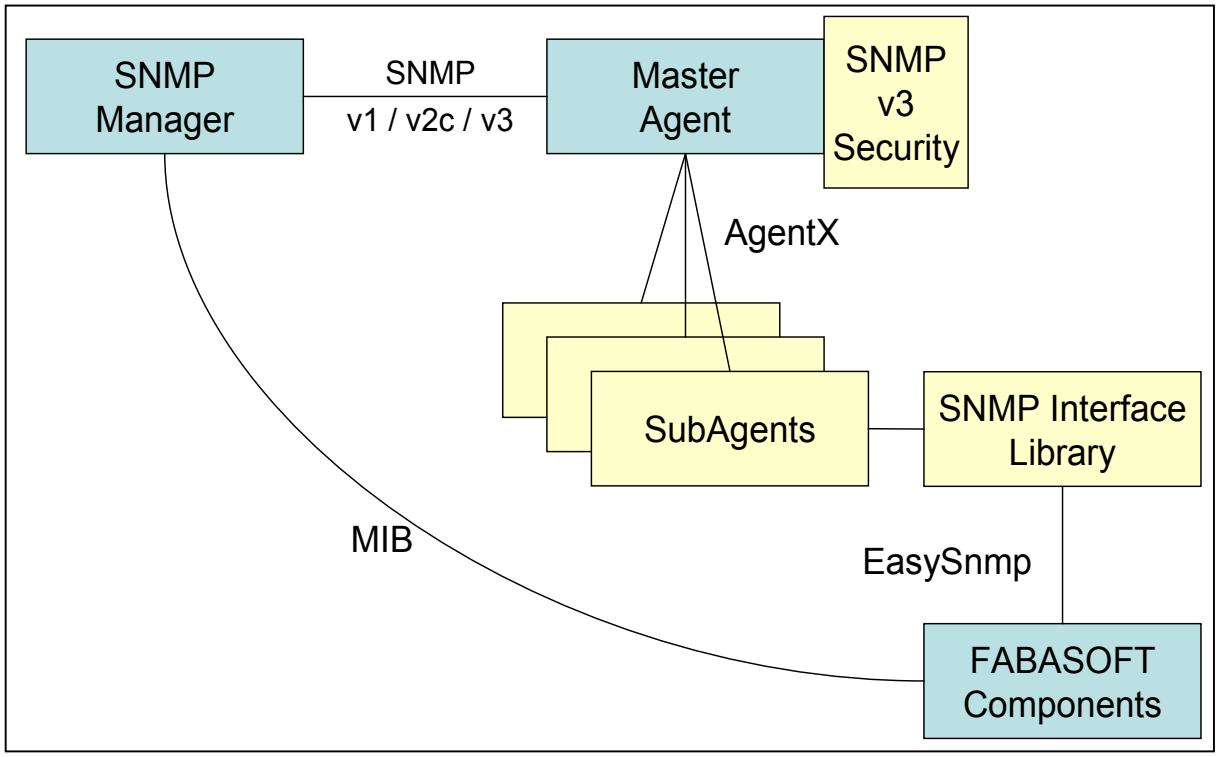

Abbildung 25: EasySnmp Umgebungsarchitektur

Die Architektur von EasySnmp erlaubt es, unterschiedlichen Applikationen (speziell vor allem Fabasoft Components) rasch und einfach SNMP Daten nach außen abzubilden. Dabei definiert die Bibliothek einfach handhabbare Schnittstellen für die Bereiche, „Scalar Objects“ (einzelne Werte), „Columnar Objects“ (tabellarische Wertestrukturen, Arrays, etc.) und für „Notifications“ (Benachrichtigungen wie Traps, Informs). Wesentlich ist dabei auch die Verwendung des lokalen Shared Memories. Aus diesem werden die jeweiligen Werte entnommen und an die in den MIBs vordefinierten OID gebunden. Wie aufmerksame Leser bereits wissen, nennt man diese Bindung ein Variable Binding. Durch die Abbildung von Werten die direkt aus dem flüchtigen Speicher kommen, sind diese Zugriffe sehr schnell.

6.2.2 Shared Memory Support (für Linux)

EasySnmp ist darauf ausgelegt, dass die Werte die abgebildet wurden bevorzugterweise direkt von einem Shared Memory Zugriff stammen. Mit den Shared Memory Konzept lässt sich die schnellstmögliche Form der Interprozesskommunikation umsetzen. Dabei schreibt ein Programm

(beispielsweise Fabasoft Components) die abzubildenen Werte in den Speicher und eine auf EasySnmp basierende Applikation liest diese Daten direkt aus dem Speicher und bildet diese nach außen per SNMP ab. Weiters ist es möglich, die Funktionalität von EasySnmp direkt von mehreren Thread zu nutzen, dafür muss lediglich der Subagent (der die Session zum Masteragenten eröffnet) im jeweiligen Thread instanziert werden.
In einem Shared Memory Bereich werden neue Ressourcen angelegt, indem man einen Schlüssel und eine Größe angibt. Der Quellcode für das Anlegen eines Shared Memory Segments sieht unter Linux beispielsweise wie folgt aus:

```
// anlagen eines neuen Elements im Shared Memory
shmid = shmget(key, size, IPC_CREATE(0555));

// falls es dieses Element bereits existiert
if (shmid == ERROR) {
  shmid = shmget(key, size, 0555);
}
```

Das Anlegen der Speicherelemente im Shared Memory geschieht normalerweise durch die Applikation, die man verwalten möchte. Aufgabe des SNMP Agentensystems ist es nun, die Zeiger auf solche Datenelemente zu verwenden und diese nach außen per SNMP zur Verfügung zu stellen. Zugriff auf den Speicher über das Wissen des Identifiers (`shmid`) möglich. Um den Zeiger auf ein Datenelement im Shared Memory zurückzukommen benützt man `shmat`.

```
// Zeiger auf Datenelement des Shared Memory anfordern
char *ptr;
if ((ptr = (char*)shmat(shmid, (char*)0,0)) == ERROR {
  ...
}
```

Der erste Parameter von `shmat` gibt, über die `shmid`, das Speichersegment an, auf welches zugegriffen wird. Der zweite Parameter wählt die Region innerhalb es Speichersegments aus. Der dritte Parameter gibt die Zugriffsart an, zum Beispiel 0 für lesenden Zugriff.

6.3 Dynamische Library

EasySnmp ist eine dynamisch zur Laufzeit integrierbare C++ Bibliothek, die unter Linux als Shared Object zur Verfügung gestellt wird. Ziel dieser Bibliothek ist, dass Standardfunktionen von SNMP sehr einfach zugänglich werden und ein Programmierer wenig weiterführendes Hintergrundwissen über Subagentensysteme benötigt. Der große Vorteil besteht darin, dass alle AgentX Subagenten zur Laufzeit den Masteragenten erweitern können, was wiederum die nötige Flexibilität in großen Systemen bringt.

6.4 Grundstruktur für AgentX Subagent (als Prozess)

```
// net-snmp includes
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
// EasySnmp includes
#include "SnmpOid.h"
#include "ScalarManager.h"
#include "TableManager.h"
#include "NotificationManager.h"
#include "TableLayout.h"

/* change this if you want to use syslog */
#define _SYSLOG_ 0

static int keep running;
using namespace std;

RETSIGTYPE stop_server(int a) {
    keep_running = 0;
}

int main (int argc, char **argv) {
  int background = 0; /* change this if you want to run in the background */
  int syslog     = 0;

  /* print log errors to syslog or stderr */
  if (_SYSLOG_) { snmp_enable_calllog(); }
  else { snmp enable stderrlog(); }

  netsnmp_ds_set_boolean(NETSNMP_DS_APPLICATION_ID, NETSNMP_DS_AGENT_ROLE,
1);
  netsnmp_ds_set_string(NETSNMP_DS_APPLICATION_ID, NETSNMP_DS_AGENT_X_SOCKET,
                        "localhost:705");

  /* run in background, if requested */
  if (background && netsnmp daemonize(1, !syslog))
      exit(1);

  /* initialize the agent net-snmp-libs */
  init_agent("testagent");

  /* ************************************* *
```

```
    * TODO:  REGISTER YOUR OWN MANAGED OBJECTS *
    * ************************************** */
  call_agent_registration_objects();

  /* read config-file testagent.conf and initialize agent */
  init_snmp("testagent");

  /* if we recevie a request to stop (kill -TERM or kill -INT) */
  keep_running = 1;
  signal(SIGTERM, stop server);
  signal(SIGINT, stop_server);
  /* you're main loop here... */
  while(keep_running) {
    /* to make runtime changes to the subagent visble to others */
    init_snmp("testagent");

    /* 0 == don't block --> 0 needs more CPU Power *
     * especially it always needs CPU Ressources   *
     * but the agent works a bit faster            */
    agent_check_and_process(1);
  }
  /* at shutdown time */
  snmp_shutdown("testagent");
}
```

Dieser Quellcode initialisiert und registriert einen Subagenten beim Masteragenten. Die dafür wesentlichen Quellcodestellen sind oben **FETT** unterlegt. Der Subagent kann sowohl als Prozess (wie in diesem Beispiel) als auch als Thread gestartet werden.
Die von uns entwickelte Library wird in diesem Beispiel in der Funktion „call_agent_registration_objects“ eingesetzt. In den folgenden Unterkapiteln wird erläutert, wie die verschiedenen Klassen (ScalarManger, TableManage, NotificationManager) verwendet werden um damit einen Systemmanagement-Subagent zu erstellen.

Der Aufbau der Bibliothek ist in die drei logische Bereiche

- ScalarManager
- TableManager (TableLayout)
- NotificationManager

gegliedert.

6.5 ScalarManager

Der Skalar Manager kümmert sich um die Registrierung und die Bearbeitung von allen Operationen auf Skalaren. Unter Skalaren verstehen die einzelnen Blätter im MIB Tree. Auf Basis der OID lässt sich, sofern die Konventionen eingehalten werden, einfach herausfinden ob

eine OID einen Skalar repräsentiert oder nicht, da all diese Identifier mit „**.0**“ enden. Es gibt leider auch Außnahmen, beispielsweise bei Tabellenindizes, bei denen die Unterscheidung nicht so einfach vorgenommen werden kann. Es ist die Aufgabe des Programmierers um die Einhaltung von Konventionen bemüht zu sein und selbst festzulegen ob ein Wert als Skalar repräsentiert werden soll.

SNMP Base Type	ASN.1 Type	SMIv1	SMIv2	C Type
INTEGER	INTEGER	Y	n	long
Integer32	INTEGER	N	y	long
Unsigned32	UINTEGER[2]	N	y	u_long
Gauge	UINTEGER[2]	Y	n	u_long
Gauge32	UINTEGER[2]	N	y	u_long
Counter	UINTEGER[3]	Y	n	u_long
Counter32	UINTEGER[3]	N	y	u_long
Counter64	INT64[4]	N	y	U64 (net-snmp)
TimeTicks	UINTEGER[5]	Y	y	u_long
OCTET STRING	OCTET STRING	Y	y	string
OBJECT IDENTIFIER	OBJECT IDENTIFIER	Y	y	SnmpOid (EasySnmp)
IpAddress	OCTET STRING (SIZE(4))	Y	y	u_long
NetworkAddress	OCTET STRING (SIZE(4))	Y	n	u_long (not used)
Opaque	OCTET STRING	Y	y	u_char
BITS	OCTET STRING	N	y	char

Tabelle 2: Verfügbare SNMP Datentypen (SMIv1/SMIv2)

Tabelle 2 zeigt die einzelnen Datentypen. Aufgrund der Verbesserungen, die in SMIv2 angebracht wurden, dient diese Version zugleich auch als Referenz für EasySnmp. Die Wrapperlibrary stellt die Datentypen Integer32, Unsigned32, Gauge32, Counter32, Counter64, TimeTicks, OCTET STRING, IpAddress und Opaque sowie BITS über die Klasse ScalarManager zur Verfügung. Der Benutzer kann mit einer Instanz dieser Klasse die einzelnen Managed Objects erstellen. Der Pool für die Managed Objects ist eindeutig pro Subagent. D.h. ein Programmierer kann mehrere Instanzen der Klasse „ScalarManager“ anlegen und diese für die Registrierung der Werte verwenden.

```
ScalarManager scalarMgr = ScalarManager();

/* shmid is known by shared memory administrator */
```

[2] UINTEGER: [APPLICATION 2] IMPLICIT INTEGER(0..4294967295)

[3] UINTEGER: [APPLICATION 1] IMPLICIT INTEGER(0..4294967295)

[4] INT64: [APPLICATION 6] IMPLICIT INTEGER(0..18446744073709551615)

[5] UINTEGER: [APPLICATION 3] IMPLICIT INTEGER(0..4294967295)

```
unsigned int *val = (int*)shmat(shmid, (char*)0,0))

SnmpOid *xoid = new SnmpOid(string("1.3.6.1.4.1.17100.2.1.1.0"));

/* register Gauge32 */
scalarMgr.registerGauge32(oid, val, false);
```

Dieses Snippet erzeugt einen SNMP-Skalar vom Typ Gauge32 und mit der OID 1.3.6.1.4.1.17100.2.1.1.0. Der erste Parameter der Funktion „registerGauge32“ gibt die OID des Managed Objects vollständig an. Es wird ein Zeiger auf eine SnmpOid Instanz verlangt. Die Klasse „SnmpOid“ ist Teil von EasySnmp und verwaltet je eine einzelne OID.
Wie schon erwähnt lassen sich alle SMIv2 Typen abbilden. Möglich wird dies durch die „register-Funktionen“:

- `registerCounter32`
- `registerInteger`
- `registerString`
- `registerGauge32`
- `registerTimeticks`
- `registerBits`
- `registerOpaque`
- `registerIpV4Address`
- `registerCounter64`
- `registerObjectId`

Die Library sieht jedoch nicht nur die Abbildung von Werten nach außen vor sondern dient auch dazu ganze Softwareapplikationen zu konfigurieren. Zu diesem Zweck gibt es eine zusätzliche Einrichtung, die es ermöglicht, auf Aktionen der Managementstationen proaktiv zu reagieren, nämlich die so genannten „CalledBackObjects“.
Am Beispiel des Typs Gauge32 würde das Quellcodestück so aussehen:

```
ScalarManager scalarMgr = ScalarManager();

/* shmid is known by shared memory administrator */
unsigned int *val = (int*)shmat(shmid, (char*)0,0))

SnmpOid *xoid = new SnmpOid(string("1.3.6.1.4.1.17100.2.1.1.0"));

/* register Gauge32 */
scalarMgr.registerGauge32(oid, val, false, new CalledBackObject());
```

6.5.1 CalledBackObjects

Ein CalledBackObject kann direkt bei der Registrierung der jeweiligen OID eingehängt werden und auf alle Aktionen reagieren, die für diese einzelne OID „von außen“ angestoßen werden. Dem Programmierer bleibt dabei die Möglichkeit, direkt mit CalledBackObjects zu arbeiten oder eine eigene Klasse zu schreiben, die von der Klasse CalledBackObject erbt und die Funktion **callback** für eigene Zwecke adaptiert. Die Funktion callback wird mit mehreren zur Verfügung stehenden Parameterlisten angeboten:

- **Übergabe eines Wertes** (normalerweise des zu setzenden Wertes) an die Funktion:

```
void CalledBackObject::callback(ParamObj &val, u_char mode);
```

Bei Verwendung dieser Callbackfunktion wird ein **ParamObj** übergeben. Ein ParamObj definiert eine einfache Struktur (C-Struct), die jeden beliebigen SMIv2 Datentyp abspeichern kann. Alle Werte bis auf den gesetzten sind NULL. Dies ermöglicht es dem Programmierer, durch einfache Verwendung von ParamObj, alle möglichen SNMP Werte zu übergeben und spart die Typprüfung und die „cast-Operation“.
Diese Funktion wird verwendet, einen Wert, der von einer Managementstation gesetzt werden soll, zu prüfen oder direkt auf diesen Eingabewert zu reagieren. Großen Komfort bietet dieser Ansatz bei der Steuerung von Applikationen durch verschiedene Aktionen der Managementstation.
Der Parameter **mode** wird verwendet um die Aktion, die angestoßen wurde, genau zu identifizieren. Bei den Modi unterscheidet EasySnmp zwischen:

`MODE_GET`	liefert den aktuellen Wert der jeweiligen OID zurück
`MODE_SET_RESERVE1`	prüft ob die Typen der zu setzenden Werte mit den Zieltypen übereinstimmen
`MODE_SET_RESERVE2`	Dieser Modul allokiert die tatsächlich benötigt Ressourcen (malloc, ...).
`MODE_SET_FREE`	Falls die SET-Operation bei einem der beteiligten Werte nicht vollständig abgeschlossen werden kann, werden alle Ressourcen wieder frei gegeben.
`MODE_SET_ACTION`	Der Modus „MODE_SET_ACTION“ erledigt die

	eigentliche SET-Operation, setzt also die Werte. Es werden gleichzeitig auch die alten Werte mitgeschrieben damit diese Aktion später wieder rückgänig gemacht werden könnte.
MODE_SET_UNDO	Der SET-UNDO Modus erlaubt es, nach einem SET-ACTION noch ein „Rollback“ zu machen, alle Ressourcen wieder freizugeben und die vorigen Werte zu verwenden.
MODE_SET_COMMIT	Dadurch wird eine SET-Operation endgültig abgeschlossen und alle alten Werte und Ressourcen gelöscht bzw. freigegeben. Die Änderungen sind permanent.
MODE_GETNEXT	Dieser Modus wird betreten falls ein GETNEXT Statement diese OID trifft. D.h. bei einem SNMPWALK wird jedes Managed Object über welches iteriert wird, zuerst mit MODE_GETNEXT betreten, dann wir der Wert der jeweiligen Variablen im Modus MODE_GET zurückgegeben und anschließend wird es wieder mit MODE_GETNEXT verlassen.

Tabelle 3: CalledBackObject Modi

Um diesen Mechanismus des atomaren Setzens weiter zu vertiefen zeigt Abbildung 26 eine SET-Operation:

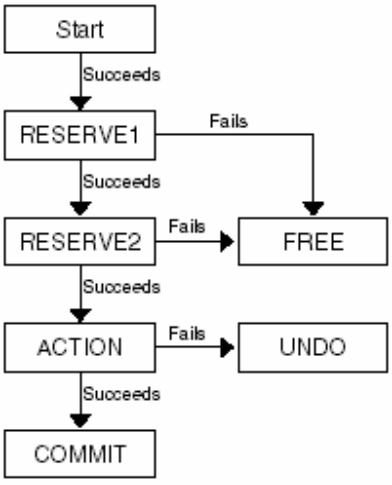

Abbildung 26: Ablauf einer SET-Operation

Es wird zu Beginn überprüft, ob alle Werte, die gesetzt werden sollen, auch tatsächlich den Zieltypen entsprechen (RESEVE1). Ist dies nicht der Fall, so wechselt der Agent in den Modus FREE, andernfalls in den Modus RESERVE2, in welchem die Ressourcen tatsächlich allokiert werden. Falls alle Ressourcen verfügbar sind, wird der Wert (die Werte) gesetzt (ACTION). Nach diesem Zustand gibt es nur mehr die Möglichkeit ein UNDO zu erzwingen und ein „Rollback“ der Werte vorzunehmen, oder die Bestätigung des Vorgangs mit COMMIT einzuleiten.

- **Übergabe eines Wertes** (normalerweise eines zu setzenden Wertes) an die Funktion:

```
void CalledBackObject::callback(SnmpOid &xoid,
                                ParamObj &val,
                                u_char mode);
```

Eine weitere Möglichkeit ist, nicht nur das ParamObj zu setzen, sondern auch eine OID anzugeben, beispielsweise um diese als TrapOID zu verwenden. Weitere Szenarien kann man sich einfach vorstellen. Hier ist die Kreativität des Programmierers gefragt.

- **Übergabe einer Watcherinfo:**

```
CalledBackObject::CalledBackObject(netsnmp_watcher_info *winfo,
                                   u_char mode);
```

Die „netsnmp_watcher_info” Struktur erlaubt es indirekt auf andere OIDs des Subagenten zuzugreifen und diese zu manipulieren. Falls man nicht direkt über den „Shared Memory“ Änderungen tätigen will, kann man so die typsichere Variante wählen.

- **Übergabe von beliebigen Werten:**

```
CalledBackObject::CalledBackObject(void *temp, u_char mode);
```

Diese Variante sollte nur verwendet werden, wenn alle anderen Varianten nicht passend sind. Der große Nachteil dieser Variante ist, dass man mit void-Zeigern (void*) die Typprüfung des Compilers verliert.

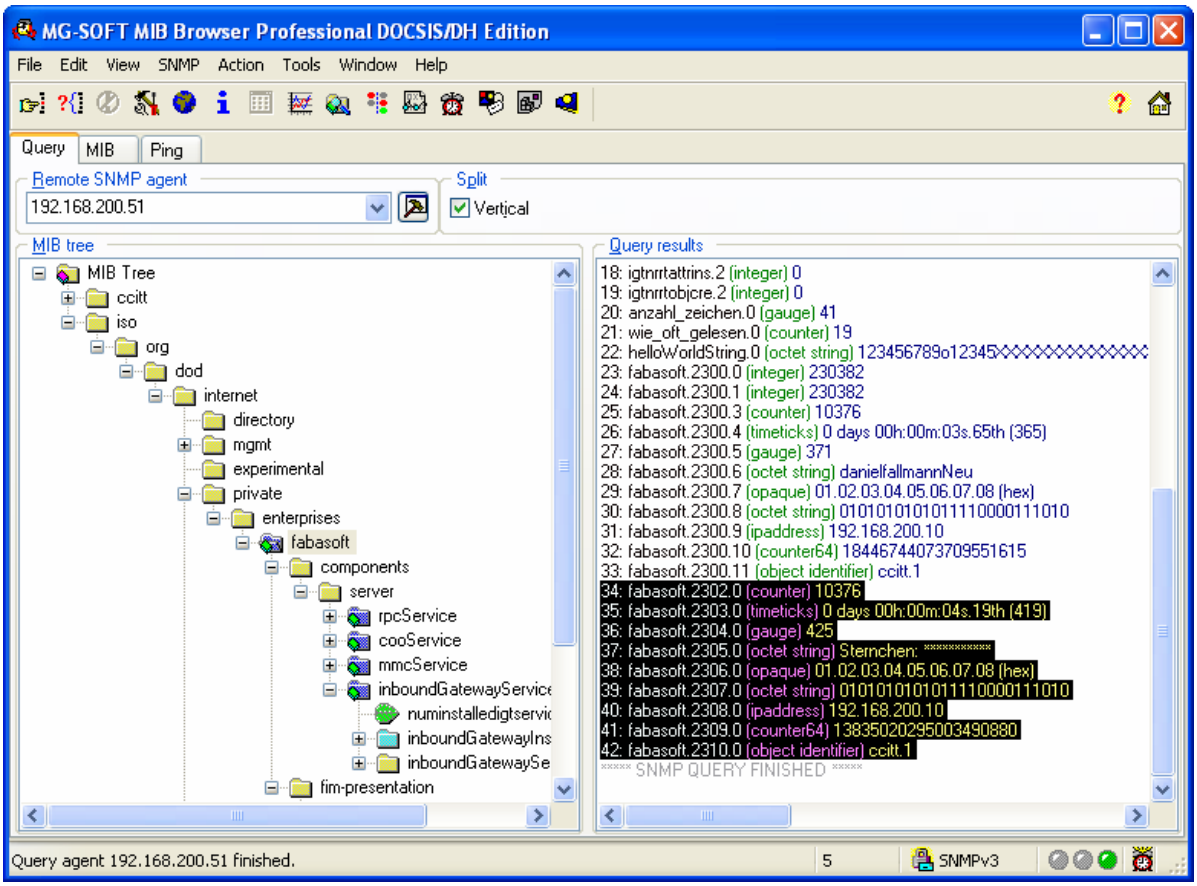

Abbildung 27: Registrierung von Skalaren

Abbildung 27 zeigt das Ergebnis eines Testtreibers, für die Registrierung, von alle Skalartypen. Mittels SNMP ist es nicht nur möglich einzelne Werte (Skalare) abzubilden, sondern auch Arrays (Tabellen) darzustellen. Im nächsten Kapitel wird erläutert wie man dies an Hand von EasySnmp macht.

6.6 TableManager

Bei SNMP unterscheidet man zwischen „Simple Tables“, diese haben einen ganzzahligen Index, und „Generic Tables“, also einer Tabelle die komplexe Indizes verwendet. Ein komplexer Index kann beispielsweise eine IP Adresse und eine Subnetzmaske sein, die benutzt werden um eine Zeile (einen Eintrag) eindeutig zu identifizieren. Da diese Spanne sehr weit reicht und es auch möglich sein muss geschachtelte (mehrdimensionale) Tabellen zu erstellen, definiert EasySnmp eine eigene Klasse „TableLayout“. Eine Instanz von „TableLayout“ kümmert sich um die Strukturdefinition der Tabelle.

„TableLayout“ implementiert zwei wesentliche Funktionen zum Erstellen von Tabellenkonfigurationen:

- **addIndex:** Durch die Parameter dieser Funktion steuert man das Aussehen des Indexes, der „beliebig“ komplex sein kann.
- **addTableStructure:** Es muss jede einzelne Spalte einzeln definiert warden. Folgende Werte sind dabei essentiell:
    - `unsigned int column:` Gibt die Spaltennummer an die dieser Eintrag näher definiert wird.
    - `int type:` Definiert den SNMP-Typ der Spalte.
    - `int writable:` „0“ gibt an dass eine Spalte nur gelesen werden darf und „1“ definiert lesen und schreiben.
    - `void *data:` Hier kann man einen Zeiger auf eine Funktion angeben die grundsätzlich immer beim Betreten dieser einen Spalte ausgeführt wird. Da die Klasse TableManager die Klasse ScalarManger verwendet ist es Möglich den angebotenen Callbackmechanismus auch für Tabellen zu nutzen.
    - `size_t data_len:` Legt den zu reservierenden Speicherplatzbedarf fest.

```
/* create new table instance */
TableLayout l = TableLayout("tabletest", true);


/* add a simple index (just an integer number) */
l.addIndex(ASN_INTEGER);

/* add our table structure */
l.addTableStructure( 2, ASN_OCTET_STR, 1, NULL, 0,
                        3, ASN_INTEGER, 1, NULL, 0,
                        4, ASN_INTEGER, 1, NULL, 0,
                        5, ASN_INTEGER, 1, NULL, 0,
                        6, ASN_INTEGER, 1, NULL, 0,
                        7, ASN_INTEGER, 1, NULL, 0,
                        8, ASN_INTEGER, 1, NULL, 0,
                        9, ASN_INTEGER, 1, NULL, 0,
                       10, ASN_INTEGER, 1, NULL, 0,
                       11, ASN_INTEGER, 1, NULL, 0,
                       12, ASN_INTEGER, 1, NULL, 0,
                       13, ASN_INTEGER, 1, NULL, 0,
                       14, ASN_INTEGER, 1, NULL, 0,
                       15, ASN_INTEGER, 1, NULL, 0,
                       16, ASN_INTEGER, 1, NULL, 0,
                       17, ASN_INTEGER, 1, NULL, 0,
                        0);
```

An diesem Beispiel sieht man, wie jede einzelne Spalte aufgebaut ist. Die erste Spalte beinhaltet den Index der Tabelle (als ASN_INTEGER). Die zweite Spalte ist vom Typ OCTET_STRING und die restlichen 15 Spalten repräsentieren je einen ganzzahligen Wert (ASN_INTEGER). Abschließend zeigt die schließende „0“ an, dass die Struktur fertig definiert ist.
Übrigens, beim Konstruktor jedes TableLayouts kann man angeben (mit dem zweiten Parameter), ob es möglich ist, eine neue Zeile zur Tabelle, zur Laufzeit, von einer Managementstation aus hinzuzufügen (true oder false).

Es ist nötig noch eine „TableManager“-Instanz zu erzeugen, mit der man die gerade beschriebene Tabelle verwaltet.

```
SnmpOid *tableOid = new SnmpOid("1.3.6.1.4.1.17100.1.1.12.2");

/* TableManager(SnmpOid *oidObj, TableLayout *layout, bool writeable);     *
 * writeable ... gibt an ob die Tabelle nur gelesen warden darf oder nicht */
TableManager table = TableManager(tableOid, &l, true);
table.addRow(0,”Test”,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15);
```

Dieses Codesnippet generiert auf Basis des „TableLayout“ eine Tabelle und fügt eine einzelne Zeile hinzu. Die Tabelle ist damit fertig definiert und kann verwendet werden.

Aufgund der Freiheitsgrade der einzelnen Paramerter, die als „va_list“ festgelegt sind, wird es möglich sehr komplexe Tabellen einfach zu erzeugen.

6.7 NotificationManager

Der „NotificationManager” kümmert sich um das Versenden von Traps.

1. Es besteht die Möglichkeit Traps direkt zu senden, d.h. die jeweilige Applikation kümmert sich darum und generiert beispielsweise parallel zu Syslogs auch Traps. Der Subagent leitet in diesem Fall den Trap an den Masteragenten weiter. Das „Dispatcher“ Subsystem des Masteragenten kümmert sich um das Weiterleiten des Traps an die definierten Zeilhosts (siehe dazu das Kapitel „SNMP aus Administratorsicht“).

```
NotificationManager notificationMgr;

SnmpOid *xoid = new SnmpOid("1.3.6.1.4.1.17100.3.3.1");
int value = 1000;

notificationMgr.sendTrapV1(xoid, &value);
notificationMgr.sendTrapV2(xoid, &value);
```

   In diesem Beispiel wird aktiv ein Trap mit der OID **xoid** und dem daran gebundenen Wert **value** ausgesandt. Die Library generiert daraus einen Standardtrap, der mindestens drei Variable Bindings beinhaltet:

   - sysUpTime
   - TrapOID
   - angehängte VarBindList (mit dieser können beliebige andere, möglicherweise betroffene, Variable Bindings mitgesandt werden)

   Wie weiter oben in dieser Arbeit bereits erläutert, sind die Trap-Formate in Version 2 und 3 anders als in Version 1 von SNMP, daher wurde dies im Beispiel unterschieden. Es werden parallel TrapV1 und TrapV2 Pakete an alle registrierten Empfänger gesandt.

2. Eine weitere Möglichkeit ist, im CalledBackObject das Senden eines Traps an einen Modus zu koppeln. Damit kann man auf alle Aktionen die eine Managementstation anstoßen kann reagieren. Der Vorteil besteht bei dieser Technik darin, dass kein Pollingmechanismus einen konkreten Wert überwachen muss, jedoch werden diese Aktionen nicht angestoßen wenn

Agenteninten Werte gesetzt werden, d.h. konkret Fabasoft Components in den Shared Memory schreibt.

3. Die dritte Möglichkeit kann entweder aktiv, durch den Programmierer, gehandhabt werden, oder durch den eingebauten Alarm Mechanismus. Bei dieser Verwendungsart wird eine Funktion programmiert, die in regelmäßigen Abständen (relativ zur SysUpTime des Agenten) überprüft, ob die in der gerufenen Funktion defninierten Schranken (Bedingungen) eingehalten werden. Man kann innerhalb dieser Funktion wiederum dafür sorgen, dass Traps gesendet werden, falls eine Bedingung nicht erfüllt ist.

```
NotificationManager notificationMgr;

/* TrapOID */
SnmpOid *xoid = new SnmpOid("1.3.6.1.4.1.17100.2.1.1.30")

/* you have to implement function callback first          *
 * winfo is a pointer to a netsnmp watcher info structure */
CalledBackObject *callbackObj = new CalledBackObject(winfo, mode);

/* watch all 60 seconds            *
 * SA_REPEAT ... continue to watch *
 * call function callback of callbackObj */
notificationMgr.registerAlarm(xoid, 60, SA_REPEAT, callbackObj);
```

Das CalledBackObject kümmert sich um den zu überprüfenden Wert. Der Zugriff wird mit Hilfe der netsnmp_watcher_info ermöglicht.

Es gibt nicht nur die Möglichkeit Werte kontinuierlich zu Überwachen. Man kann auch vereinzelt Werte nur eine gewisse Anzahl von Iterationen überwachen. Dazu schreibt man anstelle von „SA_REPEAT“ die gewünschte Anzahl der Durchläufe in das Programm.

Mit diesen drei Mechanismen lassen sich sehr komplexe Überwachungssysteme bauen.

6.8 SNMP „Hello World“ mit EasySnmp

Diese Beispiel soll Lust auf mehr machen und prägnant zeigen wofür EasySnmp sehr einfach verwendet werden kann. Die folgende MIB ist bewußt einfach gehalten:

```
FIM-HELLO-WORLD-MIB DEFINITIONS ::=
BEGIN
  IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Gauge,
    Counter32           FROM SNMPv2-SMI
    MODULE-COMPLIANCE,
    OBJECT-GROUP        FROM SNMPv2-CONF
    DisplayString       FROM SNMPv2-TC
    fim-presentation    FROM FABASOFT-FIM-MIB;
  hello-world MODULE-IDENTITY
    LAST-UPDATED "200406201000Z"
    ORGANIZATION "Fabasoft R&D Software GmbH & Co KG"
    CONTACT-INFO
      "Fabasoft R&D Software GmbH & Co KG
       Postal: Fabasoft R&D Software GmbH & Co KG
               Honauerstrasse 4, 4020 Linz
               AUSTRIA
       Tel: +43 732 606162
       Fax: +43 732 606162 609
       E-mail: support@fabasoft.com"
     DESCRIPTION
       "Provides statistic information for COO-service instances."
    ::= { fim-presentation 1 }
  anzahl_zeichen OBJECT-TYPE
    SYNTAX Gauge32
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
      "Mit diesem Wert können Sie angeben wie viele Zeichen im DisplayString
       angezeigt werden."
    ::= { hello-world 1 }

  wie_oft_gelesen OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "Dieser Wert gibt an wie oft der DisplayString bereits gelesen wurde."
    ::= { hello-world 2 }
  helloWorldString OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
      "SNMP Hello World String."
    ::= { hello-world 3 }

  warningMsg NOTIFICATION-TYPE
  OBJECTS {anzahl_zeichen}
```

```
  STATUS current
  DESCRIPTION
      "Too many characters."
  REFERENCE
      ""
  ::= { hello-world 10 }

--
-- Conformance information
--
  helloWorldConformance OBJECT IDENTIFIER ::= { hello-world 100 }
  helloWorldCompliances OBJECT IDENTIFIER ::= { helloWorldConformance 1 }
  helloWorldCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
      "Hello World Conformance."
    MODULE FIM-HELLO-WORLD-MIB
    MANDATORY-GROUPS { helloWorldGroup }
    ::= { helloWorldCompliances 1 }
  cooServiceGroups OBJECT IDENTIFIER ::= { cooServiceConformance 2 }
  cooServiceGroup OBJECT-GROUP
    OBJECTS {
            anzahl_zeichen,
            wie_oft_gelesen,
            helloWorldString
    }
    STATUS current
    DESCRIPTION
      "COO-service conformance object group."
    ::= { helloWorldGroups 1 }
END
```

Diese MIB definiert die drei Werte

- „anzahl_zeichen“ (Gauge32),
- „wie_oft_gelesen“ (Counter32)und
- „helloWorldString”.

Zusätzlich wird noch der Trap „warningMsg“ spezifiziert. Das HelloWorldSNMP Beispiel erlaubt berechtigten Benutzern die Variable „anzahl_zeichen“ zu ändern. Durch die Änderung dieser Variablen wird der „helloWorldString“ von vorne beginnend anzahl_zeichen aufgedeckt. Der String ist 40 Zeichen lang. Wenn der Benutzer einen Fehler macht und beispielsweise 41 Zeichen aufdecken möchte so wird keine Aktion angestoßen, sondern der alte Wert (anzahl_zeichen) beibehalten und eine Fehlermeldung, in Form eines Traps an den Benutzer gesandt. Die Counter32 Variable „wie_oft_gelesen“ spielt dabei eine passive Rolle und gibt an wie oft der Wert „anzahl Zeichen“ gelesen wurde. Das Szenario ist relativ einfach zu verstehen. Die wesentlichen Teile des Quellcodes sind die Registrierung der einzelnen Werte und die richtige Verwendung des CalledBackObjects.

Registrierung der einzelnen Skalare:

```
/*** registerString ***/
string *fim_val3 = new string("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx");
SnmpOid *oid_fim3 = new SnmpOid(string("1.3.6.1.4.1.17100.2.1.3.0"));
netsnmp_watcher_info *winfo = scalarMgr.registerString(oid_fim3,
                                                       40,
                                                       true,
                                                       fim_val3,
                                                       NULL);
/*** registerCounter ***/
ulong *fim_val2;
*fim_val2 = 0;
SnmpOid *oid_fim2 = new SnmpOid(string("1.3.6.1.4.1.17100.2.1.2.0"));
scalarMgr.registerCounter32(oid_fim2 ,fim_val2);

/*** regsterGauge ***/
unsigned int fim_val1 = 0;
SnmpOid *oid_fim1 = new SnmpOid(string("1.3.6.1.4.1.17100.2.1.1.0"));
scalarMgr.registerGauge32(oid_fim1,
                          &fim val1,
                          true,
                          new CalledBackObject(winfo, fim_val2));
```

Programmierung der „callback"-Funktion (Auszug aus Original):

```
void CalledBackObject::callback(netsnmp_watcher_info *wstore,
                                u_char mode,
                                int *value) {
  if (mode == MODE_SET_ACTION) {
    if (*x>=0 && *x<=wstore->max_size) {
      /* String in temp aufbauen */
      wstore->data = (void*)temp;
    } else {
        NotificationManager notificationMgr;
        /* fill trap structure notification_vars with information */
        notificationMgr.sendTrapV2(new SnmpOid("1.3.6.1.4.1.17100.2.1.2.10"),
                                   value);
    }
  }
}
```

Sofern man das Programm als eigenständigen Prozess laufen lassen will, kann man sich den in diesem Kapitel erläuterten Sourcecode als Vorlage nehmen.

An ein paar Screenshots des Managementsystems MG-SOFT Professional MIB Browser wollen wir die Funktion dieses Programms deutlich machen.

1. Nach dem Start des Subagenten und erfolgreicher Registrierung beim Masteragenten werden die Daten initialisiert auf 0 angezeigt.

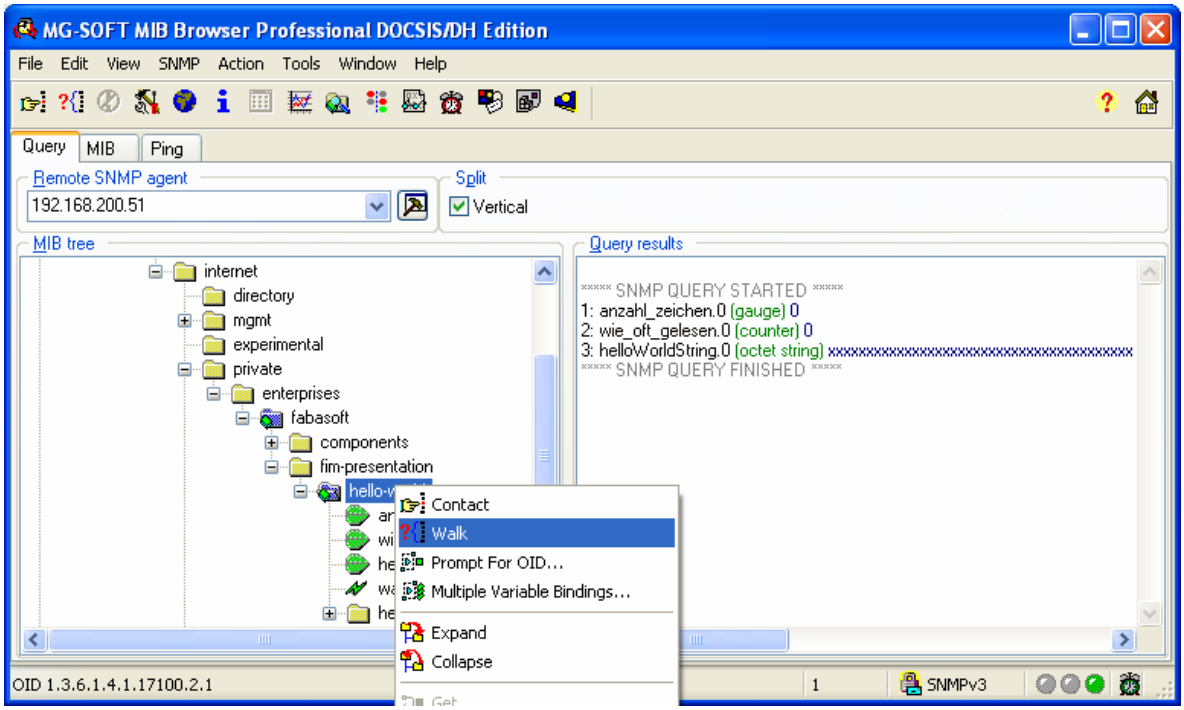

Abbildung 28: SNMP Hello World (Initialize)

2. Nachdem der Wert der Variable „anzahl_zeichen“ auf 5 gesetzt wurde, sieht man die ersten 5 Zeichen des „helloWorldStrings“ aufgedeckt.

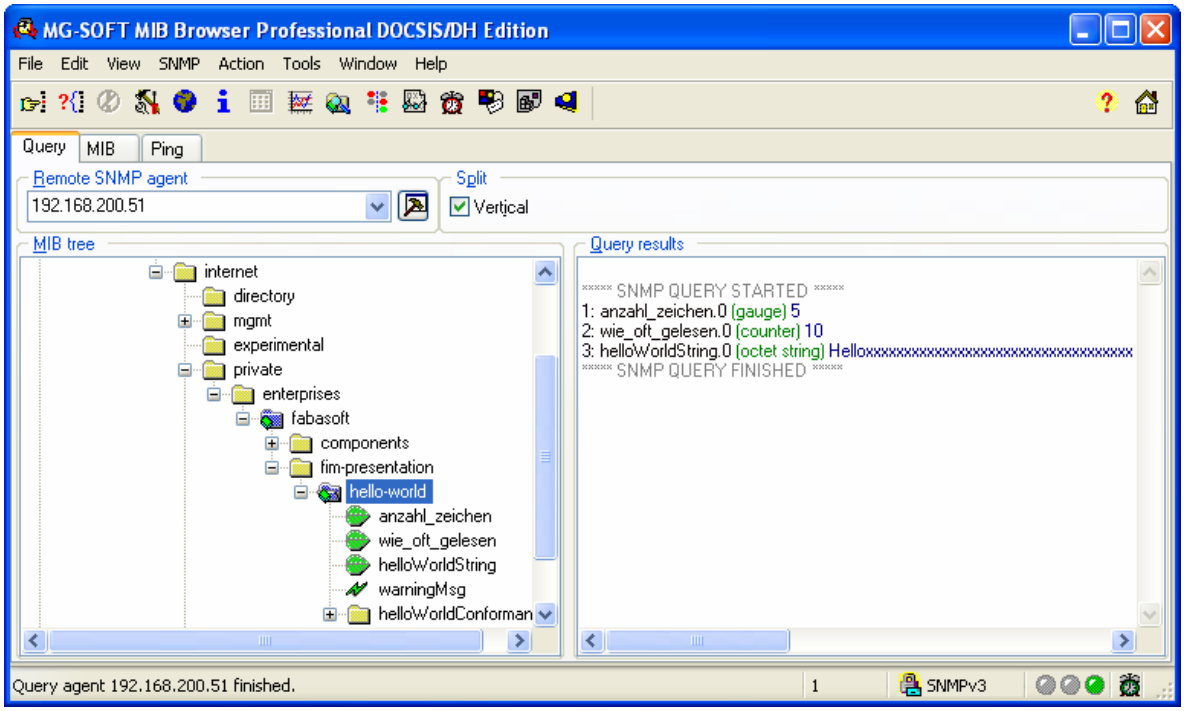

Abbildung 29: SNMP Hello World (“5”)

3. Nun deckt man alle 40 Zeichen vollständig auf.

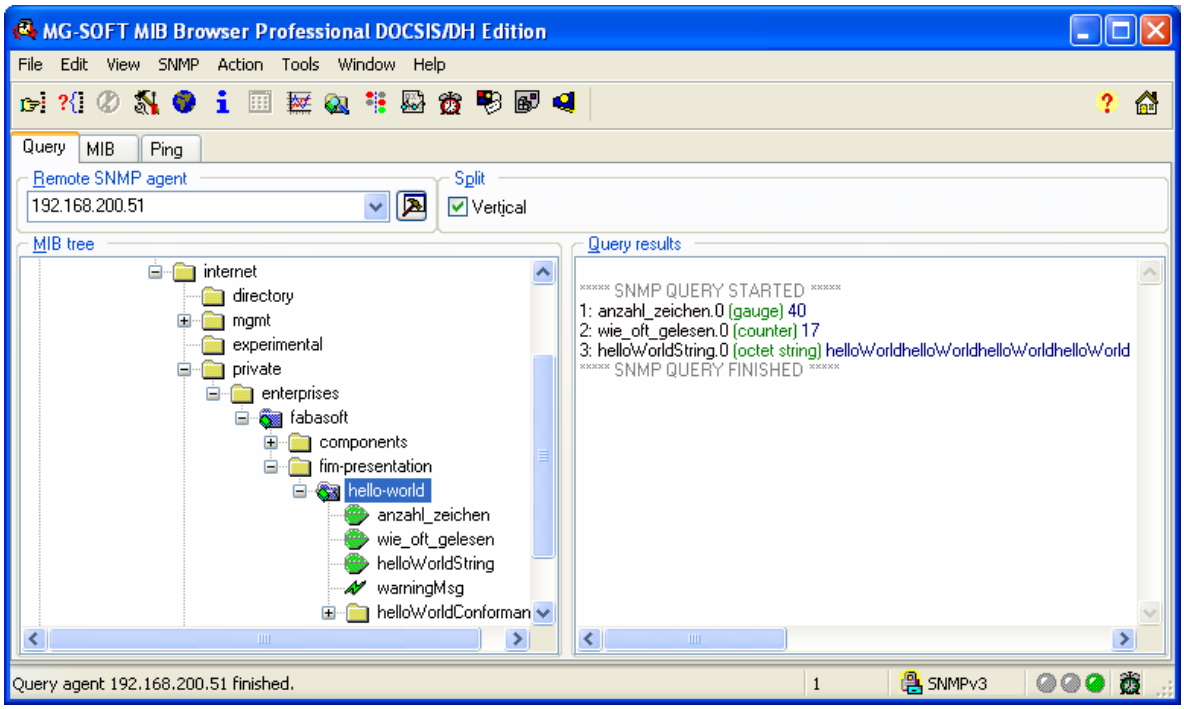

Abbildung 30: SNMP Hello World („40“)

4. Auch Fehler passieren immer wieder. Beim zuweisen der Zahl 41 an die Variable anzahl_zeichen wird ein Trap gesendet, der den Fehler beschreibt.

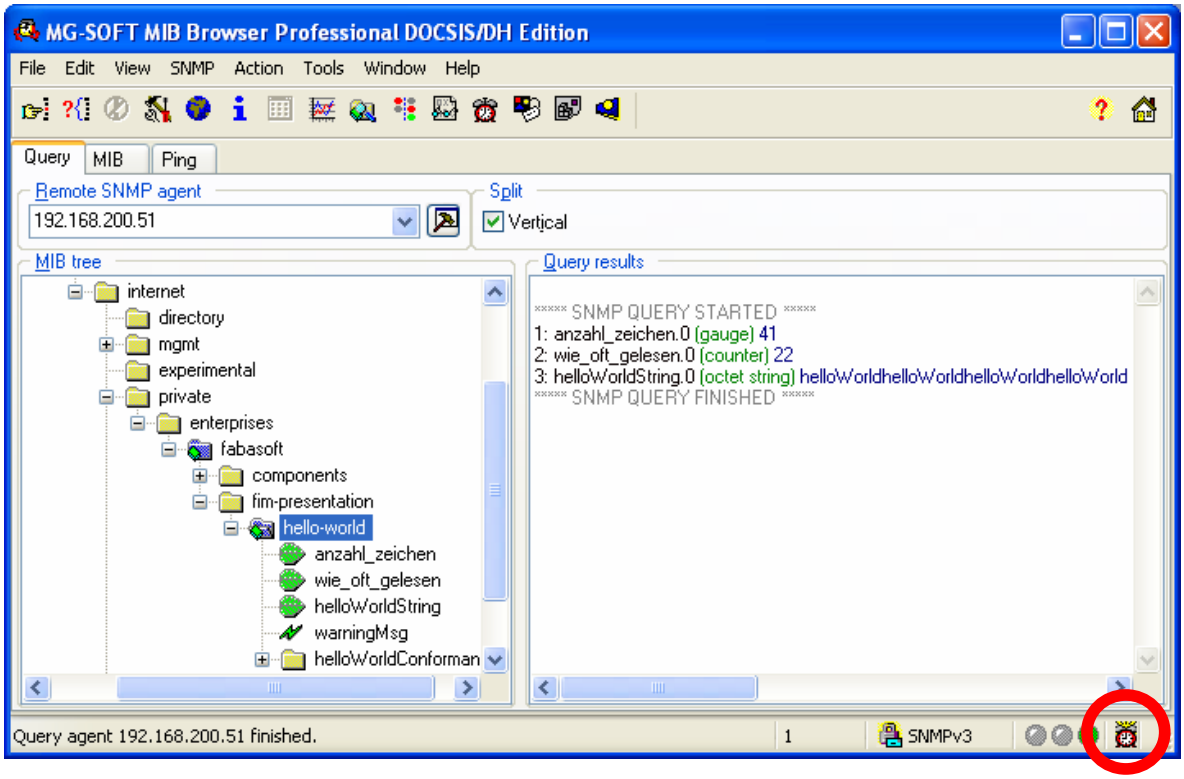

Abbildung 31: Fehler beim Zuweisen von "41"

Der folgende Screenshot zeigt die Fehlermeldung, die vom Agent an die Managementstation gesandt wurde.

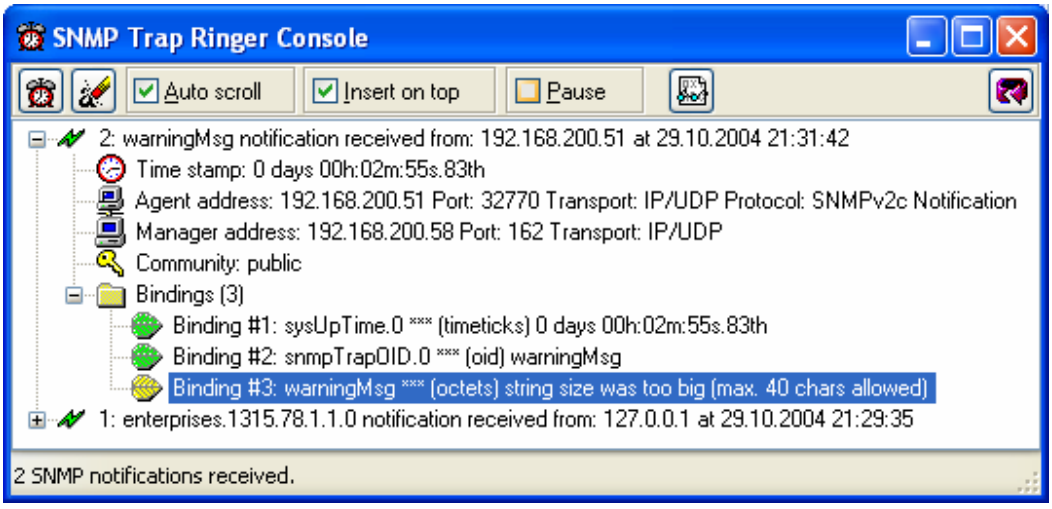

Abbildung 32: Fehlermeldung als SNMPv2c Trap

7 Arbeit im Open Source Bereich

I still think we should just switch to a 100,000
second day and to hell with the sun.
— Peter da Silva

In diesem Kapitel wird ein kurzer Überblick über die Tätigkeit im Open Source Umfeld, im Rahmen dieser Arbeit gegeben. Vor ziemlich genau einem Jahr habe ich begonnen mit der freien Bibliothek Net-SNMP zu programmieren. Zu Beginn der Einarbeitungsphase waren die unzähligen Dateien, die diese Bibliothek umfasst, für mich wie ein Buch mit sieben Siegeln. Wie in sehr vielen „offenen Projekten“, leidet unter der Freude der Entwickler am Programmieren meistens die Dokumentation. Auch in diesem Fall fand man Ende 2003 (bzw. bis heute) noch kaum eine „vernünftige“ Dokumentation, die man wie ein Buch lesen konnte.

Guter Rat war in dieser Situation teuer. Was konnte man nun tun, um möglichst schnell die Grundstrukturen der Bibliothek zu verstehen? Nach längerem hin und her und einigen Anfragen in der Net-SNMP Mailingliste, wurde der Entschluss gefasst, dass es zielführend ist, sich in den bestehenden Source Code einzuarbeiten. Zu diesem Zweck wurde die aufstrebende Entwicklungsplattform Eclipse (mit CDT Plugin, für C/C++) benutzt, um dem Projekt Herr zu werden. Es zeigte sich rasch, dass es nicht möglich war, zufriedenstellend schnell eine Textsuche über nicht indizierte Dateien zu machen. So wurde kurzerhand der bestehenden Quellcode indiziert, um dieses Probleme zu beheben. Dafür verwendet man üblicherweise ein Ctags Programm. In diesem konkreten Fall „cscope“ und das dazugehörige Frontendprogramm Cbrowser, um möglichst komfortabel arbeiten zu können. Es war, nach der Einarbeitungsphase, sehr schnell möglich, die wesentlichen Strukturen und ihre Zusammenhänge zu erkennen.
Da die Net-SNMP Library teilweise sehr komplex und verschachtelt ist, wurde zusätzlich ein freies Werkzeug namens „cflow“, das die Zusammenhänge graphisch darstellen kann, verwendet. Der Vorteil dabei ist, dass diese Werkzeuge alle Informationen direkt aus dem bestehenden Quellcode extrahieren können.

Während der Entwicklungsphase gab es anfänglich Schwierigkeiten, Designentscheidungen zu fällen, da das nötige Basiswissen für die Bibliothek, auf die aufgebaut wird, noch nicht vorhanden war. Dieses Problem wurde aber von Tag zu Tag und von Woche zu Woche

abgeschwächt, da mit der Zeit ein sehr guter Überblick erarbeitet wurde. Aufgrund der Situation in der das Projekt begonnen wurde, war Zeit nicht der kritische Faktor und so machte das Untersuchen und die Einarbeitung in die neue Thematik sehr viel Spaß. Ich möchte allerdings auch erwähnen, dass es aus Entwicklersicht doch eine der aufwändigsten Projektphasen war. Dieser Umstand wurde auch noch dadurch verschärft, dass man sich bei Fragen zwar an eine toll administrierte Mailingliste wenden konnte, man jedoch aufgrund der großen Anzahl an Anfragen immer wieder mehrere Tage warten musste, bis bei Problemen Hilfestellungen gegeben wurde. Ich möchte mich an dieser Stelle auch bei den Net-SNMP Kernentwicklern Wes Hardacker, David Perkins und Mike Silfak bedanken, die mich immer wieder unentgeltlich mit guten Ratschlägen unterstützt haben. Anfänglich gab ein ein paar kleiner Kommunikationsprobleme, die sicherlich auch durch die unterschiedliche Muttersprache und den unterschiedlichen Wissenstand hervorgerufen wurden.

Was kann man nun zusammenfassend über diese Arbeit im Zuge der Diplomarbeit sagen?
Erwähnenswert ist, dass ich mich im letzten Jahr, im Open Source Systemmanagementumfeld sehr stark persönlich weiterentwickelt habe. Die Arbeit war nicht immer einfach und so habe ich sehr viele Nächte im Labor an der Universität gearbeitet. Es gibt sehr viele bestehende, freie Softwareprojekte, die es wert sind, einen genaueren Blick hinter die Kulissen zu werfen, Net-SNMP ist sicherlich eines davon.

8 Ausblick

8.1 Entwicklungspotential

Die in dieser Arbeit entstandene Bibliothek ermöglicht die einfache Verwendung von wesnetlichen SNMP Funktionen auf sehr einfache Weise. Es wird dadurch möglich in sehr kurzer Zeit Daten per SNMP zur Verfügung zu stellen und neue Applikationen mit Managementfunktionalität zu versehen. Aufgrund des riesigen Umfangs, der Systemmanagementthematik, kann man sich sehr komplexe Szenarien überlegen die vom Einsatz eines adaptierten Produkts profitieren würden. Vor allem unter Linux steckt die „Manageability“ noch in den Kinderschuhen. Die vor knapp einem Jahr ins Leben gerufene Fabalabs Software GmbH, die mit innovativen Wissensträgern, wie der Johannes Kepler Universität, kooperiert, versucht dieser Aufgabenstellung gerecht zu werden.

Die letzten Jahre hat man immer stärker gemerkt, dass vor allem große Konzerne darauf bedacht sind die Serversysteme einfach wartbar und konfigurierbar zu machen. Ein wesentlicher Grund dafür ist aus meiner Sicht, dass die TOC (total cost of ownership) immer in den Hinterköpfen der Manager sind und kein Anbieter derzeit eine zufriedenstellende Allgemeinlösung für das Systemmangement von heterogenen Systemem parat hat.

9 Tabellenverzeichnis

10 Abbildungsverzeichnis

11 Quellen- und Literaturverzeichnis

[Ban01] Bansal Ashish: How to write dynamically loadable libraries, IBM; http://www-106.ibm.com/developerworks/library/l-shobj/index.html

[Com98] IEEE Communications Surveys; **http://www.comsoc.org/pubs/surveys**; Fourth Quarter 1998; Vol. 1 No.

[Dea01] Dean, Jeffrey: LPI LINUX Certification in a nutshell; O'Reilly 2001; ISBN 1-56592-748-6

[Ema04] EMANATE product; http://www.snmp.com/products/emanate.html; 2004

[Gol00] Goll u.a.: Netzmanagement – Einführung in das Management verteilter Systeme; http://www.stz-softwaretechnik.de/download/skripte/nm_ss2000.pdf; 2000

[Lei93] Leinwand, Allan; Karen, Fang: Network Management A Practical Perspective; Addison – Wesley Publishing Company 1993; ISBN 0 -201-52771-5

[Mar95] Marshall, T. Rose; McCloghrie, Keith: How to Manage your Network Using SNMP; PTR Prentice Hall 1995; ISBN 0-13-141517-4

[Mcc96] McCloghrie Keith (Cisco Systems) u.a.: The Simple Times; Volume 4, Number 2; http://www.simple-times.org; April 1996

[Mgs04] MG-SOFT MIB Browser Professional Version 9.0, http://www.mg-soft.si/mgMibBrowserPE.html, 2004.

[Mil96] Miller, Mark A, P.E.: Managing Internetworks with SNMP; Fourth Edition; M&T Books 1996, ISBN 1-55851-304-3

[Net04] The NET-SNMP home page; http://www.net-snmp.org; 2004

[Per97] Perkins, David; McGinnis, Evan: Understanding SNMP MIBs; PTR Prentice Hall 1997; ISBN 0-12-437708-7

[Rob00] Robins, Arnold: Unix in a nutshell; 2. Auflage; O'Reilly 2000; ISBN 3-897-21193-9

[Sag04] Solaris System Management Guide, Administration Guide; Sun Microsystems, Inc., Santa Clara; 2004

[Sch03] Schönwälder Jürgen, u.a.: Netzwerk Management; http://luca.ntop.org/Teaching/nm2004.pdf; 2003

[Sch04] Scheiderer, Jürgen: mitp – Trainingsbuch SUSE Linux Sicherheit; 2. Auflage, mitp 2004; ISBN 3-8266-1458-5

[Sch97] Schönwälder Jürgen (TU Braunschweig) u.a.: The Simple Times; Volume 5, Number 1; http://www.simple-times.org; Dezember 1997

[Sdg04] Solaris System Management Guide, Developer's Guide; Sun Microsystems, Inc., Santa Clara; 2004

[Sta03] Stallings, William: SNMP, SNMPv2, SNMPv3 and RMON1 and 2;Third Edition; Addison – Wesley Publishing Company 2003; ISBN 0-201-48534-6

[Ste90] Stevens Richard W.: UNIX Network Programming; Prentice Hall Software Series; 1990; ISBN 0-13-949876-1

[Stu03] Stump Michael: Securing SNMP: A Look at Net-SNMP (SNMPv3); SANS Institute; 2003

[Str00] Stroustrup, Bjarne: The C++ Programming Language; Special Edition; Addison – Wesley Publishing Company 2000; ISBN 0-201-70073-5

[Tan03] Tanenbaum, Andrew S.: Computer Networks; Fourth Edition; Pearson Education International 2003; ISBN 0-13-038488-7

Bemerkung: RFCs werden in dieser Arbeit, an den jeweiligen Stellen, explizit mit ihrer eindeutigen Nummer referenziert.

12 Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommen Stellen als solche kenntlich gemacht habe.

Linz, im Dezember 2004

_____

(Daniel Fallmann)

13 Lebenslauf

Persönliche Information

- **Geburtsjahr:** 1982
- **Geburtsort:** Linz
- **Staatsangehörigkeit:** Österreich
- **Name:** Daniel Fallmann
- **Wohnort:** Eidenberg (OÖ)
- **Eltern:** Helmut Fallmann
  Erna Fallmann (geb. Leonhartsberger)

Ausblidung

- **1988-1992:** Volksschule Untergeng
- **1992-2000:** Georg von Peuerbach Gymnasium (Linz)
  - Realzweig
  - Sepzialisierung auf Informatik und Mathematik (mit Mathematik Wahlpflichtfach)
  - alle 8 Jahre mit Auszeichnung bestanden
- **Matura 2000:** mit Auszeichnung bestanden
- **2000:** Führerschein der Klasse A + B
- **seit 2000:** Studium der Informatik an der Johannes Kepler Universität
- **2002:** abgeschlossene erste Diplomprüfung

Berufserfahrung

- **2001: Firma Fabasoft R&D:** 3 Monate Ferialpraxis
  - Software Engineer
  - Schwerpunkt: Portable PCs
- **2002: Firma Fabasoft R&D:** 4 Monate Ferialpraxis
  - Software Engineer
  - Konzeption und Entwicklung eines Prototypen für ein webbasiertes eLearning System (Fabasoft WBT) + Einbettung des Microsoft Agents
- **2003: Firma Fabasoft R&D:** 3 Monate Ferialpraxis
- **2004:** seit Juni mit dem Aufbau der Firma Fabalabs Software GmbH betraut (seit August Leiter der Fabalabs Software GmbH)

Weiterbildung

- **1995:** im Alter von 13 Jahren absolvierte ich bereits meine ersten Fabasoft Zertifizierungen unter Windows NT
- **2004:** LIMAK Ausbildung
    - Präsentationstechnik (bei DDr. Steinecker, Vorstandsmitglied der Energie AG)
    - Begonnene Ausbildung zum zertifizierten Projektmanager (voraussichtlicher Abschluß im März 2005)

14 Anhang

14.1 Entstandene Fabasoft MIBs

Die im folgenden textuell angeführten Fabasoft MIBs sind im Rahmen der Diplomarbeit auf Basis von vorgegebenen C-Strukturen für die verschiedenen Fabasoft Services (MMCService, COOService, InboundGatewayService, RPCService) erarbeitet worden.

Da MIBs regelmäßg auf neue Produktversionen angepasst werden müssen, darf man die hier angegebenen nur als eine Vorabversion der von Fabasoft angebotenen MIBs auffassen. Die MIBs wurden alle unter der, bei IANA (Internet Assigned Numbers Authority) registrierten, „Enterprise-OID“ 17100 eingehängt. Generell gibt es für Unternmehmen eine eigene vordefinierte Prefix ID (1.3.6.1.4.1). Den MIB „Domäne“ unter der zugewiesenen Nummer verwaltet jedes Unternehmen autonom.

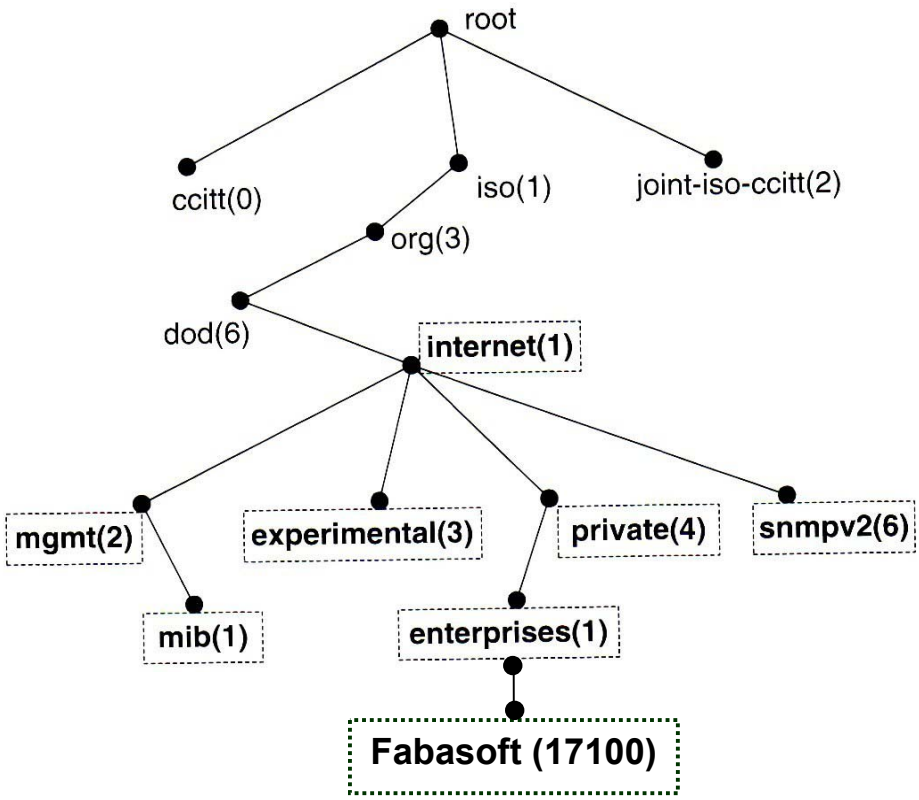

Abbildung 33: MIB-Tree Fabasoft, Quelle: adaptiert von [Per97]

Übersicht der abgebildeten Counter:

```
-- assigned by IANA (1.3.6.4.1.17100)
-- enterprise number 17100 (so whole number is 1.3.6.4.1.17100)
-- iso(1).org(3).dod(6).internet(4).enterprise(1).fabasoft(17100)
  ...
   +--internet(4)
   |
   +--enterprises(1)
      |
      +--fabasoft(17100)
         +--products(1)
            +--components(1)
               +--server(1)
               |  +--sysinfo(1)
               |  +--rpc(2)
               |  |  +-- -R-- Counter32   nrrpccalls(1)
               |  |  +-- -R-- Counter32   nrrpctrans(2)
               |  |  +-- -R-- Counter32   nrrpcthreads(3)
               |  |  +-- -R-- Counter32   nrrphysconns(4)
               |  |  +-- -R-- Counter32   nrrecvbytes(5)
               |  |  +-- -R-- Counter32   nrsendbytes(6)
               |  |  +-- -R-- Counter32   nrtotalbytes(7)
               |  |  +-- -R-- Counter32   nrrecvpackets(8)
               |  |  +-- -R-- Counter32   nrsendpackets(9)
               |  |  +-- -R-- Counter32   nrtotalpackets(10)
               |  |
               |  +--cooService(10)
               |  |  +-- -R-- Counter32   nrobjlock(1)
               |  |  +-- -R-- Counter32   nrobjunlock(2)
               |  |  +-- -R-- Counter32   nrquery(3)
               |  |  +-- -R-- Counter32   nrobjread(4)
               |  |  +-- -R-- Counter32   nrverread(5)
               |  |  +-- -R-- Counter32   nrtrans(6)
               |  |  +-- -R-- Counter32   nrverdel(7)
               |  |  +-- -R-- Counter32   nrverins(8)
               |  |  +-- -R-- Counter32   nrobjdel(9)
               |  |  +-- -R-- Counter32   nrattrins(10)
               |  |  +-- -R-- Counter32   nrobjcre(11)
               |  |  +-- -R-- Counter32   nrrtobjlock(12)
               |  |  +-- -R-- Counter32   nrrtobjunlock(13)
               |  |  +-- -R-- Counter32   nrrtquery(14)
               |  |  +-- -R-- Counter32   nrrtobjread(15)
               |  |  +-- -R-- Counter32   nrrtverread(16)
               |  |  +-- -R-- Counter32   nrrttrans(17)
```

```
|  |  +-- -R-- Counter32   nrrtverdel(18)
|  |  +-- -R-- Counter32   nrrtverins(19)
|  |  +-- -R-- Counter32   nrrtobjdel(20)
|  |  +-- -R-- Counter32   nrrtattrins(21)
|  |  +-- -R-- Counter32   nrrtobjcre(22)
|  |  +-- -R-- Counter32   rtcachehit(23)
|  |  +-- -R-- Counter32   rtcacheused(24)
|  |  +-- -R-- Counter32   nrcacheobj(25)
|  |  +-- -R-- Counter32   nrcacheload(26)
|  |  +-- -R-- Counter32   nrcacherefresh(27)
|  |
|  +--mmcService(11)
|  |  +-- -R-- Counter32   nrcontread(1)
|  |  +-- -R-- Counter32   nrcontwrite(2)
|  |  +-- -R-- Counter32   nrcontcopy(3)
|  |  +-- -R-- Counter32   nrcontdel(4)
|  |  +-- -R-- Counter32   nrrtcontread(5)
|  |  +-- -R-- Counter32   nrrtcontwrite(6)
|  |  +-- -R-- Counter32   nrrtcontcopy(7)
|  |  +-- -R-- Counter32   nrrtcontdel(8)
|  |  +-- -R-- Counter32   nrcacontread(9)
|  |  +-- -R-- Counter32   rtcachehit(10)
|  |
|  +--inboundGatewayService(12)
|     +-- -R-- Counter32   nrrtcontread(1)
|     +-- -R-- Counter32   nrrtcontwrite(2)
|     +-- -R-- Counter32   nrrtcontcopy(3)
|     +-- -R-- Counter32   nrrtcontdel(4)
|     +-- -R-- Counter32   nrrtobjlock(5)
|     +-- -R-- Counter32   nrrtobjunlock(6)
|     +-- -R-- Counter32   nrrtquery(7)
|     +-- -R-- Counter32   nrrtobjread(8)
|     +-- -R-- Counter32   nrrtverread(9)
|     +-- -R-- Counter32   nrrttrans(10)
|     +-- -R-- Counter32   nrrtverdel(11)
|     +-- -R-- Counter32   nrrtverins(12)
|     +-- -R-- Counter32   nrrtobjdel(13)
|     +-- -R-- Counter32   nrattrins(14)
|     +-- -R-- Counter32   nrrtobjcre(15)
|
+--kernel(2)
|  +-- -R-- Counter32   nrenter(1)
|  +-- -R-- Counter32   nrcall(2)
|  +-- -R-- Counter32   nrcheckaccess(3)
|  +-- -R-- Counter32   nrlock(4)
```

```
|  +-- -R-- Counter32   nrunlock(5)
|  +-- -R-- Counter32   nrsearch(6)
|  +-- -R-- Counter32   nrcommit(7)
|  +-- -R-- Counter32   nrabort(8)
|  +-- -R-- Counter32   nrpersist(9)
|  +-- -R-- Counter32   nrbackup(10)
|  +-- -R-- Counter32   nrrestore(11)
|  +-- -R-- Counter32   nrsingleobjectrefresh(12)
|  +-- -R-- Counter32   nrmultipleobjectrefresh(13)
|  +-- -R-- Counter32   nrsingleobjectloadall(14)
|  +-- -R-- Counter32   nrsingleobjectload(15)
|  +-- -R-- Counter32   nrsingleversionloadall(16)
|  +-- -R-- Counter32   nrsingleversionload(17)
|  +-- -R-- Counter32   nrmultipleobjectloadall(18)
|  +-- -R-- Counter32   nrmultipleobjectload(19)
|  +-- -R-- Counter32   nrmultipleversionloadall(20)
|  +-- -R-- Counter32   nrmultipleversionload(21)
|  +-- -R-- Counter32   nrcallintmeth(22)
|  +-- -R-- Counter32   nrcallextcommeth(23)
|  +-- -R-- Counter32   nrcallextolemeth(24)
|  +-- -R-- Counter32   nrcallextscriptmeth(25)
|  +-- -R-- Counter32   rtcachehit(26)
|  +-- -R-- Counter32   rtcacheused(27)
|  +-- -R-- Counter32   rtcacheobj(28)
|  +-- -R-- Counter32   nruser(29)
|  +-- -R-- Counter32   nrthread(30)
|  +-- -R-- Counter32   nrobjectcreate(31)
|  +-- -R-- Counter32   nrobjectdelete(32)
|  +-- -R-- Counter32   nrobjectchange(33)
|  +-- -R-- Counter32   nrversioncreate(34)
|  +-- -R-- Counter32   nrversiondelete(35)
|
+--client(3)
```

FABASOFT

Die FABASOFT-MAIN-MIB ist der zentrale Ausgangspunkt für die von IANA zugewiesene Enterprise-OID.

```
--==========================================================================
--  $Workfile: fabasoft.mib $
--  $Revision: 2 $ of $Date: 20.07.04 10:43 $ by $Author: Daniel.fallmann $
--  ------------------------------------------------------------------------
--  Copyright Fabasoft R&D Software GmbH & Co KG, 2004
--
--  Der Nutzer des Computerprogramms anerkennt, dass der oben stehende
--  Copyright-Vermerk im Sinn des Welturheberrechtsabkommens an der vom
--  Urheber festgelegten Stelle in der Funktion des Computerprogramms
--  angebracht bleibt, um den Vorbehalt des Urheberrechtes genuegend zum
--  Ausdruck zu bringen. Dieser Urheberrechtsvermerk darf weder vom Kunden,
--  Nutzer und/oder von Dritten entfernt, veraendert oder disloziert werden.
--==========================================================================

FABASOFT-MAIN-MIB DEFINITIONS ::=
BEGIN

  IMPORTS
    MODULE-IDENTITY,
    enterprises          FROM SNMPv2-SMI;

  fabasoft MODULE-IDENTITY
    LAST-UPDATED „200406201000Z“
    ORGANIZATION „Fabasoft R&D Software GmbH & Co KG“
    CONTACT-INFO
      „Fabasoft R&D Software GmbH & Co KG
       Postal: Fabasoft R&D Software GmbH & Co KG
               Honauerstrasse 4, A-4020 Linz
               AUSTRIA

       Tel: +43 732 606162
       Fax: +43 732 606162 609
       E-mail: support@fabasoft.com“
     DESCRIPTION
```

```
      „The Structure of Management Information for the Fabasoft enterprise."
     ::= { enterprises 17100 }

    -- assigned by IANA (1.3.6.4.1.17100)
    -- enterprise number 17100 (so whole number is 1.3.6.4.1.17100)
    -- iso(1).org(3).dod(6).internet(4).enterprise(1).fabasoft(17100)

    components OBJECT IDENTIFIER ::= { fabasoft 1 }
    server OBJECT IDENTIFIER ::= { components 1 }

END
```

MMCService

Die Fabasoft Components MMC-Services werden für den Zugriff und die Speicherung von Daten in den Fabasoft MMC-Stores verwendet. In den Fabasoft MMC-Stores befinden sich alle Filesystemdaten die von Fabasoft Components eingebunden werden.

```
--=============================================================================
--  $Workfile: mmcservice.mib $
--  $Revision: 4 $ of $Date: 20.07.04 10:43 $ by $Author: Daniel.fallmann $
--  -----------------------------------------------------------------------
--  Copyright Fabasoft R&D Software GmbH & Co KG, 2004
--
--  Der Nutzer des Computerprogramms anerkennt, dass der oben stehende
--  Copyright-Vermerk im Sinn des Welturheberrechtsabkommens an der vom
--  Urheber festgelegten Stelle in der Funktion des Computerprogramms
--  angebracht bleibt, um den Vorbehalt des Urheberrechtes genuegend zum
--  Ausdruck zu bringen. Dieser Urheberrechtsvermerk darf weder vom Kunden,
--  Nutzer und/oder von Dritten entfernt, veraendert oder disloziert werden.
--=============================================================================

SERVER-MMCSERVICE-MIB DEFINITIONS ::=
BEGIN

  IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Unsigned32,
    Counter32           FROM SNMPv2-SMI
    MODULE-COMPLIANCE,
    OBJECT-GROUP        FROM SNMPv2-CONF
    DisplayString       FROM SNMPv2-TC
    server              FROM FABASOFT-MAIN-MIB;

  mmcService MODULE-IDENTITY
    LAST-UPDATED „200406201000Z“
    ORGANIZATION „Fabasoft R&D Software GmbH & Co KG“
    CONTACT-INFO
```

```
    „Fabasoft R&D Software GmbH & Co KG
     Postal: Fabasoft R&D Software GmbH & Co KG
             Honauerstrasse 4, A-4020 Linz
             AUSTRIA

     Tel: +43 732 606162
     Fax: +43 732 606162 609
     E-mail: support@fabasoft.com“
   DESCRIPTION
     „Provides statistic information for MMC-service instances.“
  ::= { server 11 }

numinstalledmmcservices OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Number of Server MMC-service instances.“
  ::= { mmcService 1 }

mmcInstanceTable OBJECT-TYPE
  SYNTAX SEQUENCE OF MmcInstanceEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Table for instances of MMC-services.“
  ::= { mmcService 2 }

mmcInstanceEntry OBJECT-TYPE
  SYNTAX MmcInstanceEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Internal table entry structure.“
  INDEX { mmcid }
  ::= { mmcInstanceTable 1}

MmcInstanceEntry ::=
  SEQUENCE {
```

```
    mmcid                   Integer32,
    mmcservicename          DisplayString,
    mmcnrcontread           Counter32,
    mmcnrcontwrite          Counter32,
    mmcnrcontcopy           Counter32,
    mmcnrcontdel            Counter32,
    mmcnrrtcontread         Counter32,
    mmcnrrtcontwrite        Counter32,
    mmcnrrtcontcopy         Counter32,
    mmcnrrtcontdel          Counter32,
    mmcnrcacontread         Counter32
  }

mmcid OBJECT-TYPE
  SYNTAX Integer32 (0..1024)
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Contains the identifier for this MMC-service instance."
  ::= { mmcInstanceEntry 1 }

mmcservicename OBJECT-TYPE
  SYNTAX DisplayString
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Textual description of this MMC-service instance."
  ::= { mmcInstanceEntry 2 }

mmcnrcontread OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of contents read per second."
  ::= { mmcInstanceEntry 3 }

mmcnrcontwrite OBJECT-TYPE
  SYNTAX Counter32
```

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of contents written per second.“
    ::= { mmcInstanceEntry 4 }

  mmcnrcontcopy OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of contents copied per second.“
    ::= { mmcInstanceEntry 5 }

  mmcnrcontdel OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of contents deleted per second.“
    ::= { mmcInstanceEntry 6 }

  mmcnrrtcontread OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to read a
content.“
    ::= { mmcInstanceEntry 7 }

  mmcnrrtcontwrite OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to write a
content.“
    ::= { mmcInstanceEntry 8 }
```

```
  mmcnrrtcontcopy OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to copy contents.“
    ::= { mmcInstanceEntry 9 }

  mmcnrrtcontdel OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to deleted
contents.“
    ::= { mmcInstanceEntry 10 }

  mmcnrcacontread OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of contents per second read in the cache
       instead of routing it to a different service.“
    ::= { mmcInstanceEntry 11 }

--
-- Conformance information
--

mmcServiceConformance OBJECT IDENTIFIER ::= { mmcService 100 }
mmcServiceCompliances OBJECT IDENTIFIER ::= { mmcServiceConformance 1 }

mmcServiceCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    „MMC-service compliance module.“
  MODULE SERVER-MMCSERVICE-MIB
```

```
  MANDATORY-GROUPS { mmcServiceGroup }
  ::= { mmcServiceCompliances 1 }

mmcServiceGroups OBJECT IDENTIFIER ::= { mmcServiceConformance 2 }
mmcServiceGroup OBJECT-GROUP
  OBJECTS {
    numinstalledmmcservices,
    mmcservicename,
    mmcnrcontread,
    mmcnrcontwrite,
    mmcnrcontcopy,
    mmcnrcontdel,
    mmcnrrtcontread,
    mmcnrrtcontwrite,
    mmcnrrtcontcopy,
    mmcnrrtcontdel,
    mmcnrcacontread
  }
  STATUS current
  DESCRIPTION
    „MMC-service conformance object group.“
  ::= { mmcServiceGroups 1 }

END
```

14.1.1.1 COOService

Die Fabasoft Components COO-Services bilden alle Daten in die zugehörigen COO-Stores ab. Diese Daten werden direkt in das verwendete Datenbankmanagementsystem gespeichert.

```
--==========================================================================
--  $Workfile: cooservice.mib $
--  $Revision: 4 $ of $Date: 20.07.04 10:43 $ by $Author: Daniel.fallmann $
--  ---------------------------------------------------------------------
--  Copyright Fabasoft R&D Software GmbH & Co KG, 2004
--
--  Der Nutzer des Computerprogramms anerkennt, dass der oben stehende
--  Copyright-Vermerk im Sinn des Welturheberrechtsabkommens an der vom
--  Urheber festgelegten Stelle in der Funktion des Computerprogramms
--  angebracht bleibt, um den Vorbehalt des Urheberrechtes genuegend zum
--  Ausdruck zu bringen. Dieser Urheberrechtsvermerk darf weder vom Kunden,
--  Nutzer und/oder von Dritten entfernt, veraendert oder disloziert werden.
--==========================================================================

SERVER-COOSERVICE-MIB DEFINITIONS ::=
BEGIN

  IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Unsigned32,
    Counter32           FROM SNMPv2-SMI
    MODULE-COMPLIANCE,
    OBJECT-GROUP        FROM SNMPv2-CONF
    DisplayString       FROM SNMPv2-TC
    server              FROM FABASOFT-MAIN-MIB;

  cooService MODULE-IDENTITY
    LAST-UPDATED „200406201000Z“
    ORGANIZATION „Fabasoft R&D Software GmbH & Co KG“
    CONTACT-INFO
      „Fabasoft R&D Software GmbH & Co KG
       Postal: Fabasoft R&D Software GmbH & Co KG
```

```
              Honauerstrasse 4, 4020 Linz
              AUSTRIA

      Tel: +43 732 606162
      Fax: +43 732 606162 609
      E-mail: support@fabasoft.com“
    DESCRIPTION
      „Provides statistic information for COO-service instances.“
  ::= { server 10 }

numinstalledcooservices OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Number of Server COO-service instances.“
  ::= { cooService 1 }

cooInstanceTable OBJECT-TYPE
  SYNTAX SEQUENCE OF CooInstanceEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Table for instances of COO-services.“
  ::= { cooService 2 }

cooInstanceEntry OBJECT-TYPE
  SYNTAX CooInstanceEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Internal table entry structure.“
  INDEX { cooid }
  ::= { cooInstanceTable 1 }

CooInstanceEntry ::=
  SEQUENCE {
    cooid                   Integer32,
    cooservicename          DisplayString,
```

```
    coonrobjlock            Counter32,
    coonrobjunlock          Counter32,
    coonrquery              Counter32,
    coonrobjread            Counter32,
    coonrverread            Counter32,
    coonrtrans              Counter32,
    coonrverdel             Counter32,
    coonrverins             Counter32,
    coonrobjdel             Counter32,
    coonrattrins            Counter32,
    coonrobjcre             Counter32,
    coonrrtobjlock          Counter32,
    coonrrtobjunlock        Counter32,
    coonrrtquery            Counter32,
    coonrrtobjread          Counter32,
    coonrrtverread          Counter32,
    coonrrttrans            Counter32,
    coonrrtverdel           Counter32,
    coonrrtverins           Counter32,
    coonrrtobjdel           Counter32,
    coonrrtattrins          Counter32,
    coonrrtobjcre           Counter32,
    coonrcacheobj           Unsigned32,
    coonrcacherefresh       Counter32
  }

cooid OBJECT-TYPE
  SYNTAX Integer32 (0..1024)
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Contains the identifier for this COO-service instance."
  ::= { cooInstanceEntry 1 }

cooservicename OBJECT-TYPE
  SYNTAX DisplayString
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
```

```
    „Textual description of this COO-service instance.“
  ::= { cooInstanceEntry 2 }

coonrobjlock OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requested object locks.“
  ::= { cooInstanceEntry 3 }

coonrobjunlock OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requested object unlocks.“
  ::= { cooInstanceEntry 4 }

coonrquery OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of queries executed.“
  ::= { cooInstanceEntry 5 }

coonrobjread OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of objects, of which one or more attributes are read.“
  ::= { cooInstanceEntry 6 }

coonrverread OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
```

```
    DESCRIPTION
      „The number of objects, of which one or more attributes of a saved
version are read.“
    ::= { cooInstanceEntry 7 }

  coonrtrans OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of update transactions executed.“
    ::= { cooInstanceEntry 8 }

  coonrverdel OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of saved versions of objects deleted.“
    ::= { cooInstanceEntry 9 }

  coonrverins OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of versions of objects saved.“
    ::= { cooInstanceEntry 10 }

  coonrobjdel OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of objects deleted.“
    ::= { cooInstanceEntry 11 }

  coonrattrins OBJECT-TYPE
    SYNTAX Counter32
```

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of attributes stored.“
    ::= { cooInstanceEntry 12 }

  coonrobjcre OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of objects created.“
    ::= { cooInstanceEntry 13 }

  coonrrtobjlock OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to lock an
object.“
    ::= { cooInstanceEntry 14 }

  coonrrtobjunlock OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to unlock an
object.“
    ::= { cooInstanceEntry 15 }

  coonrrtquery OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to query objects.“
    ::= { cooInstanceEntry 16 }
```

```
coonrrtobjread OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different
     service to read one or more attributes of an object.“
  ::= { cooInstanceEntry 17 }

coonrrtverread OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different service to read
     one ore more attributes of a saved version of an object.“
  ::= { cooInstanceEntry 18 }

coonrrttrans OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different service
     to executed an update transaction.“
  ::= { cooInstanceEntry 19 }

coonrrtverdel OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different
     service to deleted a saved version of an object.“
  ::= { cooInstanceEntry 20 }

coonrrtverins OBJECT-TYPE
  SYNTAX Counter32
```

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service
       to save a version of an object.“
    ::= { cooInstanceEntry 21 }

  coonrrtobjdel OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to deleted an
objects.“
    ::= { cooInstanceEntry 22 }

  coonrrtattrins OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different
       service to store an attribute of an object.“
    ::= { cooInstanceEntry 23 }

  coonrrtobjcre OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to create an
object.“
    ::= { cooInstanceEntry 24 }

  coonrcacheobj OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
```

```
      „Cached Objects is the number of objects in the cache.“
    ::= { cooInstanceEntry 27 }

  coonrcacherefresh OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of times where the cache refreshes an object.“
    ::= { cooInstanceEntry 28 }

--
-- Conformance information
--

  cooServiceConformance OBJECT IDENTIFIER ::= { cooService 100 }
  cooServiceCompliances OBJECT IDENTIFIER ::= { cooServiceConformance 1 }

  cooServiceCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
      „COO-service compliance module.“
    MODULE SERVER-COOSERVICE-MIB
    MANDATORY-GROUPS { cooServiceGroup }
    ::= { cooServiceCompliances 1 }

  cooServiceGroups OBJECT IDENTIFIER ::= { cooServiceConformance 2 }
  cooServiceGroup OBJECT-GROUP
    OBJECTS {
      numinstalledcooservices,
      cooservicename,
      coonrobjlock,
      coonrobjunlock,
      coonrquery,
      coonrobjread,
      coonrverread,
      coonrtrans,
      coonrverdel,
      coonrverins,
```

```
      coonrobjdel,
      coonrattrins,
      coonrobjcre,
      coonrrtobjlock,
      coonrrtobjunlock,
      coonrrtquery,
      coonrrtobjread,
      coonrrtverread,
      coonrrttrans,
      coonrrtverdel,
      coonrrtverins,
      coonrrtobjdel,
      coonrrtattrins,
      coonrrtobjcre,
      coonrcacheobj,
      coonrcacherefresh
    }
    STATUS current
    DESCRIPTION
      „COO-service conformance object group.“
    ::= { cooServiceGroups 1 }

END
```

14.1.1.2 InboundGatewayService

```
--============================================================================
--  $Workfile: inboundgatewayservice.mib $
--  $Revision: 4 $ of $Date: 20.07.04 10:43 $ by $Author: Daniel.fallmann $
--  ----------------------------------------------------------------------------
--  Copyright Fabasoft R&D Software GmbH & Co KG, 2004
--
--  Der Nutzer des Computerprogramms anerkennt, dass der oben stehende
--  Copyright-Vermerk im Sinn des Welturheberrechtsabkommens an der vom
--  Urheber festgelegten Stelle in der Funktion des Computerprogramms
--  angebracht bleibt, um den Vorbehalt des Urheberrechtes genuegend zum
--  Ausdruck zu bringen. Dieser Urheberrechtsvermerk darf weder vom Kunden,
--  Nutzer und/oder von Dritten entfernt, veraendert oder disloziert werden.
--============================================================================

SERVER-INBOUNDGATEWAYSERVICE-MIB DEFINITIONS ::=
BEGIN

  IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Unsigned32,
    Counter32           FROM SNMPv2-SMI
    MODULE-COMPLIANCE,
    OBJECT-GROUP        FROM SNMPv2-CONF
    DisplayString       FROM SNMPv2-TC
    server              FROM FABASOFT-MAIN-MIB;

  inboundGatewayService MODULE-IDENTITY
    LAST-UPDATED „200406201000Z“
    ORGANIZATION „Fabasoft R&D Software GmbH & Co KG“
    CONTACT-INFO
      „Fabasoft R&D Software GmbH & Co KG
       Postal: Fabasoft R&D Software GmbH & Co KG
               Honauerstrasse 4, A-4020 Linz
               AUSTRIA
```

```
      Tel: +43 732 606162
      Fax: +43 732 606162 609
      E-mail: support@fabasoft.com“
    DESCRIPTION
      „Provides statistic information for InboundGateway-service instances.“
  ::= { server 12 }

numinstalledigtservices OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Number of Server InboundGateway-service instances.“
  ::= { inboundGatewayService 1 }

inboundGatewayInstanceTable OBJECT-TYPE
  SYNTAX SEQUENCE OF InboundGatewayInstanceEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Table for instances of InboundGateway-services.“
  ::= { inboundGatewayService 2 }

inboundGatewayInstanceEntry OBJECT-TYPE
  SYNTAX InboundGatewayInstanceEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    „Internal table entry structure.“
  INDEX { igtid }
  ::= { inboundGatewayInstanceTable 1 }

InboundGatewayInstanceEntry ::=
  SEQUENCE {
    igtid              Integer32,
    igtservicename     DisplayString,
    igtnrrtcontread    Counter32,
    igtnrrtcontwrite   Counter32,
```

```
      igtnrrtcontcopy    Counter32,
      igtnrrtcontdel     Counter32,
      igtnrrtobjlock     Counter32,
      igtnrrtobjunlock   Counter32,
      igtnrrtquery       Counter32,
      igtnrrtobjread     Counter32,
      igtnrrtverread     Counter32,
      igtnrrttrans       Counter32,
      igtnrrtverdel      Counter32,
      igtnrrtverins      Counter32,
      igtnrrtobjdel      Counter32,
      igtnrrtattrins     Counter32,
      igtnrrtobjcre      Counter32
    }

  igtid OBJECT-TYPE
    SYNTAX Integer32 (0..1024)
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
      „Contains the identifier for this InboundGateway-service instance."
    ::= { inboundGatewayInstanceEntry 1 }

  igtservicename OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Textual description of this InboundGateway-service instance."
    ::= { inboundGatewayInstanceEntry 2 }


  igtnrrtcontread OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to read a
content."
```

```
    ::= { inboundGatewayInstanceEntry 3 }

  igtnrrtcontwrite OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to write a
content."
    ::= { inboundGatewayInstanceEntry 4 }

  igtnrrtcontcopy OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to copy contents."
    ::= { inboundGatewayInstanceEntry 5 }

  igtnrrtcontdel OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to deleted
contents."
    ::= { inboundGatewayInstanceEntry 6 }

  igtnrrtobjlock OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to lock an
object."
    ::= { inboundGatewayInstanceEntry 7 }

  igtnrrtobjunlock OBJECT-TYPE
    SYNTAX Counter32
```

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to unlock an
object.“
    ::= { inboundGatewayInstanceEntry 8 }

  igtnrrtquery OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to query objects.“
    ::= { inboundGatewayInstanceEntry 9 }

  igtnrrtobjread OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to read
       one or more attributes of an object.“
    ::= { inboundGatewayInstanceEntry 10 }

  igtnrrtverread OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to
       read one ore more attributes of a saved version of an object.“
    ::= { inboundGatewayInstanceEntry 11 }

  igtnrrttrans OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to
```

```
      executed an update transaction."
  ::= { inboundGatewayInstanceEntry 12 }

igtnrrtverdel OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different service
     to deleted a saved version of an object."
  ::= { inboundGatewayInstanceEntry 13 }

igtnrrtverins OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different service to
     save a version of an object."
  ::= { inboundGatewayInstanceEntry 14 }

igtnrrtobjdel OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different service
     to deleted an objects."
  ::= { inboundGatewayInstanceEntry 15 }

igtnrrtattrins OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of requests routed to a different service
     to store an attribute of an object."
  ::= { inboundGatewayInstanceEntry 16 }
```

```
  igtnrrtobjcre OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of requests routed to a different service to create an
object.“
    ::= { inboundGatewayInstanceEntry 17 }

--
-- Conformance information
--

  inboundGatewayServiceConformance OBJECT IDENTIFIER ::= {
inboundGatewayService 100 }
  inboundGatewayServiceCompliances OBJECT IDENTIFIER ::= {
inboundGatewayServiceConformance 1 }

  inboundGatewayServiceCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
      „InboundGateway-service compliance module.“
    MODULE SERVER-INBOUNDGATEWAYSERVICE-MIB
    MANDATORY-GROUPS { inboundGatewayServiceGroup }
    ::= { inboundGatewayServiceCompliances 1 }

  inboundGatewayServiceGroups OBJECT IDENTIFIER ::= {
inboundGatewayServiceConformance 2 }
  inboundGatewayServiceGroup OBJECT-GROUP
    OBJECTS {
      numinstalledigtservices,
      igtservicename,
      igtnrrtcontread,
      igtnrrtcontwrite,
      igtnrrtcontcopy,
      igtnrrtcontdel,
      igtnrrtobjlock,
      igtnrrtobjunlock,
      igtnrrtquery,
```

```
      igtnrrtobjread,
      igtnrrtverread,
      igtnrrttrans,
      igtnrrtverdel,
      igtnrrtverins,
      igtnrrtobjdel,
      igtnrrtattrins,
      igtnrrtobjcre
    }
    STATUS current
    DESCRIPTION
      „InboundGateway-service conformance object group.“
    ::= { inboundGatewayServiceGroups 1 }

END
```

14.1.1.3 RPC Service

```
--==========================================================================
--  $Workfile: rpcservice.mib $
--  $Revision: 6 $ of $Date: 20.07.04 10:43 $ by $Author: Daniel.fallmann $
--  --------------------------------------------------------------------------
--  Copyright Fabasoft R&D Software GmbH & Co KG, 2004
--
--  Der Nutzer des Computerprogramms anerkennt, dass der oben stehende
--  Copyright-Vermerk im Sinn des Welturheberrechtsabkommens an der vom
--  Urheber festgelegten Stelle in der Funktion des Computerprogramms
--  angebracht bleibt, um den Vorbehalt des Urheberrechtes genuegend zum
--  Ausdruck zu bringen. Dieser Urheberrechtsvermerk darf weder vom Kunden,
--  Nutzer und/oder von Dritten entfernt, veraendert oder disloziert werden.
--==========================================================================

SERVER-RPCSERVICE-MIB DEFINITIONS ::=
BEGIN

  IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Unsigned32,
    Counter32           FROM SNMPv2-SMI
    MODULE-COMPLIANCE,
    OBJECT-GROUP        FROM SNMPv2-CONF
    DisplayString       FROM SNMPv2-TC
    server              FROM FABASOFT-MAIN-MIB;

  rpcService MODULE-IDENTITY
    LAST-UPDATED „200406201000Z“
    ORGANIZATION „Fabasoft R&D Software GmbH & Co KG“
    CONTACT-INFO
      „Fabasoft R&D Software GmbH & Co KG
       e-Government-Service
       Postal: Fabasoft R&D Software GmbH & Co KG
               Honauerstrasse 4, A-4020 Linz
```

```
              AUSTRIA

      Tel: +43 732 606162
      Fax: +43 732 606162 609
      E-mail: support@fabasoft.com“
    DESCRIPTION
      „Provides statistic information for RPC-service instances.“
   ::= { server 1 }

 numinstalledrpcservices OBJECT-TYPE
   SYNTAX Unsigned32
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
     „Number of Server RPC instances.“
   ::= { rpcService 1 }

 rpcInstanceTable OBJECT-TYPE
   SYNTAX SEQUENCE OF RpcInstanceEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
     „Table for instances of RPC-services.“
   ::= { rpcService 2 }

 rpcInstanceEntry OBJECT-TYPE
   SYNTAX RpcInstanceEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
     „Internal table entry structure.“
   INDEX { rpcid }
   ::= { rpcInstanceTable 1}

RpcInstanceEntry ::=
   SEQUENCE {
     rpcid                  Integer32,
     rpcservicename         DisplayString,
     rpcnrcalls             Counter32,
```

```
      rpcnrtrans              Counter32,
      rpcnrthreads            Unsigned32,
      rpcnrrphysconns         Unsigned32,
      rpcnrrecvbytes          Counter32,
      rpcnrsendbytes          Counter32,
      rpcnrrecvpackets        Counter32,
      rpcnrsendpackets        Counter32
    }

--
-- RPC Performance counters
--

  rpcid OBJECT-TYPE
    SYNTAX Integer32 (0..1024)
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
      „Contains the identifier for your RPC-service instance.“
    ::= { rpcInstanceEntry 1 }

  rpcservicename OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Textual description of this RPC-service instance.“
    ::= { rpcInstanceEntry 2 }

  rpcnrcalls OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of remote procedure calls executed by a server process.“
    ::= { rpcInstanceEntry 3 }

  rpcnrtrans OBJECT-TYPE
    SYNTAX Counter32
```

```
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of remote procedure call transactions executed by
     a server process.“
  ::= { rpcInstanceEntry 4 }

rpcnrthreads OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of threads currently allocated in a server
     process for remote procedure call transactions.“
  ::= { rpcInstanceEntry 5 }

rpcnrrphysconns OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of currently used physical connections.“
  ::= { rpcInstanceEntry 6 }

rpcnrrecvbytes OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of bytes received by a server process as
     input parameter of remote procedure calls.“
  ::= { rpcInstanceEntry 7 }

rpcnrsendbytes OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „The number of bytes sent by a server process as output
```

```
      parameter of remote procedure calls."
    ::= { rpcInstanceEntry 8 }

  rpcnrrecvpackets OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of network packets received by a server
       process as input parameter of remote procedure calls."
    ::= { rpcInstanceEntry 10 }

  rpcnrsendpackets OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „The number of network packets sent by a server process
       as output parameter of remote procedure calls."
    ::= { rpcInstanceEntry 11 }

--
-- Conformance information
--

  rpcServiceConformance OBJECT IDENTIFIER ::= { rpcService 100 }
  rpcServiceCompliances OBJECT IDENTIFIER ::= { rpcServiceConformance 1 }

  rpcServiceCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
      „RPC-service compliance module."
    MODULE SERVER-RPCSERVICE-MIB
    MANDATORY-GROUPS { rpcServiceGroup }
    ::= { rpcServiceCompliances 1 }

  rpcServiceGroups OBJECT IDENTIFIER ::= { rpcServiceConformance 2 }
  rpcServiceGroup OBJECT-GROUP
    OBJECTS {
```

```
      numinstalledrpcservices,
      rpcservicename,
      rpcnrcalls,
      rpcnrtrans,
      rpcnrthreads,
      rpcnrrphysconns,
      rpcnrrecvbytes,
      rpcnrsendbytes,
      rpcnrrecvpackets,
      rpcnrsendpackets
    }
    STATUS current
    DESCRIPTION
      „RPC-service conformance object group.“
    ::= { rpcServiceGroups 1 }

END
```

14.1.1.4 Kernel

Der Fabasoft Components Kernel stellt im Webservice das Objektmodell zur Verfügung. Es wird auch der Zugriffsschutz festgelegt, Eigenschaften überprüft, Transaktionen ausgeführt, Methodenaufrufe zugeteilt und auch die Objektplazierung vorgenommen.

```
--============================================================================
--  $Workfile: kernel.mib $
--  $Revision: 8 $ of $Date: 24.07.04 11:43 $ by $Author: Daniel.fallmann $
--  ----------------------------------------------------------------------------
--  Copyright Fabasoft R&D Software GmbH & Co KG, 2004
--
--  Der Nutzer des Computerprogramms anerkennt, dass der oben stehende
--  Copyright-Vermerk im Sinn des Welturheberrechtsabkommens an der vom
--  Urheber festgelegten Stelle in der Funktion des Computerprogramms
--  angebracht bleibt, um den Vorbehalt des Urheberrechtes genuegend zum
--  Ausdruck zu bringen. Dieser Urheberrechtsvermerk darf weder vom Kunden,
--  Nutzer und/oder von Dritten entfernt, veraendert oder disloziert werden.
--============================================================================
COMPONENTS-KERNEL-MIB DEFINITIONS ::=
BEGIN

  IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Counter32           FROM SNMPv2-SMI
    DisplayString       FROM RFC1213-MIB
    --OBJECT-GROUP        FROM SNMPv2-CONF
    statistics          FROM COMPONENTS-STATISTICS-MIB;

  kernel MODULE-IDENTITY
    LAST-UPDATED „200407101732Z“
    ORGANIZATION „Fabasoft AG“
    CONTACT-INFO
      ORGANIZATION „Fabasoft R&D Software GmbH & Co KG“
    CONTACT-INFO
      „Fabasoft R&D Software GmbH & Co KG
       e-Government-Service
```

```
      Postal: Fabasoft R&D Software GmbH & Co KG
              Honauerstrasse 4, A-4020 Linz
              AUSTRIA

      Tel: +43 732 606162
      Fax: +43 732 606162 609
      E-mail: support@fabasoft.com“
     DESCRIPTION
       „Provides statistic information about the kernel.“
  ::= { components 2 }

numComponentsKernel OBJECT-TYPE
  SYNTAX INTEGER
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Number of kernel instances.“
  ::= { kernel 1 }

componentsKernelTable OBJECT-TYPE
  SYNTAX SEQUENCE OF ComponentsKernelEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    ““
  ::= { kernel 2 }

componentsKernelEntry OBJECT-TYPE
  SYNTAX ComponentsKernelEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    ““
  ::= {componentsKernelTable 1}

ComponentsKernelEntry ::=
  SEQUENCE {
    id                        DisplayString,
    strServerKernelName       DisplayString,
```

```
    nrenter                    Counter32,
    nrcall                     Counter32,
    nrcheckaccess              Counter32,
    nrlock                     Counter32,
    nrunlock                   Counter32,
    nrsearch                   Counter32,
    nrcommit                   Counter32,
    nrabort                    Counter32,
    nrpersist                  Counter32,
    nrbackup                   Counter32,
    nrrestore                  Counter32,
    nrsingleobjectrefresh      Counter32,
    nrmultipleobjectrefresh    Counter32,
    nrsingleobjectloadall      Counter32,
    nrsingleobjectload         Counter32,
    nrsingleversionloadall     Counter32,
    nrsingleversionload        Counter32,
    nrmultipleobjectloadall    Counter32,
    nrmultipleobjectload       Counter32,
    nrmultipleversionloadall   Counter32,
    nrmultipleversionload      Counter32,
    nrcallintmeth              Counter32,
    nrcallextcommeth           Counter32,
    nrcallextolemeth           Counter32,
    nrcallextscriptmeth        Counter32,
    rtcachehit                 Counter32,
    rtcacheused                Counter32,
    rtcacheobj                 Counter32,
    nruser                     Counter32,
    nrthread                   Counter32,
    nrobjectcreate             Counter32,
    nrobjectdelete             Counter32,
    nrobjectchange             Counter32,
    nrversioncreate            Counter32,
    nrversiondelete            Counter32
  }

id OBJECT-TYPE
  SYNTAX DisplayString
```

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Contains the identifier for your kernel instance.“
    ::= { componentsKernelEntry 1 }

  strComponentsKernelName OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Textual description of this kernel instance.“
    ::= { componentsKernelEntry 2 }

  nrenter OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Actions Called/sec is the number of actions called on objects per
second.“
    ::= { componentsKernelEntry 3 }

  nrcall OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Actions Called/sec is the number of actions called on objects per
second.“
    ::= { componentsKernelEntry 4 }

  nrcheckaccess OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Access Checks/sec is the number of access checks on objects per
second.“
```

```
    ::= { componentsKernelEntry 5 }

  nrlock OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Locked/sec is the number of objects locked per second.“
    ::= { componentsKernelEntry 6 }

  nrunlock OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Unlocked/sec is the number of objects unlocked per second.“
    ::= { componentsKernelEntry 7 }

  nrsearch OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Queries/sec is the number of queries executed per second.“
    ::= { componentsKernelEntry 8 }

  nrcommit OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Transactions Committed/sec is the number of transactions committed per
second.“
    ::= { componentsKernelEntry 9 }

  nrabort OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
```

```
    DESCRIPTION
      „Transactions Aborted/sec is the number of transactions aborted per
second.“
    ::= { componentsKernelEntry 10 }

  nrpersist OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Transactions Persisted/sec is the number of transactions persisted per
second.“
    ::= { componentsKernelEntry 11 }

  nrbackup OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Transactions Backed Up/sec is the number of transactions backed up per
second.“
    ::= { componentsKernelEntry 12 }

  nrrestore OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Transactions Restored/sec is the number of transactions restored per
second.“
    ::= { componentsKernelEntry 13 }

  nrsingleobjectrefresh OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Refreshed (Single)/sec is the number of objects refreshed in a
single call per second.“
```

```
    ::= { componentsKernelEntry 14 }

  nrmultipleobjectrefresh OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Refreshed (Bulk)/sec is the number of objects refreshed in a
bulk call per second.“
    ::= { componentsKernelEntry 15 }

  nrsingleobjectloadall OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Completely Loaded (Single)/sec is the number of objects
completely loaded in a single call per second.“
    ::= { componentsKernelEntry 16 }

  nrsingleobjectload OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Partially Loaded (Single)/sec is the number of objects
partially loaded in a single call per second.“
    ::= { componentsKernelEntry 17 }

  nrsingleversionloadall OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Versions Completely Loaded (Single)/sec is the number of versions
completely loaded in a single call per second.“
    ::= { componentsKernelEntry 18 }

  nrsingleversionload OBJECT-TYPE
```

```
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Versions Partially Loaded (Single)/sec is the number of versions
partially loaded in a single call per second.“
    ::= { componentsKernelEntry 19 }

  nrmultipleobjectloadall OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Completely Loaded (Bulk)/sec is the number of objects
completely loaded in a bulk call per second.“
    ::= { componentsKernelEntry 20 }

  nrmultipleobjectload OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Objects Partially Loaded (Bulk)/sec is the number of objects partially
loaded in a bulk call per second.“
    ::= { componentsKernelEntry 21 }

  nrmultipleversionloadall OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Versions Completely Loaded (Bulk)/sec is the number of versions
partially loaded in a bulk call per second.“
    ::= { componentsKernelEntry 22 }

  nrmultipleversionload OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
```

```
    DESCRIPTION
      „Versions Partially Loaded (Bulk)/sec is the number of versions
partially loaded in a bulk call per second.“
    ::= { componentsKernelEntry 23 }

  nrcallintmeth OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Internal Methods Called/sec is the number of internal methods called
per second.“
    ::= { componentsKernelEntry 24 }

  nrcallextcommeth OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „External COM Methods Called/sec is the number of external COM methods
called per second.“
    ::= { componentsKernelEntry 25 }

  nrcallextolemeth OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „External OLE-Automation Methods Called/sec is the number of external
OLE-Automation methods called per second.“
    ::= { componentsKernelEntry 26 }

  nrcallextscriptmeth OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „External Script Methods Called/sec is the number of external ActiveX-
Scripting methods called per second.“
```

```
    ::= { componentsKernelEntry 27 }

  rtcachehit OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „% Cache Hit Rate is the percentage of read requests resolved in the
cache.“
    ::= { componentsKernelEntry 28 }

  rtcacheused OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „% Cache Used is the percentage of space used in the cache.“
    ::= { componentsKernelEntry 29 }

  rtcacheobj OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Cached Objects is the number of objects in the cache.“
    ::= { componentsKernelEntry 30 }

  nruser OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      „Users is the number of current user contexts.“
    ::= { componentsKernelEntry 31 }

  nrthread OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
```

```
  DESCRIPTION
    „Threads is the number of current worker threads.“
  ::= { componentsKernelEntry 32 }

nrobjectcreate OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Objects Created/sec is the number of objects created per second.“
  ::= { componentsKernelEntry 33 }

nrobjectdelete OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Objects Deleted/sec is the number of objects deleted per second.“
  ::= { componentsKernelEntry 34 }

nrobjectchange OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Objects Changed/sec is the number of objects changed per second.“
  ::= { componentsKernelEntry 35 }

nrversioncreate OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    „Versions Created/sec is the number of versions created per second.“
  ::= { componentsKernelEntry 36 }

nrversiondelete OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
```

```
    STATUS current
    DESCRIPTION
      „Versions Deleted/sec is the number of versions deleted per second.“
    ::= { componentsKernelEntry 37 }


--
-- Conformance information
--

  kernelConformance OBJECT IDENTIFIER ::= { kernel 100 }
  kernelCompliances OBJECT IDENTIFIER ::= { kernelConformance 1 }

  kernelCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
      „kernel compliance module.“
    MODULE COMPONENTS-KERNEL-MIB
    MANDATORY-GROUPS { kernelGroup }
    ::= { kernelCompliances 1 }

  kernelGroups OBJECT IDENTIFIER ::= { kernelConformance 2 }
  kernelGroup OBJECT-GROUP
    OBJECTS {
      nrenter,
      nrcall,
      nrcheckaccess,
      nrlock,
      nrunlock,
      nrsearch,
      nrcommit,
      nrabort
      nrpersist,
      nrbackup,
      nrrestore,
      nrsingleobjectrefresh,
      nrmultipleobjectrefresh,
      nrsingleobjectloadall,
      nrsingleobjectload,
```

```
      nrsingleversionloadall,
      nrsingleversionload,
      nrmultipleobjectloadall,
      nrmultipleobjectload,
      nrmultipleversionloadall,
      nrmultipleversionload,
      nrcallintmeth,
      nrcallextcommeth,
      nrcallextolemeth,
      nrcallextscriptmeth,
      rtcachehit,
      rtcacheused,
      nrcacheobj,
      nruser,
      nrthread,
      nrobjectcreate,
      nrobjectdelete,
      nrobjectchange,
      nrversioncreate,
      nrversiondelete
    }
    STATUS current
    DESCRIPTION
      „kernel conformance object group.“
    ::= { kernelGroups 1 }

END
```