Computer Communications and Networks

Springer-Verlag London Ltd.

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

Also in this series:

An Information Security Handbook
John M.D. Hunter
1-85233-180-1

Multimedia Internet Broadcasting: Quality, Technology and Interface
Andy Sloane and Dave Lawrence (Eds)
1-85233-283-2

The Quintessential PIC Microcontroller
Sid Katzen
1-85233-309-X

Information Assurance: Surviving in the Information Environment
Andrew Blyth and Gerald L. Kovacich
1-85233-326-X

UMTS: Origins, Architecture and the Standard
Pierre Lescuyer (Translation Editor: Frank Bott)
1-85233-676-5

Designing Software for the Mobile Context
Roman Longoria (Ed)
1-85233-785-0

Peers in a Client-Server World
Ian J. Taylor
1-85233-869-5
(*publication due September 2004*)

Kundan Misra

OSS for Telecom Networks

An Introduction to Network Management

Springer

Kundan Misra, BSc (Hons), LLB, PhD
Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

Series editor
Professor A.J. Sammes, BSc, MPhil, PhD, FBCS, CEng
CISM Group, Cranfield University, RMCS, Shrivenham, Swindon SN6 8LA, UK

British Library Cataloguing in Publication Data
Misra, Kundan
OSS for telecom networks : an introduction to network management. - (Computer communicat and networks) 1. Telecommunication systems - Management 2. Management - Computer progrions I. Title :ams
621.3'85'068
ISBN 978-1-85233-808-4

Library of Congress Cataloging-in-Publication Data
Misra, Kundan.
OSS for telecom networks : an introduction to network management / Kundan Misra.
p. cm
Includes bibliographical references and index.
ISBN 978-1-85233-808-4 ISBN 978-0-85729-400-5 (eBook)
DOI 10.1007/978-0-85729-400-5
1. Telecommunication systems—Management—Data processing. 2. Telecommunication systems—Quality control. 3. Computer systems. I. Title: Operational support systems for telecom networks. II. Title.

TK5102.84.M57 2004
384'.068'4—dc22 2004041727

Computer Communications and Networks ISSN 1617-7975

ISBN 978-1-85233-808-4

springeronline.com

Originally published by Springer-Verlag London Limited in 2004

Typesetting: Gray Publishing, Tunbridge Wells, UK

34/3830-543210 Printed on acid-free paper SPIN 10956991

Preface

This is a book about management of telecommunications networks. Our goal is to explain what is involved in day-to-day network management. In particular, we answer the questions:

1. What is a network management operational support system (OSS)?
2. What does network management OSS software do?

Telecom technologies are addressed at a high level only, to maintain the focus on network management OSS. Therefore, the book will be useful to those without an engineering or telecom background who just want to learn about network management OSS. At the same time, the book will be useful to engineers who are familiar with some areas of telecommunications but not with network management OSS.

In preparing the book, we have kept in mind the author's experience of not many years ago when, with no knowledge of telecommunications, he joined a network management OSS vendor. At that time, the author keenly felt the need for a book on network management OSS which could clearly introduce network management OSS and explain how it worked. He also observed that in order to get these basic questions answered, he was required to read hundreds of pages on telecom technologies such as asynchronous transfer mode (ATM), synchronous optical network (SONET), code division multiple access (CDMA), etc. In fact, as a telecom and network management OSS beginner, it was difficult to find any suitable book that focused on network management OSS.

As far as possible, we spare the reader details of specific network technologies. Of course, some introduction to specific network technologies is necessary, because it is hard to talk about any type of management without some knowledge of what is being managed. We give an especially long introduction to ATM technology as it is such a dominant transmission technology.

We cover many excellent OSS products in this book but acknowledge that others have been left out. We do not claim that this book is comprehensive in describing the best OSS products on the market, although the ones that we do describe are among the best. Many OSS vendors were extremely helpful when approached for assistance. We received more offers to examine particular products in detail than we had time to accept. Many more vendors assisted with and influenced this book than the number of products that are explicitly described. We strongly recommend that

service providers who are considering purchasing an OSS system closely research the companies mentioned in this book, in addition to the other OSS vendors which we did not have time to mention or discuss. We hope to cover more products and technologies in future editions.

Acknowledgements

The editors of two publishing companies drove this book by providing encouragement, support and the occasional gentle prodding to the author. I heartily thank both Helen Callaghan and Jenny Wolkowicki of Springer. It has been a lot of fun working with both of you and I hope that we can do it again. Thank you Marjorie Spencer and Jessica Hornick for your smarts and energy!

I had the inspiration for this book while gazing at a contract law textbook belonging to Elizabeth Dobbin, which was written by one of Australia's pre-eminent law firms. I thought, "What an achievement! Could we do the same thing for OSS?" Elizabeth encouraged me and soon I had a contract for the book ... the first of two.

I was introduced to OSS in a trial by fire at Clarity International, where I met many hardworking and switched-on people who continue to make a big impact on the telecom industry through OSS software: Angela Dickensen, Peter Georgopoulos, Kon Georgopoulos, Amanda Stevens, Katie McGeachie, Steve Parry, Bruce Creswell, Kylie Palmer, Nick Lochrin and many others but, above all, the inimitable Tony Kalcina, CEO – a brilliant communicator and a great leader.

Many other companies and people at those companies played a big role in the writing of this book: first of all, AdventNet, especially Sridhar Vembu, CEO. Sridhar's company provided massive help. Special thanks to Ashutosh Agarwal and Dev Anand for your efficiency and enthusiasm.

I list the following in alphabetical order, as so many people were forthcoming with their knowledge and creativity:

- ADC, especially Trish Christo and Amy Ward
- BoldTech Systems
- CDMA Development Group, especially Khevin Curry
- Cisco, especially Olivier Dahon
- Comnitel Technologies
- Cramer Systems, especially Robert Curran and Brian Buggy
- Flavio Spolidoro of Empresa Brasileira de Telecomunicacoes, Brazil, and Noemi Rodriguez of Departamento de Informatica, PUC-Rio, Brazil
- GE Smallworld, especially Barnaby Perks
- Granite Systems, especially Justin van der Lande
- Information Processing Limited, especially Paul Tinkler and Mike Johnson
- Intrarom SA of Romania
- Martin Group, especially Dudley Johnson

- the MISA project, especially Joan Serrat
- the great people in MISA's successor project, WINMAN, including Alex Galis at University College London and especially Bruno Mayersohn and Anat Schwartz of Team Telecom International Ltd (TTI)
- Net Predict especially Bjorn Frogner
- Open Telecommunications, especially Stephan Wickham, Carl Hampel, Bill Rogers, Alastair Dodwell and David Morton
- Syndesis, especially Lynne Godfrey
- Veaceslav Sidorenco of the Technical University of Moldova
- Telcordia, especially Pamela Hakim, Sharon Martin and Alex Poylisher, who also had several long talks with me at the University of Warwick
- Telemanagement Forum
- UMTS World, especially Petri Possi – thanks for the laughs and I hope to see you in Australia some time soon
- Varros, especially Al Syed
- Visionael, for stimulating conversations with several of your staff at two Telemanagement World conferences.

Shravani Dutta lent her infectious enthusiasm during the latter part of the project. Derwyn Rowlands gave encouragement at a critical juncture midway. Berndt Farwer of the University of Hamburg lent camaraderie. Berndt also gave me a sense that "we're in this together" while he was approaching the finish line for a book on which he was independently working.

Anushka Rao checked the Preface and Acknowledgements, and was always a joy to discuss the book with.

Many people at the Department of Computer Science at the University of Warwick assisted directly and indirectly. My PhD supervisor Saraswati Kalvala and personal advisor Raja Nagarajan showed nerves of steel while watching me write this book and a PhD simultaneously. Ananda Amatya and Ranko Lazic cheered me along the way. The prolific Doron Peled helped me enormously in the initial approach to Springer.

Springer's anonymous reviewers gave many clear and insightful comments on the manuscript at several stages. Vishal Thakur additionally reviewed the book with a super-finetoothed comb. Vishal was supported by Manmeet Singh, who smoothed the review process. The broad general engineering knowledge and massive technical writing experience of all reviewers gave me confidence in the book from an early stage.

Kundan Misra
April 2004, London

Contents

1 Introduction

If you are involved in the telecommunications industry in any way, then you have picked up the right book. As telecom networks and, in particular, network elements become more automated, telecom businesses will increasingly be run through their operational support systems (OSS). The management of telecom networks at an operational level is also becoming more automated and the locus of this automation is OSS software.

The trend towards automation is motivated by the same issues as automation in any industry. If software can be made to do what a human does, then money will be saved. In addition, telecom service providers are finding that they need to put more value and differentiation into their services in order to drive revenue. This often requires more complexity in services. To offer complex services in scale, software must be in place to manage those services. Human intervention can only be necessary when exceptional situations arise. Hence it is easy to see that OSS software will play a key and increasing role in driving revenue for telecom service providers in the future.

Keith Willits, Founder of the Telemanagement Forum, said at the 2002 Telemanagement World Conference in Nice that the telecom industry is increasingly like the automobile industry. The telecom industry is highly automated and operational control is becoming more centralised. To achieve efficiency, scalability and cost control, management and control of operations is being delegated as much as possible to OSS software.

What *is* this OSS software? What does it look like? What can it do? What do its users do all day? How can it be extended? We seek to de-mystify OSS software in this book by showing readers what it is all about.

We shall also describe and discuss some of the goings-on amongst OSS software vendors, such as partnerships, evolution of products and the basic features of an array of OSS products.

A common refrain in the OSS industry is that "there are no green fields!" This means that networks are never built from scratch but, rather, incrementally evolve over many years. Correspondingly, their management systems must also incrementally evolve and accumulate. The result is that most telecom service providers have a multiplicity of management systems which do not talk to each other. There is a need for humans to manually take information from one system to another ("swivel-chair integration"). This raises the issues of end-to-end management and unified management.

K. Misra et al. (eds.), *OSS for Telecom Networks*

1.1 End-to-end Management and Unified Management

End-to-end management is the ability of a system to manage all steps of a process, starting with the customer order, continuing through to activation of the service, regular billing of the service and termination of the service. Many service providers have different systems for these steps and need humans to "glue" them together.

Service providers are finding that integrating their various systems can save much money. Equally important, system integration allows the service provider to do things they could not do before, such as telling customers that their service will be available within a few hours, rather than days or weeks. Of course, in integrating systems, interfaces are needed and common standards are preferable so that custom interfaces need not be built for every pair of systems.

Expensive integration work is currently the norm in achieving end-to-end management. However, organisations such as the Telemanagement Forum encouraging the adoption of standards. They are also facilitating joint efforts between OSS vendors and service providers to develop, implement and adhere to interface standards so that software from different vendors can be clicked together like Lego blocks.

Unified management is about using a single system to manage network infrastructure. For example, many network operators have many systems for their network inventory – and some of those systems include Excel spreadsheets, Post-it notes and assorted other paraphernalia. Yet, there is great value in having network inventory in a single system. For example, end-to-end service provisioning is not possible without unified inventory management. The provisioning and activation of appropriate network resources cannot be automated unless all network inventory is available and stored in a single format or, at least, is in electronic form and accessible from a single system.

1.2 Standardisation

It can never be expected that a single system or approach will solve all problems, or even be appropriate. This is especially so for network management software because telecom networks are constantly growing and changing. The business and strategic expense of having ineffective management software or of being unable to manage the latest telecom network technology can be enormous. So extensibility is a must. Many OSS vendors believe that a stronger case can be made for their products' value-add if they can ensure that their software is "future-proof" to some degree. This is where application programming interfaces (APIs) enter the picture. Many OSS systems implement APIs and examples will be described in this book.

2 Network Management OSS Overview

2.1 Introduction

Operations and maintenance (O&M) preceded modern operational support systems (OSS) software and its associated network management model. O&M referred to control and supervision of telecom networks. O&M has now developed to the extent that it is included under the head of network management, which is broader than merely control and supervision. It also includes fault management, traffic management, service provisioning and service management.

The network management OSS software industry is growing because carriers are realising that O&M, under the broader meaning, is critical to their capability to offer diverse services and to extract maximum return on investment (ROI) from their infrastructure. The capital investment required for advanced network management systems is a fraction of that required for network infrastructure. At the same time, a network management system must dovetail with operational processes. Introducing a new network management OSS often causes an operator to see that their operational processes also need an overhaul. Changing processes requires serious management commitment because it involves the way in which people do their work on a daily basis.

Business process re-engineering (BPR) can be expensive, especially in terms of the time required of upper management. The re-engineering gurus are the first to admit that re-engineering is very hard. However, by streamlining processes and eliminating unnecessary costs, the eventual cost saving from re-engineering can be huge. Further, a successful BPR project in a telecom operator will provide a solid base from which a wide variety of advanced telematic services can be offered long into the future.

On the supplier side, there are now many vendors of network management OSS software offering functionality far beyond what has traditionally been bundled by telecom hardware vendors with their network gear. Telecom hardware vendors have a powerful position after they have made a sale to an operator, and can use this to sell OSS software that is tightly integrated with their hardware. These vendors are also investing heavily in OSS, which they often bundle in sales of networking equipment. A vendor who has won a bid for network equipment, and who then also competes for the OSS component of the bid can deny essential information about their

K. Misra et al. (eds.), *OSS for Telecom Networks*

network hardware to competing OSS vendors. The result is often that vertically integrated network hardware/OSS vendors can shut smaller vendors out of OSS bids, reducing competitiveness in the OSS space and perpetuating a vertically integrated structure in the telecom industry. Such tactics are not good for the telecom industry in the long term, because carriers will often be denied the opportunity to consider seriously the product offerings of the specialist, non-vertically integrated OSS vendors.

The majority of OSS systems are developed in-house by telecom carriers themselves. However, the "buy or build" debate has gained prominence. This is the question whether OSS software should be bought "off-the-shelf" or developed ("built") in-house. Carriers are beginning to realise that the expenditure required in developing and maintaining an OSS in-house is in the medium-to-long term far greater than the cost of buying an OSS from a specialist vendor. This is especially so because the requirements of OSS today are far more sophisticated than in the past. It is a drain on resources for a telecom carrier to develop and maintain an OSS in-house because a telecom operator is not primarily a software development firm. Of course, buying an OSS "off-the-shelf" has the downside of not being as customised as an in-house system can be. At the same time, this problem places an opportunity for product differentiation at the feet of OSS vendors. The challenge for OSS vendors is to develop software:

1. That is useful in its own right so that they are not effectively acting as contract software developers by having to write large amounts of custom code each time a telecom carrier wants to buy their product; and
2. That is customisable and extensible to a reasonable extent since no two carriers will have the same network or the same business needs.

2.2 Telecom network operations and maintenance

Operations and maintenance are those business processes which enable the network infrastructure to provide customers with services that are profitable for the carrier. Thus, O&M is essentially the discipline of delivering value and extracting profit from a network infrastructure.

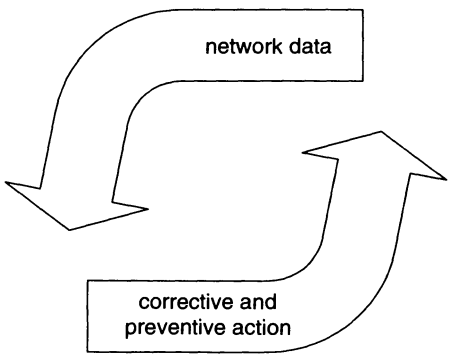


Figure 2.1 Operation and maintenance.

Operations include subscriber management, charging, signalling, traffic management and service management. Of these five, OSS vendors have least control over signalling, because signalling functionality is generally built into the fabric of a telecom network. On the other hand, maintenance involves (a) preventive action, through measurement and supervision, and (b) corrective action, such as finding the location of a fault ("fault localisation") and fault correction. See Figure 2.1.

Effective operation and maintenance together require that the operator communicate with the traffic-carrying network elements. These network elements must be able to inform the operator of alarm details, including the time and location of fault, and the status and performance of the network elements (NEs). The operator must be able to order network statistics and reroute traffic and must have as much flexibility as possible in implementing changes to ensure optimal functioning of the network.

2.2.1 Operations

The first of the five areas of operational management is subscriber management. This includes connecting new subscribers and ensuring that services are provisioned and available. The second area of operational management is charging, which includes collecting billing data. This is the most obvious foundation of the operator's revenue-earning scheme and is carried out several times in any 24-hour period.

The third and fourth aspects of operational management are traffic management and signalling. Traffic management distributes traffic in a way that improves grade of service and network utilisation, while minimising the possibility of network overload. Signalling is used for most traffic switching, and activation and disconnection of services in public networks.

Service management is about management of the other four areas of operational management but at the services level of abstraction.

2.2.2 Maintenance

Preventing faults before they arise requires close supervision and accurate measurement of many network parameters. Systematic maintenance requires simple network monitoring methods and many measurements in order to prevent faults. The network operator must check that parameters, such as bit error ratio (BER) and probability of congestion, are within a tolerance range. This range is set to assure delivery of promised services at the agreed level of certainty.

If a hardware fault is detected, the operator's staff first localise the fault and then allow software to pinpoint the fault. Next, corrective action is taken to eliminate the fault. The appropriate troubleshooting and fault localisation procedure depends on the nature of the fault and the type of equipment. Before modern OSS software, this process was manual, requiring many skilled, and therefore expensive, staff. Nowadays, software agents installed in network elements automate a large part of the process. Further, centralised network management software is equipped with fault correlation algorithms. These process the data received from distributed software agents and reduce the manual effort involved in fault localisation, diagnosis and correction.

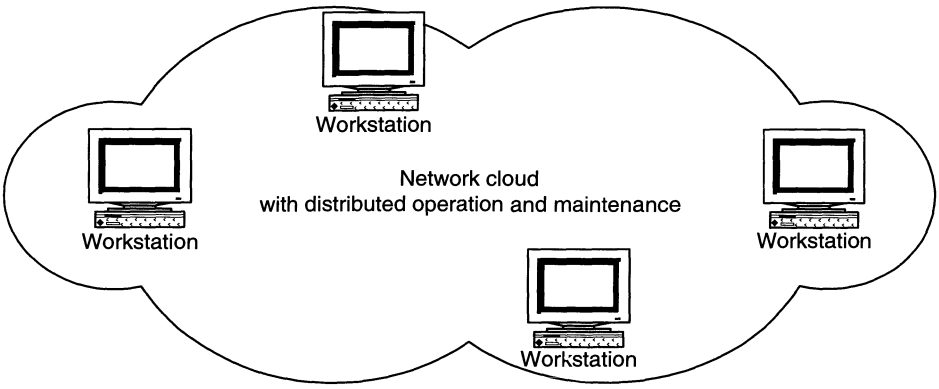


Figure 2.2 Distributed operations and maintenance functionality.

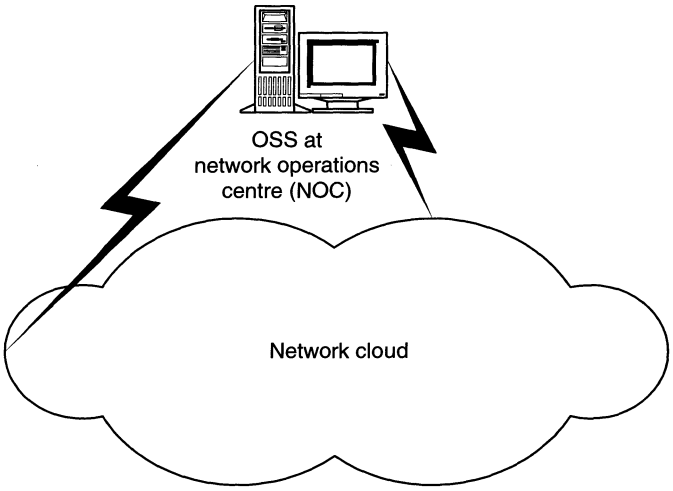


Figure 2.3 Centralised operational management.

In the past, O&M was decentralised, and many operators still prefer this (Figure 2.2). The bulk of an operator's costs are involved in O&M work. Associated costs include staff, decentralised and duplicated facilities and redundancy ("overdimensioning") in the network. Overdimensioning is the surest but most expensive way to ensure a high grade of service. In addition, owing to the large amount of expertise needed to maintain a network because of the usually vast variety of types and makes of equipment in the network, staff costs for O&M are high. Unfortunately, decentralising involves resource duplication, added costs and inefficiencies in communicating with staff.

An OSS of today can control and supervise several sets of network equipment in a centralised management model (Figure 2.3). Some telecom professionals broaden the term OSS to include billing systems and customer relationship management (CRM) software, although billing and CRM are beyond the scope of this book (Figure 2.4).

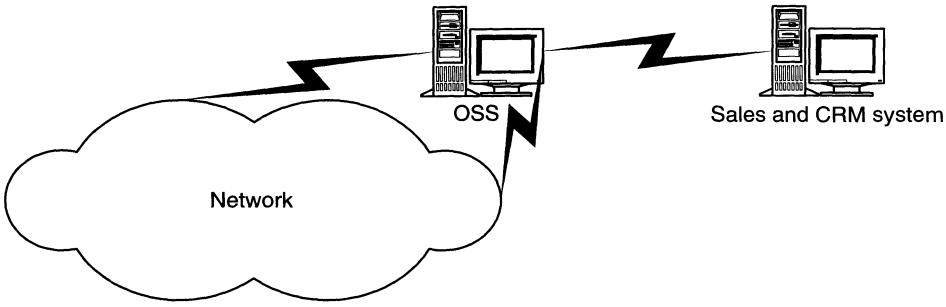


Figure 2.4 Customer management.

Centralised operational management by an OSS has many business advantages for a service provider. Many functions can be efficiently handled from just a handful of locations so that fewer staff are needed. Although the advent of OSS systems allowed centralisation.

The benefits of centralised operational management have induced many companies in the telecom industry to invest in OSS software. Several industry conferences are devoted to industry-wide collaboration for improving OSS. OSS industry bodies, in particular the Telemanagement Forum (TMF), organise many of these conferences. Some TMF initiatives will be discussed in this book.

2.2.3 Customer Management

Customer management encompasses the management of charging information and subscription management. These functions include:

- Provisioning and termination of services according to new and expiring subscriptions; and
- Customised services, which are particularly sold to corporate customers.

Sales staff communicate with a customer database from which instructions to the OSS are generated. The OSS then carries out the work needed to deliver services by sending commands to the software installed at the local exchanges. For example, a command might be to activate a service for a new subscriber.

2.2.4 Managing Circuit-switched Networks

The operation and maintenance centres that handle the element and network levels are called the Operation and Maintenance Centre (OMC) and Network Operations Centre (NOC) respectively.

Any network will generally have many OMCs but may have only one NOC (Figure 2.5). Further, each OMC may be responsible for several local exchanges and use one or more OSS. The NOC is focused on network management functions such as network monitoring, traffic management and signalling management. The OMCs are focused on element management, the basic functioning of the exchanges, signalling and traffic management, and element mediation.

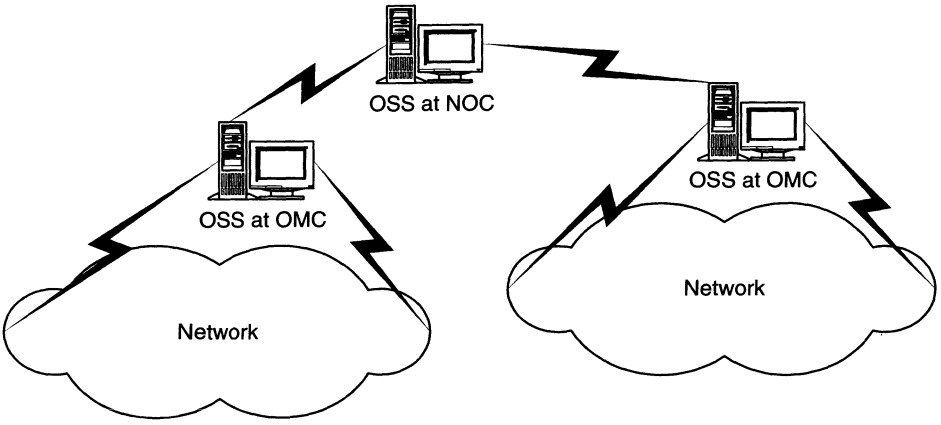


Figure 2.5 Element and network management.

2.2.5 Element Management

The basic functions of an OSS include:

- drawing network maps for the use of human staff, from the equipment database
- alarm management
- command management
- translation of operational commands into communication protocols.

The network map gives network management staff an overall view of the network. Network maps are usually hierarchical with the highest level showing the entire network. The user can drill down to see details of local areas. The highest level gives a survey and also shows critical alarms and NEs represented by symbols. Different severity levels for alarms issued by telecom equipment in the network are indicated through colour and symbols. An alarm is a signal issued by a NE that some intervention is needed for the NE to return to normal functioning. The severity of an alarm depends on the risk that it poses to service. For example, failure of the primary power supply to a switch might have a lower severity level than a multiplexer failure.

Alarm or fault management include receiving, processing and displaying alarms as they are received from exchanges and other network equipment. The Telecommunications Management Network (TMN) model of the telecom standards section of the International Telecommunications Union (ITU-T) defines alarm levels (Table 2.1).

Table 2.1 Alarm severity levels.

Severity level	Description
Critical	Immediate action needed
Major	Act as soon as possible
Minor	Watch situation carefully
Warning	Investigate during next scheduled maintenance

Colour-coding of alarms allows the severity level of an alarm to be determined immediately. An OSS should allow network operations staff to drill down to more detailed information about a particular alarm. All alarms are logged and stored for later reference or printing. Alarms can be:

- Unacknowledged or acknowledged by network operations staff: an acknowledged alarm has been recognised and some corrective action has been taken, such as the issue of a "trouble ticket" which causes a technician to visit the site of the fault in the network; or
- Resolved or inactive: an alarm with resolved status has been corrected while an inactive alarm has not deliberately been resolved but the alarm has ceased to indicate the original fault.

The staff can control and supervise individual network elements by sending commands to them and interpreting their responses. Management commands can be sent for immediate execution, or they can be timed for execution at a particular future time. The response of the exchange – whether immediate or delayed – is directed to the appropriate destination in the OSS at the NOC through a visual computer display, a file, a mailbox, a printer or a combination of media.

2.3 Traffic Management

In addition to the basic functions of managing the local exchanges and managing other NEs, an OSS has functions for traffic measurement and gathering of statistics. Measurement results can be processed, stored and displayed to form the basis of traffic and resource planning performed in an OMC.

Traffic management is simpler at the OMC level than at the NOC level. At the OMC level, the following types of traffic are measured:

- local subscribers to subscribers in the same exchange, or local calls
- local subscribers to subscribers in another exchange, or outgoing trunk calls
- subscribers in another exchange to local subscribers, or incoming trunk calls
- transit calls, which are a through-connection between a pair of subscribers, both of whom are outside the local exchange.

At the NOC level, there are more views of network traffic and more statistics to consider because any NOC is responsible for regions covered by several OMCs.

2.3.1 Measuring Traffic Distribution

Traffic distribution is the relative amount of traffic along different paths, shown in Figure 2.6. In particular, the distribution of traffic along different paths between a particular pair of exchanges will be of interest, such as link 1 → link 4 versus link 1 → link 2 → link 3 between Exchanges 1 and 2 in Figure 2.6. By shifting traffic from one path to another, the traffic distribution might be evened out while the overall result of traffic passing between Exchanges 1 and 2 is unchanged.

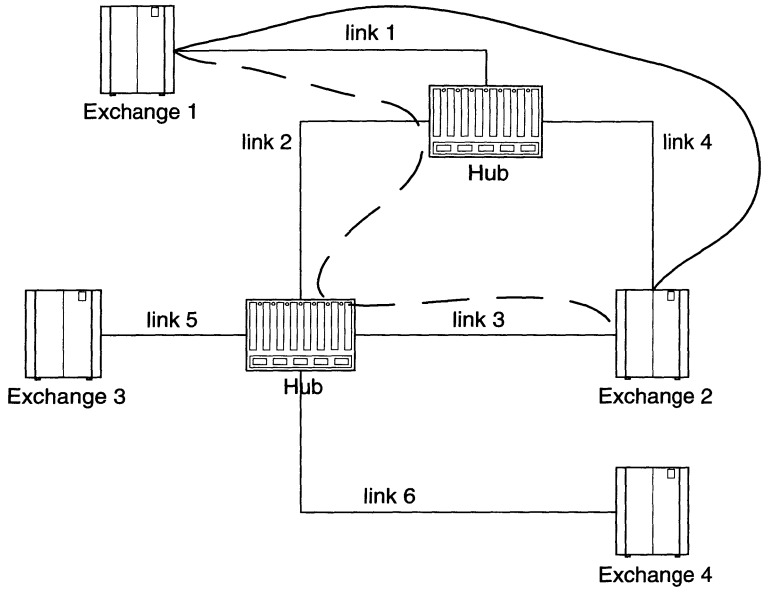


Figure 2.6 Alternative paths for the same endpoints.

Most exchanges have traffic measurement functionality. Measurement is initiated by the OSS and log files are received and processed by a traffic management module in the OSS where the results can be displayed graphically or as raw numerical data. Apart from traffic monitoring, OMC functions include:

- data registering
- creating and maintaining logs of network events
- issuing trouble reports.

These functions simplify routine tasks in an OMC. Data registers are helpful as staff can look up exchange data, find out which software versions a specific exchange is loaded with and figure out the configuration of that software.

Work carried out in an OMC is logged for general audit purposes, so that there is a record of who made which changes and when. The OSS can also support the staff by simplifying the handling of trouble reports. Even without further automation, simply by having text-based fields in software, mundane administrative tasks such as form-filling can be simplified using cut and paste.

2.3.2 Network Management at the NOC

OSS which present the network at various levels of abstraction are becoming more common. These are known as hierarchical OSS, and the NOC is typically installed at a higher level of abstraction than the OMCs (see Figure 2.5). The NOC has different functions from an OMC and has some network management advantages over the OMC. Having a NOC means that OMCs need only be staffed during normal business hours, to avoid paying higher wage rates for unusual working hours. When the

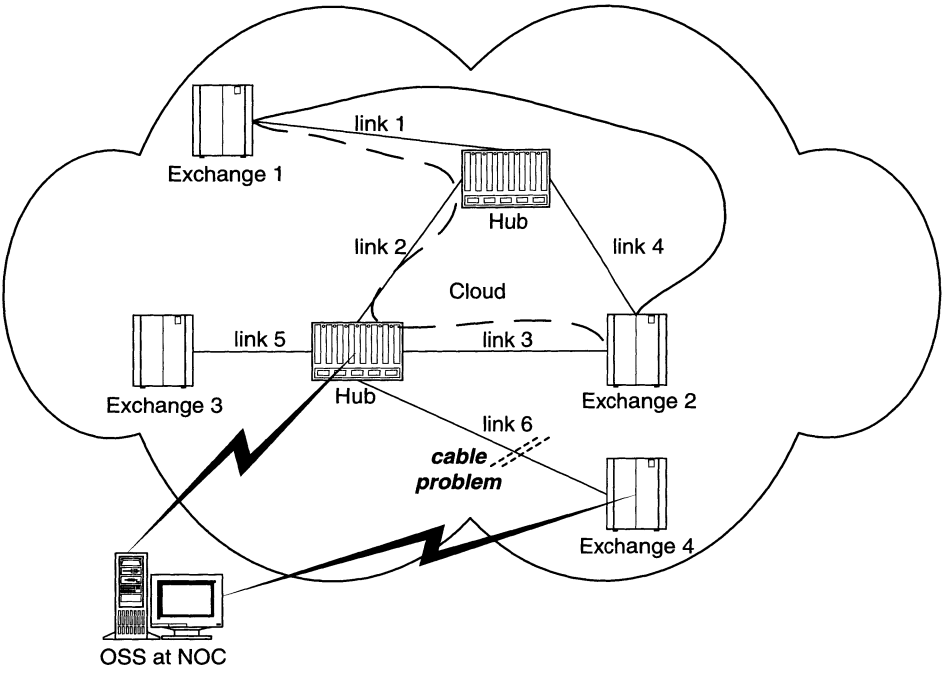


Figure 2.7 Fault detected by OSS at NOC.

OMCs are not staffed, urgent alarms from exchanges managed by an OMC are directed to the NOC.

Since the NOC is connected to all OMCs, the NOC has indirect access to all managed NEs. Thus, staff at the NOC are given an overall view of the telecommunications network, which generally comprises several bearer networks. Because of the dependency of a large number of circuits on bearer networks, a fault in the transport network can affect a large number of services.

The main NOC functions are:

- traffic management, including traffic monitoring and analysis
- network surveillance
- signalling management.

In Figure 2.7, the OSS installed at the NOC has detected a problem in link 6. Combined with information such as traffic data, the cause of the fault may be determined to be, say, a traffic overload in the case of excessive throughput or a cable break in the case of zero throughput.

Traffic data are collected periodically from local exchanges, every 5–10 minutes, and made available to NOC staff. The data are analysed by software installed at the NOC.

Traffic management functionality includes setting thresholds for alarms in exchanges and on paths. It also includes assisting network management staff in making rerouting and redistribution decisions such as restricting traffic along some

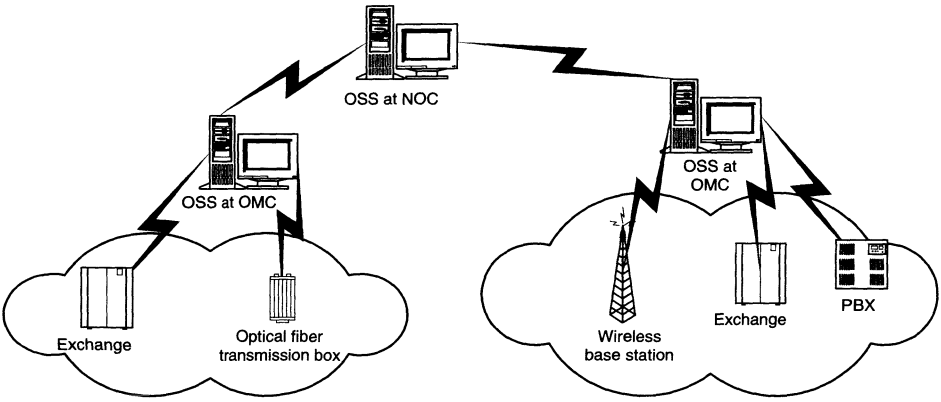


Figure 2.8 Network surveillance – the NOC has an integrated view of the entire network, unlike the OMCs.

paths or through some exchanges. An OSS that is advanced in traffic management functionality would be able to make decisions on the basis of network data that have been collected and, on the basis of those decisions, issue rerouting commands to a local exchange.

Network surveillance includes:

- network supervision
- furnishing network operations staff with an overall view of the NEs and network processes that comprise the network, including switching, transport and signalling.

The OMC presiding over the part of a network in which an alarm originates sends any urgent alarms to the NOC for alarm filtering. The network surveillance function at the NOC compiles these alarms and often applies rule-based analysis to determine action to be taken. This rule-based analysis is part of the expanding domain of policy-based management (PBM).[1]

2.4 Management of Transport Networks

The last two decades have seen great strides forward in operators' ability to manage and control the transport network efficiently. Manageability was improved by the introduction of the synchronous digital hierarchy (SDH) transport standard. Network elements based on earlier technology were inflexible to the extent that service provisioning and activation typically took weeks.

The predecessor of SDH is called plesiochronous digital hierarchy (PDH). Plesiochronous means "almost synchronous". In particular, the difference between two signals is constrained to some limited time period.

The introduction of SDH included new flexible and controllable NEs, such as digital cross-connects (DXCs) and add/drop multiplexers (ADMs). The importance of

[1] During 2001, the author participated in a Telemanagement Forum "tiger team" developing guidelines for policy-based management mechanisms for OSS.

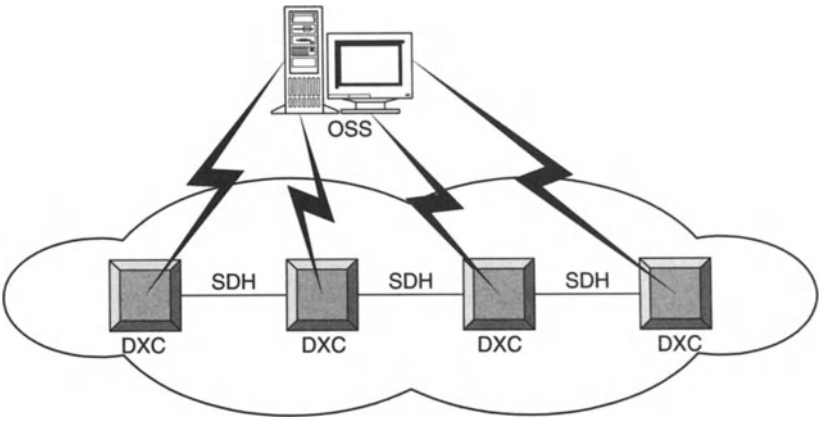


Figure 2.9 SDH network.

these developments was not lost on OSS vendors such as Clarity International, whose products have notable explicit support for these technologies. This will be seen in discussion of features of the Clarity Configuration Manager product in Chapter 11. Like many modern OSS, Configuration Manager embraces the combination of SDH, DXCs and OSS functionality.

The advantages of SDH technology include:

- fast provisioning and activation of capacity from a central OSS
- automatic rerouting in case of a cable break, or some other transport problem
- round-the-clock quality of service (QoS) monitoring
- the possibility of optimizing route selection in the network after the network has been analysed for bottlenecks, and for over-sensitivity to particular kinds of disturbance.

As mentioned in the Introduction, telecom networks nearly always include a combination of technologies through their evolution over several generations of transmission technology. Therefore, PDH often appears alongside SDH in the same network. Modern OSS must support NEs that are based on SDH and DXCs, but they can also communicate with PDH equipment using appropriate mediation devices. In fact, it is even possible to continue using the OSS that was in place before SDH was implemented. In that case, the older OSS would communicate with the newer through some interface. This unification of network management systems, including OSS, has been an important contribution of OSS vendors and their system integration partners over the last decade. There are still many service providers who would enjoy great cost savings if they carried out such a unification project. Such a project should be carried out in partnership with an OSS vendor with the appropriate software. Such an OSS vendor should also have experience in building interfaces between inherited (or "legacy") management systems, and their own OSS product.

Upon sale of an OSS software product to a telecom carrier, an OSS vendor will commonly find a range of network hardware in use by the carrier which the carrier

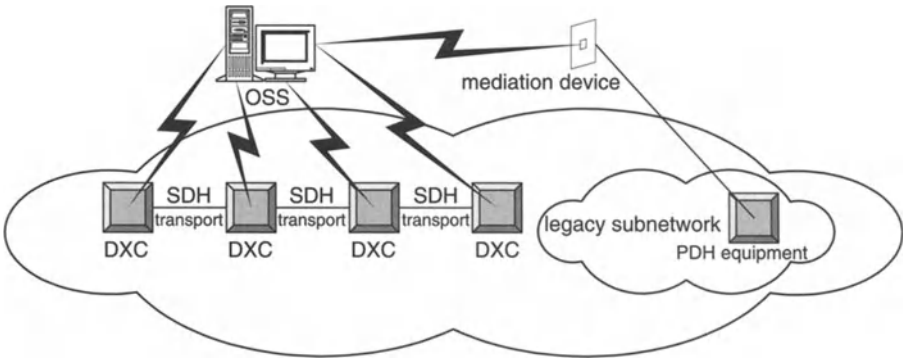


Figure 2.10 SDH network with legacy PDH subnetwork.

wants to continue to use. Such hardware may even use a proprietary protocol which is outside industry standards. The OSS vendor will then set to work building an interface to that hardware. If the carrier still has the documentation for that legacy hardware, then the task of building the interface is made easier. If the carrier does not have the documentation, then the specifications for its protocol need somehow to be worked out, and there are methods for doing this although it can be time consuming and expensive. OSS vendor Clarity International has done just this for many carriers the world over. The carrier must weigh the value of keeping the legacy hardware, often more than 20 years old, with the cost of building custom interfaces so that it can be brought within the purview of a new OSS system.

2.5 Configuration Management

OSS functionality for configuration management includes:

1. installation and configuration
2. path and capacity implementation
3. network protection.

In the case of installation and configuration, an OSS chiefly includes support functions for installing network equipment.

Many support functions seek to provide NEs with accurate input data and to maintain the OSS inventory database. This picture of input data and equipment is used by the OSS in calculations that are made when traffic is rerouted or when leased lines or other telecom traffic-carrying capacity are installed.

When equipment is installed in the network, the OSS database is updated with an indication of service status, which says whether or not the equipment is free or blocked. The database can be synchronised with the current equipment status, either using incoming SDH or PDH signals or using a clock.

When a leased line is to be installed, the operator specifies the termination points, which are called the "A end" and "B end" in the terminology of some OSS systems.

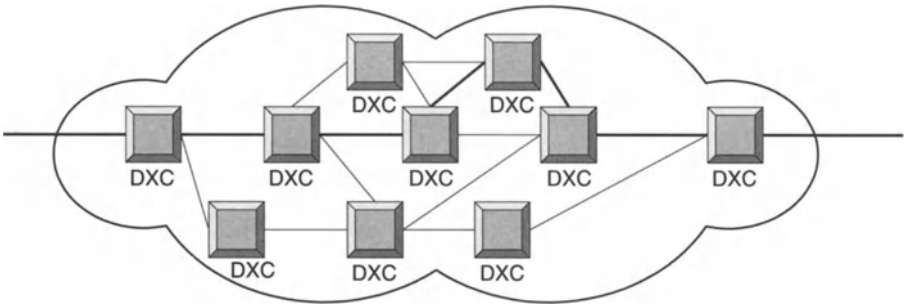


Figure 2.11 Path between specified endpoints.

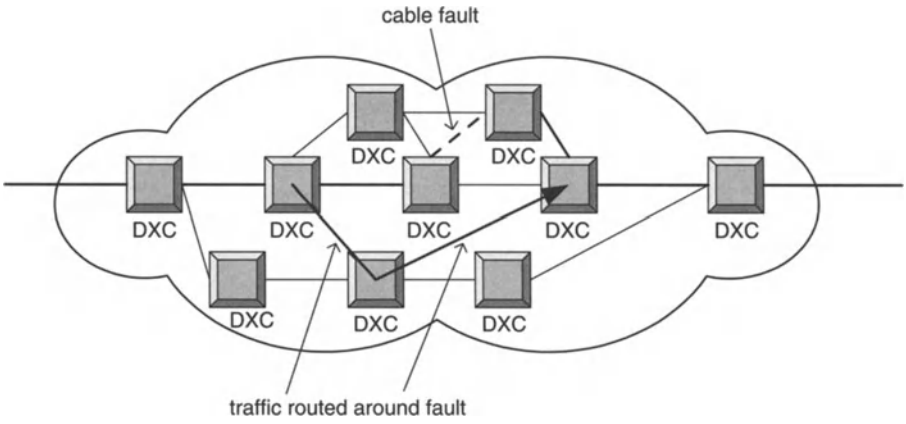


Figure 2.12 Traffic routed around cable fault.

In calculating the path, the OSS uses an algorithm and all available network data – including factors such as obtainable resources and network quality – to calculate the optimal path between nodes. The operator can specify a cost parameter for any given path and this will be taken into account by the OSS in suggesting an optimal route. In Figure 2.11, we see that the route suggested is not necessarily the most direct route. This might be because the more direct route has already been provisioned for a higher paying customer, or because some links on the more direct route are already operating at a high percentage of their capacity.

The OSS suggests a path and, if the operator accepts it, then the OSS sends the relevant commands to the NEs concerned making the line ready to use.

Using time-related rerouting, the operator can specify that one path be used during certain times of the day, week or month while another path be used at all other times. More complicated time-related routing specifications could also be defined. Most configurations include network protection options to enable traffic to be rerouted in the event of a cable fault, while not breaching other service parameters imposed on network traffic.

In some networks, the NEs detect a cable break and send that information to the OSS. The OSS then calculates that particular paths have been affected (namely

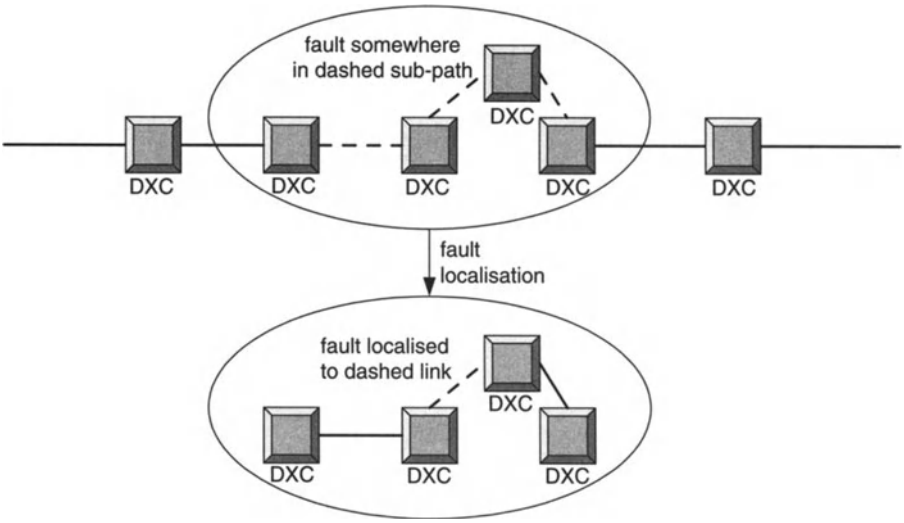


Figure 2.13 Fault management function – localisation of the fault in Figure 2.12.

the paths between the NEs which detected the cable break) and the OSS will send a command to reroute traffic. Some OSS can be programmed to suggest the best stand-by path at the time the main path is provisioned or after a fault occurs.

Situations in which the quality of a leased line proves to be too low may also occur. The OSS then triggers an alarm before any rerouting is initiated. A command by the OSS to NEs to reroute traffic is generally executed within seconds.

2.6 Fault Management

Fault management comprises network monitoring, fault isolation and network testing. Staff at the NOC can decide which details of a fault are to be reported to their manager, reported to the customer and stored in case there is a need for future reference. Staff at the NOC also define the events and criteria which will trigger an alarm and how alarms should be displayed on the alarm list and at the workstation. We shall see just how detailed these options can be in the discussion of the fault management product from AdventNet in Chapter 12.

The OSS can localise the fault after an alarm is triggered. If a DXC issues an alarm, an OSS staff member sitting at his/her workstation can detect and identify a faulty printed board assembly, for example by scrolling the network picture for the DXC concerned. A work order would then be raised to replace the faulty board. A different procedure would be more appropriate for an increased BER since increased BER might be caused by a fault in a signal regenerator. Once it is known that the BER has increased, measurements would reduce the number of candidates for faulty regenerators. The OSS would automatically trigger a battery of tests or diagnostics on the candidates to determine exactly which is faulty and what is the fault.

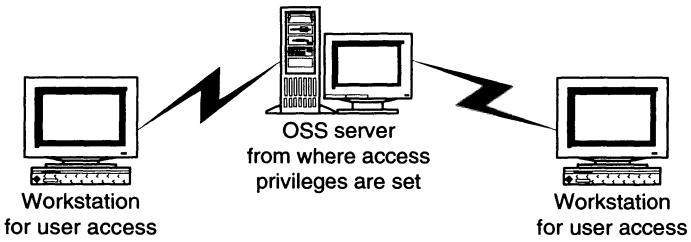


Figure 2.14 Security management.

2.7 Security

Security administration involves restricting the use of the OSS and other network management resources. Network administrators can specify security requirements using additional OSS features. We shall see some security administration features available in some OSS implementations mentioned in this book.

Security is implemented as in any multi-user software package. Each user is assigned to a group, each of whose members has default access privileges. For situations when the network is accessed simultaneously by staff at different locations, such as at the NOC via the OSS and at a workstation terminal at a local exchange, a locking mechanism is necessary to avoid conflicting instructions being given to network equipment. There must also be protection against interference from the embedded communication channel (ECC) of SDH. Data flowing in the ECC are part of the telecom network traffic. It must be separated out so that it does not have the effect of issuing unintended and unwanted commands to the OSS.

2.8 Network Planning Support

The OSS can support the operator in the modification, extension or design of networks by evaluating capacity limits. Simulation helps in network planning and design. To determine the new or modified network capability, the OSS can simulate network behaviour in specific traffic situations. This is done to determine how the network will behave in such situations. The OSS might also be able to simulate how services will be impacted by a fault.

Simulation can also find unused capacity in an existing network. It can be used to find design flaws in an existing network that are responsible for recurring performance problems such as bottlenecks.

2.9 Summary

The purpose of this chapter was to introduce the main aspects of OSS that will be covered in later chapters. An overview was given of each of the following topics:

- telecom network operations and maintenance
- traffic management

- management of transport network infrastructure
- telecom network configuration management
- fault management for telecom networks
- telecom network security
- support for telecom network planning.

3 ATM Network Management – The ATM Forum Model

3.1 Introduction to Asynchronous Transfer Mode (ATM)

Asynchronous transfer mode (ATM) is a cell-based transmission or transport technology. Cell-based means that information is carried in packages (or "cells") of some particular size. Transmission or transport is the carrying of information through a telecom network from a sender to a receiver. ATM works by setting up a connection between the endpoints for the duration of a communication session. It emerged with the aim of becoming the basic infrastructure for all broadband communications. ATM integrates essential features, such as QoS and traffic loss prioritisation. ATM scales to very large telecom networks without a loss in quality. The scalability of ATM along with its widespread implementation has made it a "future-proof" option for telecom network infrastructure.

ATM is still the technology in most widespread use around the world for the core of telecommunications transport infrastructures.[2] Nonetheless, network management remains a major challenge faced by ATM technology. ATM networks require more management procedures than pre-ATM networks. For example, ATM management does not concentrate only on the physical level, that is, monitoring the traffic through a wire or the status of a switch port, with a view to dealing with defects, failures and malfunctions in wires, switches and other hardware. In fact, ATM focuses mainly on the logical level, with the overall goal of offering the preferred QoS and traffic control mechanisms. Further, management procedures used in traditional local area networks (LANs) allowing resource sharing are not sufficient for managing very high-speed virtual channel connections, which does not bode well for the future. Finally, ATM administration must address implications of multimedia networks carrying voice, data, video and audio.

Tasks related to ATM management are complicated by the complex and heterogeneous nature of the networking environment. In fact, most current networks comprise diverse equipment from different vendors, and new technologies and services add to the complexity of the network. Therefore, ATM management applications must support the complex procedures imposed by ATM technology, while at the same time being suited to composite networks. Note that heterogeneity issues do

[2] For example, see Tim Greene's *Network World* article that appeared in my email Inbox on 5 September 2002: "ATM still rules the core".

K. Misra et al. (eds.), *OSS for Telecom Networks*

arise in purely ATM networks, from differences in implementations of ATM technology in equipment from different vendors.

The Network Management Working Group of the ATM Forum has developed a structured approach to managing ATM networks in a layered model. This model is designed for both public and private networks, and it includes interfaces to:

- Simple Network Management Protocol (SNMP)
- Common Management Information Protocol (CMIP)
- Proprietary management systems.

The model also allows for distribution of management information over ATM networks through Operations and Maintenance (O&M) cells. Several other management models have been independently proposed as potential solutions for integrated management, most notably the TMN model.

Vendor independent network management is an ongoing goal of the OSS community. "Vendor independence" refers to independence from which telecom network hardware is implemented by a carrier. An OSS should have the same functionality regardless of whether Cisco, Nortel, Siemens or any other company is chosen for the physical hardware. Otherwise, different hardware requires a different OSS and installing new hardware from different vendors leads to a patchwork of OSSes. Vendor independence can be achieved by getting hardware vendors to implement a common protocol through which OSS software can talk to the hardware just as all computers on the web can talk to each other because they all use TCP/IP.

A protocol has been proposed to achieve vendor independence in managing ATM switches, called the General Switch Management Protocol (GSMP). Vendor independence is achieved when functionalities provided by some technology are provided by equipment regardless who supplies that equipment. However, GSMP is limited in scope to a small set of the capabilities of ATM switches. Distributed approaches, such as Common Object Result Broker Architecture (CORBA) and Telecommunication Information Networking Architecture (TINA), also seek to remedy the problem of heterogeneity technology implementations by different vendors.

3.2 ATM Configuration Management

ATM has virtual paths (VPs) for telecom network traffic. VPs are decoupled from physical transmission paths and physical interfaces. ATM has a double-multiplexing structure which allows virtual trunk routes to be created at the VP layer of an ATM network. These VP routes can carry traffic in the virtual channel (VC) layer from different end-users. Indeed, many thousands of VCs can be multiplexed on to any VP.

The physical and virtual layers of ATM can be mapped on to a functional architecture with:

- trail termination functions *within* layers that are independent of payload
- functions *between* layers that are also independent of payload.

A single link of an ATM virtual connection is called a "connection".

A virtual path connection (VPC) comprises many virtual path links (VPLs) connected end-to-end. Each VPL may run over any of a variety of technologies, such as

Table 3.1 Summary of ATM channels and links.

Link/connection	Comprises
Virtual path connection (VPC)	VPLs end-to-end
Virtual path link (VPL)	A link on one of a variety of technologies, such as ADSL, SDH or PDH
Virtual path connection (VPC)	Bundle of many VCLs
Virtual channel connection (VCC)	VCLs end-to-end

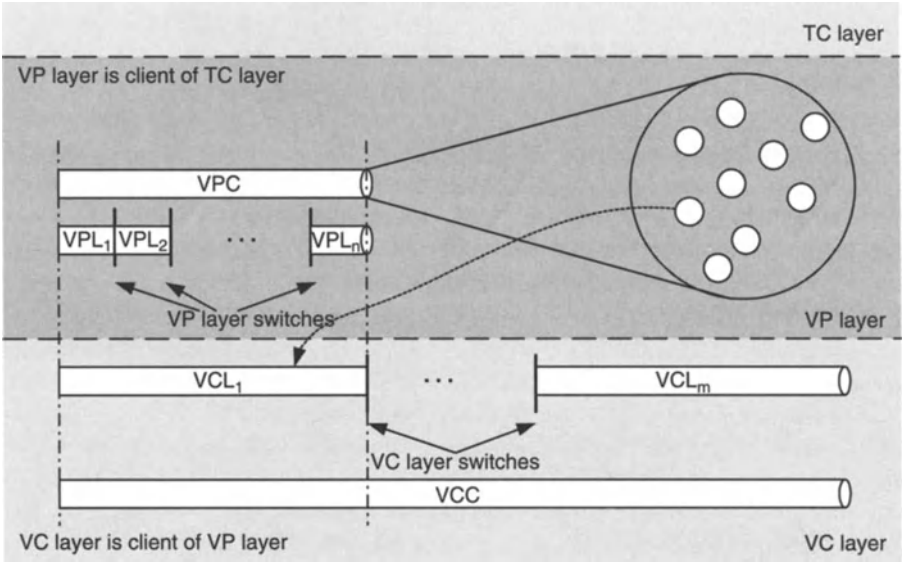


Figure 3.1 ATM channels and links.

asynchronous digital subscriber line (ADSL), SDH or PDH. VPLs are connected to one another by VP layer switches. Each VPC contains a bundle of many virtual channel links (VCLs). A number of VCLs connected end-to-end forms a virtual channel connection (VCC). VCLs are also connected to one another by VP layer switches. The relationship between the different type of ATM channels and links is summarised in Table 3.1 and Figure 3.1.

The transmission convergence (TC) layer inserts and deletes cells that match the VP traffic to the physical transmission. Each cell has a header which includes information about the cell. The TC layer uses the header error correction (HEC) field to detect and, if possible, correct header errors. Cells are discarded when their header errors cannot be corrected.

The VP layer is the client of the TC layer. The virtual connections of the VP layer are called VP layer *trails*. The VP *adaptation* functions encompass:

- multiplexing and demultiplexing the virtual connections of the VP layer
- handling the O&M flows for the VP layer.

The trail termination functions of the VP layer handle the end-to-end O&M flows. Those VCs that carry O&M flows also terminate in the VC layer.

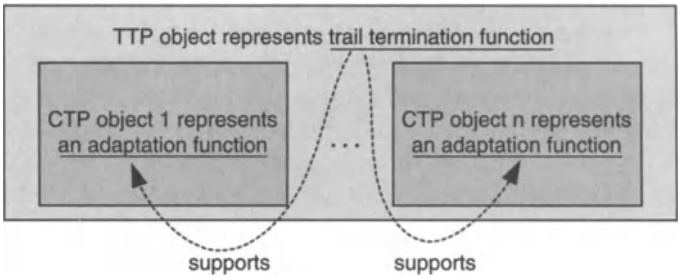


Figure 3.2 Relationship between TTP and CTP objects.[3]

Table 3.2 Sameness of TTP and CTP objects at various layers.

Layer	TTP	CTP
TC	◆	♣
VP	♠	♣
VC	♠	♣

Now, the VC layer is the client of the VP layer (although the "C" in VC is for channel, not client!). The virtual connections of the VC layer are also called trails. VCs carry their own O&M flows.

Trail termination points (TTPs) represent trail termination functions in the ATM functional architecture. On the other hand, connection termination point (CTP) objects represent the adaptation functions – whether in the VC layer or VP layer. The adaptation functions multiplex and demultiplex trails in the VC layer.

Trail termination functions support adaptation functions and so TTP objects contain CTP objects (see Figure 3.2). The TTP objects of the VC and VP layers are similar and the CTP objects of those layers are also similar. On the other hand, the TTP objects of the TC layer are different (see Table 3.2).

TTP objects in the TC layer raise an alarm if the trail termination functions that they model cannot extract cells from the incoming physical transmission. To activate or deactivate the scrambling of ATM cells on a physical link, the operations system of an ATM switch must vary the appropriate attribute for the link in its TTP object in the TC layer.

There are often limits on the VC/VP identifier of ATM connections. The capacity that is available for new connections is also limited. ATM access profiles describe either:

- the profile of the VPs that are supported by a trail on the TC layer, or
- the profile of the VCs that are supported by a trail on the VP layer.

Access profiles are contained in the appropriate TTPs.

The operations system of an ATM switch performs a "*get*" operation on the profile object within a TTP object to determine the number of identifier bits and ATM connections that are supported. To discover the TTP object to which an interface refers, the operations system of an ATM switch can perform a *get* on the pointer attribute.

[3] The three dots at the middle of the diagram are an ellipsis.

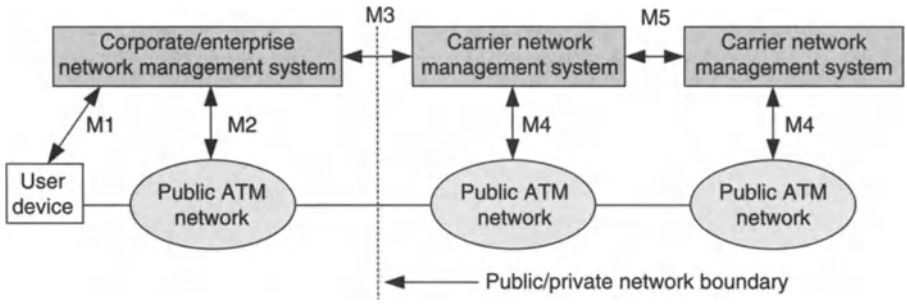


Figure 3.3 ATM Forum Management model.

Each ATM cross connection is represented by a cross connection object. These objects are contained in the fabric object that represents the cross connection matrix. To create a cross connection object, the operations system of an ATM switch sends a *connect* action to the ATM fabric. This action identifies the CTP objects at the ends of the connection or includes a description of the ATM traffic if there is no CTP and the latter allows an appropriate CTP object to be created. To remove a cross connection, the operations system sends a *disconnect* action to the ATM fabric.

3.3 The ATM Forum Management Model

The Network Management Working Group of the ATM Forum has developed an end-to-end generic management model that encompasses private and public networks and lays out standards for interworking between them. The model defines gateways between SNMP and CMIP systems, and between standards-based and proprietary systems. Five key management interfaces are defined in this model, labelled M1–M5.

M1 is concerned with the management of the end-user equipment connecting to either private or public switches. M2 undertakes management of private ATM switches and networks. Private ATM network management is addressed through M1 combined with M2. M4 deals with their public ATM switches and networks. M3 is the link between private and public networks, used for exchanging fault, performance and configuration information. Finally, M5 supports interactions between any two public networks. The definition of these interfaces allows a complete management service, ranging from a global view of the network (M5 management interface) to the management of individual elements (M1 management interface).

In some cases, several management interfaces use the same information from a management information base (MIB) tree.[4] We now elaborate on the management interfaces but omit M5 as significant work remains to be done on this standard (also see Table 3.3).

[4] See Chapter 4 on SNMP for a quick introduction to MIBs.

Table 3.3 Management interfaces.

Management interface	Description
M1	Interface between corporate NMS and user device
M2	Interface between corporate NMS and private ATM network
M3	Interface between NMS
M4	Interface between carrier NMS and public ATM network

3.3.1 M1/M2 Interfaces and the ILMI Implementation

We explain the M1 and M2 interface by describing the Interim Local Management Interface (ILMI), which is an implementation of the M1/M2 interfaces. ILMI enables the exchange of status, configuration, accounting and control information between any two ATM devices – such as two ATM switches – across a user-to-network interface (UNI). For ILMI to function, every ATM switch or network terminator and every ATM network that deploys a public or private network UNI must be equipped with a UNI Management Entity (UME) which supports an ILMI MIB. Two adjacent (or peer) UMEs can communicate using the common attributes provided by the ILMI. By sending SNMP commands, a UME may obtain or modify (if the object is indeed modifiable) information contained in its ILMI MIB.

The ILMI MIB is hierarchically organised (Table 3.4). It contains information concerning each group listed in Table 3.4. Also defined are functions that allow retrieval and handling of information in the ILMI MIB. The ILMI has been deployed by some vendors to perform management tasks across the UNI for some devices. However, since the ILMI provides a solution that is applicable only at the UNI, it cannot support the management tasks that are involved in a network comprising a range of ATM devices. Thus, on its own, the ILMI does not provide the capability to manage multi-vendor ATM networks.

3.3.2 M3 Customer Network Management Interface

The M3 MIB defines objects for the customer portion of a public network. The M3 interface, also known as the Customer Network Management (CNM) interface, defines the interaction between the customer and carrier management systems in

Table 3.4 Structure of the ILMI MIB.

Group	Parameters
Address group	Address of terminator
Network prefix group	ATM address prefix
Virtual path connection group	Status, traffic descriptors, service parameters
Virtual connection group	Status, traffic descriptors, service parameters
ATM-layer statistics group	Total cells transmitted, total cells received, total cells dropped
ATM-layer group	Number of VPCs, maximum number of VPCs, number of VCs, maximum number of VCs
Physical-layer group	Port type, media type

order to give the customer a view into the carrier's network. Ultimately, carriers plan to extend their CNM offerings to give network managers real-time control over the device services that they use. The most common uses of CNM for ATM networks are testing, receiving event notifications, defining traps, administering and generating trouble reports, supporting security features, and retrieving information on configuration, usage and performance from the M3 MIB.

CNM applications exemplify the classical manager/agent model. A CNM application acts as a manager that communicates with a CNM agent residing in an ATM network. The CNM agent is capable of supplying the required management information and carrying out management tasks related to the managed resources. It communicates with the carrier's network management system (NMS), by which we mean all management functions supported by the carrier. Therefore, the CNM agent usually supports only the portion of these management functions that pertain to the service provided to the customer. The interface between a CNM agent and NMS should be based on open standards, in order to ensure that the CNM agent may take advantage of new functions that are added to the NMS. The CNM service initially supported by carriers targeted Cell Relay Service customers. CNM services are therefore built based on Cell Relay Service and ATM network requirements.

3.3.3 M4 Interface

The M4 interface specifies requirements for managing individual network elements. It defines a protocol-independent MIB that identifies the information items to be exchanged between ATM devices and the system that manages them. This MIB is a *logical* information tree, consisting of *managed entities*, and it is intended as a framework for the development of protocol-specific MIBs, such as those based on SNMP or CMIP, that would potentially permit the management of ATM networks with heterogeneous devices.

Several aspects of the M4 interface are similar to the five-layer TMN model. M4 addresses the interactions between the lower three element, element management and network management layers. The M4 specification also addresses the fault, performance, configuration and security areas of ATM management. Thus, M4 is a complete specification that accommodates the management of current and future public ATM networks. Early versions of the M4 interface did not, in practical terms, significantly impact ATM network management. This may have been due to the complexity of the specification, which made it hard for M4 to challenge SNMP-based implementations.

3.4 Wireless ATM

A wireless ATM network (W-ATM) comprises mobile terminals, base stations, ATM switches and concentrators (collectively, "nodes") and fixed terminals. Mobile terminals access the network through a radio link. Base stations are the interface between the wired and wireless parts of the network, and can have switching capabilities.

ATM nodes comprise the basic infrastructure of the broadband integrated services digital network (B-ISDN) core network. ATM nodes must support mobility services and also perform all the usual functions of ATM nodes in a land-based network. In W-ATM, information flows enter the network at the UNI and leave at the destination UNI. Thus, the network is homogeneous and does not distinguish fixed from mobile users. In other mobile network architectures, the B-ISDN connection terminates at a base station *within* the network, thus drawing a distinction between fixed and mobile users.

Mobile terminals being within the B-ISDN network leads to some network management problems. In particular, the approach to mobility management is specific to W-ATM. Performance issues in W-ATM mobility management rests almost entirely on the problem of handover and these problems have not yet been convincingly solved. It is not certain whether W-ATM will have a significant role in future wireless networks.

3.5 Conclusion

Standardisation efforts towards a unified framework for managing ATM networks have largely been limited either to general guidelines or specific areas of ATM management (such as the UNI via the ILMI specification). Therefore, most vendors remain attached to traditional management methods such as SNMP and extend these to provide their own proprietary solution. As a result, the development of an integrated platform for managing a heterogeneous, multi-vendor network requires considerable implementation effort, usually building upon some generic commercial management platform.

Web-based network management (WBM) is an increasingly viable alternative to the monolithic, integrated platform approach to managing ATM networks. Technologies such as Java combined with distributed object technologies, such as CORBA, make it possible to develop distributed WBM applications that deal efficiently with the implications of managing heterogeneous devices. Both the WBM and integrated platform approaches require a high-level information model that contains the management information.

The choice between an integrated, general-purpose management platform and a WBM depends on the carrier under consideration. WBM offers a lightweight implementation with low installation and operation costs. It can manage networks remotely and depends on open standards and interfaces that enjoy broad industry support. On the other hand, a network operator with a large investment in network equipment, and the resources to support a complex platform, may prefer an integrated management platform which closely accommodates their particular needs. This approach will be sure to provide efficient and consistent management throughout the network. The downside is that building on that platform in the future – in line with the inevitable evolution of the network infrastructure – will generally be more disruptive and expensive than an approach based on open standards.

4 Network Management Using SNMP

4.1 Introduction

The Simple Network Management Protocol (SNMP) is a protocol designed for communicating with network elements, such as switches and multiplexers, and for managing their operation. Managing a network element includes detecting and diagnosing faults, provisioning capacity, activating service and suspending and re-activating service.

SNMP was introduced with the larger goal of allowing standardised, automated network management. The motivation for standardisation of management protocols is that independent vendors of

- NMSs
- SNMP-managed devices and
- other network equipment

may develop and supply devices that can interwork and communicate with other NMS and OSS software that support SNMP.

SNMP has been fairly successful in achieving its aims. The chief reason for the rise of SNMP has been its simplicity: implementing SNMP management in a network device is more straightforward than most other approaches to network management. In addition, SNMP was successfully "sold" by the Internet Engineering Task Force (IETF) as the chief Internet standard for network management. The IETF is influential in the Internet engineering and telecom communities. In the early days of SNMP, developing applications based on SNMP required significant effort to manage the variety of networked devices in use. This problem faded away as more development and other tools became available to support SNMP. SNMP is now the dominant communications protocol for end-to-end management of internetworking devices.

SNMP facilitates communication between a managed device and the user of an SNMP management application. A managed device is any device with an SNMP agent. The SNMP agent is stored on a managed device. The agent gives access to data, or managed objects, that describe the state of a managed device. Through this access, the SNMP manager or management application can monitor and control the device.

K. Misra et al. (eds.), *OSS for Telecom Networks*

The Oulu University Secure Programming Group of Finland has found several vulnerabilities in the decoding and processing of SNMP trap messages and SNMP request messages by SNMP managers. There are vulnerabilities in the decoding and the subsequent processing of SNMP trap messages and SNMP request messages.[5] Vulnerabilities include denial-of-service conditions, format string vulnerabilities and buffer overflows. These may in turn result in service interruptions and may allow an attacker to gain access to the affected device. Owing to slight differences in SNMP implementations, the specific outcome arising from the vulnerabilities varies from product to product.

The development of the second version of SNMP (SNMPv2) standard addresses the lack of security of SNMP using mechanisms which focus on privacy, authentication and access control. SNMPv2 also allows more complex specification of variables and has some management commands not available in SNMP. SNMPv2 and SNMPv3 are now widely supported in new telecom equipment. Several of the products mentioned in this book, such as TEMPo by Tenor Networks, and AdventNet's products support many features of SNMPv2 and SNMPv3.

4.2 Object Management

Communication between an SNMP manager and a managed device occurs via SNMP in abstract units called Protocol Data Units (PDUs). The communication SNMP manager and a managed device are typically encapsulated in User Datagram Protocol (UDP) packets. There are four main kinds of operation between managers and agents:

1. The manager can perform a *get* (or read) to obtain information from the agent about an attribute of a managed object.
2. The manager can perform a *get-next* to do the same for the next object in the tree of objects on the managed device.
3. The manager can perform a *set* (or write) to set the value of an attribute of a managed object.
4. The agent can send a *trap*, or asynchronous notification, to the manager telling it about some event on the managed device.

SNMP management is achieved by making an SNMP call to the managed device in order to access a managed object. A managed object may be regarded as a value in a database of managed objects in the device. This database is maintained by the device and is populated with parameters that are relevant to the status, performance and configuration of the device (see Figure 4.1).

To specify managed objects to the SNMP agent, the SNMP manager or management application uses a rigorous naming syntax to specify variables. An object name is an object identifier (object ID or OID), which is like a serial number uniquely identifying an object to an SNMP agent.

[5] These flaws have been documented by the CERT Coordination Center of the Software Engineering Institute operated by Carnegie Mellon University.

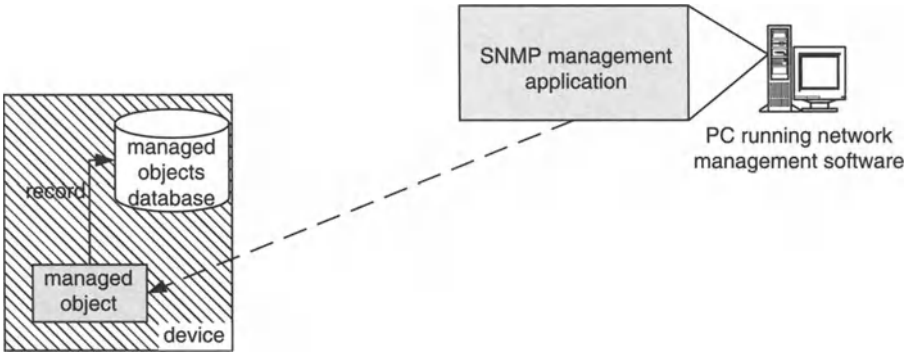


Figure 4.1 Managed object database in an SNMP enabled device.

The SNMP manager must know the type of product that it is communicating with, usually by model number, so that it can determine whether or not the managed object being requested is available on the device and what type of information the managed object contains, whether it is information on current faults, past faults, activity, available capacity or some other information. This information is held by the NMS in databases for use in actions that are involved in managing the device. The NMS may present a map to the network management application with icons representing the managed objects. Alternatively, the managed objects and their values may simply be provided in text format. When the user clicks on an icon they are prompted to define the managed object being requested. Associated with the icon are the Internet Protocol (IP) address and product type of the device. When the user requests information from a product, the NMS examines a definition of all managed objects that can be retrieved using the specified product type. The user is then given those managed objects and prompted to define each in turn.

4.3 Management Information Base

The MIB of a device is a definition of the format of the database of managed objects within the device. The MIB is a list of managed objects together with a description of the kind of information associated with each managed object. MIBs for some products or functions are sometimes predefined in accordance with standards resulting from work by consortia of vendors that are published to the telecom community. Such work is supervised by the IETF, which release MIB definitions as requests for comment (RFCs) to the Internet engineering community.

MIB definitions are written using Abstract Syntax Notation 1 (ASN.1), which is a standard program definition syntax, so that the definitions are standardised. It is important that definitions are standardised to avoid ambiguity and so that MIB implementations in telecom products are compatible with SNMP managers which make requests for data from MIBs in the day-to-day operation of telecom networks. The NMS is usually sent a copy of the MIB at configuration time, so that it has a definition of the database for all product types that will be managed, such as workstations,

file servers, routers and multiplexers. The managed device uses a copy of the MIB to provide the managed object information upon request by a management application such as an NMS. In the case of the managed device, the firmware is coded so that management information collected from different parts of the product is stored in a database whose tables correspond to the MIB definition.

The NMS uses an MIB to identify the nature of a managed object within a class, or category of managed device. This information is used to fetch the contents of the managed object from an instance. An "instance" of a managed object is simply an actual device found in the network or a subset of the functionality of a *particular* device. The device uses the MIB to present the contents of the managed object upon request. Transactions between the NMS and managed device are based upon a master–slave relationship. The device cannot send unsolicited information to the NMS, unless the information is in the form of a *trap*.

Each SNMP agent provides access to objects that are specific to a particular type of device. To enable the SNMP manager, or management application, to operate intelligently on the available device-related data, the manager must know the names and types of objects on the managed device. This is made possible by MIB modules. MIB modules are specified in MIB files which are usually provided with managed devices. For example, RFC1213-MIB, also known as MIB-II, is a MIB module which is typically supported by all SNMP agents on Transmission Control Protocol/Internet Protocol (TCP/IP)-enabled devices or systems. This MIB file contains a description of the object hierarchy on the managed device, in addition to the name, or OID, syntax and access privileges for each variable in the MIB. For example, when the MIB module is loaded in an MIB browser, the label of the variable, such as sysDescr, can be used to identify it, since the MIB browser can use the MIB module to translate this label to an OID by querying the MIB database.

A key aspect of MIB is that only the types of object on the managed device are specified by the MIB, and not the specific objects/instances. For example, ifInOctets in RFC1213-MIB[6] specifies a type of object for a number of input octets on an interface. However, the specific objects or instances of that type are specified as ifInOctets.1, ifInOctets.2, ... , ifInOctets.n, where n is number of interfaces to the managed device.

SNMP (version 1) supports five different kinds of message: *GetRequest*, *SetRequest*, *GetNextRequest*, *GetResponse* and *Trap*. A single SNMP message is a PDU, whose format is in accordance with the ASN.1 notation. The PDUs are translated into binary format using Basic Encoding Rules (BER). Binary format is the most basic form in which commands can be expressed and is the form in which they are directly understood by the managed device. SNMP request messages are sent from managers to agents. Request messages can poll the agent for current performance or configuration data, ask for the next SNMP object in an MIB or modify configuration settings. Polling is where a request for information is sent electronically at regular intervals, such as every second or every minute. The purpose of an SNMP agent is to decode request messages reliably and process the resulting application data.

To specify an object to an SNMP agent, both the Object Name or ID (OID), which defines the type of object, and the instance, the specific object of the given type,

[6] Request for Comment number 1213, published by the IETF.

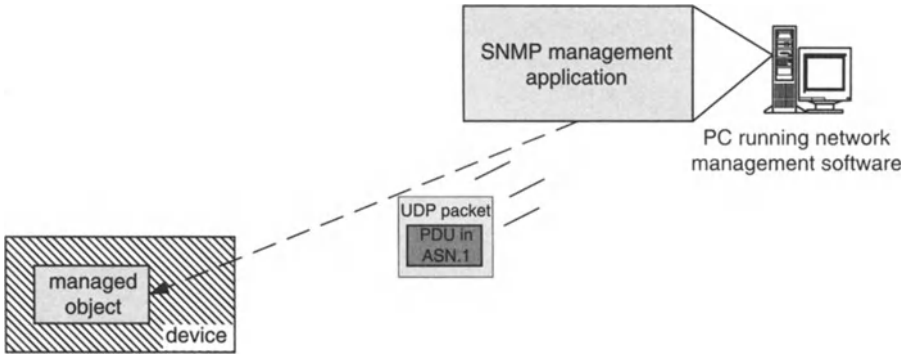


Figure 4.2 SNMP management application communicating with a managed object.

must be provided. An OID is obtained from the MIB and an instance must be added to this. Particular object instances are important as they are real components of the device, whereas an OID is merely an identifier. For non-tabular or "scalar" objects the instance number is 0, for example, "sysDescr.0". For tabular objects, the instance is defined by the MIB and is a sequence of one or more variables, for example, "ifInOctets.2", as mentioned above, or "tcpConnState.179.74.15.126.1192.225.226.126.197.80" where "tcpConnState" means TCP connection state.

Rather than specifying the instance number, one can also use *get-next* while specifying only the OID from the MIB, such as sysDescr. This will provide the first instance of that type from the SNMP agent. This method can be used for all types of managed object.

When using an MIB browser, the user selects the MIB node of interest and then selects *get-next*. Alternatively, the user selects an MIB node, explicitly enters the instance in which they are interested after the end of the OID and uses *get*.

The AdventNet SNMP MibBrowser allows loading of MIBs, MIB browsing, walking through the MIB tree, searching MIBs and all other common SNMP-related functions. The hierarchical structure of some MIB databases gives them a tree structure which an operator can find it convenient to walk through to check the status of various managed objects. The data that are available through an SNMP agent in a managed device can be viewed and manipulated through the MibBrowser.

Figure 4.3 shows the main window of the MibBrowser. The operations that can be performed using the MibBrowser are available in the toolbar. The left pane has the MIB tree, which is a structure through which all MIBs that are loaded can be viewed. The MIB tree component enables the user to traverse the tree, view the loaded MIBs and see the definition for each node. The right pane has labeled text fields to specify basic parameters such as Host and Community and it has a Result text display area in which to view results of requests sent to the managed object.

In addition to browsing the loaded MIBs, the MibBrowser can be used to view and operate on the data that are available through an SNMP agent. The MibBrowser can be configured in a way that is consistent with the carrying out of SNMP operations. Common parameters are set in the MibBrowser "Mib Settings" frame. The "General" frame is displayed in Figure 4.4. General settings include the basic SNMP protocol

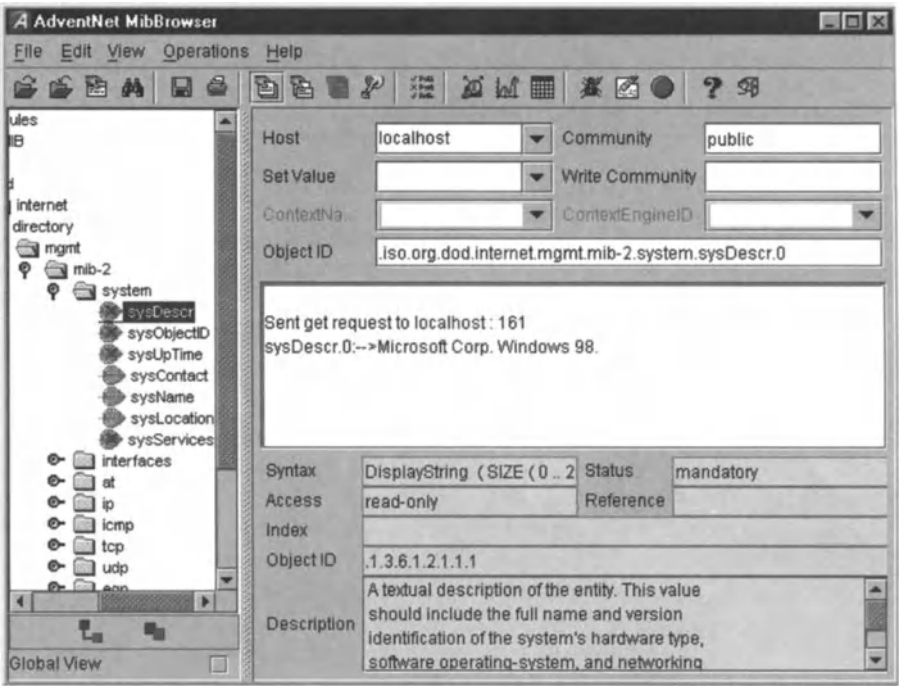


Figure 4.3 MibBrowser main window.

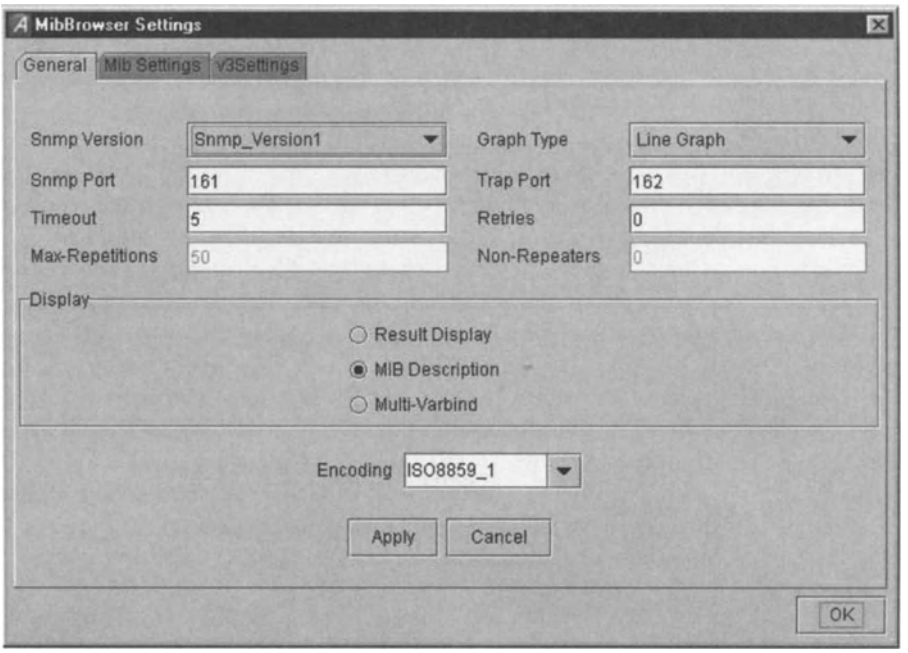


Figure 4.4 General settings of the MibBrowser.

Table 4.1 General setting parameters.

Options	Default values	Other values
SNMP version	v1	v2c or v3
SNMP port	161	Any user-defined port
Timeout	5 s	Any user-defined value
Max-repetitions	50	Any user-defined value
Graph type	Line graph	Bar type
Trap port	162	User-defined value
Retries	0	User-defined value
Non-repeaters	0	User-defined value

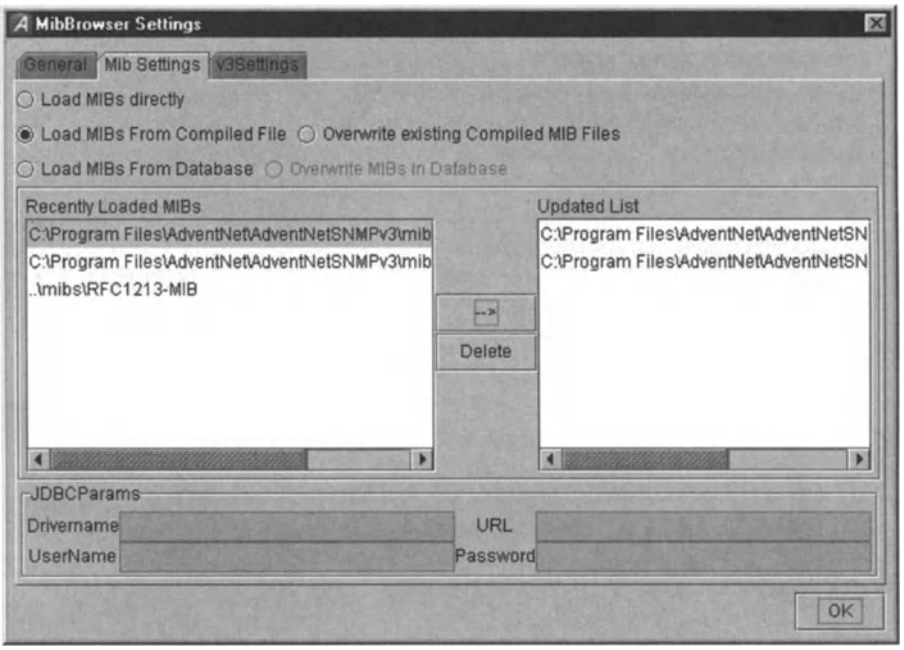


Figure 4.5 MIB parameter settings.

options such as SNMP version and port, options in the Display group box and the Encoding drop-down list box.

The various options available in the General settings for basic protocol options related to SNMP are given in Table 4.1.

The Display setting alters the view of the main window of the MibBrowser. The Encoding option refers to the transmission format of information. The MibBrowser supports ISO8859_1 transmission format. The MIB Settings options focus on the loading of MIBs in the MibBrowser (see Figure 4.5).

Recently loaded MIBs are those MIBs that were loaded in all previous sessions of the MibBrowser. If an SNMP management application supports SNMPv3, then a particular screen is needed to enable the user to specify settings that are particular to SNMPv3 (see Figure 4.6 and Table 4.2).

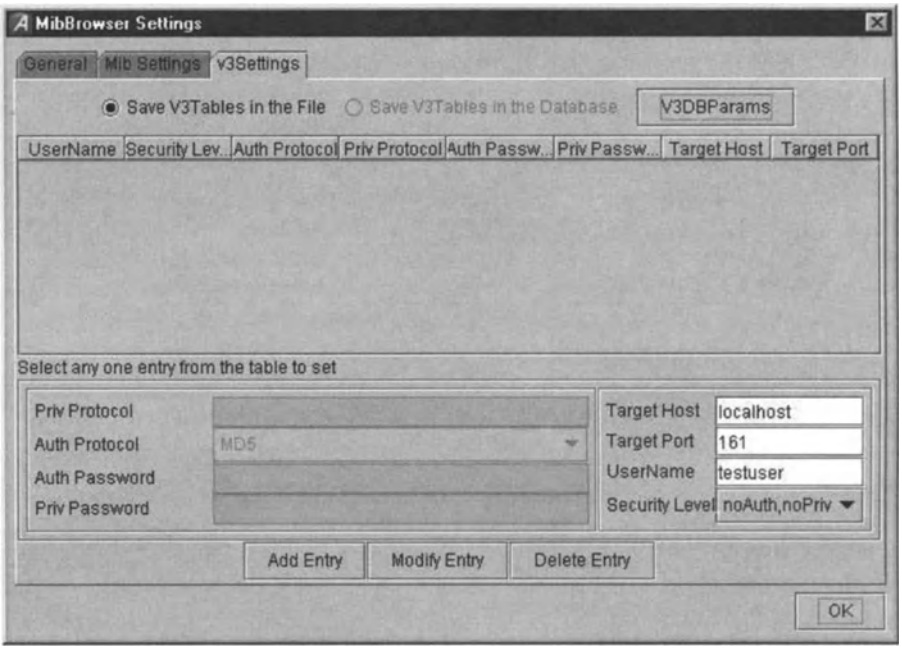


Figure 4.6 SNMPv3 settings.

Table 4.2 SNMPv3 parameters.

Options	Default values	Other options
Privacy protocol	CBC-DES (if privacy is chosen in security level)	Not available
Authentication protocol	MDS (If authorisation is chosen in security level)	SHA
Authentication password	Any user-defined value	–
Privacy password	Any user-defined value	–
Target host	Local host	Any host with SNMPv3 agent or proxy agent
Target port	161	Any user-defined port
User name	Null	Any user-defined value
Security level	noAuth, noPriv	Auth noPriv and Auth Priv

The SNMPv3 settings frame can be viewed as in three parts. The SNMPv3 table can be stored as a serialized file or as a database. Operation is faster when table entries are stored as a serialized file. To store SNMPv3 user information in database format, the necessary database parameters are entered in a dialog box (see Figure 4.7). If the database connection is established then all user information that is entered will be saved in the database. Storing table entries in database format offers the advantages of scalability and accessibility.

MIB files are usually read and parsed into MIB modules and displayed in the MIB tree. In the case shown in Figure 4.8, a MIB is always parsed before it is loaded. This can be time consuming, especially when the MIB files are large. To improve performance, the user may select the alternative option of loading from compiled MIBs (see the radio button selected in Figure 4.8).

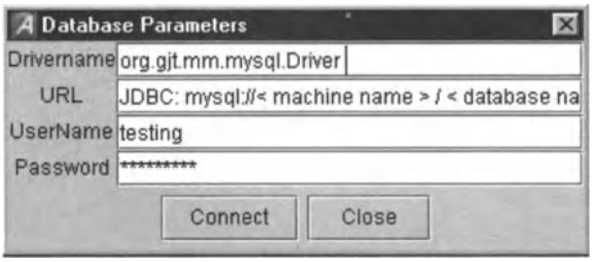


Figure 4.7 Database parameters.

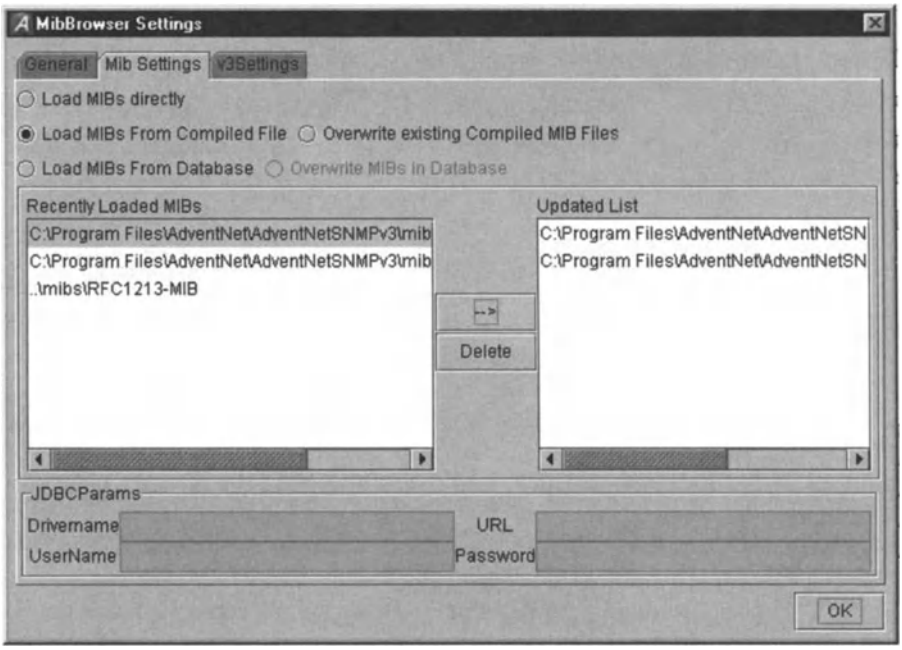


Figure 4.8 MibBrowser settings.

MIB files can also be loaded from any Relational Database Management System (RDBMS) such as SQL Server, MySQL or Oracle. The MibBrowser uses Java Database Connectivity (JDBC) for database connectivity, although it should be noted that this is specific to AdventNet and certainly not all SNMP management applications will use JDBC.

The MibBrowser allows the user to carry out the usual SNMP operations:

- retrieving Data – *get, get-next, get-bulk*
- altering Variables – *set*
- receiving Unsolicited messages – *trap*.

To perform any basic operation, the OID, instance, hostname and community string must be specified. To specify an object to an SNMP agent, the OID and instance

must be provided. From the MIB, the user can obtain the OID and to that an instance must be added in order to identify completely the object of interest.

The *get* operation is performed to obtain one or more values from the managed objects. This operation receives a list of all objects under the selected MIB object, or the specified values under a specific object if the MIB node and instance are specified.

SNMP *get-next* is similar to the SNMP *get* operation, but retrieves the value of the next OID in the tree. This operation is convenient when traversing the MIB tree, and will get the next object after the specified object. Alternatively, it will retrieve the specific object instance, just like *get*, if an MIB node is specified. With *get-next*, specifying the instance is optional.

The *get-bulk* operation is used to retrieve voluminous data from a large table. A *get-bulk* request is executed by providing an OID along with a maximum repetitions value and a non-repeaters value. The *get-bulk* operation is not available in SNMPv1, only in SNMPv2c and v3.

The *set* operation is used to modify variables from an existing value or to provide a value where there is none. However, most network devices have a default value that is maintained by the agent.

4.4 Traps

SNMP is equipped with a mechanism called trapping for sending unsolicited management information from a managed device to the NMS. Under predefined conditions, such as a power failure or the device having throughput corresponding to 80% of its capacity, the managed device will issue a trap to the manager announcing such a condition. The standard traps are *TRAP 0* through *TRAP 5*. All other traps come under *TRAP 6* and these are product-specific or enterprise traps. By "enterprise" we are referring to the vendor of the device in question.

Transactions between an NMS and managed device are usually made where the NMS makes a request and the managed device responds. This strategy works when the NMS periodically obtains management information on a "must know immediately" basis. Under certain conditions, the managed device must inform the NMS of management information, such as in the case of faults or alarms. There is also the option of the NMS periodically polling all devices for fault and alarm information, but this is undesirable for the following reasons:

- Latency/delay can be considerable and increases proportionally with the number of devices being polled on the network.
- The effect would be to overload the network with management traffic for the polling of devices which, more than 95% of the time, have no valuable information to provide.

Hence the ability to inform the NMS of traps is distributed, with managed objects being capable of sending traps as and when they arise.

In order to receive the incoming traps to the specified port, and to view and parse traps, the MibBrowser has a Trap Viewer. The Trap Viewer can listen to one or more ports at a time and the traps can be sent to the Trap Viewer from any host (see Figures 4.9 and 4.10).

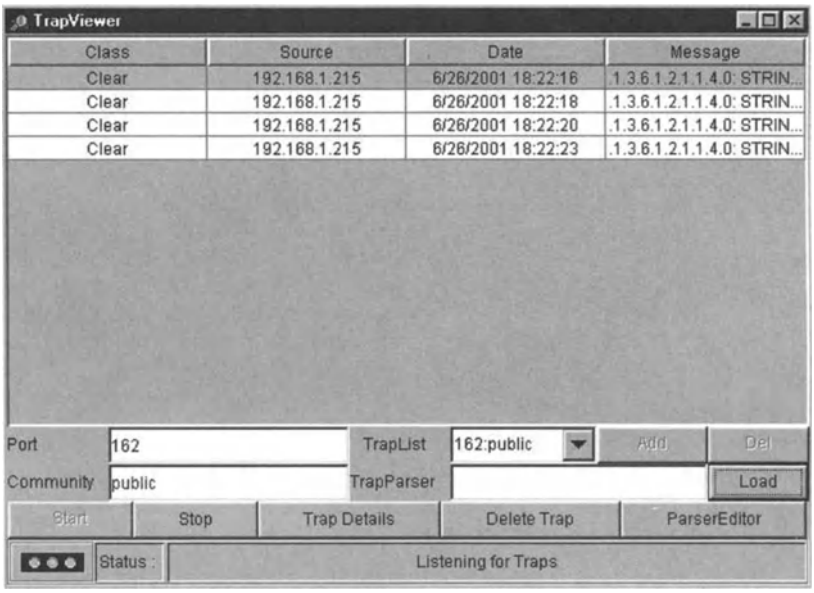


Figure 4.9 Trap Viewer providing summary trap information.

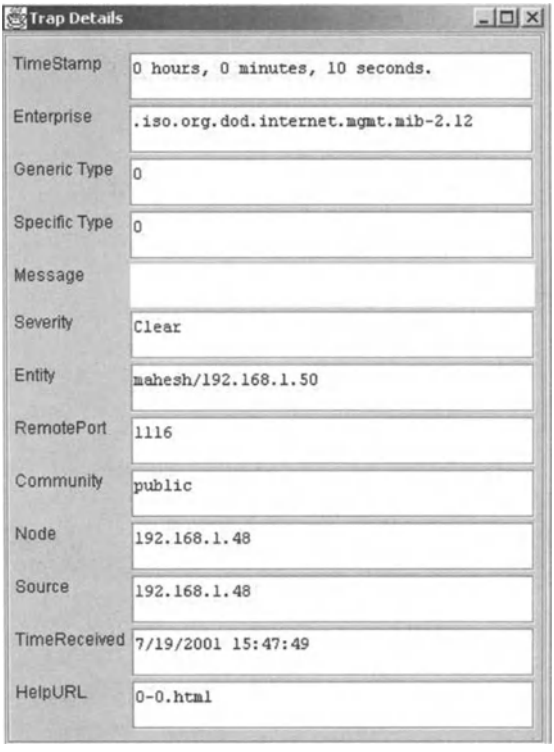


Figure 4.10 Trap details.

Table 4.3 Parameters of trap details.

Trap parameter	Description
TimeStamp	Indicates the number of hundredths of a second that have elapsed since the (re)start of the SNMP agent and the sending of the trap. It shows the value of the MIB-II variable sysUpTime converted into hours, minutes and seconds. (MIB-II is a more recent version of MIB, which caters for more detailed information on managed objects)
Enterprise	Shows the OID of the management enterprise that defines the trap message
Generic Type	The Generic type value is categorised and numbered 0 to 6: 0 = coldStart, 1 = warmStart, 2 = linkDown, 3 = linkUp, 4 = authenticationFailure, 5 = egpNeighborLoss; the trap type value 6 is identified as an enterprise-specific value. This field shows the value based on the type of trap
Specific Type	The specific trap type indicates the specific trap as defined in an enterprise-specific MIB. If the Generic type value is 6 then this field shows a value > 0. If the Generic type value is a value other than 6, then the field shows a value 0
Message	By default contains the variable values ("bindings") in the Trap PDU. This can be substituted with message text
Severity	Shows the severity or the intensity of the trap. They could be 0 = all, 1 = critical, 2 = major, 3 = minor, 4 = warning, 5 = clear and 6 = info
Entity	IP address of the source of the trap
RemotePort	Port from which the trap was sent by its source
Community	The community string is displayed here, and indicates which group of users has an interest in the trap. The "public" community includes all users
Node	MIB node of the source of the trap
TimeReceived	Date and time when the trap was received
HelpURL	The URL where more details of the received trap are available

In Table 4.3, the various parameters which define a trap – and which are provided by the AdventNet MibBrowser – are listed and explained.

4.5 Configuring Notification Reception of SNMP Traps

The Fault Management Server (FMS) product,[7] which uses the MibBrowser, can be configured to receive traps on one or more ports. Ports are set to listen for traps by parameters being entered in a configuration file. By default, FMS is configured to listen at port 162, with the entry

```
<TRAP_PORT>
  <PORT
  trapport="162"/>
</TRAP_PORT>
```

Multiple Ports can be specified using the comma separator. For example, to listen for traps at ports 8001, 8002, 8003, 8004, the entry will be

```
<TRAP_PORT>
  <PORT
  trapport="8001,8002,8003,8004"/>
</TRAP_PORT>
```

[7] We address FMS in much more detail in Chapter 12.

Trap ports can be configured at runtime, as part of the trap parser's dynamic configuration.

The existence of traps is often relevant to other management applications that are relying on the base fault management product. For example, a trap stating that a switch is running at 80% of capacity is valuable information to a service provisioning application, because service for new users should be activated at alternative switches if any are available. A trap stating that power has failed at a multiplexer is relevant to service level agreement (SLA) management applications as customers may be affected by the fault who are entitled to service credits as a result. These are called vertical dependencies between products, based on APIs, that were discussed in the Introduction. So, when traps are received by FMS, it may be necessary that other applications are made aware of those traps. The trap observer mechanism makes this possible. The specific application that must be notified should register with the Event Manager module of FMS as a trap observer. Once an application is registered, a notification will be sent to it whenever a trap is received.

An application registers as a trap observer by implementing the TrapObserver API and, in particular, by using the register method.[8] Notification is executed by invoking a method in the observer application, the implementation class of TrapObserver. Notification made in SNMP PDU format contains trap PDU information. To de-register as a trap observer, registered observers use an EventObserver method. After an application has de-registered, no notification will be sent to it when FMS receives a trap.

FMS uses a feature called "common trap receiver". This allows more than one application to listen for traps at the same port(s) simultaneously. Without common trap receiver, when a management system is configured to listen for traps at a particular port, no other application would be able to listen for traps at that port.

Any network management software application can register for multiple ports. To do this, the developer of the application would first write a class implementing a "SocketListener" API.[9] Next, this class is registered with the trap receiver mechanism for the desired ports, using either the method for registering with a single port or the method for multiple ports. Then, when a trap is received, a method is invoked on a "SocketListener" interface. The PDU encapsulates the trap information and passes on that information as an argument.

4.6 Conclusion

The major benefit of SNMP is that it is vendor-independent, and so helps in achieving all of the benefits expected from an industry standard. The development of tools

[8] We assume some basic knowledge of object-oriented programming. It is enough for the reader to know what a "class" is and what a "method" is.

[9] A "socket" is an abstract pipe between which pieces of software communicate. To "listen" at a socket is to wait for incoming data from that socket.

for SNMP has enhanced the usefulness of the protocol. Now, no network hardware would fail to include SNMP support in their products. SNMP is enjoying a virtuous cycle and so we can expect broadening and increasingly advanced support for the more recent versions of SNMP, both on the management side and on the managed device side.

We thank AdventNet for allowing us to use its material and for other assistance with this chapter.

5 Network Management Using Telnet/CLI and TL1

5.1 Introduction to CLI

Command Line Interface (CLI) has traditionally been an essential interface to any network device making CLI the de facto interface for communication with devices via a serial port. Typically, a terminal would be connected to serial ports for monitoring device status, for diagnostic commands to isolate the hardware problems or for recording and notifying of alarms.

The originators of CLI intended it to be a low-overhead way of managing devices. This is reflected in its lightweight design. By modern standards, it is certainly extremely lightweight given that it is command line driven! In their pure form, CLI commands are an American Standard Code for Information Interchange (ASCII)-based human-to-machine language with intuitive meanings, but without any standardized way in which these commands could manage devices. That is, the meaning of a CLI command for device X would be different for device Y, if that command was defined at all for device Y. Nonetheless, a niche was carved out for CLI in device management by its ease of use for network management staff and the ease with which it could be quickly implemented and integrated into a network device.

Despite advances in network management technologies, and the advent of other popular forms of management protocols since CLI, such as SNMP and Transaction Language One (TL1), there are many devices which can only be used with CLI. As CLI has been around for so long, it is still the most prevalent interface for management connectivity to devices.

5.2 Keeping CLI relevant

Owing to the popularity of the Internet and other IP networks, CLI support was broadened to TCP and other transport protocols, though it was originally designed for serial links only. This allowed CLI users to continue to use the same CLI commands from a remote, web-based terminal.

Device management can be a very big task when it involves managing devices with large numbers of commands. The AdventNet CLI API simplifies this via APIs, which hide the complexity of the commands from the user. The API also allows applications to be built to manage any type of CLI device and to perform network

K. Misra et al. (eds.), *OSS for Telecom Networks*

management tasks such as monitoring network status, collecting network statistics and issuing network device control commands. The CLI API also allows the CLI protocol to be run over any transport protocol.

The CLI APIs can be used to build a management console for device management where:

- Devices are remotely managed from a terminal over IP in an Ethernet-based network, where the terminal is connected to a data switch via Telnet or TCP.
- A terminal has local access via a serial port to a telecom switch.
- A remote terminal has remote Hypertext Transfer Protocol (HTTP) access to a Wide Area Network (WAN) router.

We give an overview of the CLI Browser module of AdventNet's product to give an idea of how CLI is seen from the viewpoint of network management OSS.

5.3 CLI browser

CLI Browser is a graphical user interface (GUI)-based network management application, which can manage any CLI device. Its features include the ability to load and use files, Command Set and Data Set, with different sets of input messages and different configuration settings.

The CLI Browser can be used to query network devices, such as switches and routers, and to receive data in responses. It allows the user to view and operate on data available through a CLI agent on a managed device.

Application developers use Extensible Markup Language (XML) to write Command Set and Data Set files for commands that are to be sent to the device. The Command Set and Data Set files are loaded into the CLI Browser from the data directory. Telnet is used as a standard transport protocol to query remote devices. Thus, Telnet and CLI work together wherever CLI is used.

To communicate with network devices, a user would start the CLI Browser, connect to any remote device and select a valid CLI Command in the "Input CLI Command" command line on the right pane. The response is received from the device after sending the command by clicking the Send button. The response received is displayed in the "Response Message" text area.

The CLI Browser can load and use different files with different sets of input messages. The Input CLI Command field indicates what operation is being carried out.

The main window of the CLI Browser is shown in Figure 5.1. The left pane has the CLI Tree which is used to load the CLI input messages in XML format. The right pane has details of the selected CLI input message and text area to display the response received from the agent.

Communication can be established with devices by setting various connection parameters. In Figure 5.2, Telnet is set as a standard protocol at port number 23 to establish connection with a device.

The remote host parameter specifies to which system/device the session is to be established. Typically, the user can set this to "localhost" in order to establish a session with his or her own system, provided that system supports Telnet, which means that it must be running a Telnet server. The port number parameter specifies the

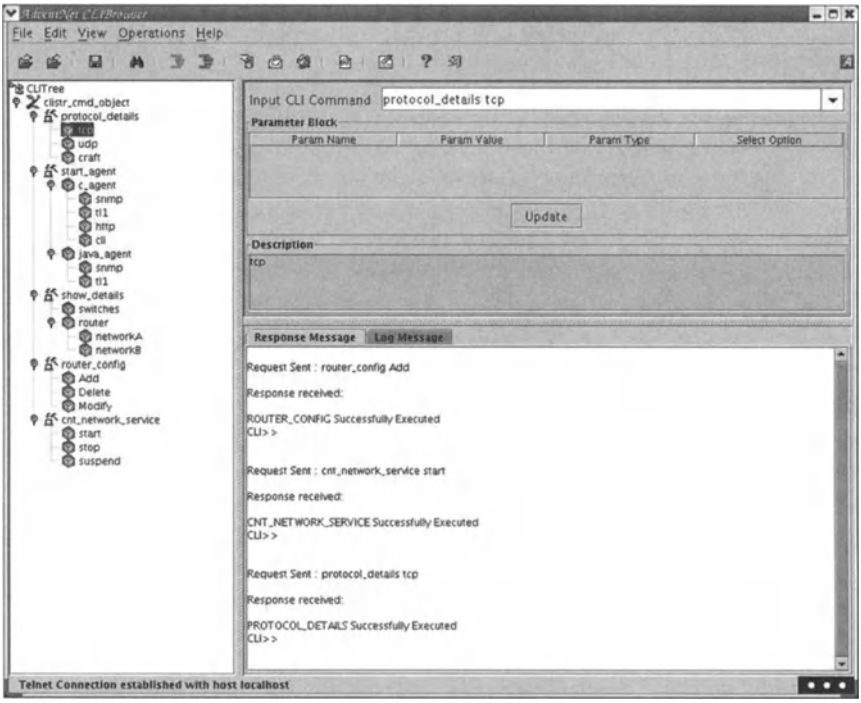


Figure 5.1 Screenshot – CLI Browser main window.

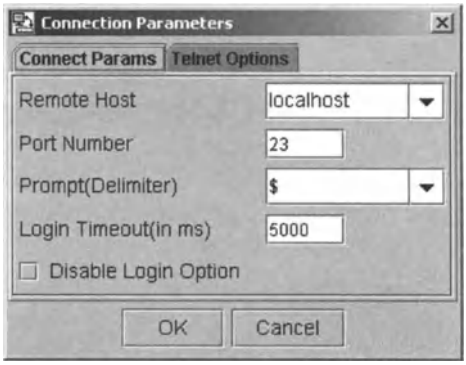


Figure 5.2 Screenshot – CLI Connection parameters.

port in which Telnet server is running. By default, this has the value 23, which is the standard Telnet port.

5.4 Introduction to TL1

Transaction Language One (TL1) is the most widely used management protocol in telecommunications. It manages most broadband and access networks and is

increasingly being used for newer management applications, such as end-to-end service provisioning. Despite its low profile, TL1 is the most widely implemented vendor-independent telecom management protocol.

The three basic requirements for a management interface are:

1. the ability to develop new interfaces quickly
2. the ability to upgrade the management interface to an existing element as new features are added to it
3. the ability of a deployed interface to keep up with performance and usability requirements of the management systems and network elements that it connects.

TL1 interfaces are character-based and designed to be comprehensible by human operators so that they can be rapidly developed and easily updated. Special decoders are not needed for debugging and new messages can be added easily. This contrasts with the complex environments that are needed to develop and maintain other protocol interfaces, for which development cycles of 9–12 months cause the interfaces to fall behind hardware enhancements. Finally, TL1 is scalable, as it is capable of working with highly scalable management systems such as Bellcore's standard network monitoring and analysis system.

End-to-end management support for networks of TL1 devices is desirable, including device agent and management applications along with full support for TL1 device management. In the case of AdventNet, this support includes:

- open APIs and Java bean components for managing TL1 devices
- a TL1 web-based craft interface on the client side
- northbound OSS TL1 interface on the server
- a multi-protocol configuration engine with TL1 support
- support for TL1 devices in the fault, configuration, assurance, provisioning and security management (FCAPS) and service management applications.

As with SNMP and CLI, the TL1 protocol facilitates communication between a managed device, a device with a TL1 agent and a TL1 manager. The TL1 agent on the managed device serves to provide access to data stored on the managed device. The TL1 manager uses this access to monitor and control the managed device. TL1 protocol communicates via TCP with which data are sent and received as TL1 messages in the form of a byte stream.

We give an overview of the TL1 Browser from AdventNet to show how TL1 is used by network management OSS.

5.5 TL1 browser

The TL1 Browser is a GUI application that can be used to query TL1 devices. It allows the user to view and operate on data that are available through a TL1 agent on a managed device, and manage the device. After starting the browser, a user can connect to any TL1 device and enter any TL1 command in the input message field.

To permit a better view of the data available on the TL1 agent, a TL1 file is usually provided with the manager application. This TL1 file contains messages requesting data such as recent activity and faults which can be used to query an agent.

Figure 5.3 Screenshot – TL1 Browser main window.

The TL1 Browser has the capability to load and use multiple files with different sets of input messages. The command code in the input message indicates the operation to be carried out on the agent side. Two types of file can be loaded:

- Text files, comprising input messages in separate lines: they are displayed as a list, and any message from the list can be selected and sent to the device.
- TL1 command set (TCS) files, comprising input messages in XML format: only the syntax of the command code is taken from the TCS file. The values of other fields in the input message are taken from the data file.

The primary window of the TL1 Browser when an XML file is loaded to a tree is shown in Figure 5.3. The left pane contains the TL1 Tree and the list of files. The TL1 Tree is used to load the TL1 input messages in XML format, and the list is used to load the TL1 input messages in text format. The right pane has details about the selected TL1 input message and the text area displays responses received from the agent.

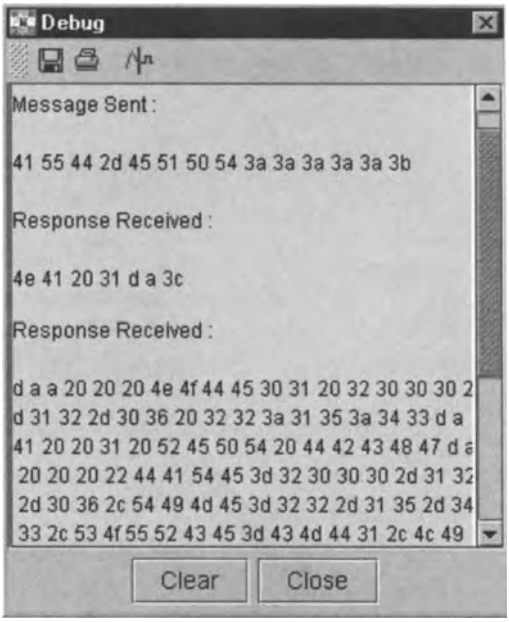


Figure 5.4 Screenshot – Debug window.

The input message consists of the following fields which can be edited and stored to the data file:

- Command Code
- Target ID
- Access ID
- C-Tag or Correlation tag
- General block
  —Order No.
  —Date
  —Time
- Contingency flag (C-flag)
- Indirect data retrieval (IDR) indicator
- Message Payload Block
- a text area for displaying the description of the input message
- a tabbed text area for displaying the response received from the agent, to display autonomous alarms from the agent and then the log messages.

The TL1 Browser is provided as a bean component in the API package which comes with AdventNet's Configuration Management Server. Some of the functions that can be performed through the TL1 Browser are:

- multiple command set and data set loading
- setting the parser options for a particular session
- support for sending messages as strings
- support for receiving partial TL1 messages

Figure 5.5 Screenshot – Decoder window.

- support for debugging TL1 messages
- support for checking drop out connections
- support for receiving Autonomous messages.

The TL1 Browser can be used to debug the messages transferred between the TL1 Browser and a TL1 device. The debug window can be used to check messages sent and received between the TL1 Browser and agent. The decoder translates hexadecimal messages into ASCII text.

5.6 Summary

This chapter has introduced and given an overview of two major network management protocols: CLI and TL1. CLI is an old protocol but is still widely used. We have explained some of the measures that have been taken to allow CLI to remain in use for so long. The aim has been to allow the reader to understand the role of CLI and TL1, and also to have a basic technical understanding of these protocols. Finally, the functionality of modern network management OSS applications that support CLI and TL1 has been explained.

6 Service Provisioning and Activation

6.1 Introduction

Telecom service provisioning is the making available of resources necessary for a service by allocating those resources in a carrier's network. Service activation is the switching on – or "going live" – of the service. Service provisioning comprises a range of activities and these must be carried out in the correct sequence.

Starting with an order received from a customer, the provisioning process involves a chain of events resulting in the service being activated at the network level and the billing system registering the new customer.

When a customer requests a new service or a change to an existing service, several disparate databases must be changed and NEs (re)configured. The customer's order is decomposed into a set of components and that set matches some group of network resources. The identification and allocation of the necessary NEs which allow the network to provide the customer with the service requested require real-time information on the configuration and capacity of each NE involved. Some NEs may require work to be carried out on them before they can support the new service.

Service provisioning is central to the service provider's business because a revenue-generating service is only available once it has been provisioned. Hence network management staff must have at their disposal end-to-end monitoring and control of network configuration. The number of systems involved in services is increasing and systems in different enterprises are interdependent owing to mergers, takeovers, joint ventures and partnerships. Thus, manual – and even custom automated – interconnection is not practical owing to expense and complexity. Standardisation of interfaces for interoperability is one of the few viable answers to this problem. Some of the products mentioned in this book make heavy use of interfaces based on standard technologies such as XML, Java and propriety – but open – APIs.

The Telemanagement Forum (TMF) has done more than any other organisation to promote the use of standard interfaces for OSS software. The TMF's next-generation OSS (NGOSS) initiative relies on XML and CORBA. At the same time, the OSS-J initiative under the auspices of the TMF and spearheaded by Sun Microsystems is based on Java. The device independence of these technologies means that they are well suited to helping achieve OSS interoperability. The wireless application protocol (WAP) was one of the first practical technologies to be built on XML, and XML

K. Misra et al. (eds.), *OSS for Telecom Networks*

was chosen owing to WAP's purpose of allowing communication with a range of types of wireless device.

Provisioning tasks that require interaction between the service provider and external organisations include:

1. provisioning of leased circuits
2. provisioning of IP components
3. interconnection requests from other operators
4. local loop unbundling requests
5. carrier preselection orders
6. number portability requests.

As services become more complex (e.g. with operators using creative product design to derive return on investment in 2.5G and 3G mobile infrastructure and licences), the number of components and partners in the end-to-end provisioning process can only increase. The extensibility of XML makes it well suited to the open-endedness of the universe of future services. XML is also powerful enough to play a role in implementing policy-based management mechanisms.

The need for interoperability is enhanced by the tendency for vendors in the young and innovative OSS industry to focus on different parts of the total OSS picture. For example, Clarity products are very strong in circuit design functionality whereas Cramer and Alcatel products are strong in inventory database synchronisation. Both of these functional domains are critical for trouble-free service provisioning. Thus, a carrier who wants true "best-of-breed" in each OSS functionality domain will certainly be interested in technologies which ease burdens in achieving seamless interoperability between OSS products from different vendors.

Service provisioning may require work to be done, such as "soft" work as in the design of a circuit, or field work such as making a trip to the customer premises to install a piece of equipment. The less field work is involved, the faster and cheaper is the provisioning process.

When all hardware is in place, service provisioning might merely involve marking a particular path, logical circuit or bandwidth as no longer available. Then, when another new service is to be provisioned, the required resource will be sought in the network. If the resource exists, then the only question is whether it is already allocated. Checking whether it is allocated – and, if it is not allocated, then allocating it – is a process that can ultimately be automated. The extent to which it can be automated depends on how accurate the inventory system is and how accessible it is to the provisioning software. Of course, if the necessary resources do not exist, then field work may have to be carried out or the customer might simply be informed that the service cannot be provided because the provider is "sold out" of the necessary resources.

We often see service provisioning under the heading of service management, which, in turn, is closely related to service level agreement (SLA) management. Once we are in the domain of SLAs, we are in the domain of performance. This is why one will sometimes see alarm management, performance reporting, SLA management and service provisioning all in a single OSS solution.

A service level agreement is a contractual definition of the relationship between a service provider and one of its customers. A static SLA is negotiated periodically

(such as monthly or quarterly) whereas a dynamic SLA negotiates the service at the time a service is demanded. In a dynamic SLA, the request and the negotiation take place using a network signalling protocol.

SLAs are very important in telecommunications, although it is thought that there will be a decline in SLAs and a move towards guaranteed service levels. After all, if you purchase a certain quality of service, then you want that service – you do not want credits for more of the same defective service.[10] In any case, OSS vendors in the service provisioning space have had to offer SLA management, which means that SLA breaches must be measurable. The result has been that a number of low-level functions (alarm and performance management) are sometimes seen in service management products.

Service provisioning products typically span a great many functionalities, whether or not they offer SLA management. For example, the Telcordia Service Provisioning and Activation suite includes:

1. Network configuration management, which assists in making decisions regarding network and service configuration, as well as assisting in assigning circuits and inventory management.
2. Customer access and location management, which merges service location information with the access data network for each location.
3. Customer number management – in particular, one should consider numbers (e.g. telephone numbers, in the usual sense) as a resource which, like any resource, is allocated in the provisioning process.
4. Service delivery for real-time, flow-through order tracking and service delivery.
5. Work item management, which involves the prioritisation of work items and the assignment of work items to technicians.
6. Unified order management, from customer order through service provisioning and on to billing.
7. Hybrid fibre–coaxial cable (HFC) element management for end-to-end management of HFC networks.
8. Service cutover management, allowing transition of services between access networks – particularly for service cut over from an old to a new network.
9. SS7 network activation management for network configuration, fault management, routing data and local number portability.
10. Element management – including element activation – in multi-vendor transport networks.
11. Switch element activation, for automated activation of switch services.
12. Customer network management, which is where customers are given control over certain parts of the services which they are buying, including Centrex services management.[11]
13. Switch load balancing – which is the proactive transferring of line equipment for the purpose of easing traffic burdens.

[10] An early 2002 issue of *Capacity* magazine – published in the UK – quoted this argument from several enterprise and wholesale customers of service providers.

[11] Centrex (CENtralised EXchange) comprises PBX-type services provided directly to a subscriber from the carrier's (centralised) exchange.

Most of these 13 functionalities are provided by separate products offered by Telcordia. This list gives an idea of the number of functions that are involved in service provisioning and activation. While Telcordia is not a typical "OSS vendor", in this relatively new industry, it can be hard to identify any OSS vendor as typical.

The following are involved in provisioning support through all phases of provisioning, from user requirement identification through satisfaction and validation that the requirement has been satisfied:

1. Order entry.
2. Service request processing:
   (a) reviewing, validating and logging telecommunications service requests
   (b) service design
   (c) writing service orders and work orders that are needed to implement a service, after receiving a service request from a user
   (d) creating or updating a customer account in the configuration management database
   (e) provisioning the service
   (f) updating the provisioning status of the requested service in the configuration management database
   (g) providing status reports on service requests.
3. Implementation support: verify the implementation of service requirements and document changes to those requirements; update the configuration management database for service implementation status and coordinate customer acceptance of new service.
4. An interface to the end-user equipment.
5. Billing and SLA reporting.
6. Workflow management, which includes ongoing process monitoring, and updating and improving the provisioning process: recommend procedural, organisational and operational changes to the telecommunications provisioning process with the objectives of reducing the associated cost, time delay, personnel and inefficiencies.

6.2 The Provisioning Bottleneck

Provisioning was once a labour-intensive process involving many systems and people. However, many parts of a given provisioning process can be largely automated. For example, in the optical domain, new transport technologies allow bandwidth provisioning times to be cut from months to days. The opportunity for OSS vendors is in lifting rapid provisioning from network capacity to entire services.

A service provider can gain a competitive advantage with fast service activation. Of course, once some service providers gain this ability, there is a competitive pressure on all service providers to follow. In achieving rapid service provisioning, information transfer can be complex because services may span multiple carriers.

Many predict that customers of the future will be able to order services online and request changes to their services online. Indeed, as long ago as late 2000, the

California-based software vendor Xacct was doing brisk business in the USA and Asia in selling the front-end web interfaces for customer self-provisioning.[12] For example, a customer with geographically dispersed offices requiring additional bandwidth for a company-wide videoconference will seek to be able to order and turn on the service with little notice to the carrier. This is similar to how a customer can order a voice conference service today. The carrier would then create the service and apply the change to the current billing cycle.[13]

With growth in data services and strong competition for customers, carriers and service providers need management systems that allow rapid creation and modification of end-to-end network services that are integrated with their business management functions. The TMN management architecture model of the ITU-T breaks down the activities associated with managing a telecom network by defining functional blocks and interfaces between those blocks.

Network hardware sold by equipment vendors is accompanied by a network element management system (EMS). To allow integration, the EMS must implement a (preferably open) northbound API that allows the service provisioning, fault and performance monitoring functions to be accessed from network and service management applications. To achieve support for the many operating system platforms and languages that are used to develop applications, the OMG developed CORBA, which allows distributed applications to communicate with each other in a platform- and language-independent way. Using CORBA to define the interfaces between the layers of the TMN model, carriers can integrate the applications and functions provided by the EMS.

When provisioning a new service, the carrier or service provider enters a contract with the customer to ensure that the service delivered meets a defined set of service level metrics.[14] This contract is embodied in the SLA, which includes metrics such as availability, latency (delay) and loss. In order to report on these and other parameters, the equipment and EMS must collect, store and present accurate statistics that can be used to define the terms of the SLA. This data must reliably be collected for each managed element within the EMS domain, and be made accessible to external reporting and billing applications.

6.3 Service Management in the Intelligent Network

The intelligent network (IN) is a way of implementing services in the main telecom network (the public switched telephone network, PSTN) using instructions and data in computing equipment that is remote from the exchanges. The computing

[12] They were fairly successful, although a little ahead of their time. As Tony Kalcina, CEO of the OSS vendor Clarity International, said to a senior member of Xacct in late 2000, "If the backend connectivity were in place, you might have stumbled upon the holy grail of OSS there." Back then, only a few service providers had begun to put in place a unified system with management control across all network elements. Consciousness of the desirability and eventual need for one-touch service provisioning was nowhere near as widespread amongst telecom carriers as it is today.

[13] By the time this book is published, several service providers may already have this capability.

[14] Also see Chapter 7.

equipment is intelligent, with the intelligence residing in the instructions whose detailed content is based on data. The control programs in the switches provide basic services such as connectivity of network bearers.[15] In an "unintelligent" network, services such as 1800 and call forwarding can be implemented by adding service logic and data to the program installed in each switch. However, the switch-based approach is prohibitive in development time, cost and management effort. These are all reduced by an IN architecture. Further, the range of services that can be implemented is much larger owing to the greater flexibility of a computing device.

The idea of the intelligent network is that the infrastructure is not constrained to providing just one type of service. Thus, a service management system that takes advantage of the intelligent network should be able to change its configuration dynamically so that it may support many services with little redevelopment. It should be flexible enough to be able to manage unpredictable network growth, and it should be able to manage network elements and application servers.

The intelligent network comprises service control points (SCPs), service nodes (SNs), compact service nodes (CSNs), a service creation environment (SCE) and a service management system. Intelligence becomes more distributed when multiple SCPs and SDPs are involved in processing a call. Multimedia services require IN support when multiple calls are involved in a service. Thus, in the long term, distributed intelligence both in the core network and also in peripheral devices, will be basic to providing services.[16]

The SCP provides run-time functionality for intelligent network services. When an exchange needs to process an intelligent network call, it passes control of that call to an SCP via the SS7 network. The SCP runs the service required by the call and returns call routing and call state information back to the exchange to process the call. SNs and CSNs handle tasks such as voice announcements and dual-tone multi-frequency (DTMF) decoding. Intelligent network services are created by service developers using the SCE. This can be regarded as the communications service analogy to integrated development environments (IDEs), which are used by software developers. The SCE is used to design and develop the executable files and custom database for the service, and these tasks are collectively referred to as "implementation" of the service.[17]

The implementation of a service is installed on an SCP to be run and is also installed on the service management system in order to send provisioning requests to the SCP databases. The service management system is used to provision and

[15] Alcatel has a strong history in developing IN infrastructure. Incidentally, Clarity International's VP of R&D (Dr Steve Parry) was a senior manager in Alcatel's IN group before he joined Clarity. Many senior people in successful OSS vendors had a significant career in the telecom industry before moving to a pure OSS vendor.

[16] The Telecommunications Information Networking Architecture (TINA) is an open standard intended to support services based on distributed computing.

[17] The ability to use software to design and implement services with almost infinite variations has spawned the new field of telecom service engineering. Important work has been done in this area at the Telecommunications Systems Group of the Department of Electronic and Electrical Engineering at University College London and the Telecommunications Laboratory of the Department of Electrical Engineering and Computer Science at the National Technical University of Athens.

manage custom data for intelligent network services present on the SCPs. It receives the definition of the information needed to provision the service from the SCE. The data necessary to provision the services can be sent to the service management system via manual entry or by an upstream system. The data are validated and routed to the appropriate SCPs by the service management system.

An example of a network owner adding intelligence to its network was the sale by Nokia to Globe Telecom (Philippines) of a mobile switching centre (MSC) with a capacity of 150 000 subscribers. This expanded the capacity of Globe's Manila MSC from 120 000. Nokia also supplied a Home Location Register and the Nokia mobile IN solution. This solution added to Globe's ability to provide new value-added services to GSM (Global System for Mobile Communications) subscribers. In particular, Nokia's IN solution was seen as a step towards more efficient service provisioning.

6.4 Measuring QoS at the Customer Layer

Providing services of a particular quality requires the monitoring of low-level network metrics since the behaviour of the network is ultimately the basis of the services that a customer receives. However, network-level metrics must be translated into customer-layer service quality metrics. This was once a labour-intensive task, but most service providers are now working towards automating the conversion of network-level metrics to service-quality metrics.

In the past, verification testing could only be done by skilled field technicians and this manual work was a prerequisite to service activation. For each service type, standard procedures were followed and the network was optimised – typically for telephony and low-speed data transmission. With the advent of multimedia and the delivery of large amounts of content, it has become harder to manage network performance with a view to achieving a certain QoS as experienced by the end-user.

Service providers cannot ignore end-to-end customer QoS because it is directly related to customer satisfaction and customer churn. Installing test units for each service at each customer site would give complete visibility but would be costly and hard to manage. More realistic is to use network monitoring hardware and software to collect network performance data. Different aspects of services critically rely on various network performance parameters, and so correlation and trend analysis methods can be used to estimate end-user QoS. In mobile networks, each call generates a large amount of QoS data and so correlation methods are most advanced in mobile networks.

Performance data collected at lower network levels can be cascaded up to the customer layer by correlating lower level network events with relationships between the network and services. To avoid provisioning services that are not supported by resources, the service provider uses QoS metrics that apply across network layers. A typical optical network has at least three interdependent layers:

1. fibre-optic cables at the pure physical layer
2. optical channels on another physical layer
3. data transmission protocol [such as synchronous optical network (SONET), Ethernet, IP] on another layer.

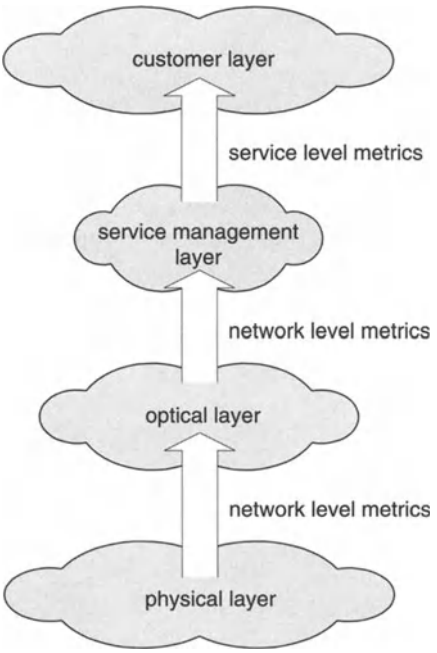


Figure 6.1 Correlation of performance of lower network layers with QoS at the higher customer layer.

The third layer is subdivided below the customer layer. Each layer is dependent upon the layers beneath it. If the SONET overhead indicates bit errors in a path on the data transmission layer, packets can be lost which would affect real-time applications. Often there is an explosion in the effect of lower level network layer errors on higher layers. That is, as a fault cascades upwards, its effect multiplies, potentially by thousands of times.[18]

A strategic layer can sit between the lower layers and the customer layer where data collected from the network and service quality converge. This is the integrated service layer that unifies performance management across SONET, Ethernet, IP and other optical network services.

In optical networks, photonic switches and other all-optical network elements suffer from a lack of built-in performance-monitoring capabilities. This motivates a performance management layer between the optical and customer layers to correlate optical domain metrics with QoS metrics (such as BER or packet loss) with end-user quality (such as voice quality or availability). Measurements taken in the optical domain use statistical sampling on optical channels to measure the three components of wavelength character (signal power, noise and drift). This correlation must be made with proper equipment monitoring in the core and at the network edge. Simultaneous measurements of signal power, optical signal-to-noise ratio, BER, slips

[18] This was emphasized in a presentation given by a member of Astracon at Telemanagement World, Nice, in 2002.

and framing errors are brought back into the integrated performance management layer for correlation, analysis and reporting at the customer layer.

Network data are organized in various layers for better correlation at the application level and greater visibility of end-to-end QoS. Conforming to a standard procedure at each stage of network evolution helps ensure service quality. The service quality management (SQM) process has:

1. a service-provisioning request interface which returns an acceptance message saying that the service qualifies or is permitted
2. an SLA reporting interface.

The service-provisioning request supports on-demand service provisioning requests and uses a common reference model such as a cable and fibre management system or circuit inventory database. At that interface, the service qualification mostly involves field-testing units. However, medium and large carriers use remote-testing units that are managed at the NOC. The new technologies of IPv6 and GMPLS favour a self-provisioning and self-qualifying end-user service management process. Such a process allows end-users to customise their service profile. For mass deployment, service providers should implement a centralised and automated end-to-end remote performance monitoring system for visibility over the customer experience.

After qualifying a service, the service is monitored from the NOC. If a fault is detected then the service must be requalified. Thus, in a well-designed business process, service qualification and monitoring will occur at the same site. This saves time in troubleshooting and reconfiguring the network after requalification.

The SQM process demands collection of QoS metrics and SLA performance reporting. In an optical network, fibre commissioning and monitoring are carried out using optical time domain reflectometer (OTDR) measurements. Once fibre is commissioned, remote monitoring is best managed from the NOC. At the optical layer, spectrum analysers can be used for service qualification and monitoring. "Bringing-into-service tests" are carried out at the service layer. Monitoring is carried out by a remote management system. Ideally, a single unit would support a range of services including SONET, GigE, ATM and IP.

The multi-layer SQM approach benefits SLA management where (technologically independent) SLA parameters are specified for each customer and are measured for each service type.

The multi-layer SQM approach provides transparent wavelength services. Correlating optical layer QoS with the unified service layer adds value to service providers with dense wave division multiplexing (DWDM) networks because they need not force customers to use a specific service (e.g. GigE) just so that they may have monitoring. Using historical data and correlation of current data, the service provider can determine how degradation in network performance at the optical layer affects customer services.

6.5 Real-time Service Controls

Real-time service controls are most relevant in the potentially dynamic area of broadband mobile/wireless services and, in particular, broadband mobile IP services.

This is because broadband mobile IP lends itself to innovative services driven by unique marketing imperatives, which have been well publicised in the telecom industry press. It is still early days for mobile IP services in that operators are only now learning what drives revenue in this domain. Therefore, service providers realise that they must spread a wide net by offering a variety of services in order to appeal to a variety of (potential) desires of (potential) customers.[19] At the same time, it is necessary to build flexibility into service offerings, create and enforce value-based billing models and track and measure usage. The market for IP services is dynamic, partly because it is new and partly because it is not well understood by either service providers or consumers or enterprise customers. Thus, an IP service provider's business infrastructure must be able to adapt to the dynamism in the value attributed to each IP service as consumers are found to be enthusiastic or not so enthusiastic about innovative service offerings. This need to support dynamism is a challenge.

Potential solutions are:

1. Agent-based billing software which uses a decentralised architecture, with content providers establishing separate data links to the service provider's central billing system. These systems receive access requests, validate access and then bill for services sequentially but not in real time. Software developers must integrate applications and content services into the billing software using APIs.
2. Real-time service control systems which unify system policies, customer access profiles and billing information on a centralised server that is managed by the carrier. Real-time service control systems provide a single control point between the carrier network, content providers and the end-user. Such systems enable carriers to maintain control over services, customer relationships and billing, while offering a wide service.

A central challenge for wireless data service carriers – maintaining control of user access to content and applications while offering personalisation and freedom of choice – is reflected in the developing service/billing models. Models used in fixed Internet service and at opposite ends of the spectrum are the "walled garden" and the "open garden" models. Under the walled garden model, users are restricted to a menu of services, within which users may use any service for some fixed price.[20] This model can entice new customers to wireless IP services. Under the open garden model, users may access all applications and content each for a premium. Charging schema may be based on time, content, QoS or other criteria. Between the walled garden and open garden extremes, users may define service boundaries and decide on the value of the services that they wish to buy based on charging parameters. Carriers will sign up customers under the walled garden model and seek to move customers through the middle ground to the open garden. A real-time service

[19] When designing services with new technologies, marketing executives must seek desires felt by consumers which consumers themselves do not yet know that they have. This is a result of new technologies permitting services which did not previously exist and so which are beyond the imagination of many would-be customers.

[20] This is similar to the model of some ISPs where particular web content is available exclusively to subscribers.

control system allows a carrier to support all models simultaneously because any particular customer can be signed for a unique, customised service combination.

The success of SMS prompted service providers extend the service. Other wireless data applications, such as corporate intranet access, entertainment information, driving directions, weather, sports and business updates have been rolled out to early adopters. It seems that a number of market-driving applications may appear. Before its launch, Orange performed extensive market research and technical engineering work to develop multimedia messaging service (MMS) for sending photographs, video snippets and sound bites to friends and colleagues from a camera-equipped mobile "phone".

Convergence of voice and data demands that wireless carriers deploy a single system that delivers applications and manages, controls and measures the usage of each application. The personal nature of handheld devices is augmented by the control that service providers can exert over menus and offerings.

Real-time service control systems centralise and externalise access controls from a carrier's content and applications. This approach minimises integration effort and deployment costs and simplifies ongoing system administration. As services are added and modified, application servers do not need to be changed. This is important in a heterogeneous environment with many types of servers and applications from multiple vendors. With these solutions, providers can use any application or server. Carriers are not restricted in their choice of application provider and are not forced to wait for the application provider to develop the agent plug-ins that are needed for billing.

Real-time service control solutions also deliver advantages such as single user sign-in. The profile- and policy-based access control allows carriers to customise offerings user-by-user. Profiles allow service providers to package services for specific market segments and personalise access. Policies allow control over access to selected content and applications for new revenue streams and manage and control service quality.

6.6 Self-service Provisioning

Self-service is now available for communications network services, reducing operating costs for service providers. Over the past few years, several service providers have built customised systems that allow end-customers and channel partners to serve themselves. These systems incorporate transaction-management applications that automate what had been paper-intensive and error-prone manual processes across the provider's back-office OSS. The business processes involved included order processing, network provisioning, service activation, billing and customer care. Custom development work allowing the corresponding OSS to communicate allows a continuous, automated workflow. By adding a web-based interface for customers (e.g. Xacct), automated ordering and service provisioning can be achieved.

Service providers following this path early in 2002 included Capsule Communications and Talk America. These service providers enjoyed customer growth, revenue growth and savings in operating costs.

Service providers that can purchase similar COTS systems from DSET Corporations merger partner ISPsoft, from Netcracker, from Netcracker's integration partners (such as Quallaby), from TeleGea and from TeleGea's integration partners (such as Aplion Networks and Telution).

Web-ordering and service-management portals are available to reseller channels and end-users. Automation and flow-through features simplify sales and customer-support training. Other benefits include more immediate service activation. Allowing retail and wholesale customers to shape design their own data and communications services is in itself a marketing differentiator, but at the same time is a great cost saver by reducing the need for customer service, sales and marketing staff.

Continuing with the experience of Capsule Communications, in 1999 it was primarily a telecom wholesaler with little dependence on OSS software for automating administrative work. Automation through integrated OSS software functionality allowed the company to move down to the retail level and sell to thousands of smaller customers. Importantly, automation minimized the costs of dealing with so many customers and provided a scalable operational framework, allowing rapid growth without degrading the services delivered to existing customers. In 2001, almost 90% of Capsule's orders were processed electronically, of which 70% were from consumer or small businesses.

The electronic process is as follows. The initial order is turned into an account automatically and the data in the order are verified and then shared with service activation. When an order is generated, the system applies multiple processes to the information. These include credit scoring, third-party verification and feature comparisons with the customer's previous incumbent carrier account. The order is then forwarded to the provisioning OSS.

Network management, credit and collections, billing, customer care and all OSS software works together so that more than half of customer care is electronic. The company found that all of this reduced salary and administration costs even while its customer base and the quality of its services was increasing. The company achieved its results with three developers and a handful of system managers using Microsoft SQL Server, ASP Server and Visual Basic development software. Longer term plans called for porting the applications to a back office based on IBM Informix databases which allow real-time rating and billing of services.

Capsule used a web-hosting architecture which separated out web, applications and database servers as separate tiers, and then divided the entire system into the functional groups:

1. order transactions
2. order status e-mails to customers
3. messaging to network provisioning systems.

Online invoice and credit card payment services were introduced as part of the first functional group. The web "tier" when equipped with conditional access firewalls allowed three tiers of front-end web servers: a "mass marketing" site for end-customers, a site for agents and other partners and an internal site which comprised the internal intranet and the other two sites.

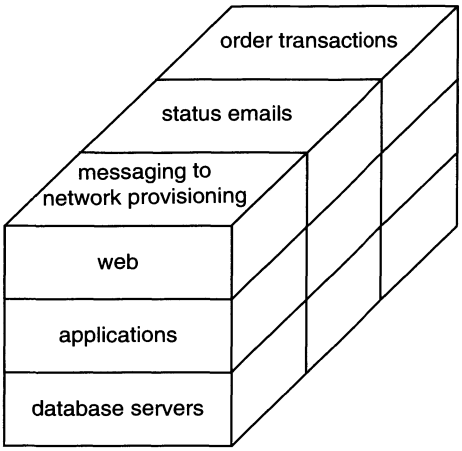


Figure 6.2 Three-tier architecture with three functional groups.

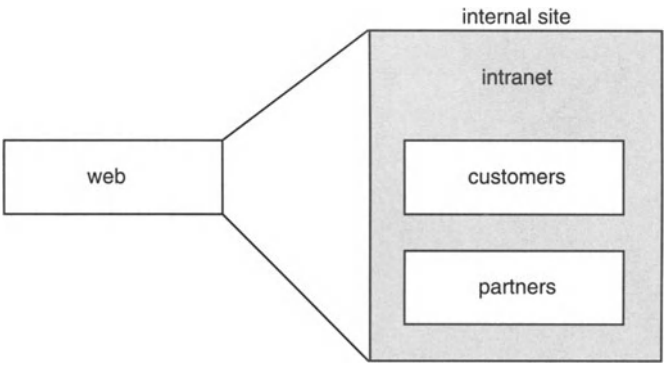


Figure 6.3 Three tiers in web server.

TMF predicted in the e-Business Telecoms Operating Map (eTOM) that service providers would want to brand and push the web interface to end-customers. This is being taken a step further with service providers encouraging their channel partners to push the web interface to their own customers after rebranding the web interface provided by the service provider. This creates an all-electronic supply chain which extends to the end-user.

The Capsule Communications project is an example of a solution that comprised a panoply of best-in-breed COTS products alongside significant in-house development and implementation. Whether a service provider should seek to achieve the gains enjoyed by Capsule using an in-house solution or a COTS solution depends upon the state of its systems at the time it decides to revamp its OSS functionality. Note that Capsule went from a mostly paper-based office to one that was very much electronic. This meant that it was able to start from scratch with compatible systems and software, such as SQL Server, ASP Server and Visual Basic. (In this sense, it was

a rare site called a "green field".) Most service providers will not find themselves in such an enviable position.

Component object-oriented software provided by Netcracker, TeleGea, Telution and other vendors has helped to advance the enterprise operations and management of the service provider. In particular, the COTS software packages of these vendors allow each step in the end-to-end workflow to be assigned to a specialist software module. By contrast with Enterprise Resource Planning (ERP) software, with which many CIOs have been disappointed, integrated enterprise (including network) resource control has been of enormous benefit to service providers.[21]

For a solution to improve order processing, it must communicate with order management, network-inventory management, provisioning and activation, and this is what TeleGea does. In the other direction, the Aplion's Network Virtuoso multiservice-provisioning solution improves network provisioning by communicating with Emporium. The "point, click, deliver" enabled by Emporium relieves the service provider of having to know about router tables, packet analysis and so on. TeleGea hooks into the underlying OSS whereas Aplion allows customers to design or reconfigure a complex service. It is the hooking into the underlying OSS that allows rapid design or reconfiguration and end-to-end activation of a service.

Other vendors who focus on linking order processes to other service and customer management systems downstream include Alopa Networks, Axiom Systems, Connexn Technologies, NightFire Software, Open Telecommunications North America, Orchestream, Vitria Technology and webMethods.

Most of these vendors are making their workflow applications malleable by using Java 2 Platform Enterprise Edition (J2EE). For example, a software object might represent a field in an order-entry form or it might represent a process or combination of processes. Object architectures allow non-technical staff at the service provider to *create* automated transaction processes by graphically arranging boxes. Such functionality allows a closer match to any carrier's particular workflow method. This lessens the impact of the justification commonly thrown up for a customised in-house solution: that there is no COTS OSS solution available that exactly fits the service provider's business processes.

TeleGea allows the service provider to name a service plan, define the description that the customer will see, associate the service with specific equipment, features and service levels, and establish plans, features and pricing bundles for various markets. Using this functionality, the service provider can create an ordering portal equipped with a catalogue for its customers. A service provider or carrier can then offer these capabilities to its resellers and agents so that they can offer them to *their* customers. This provides a strategic advantage for the resellers of services, which of course greatly helps the provider at the head of the chain (or the top of the pyramid).

We have touched on a wide range of specialist COTS OSS solutions. It is hardly feasible for a service provider to spend the time developing advanced software

[21] Unified management, as enabled by integrated OSS systems, is a kind of telecom-specific ERP which has delivered real benefits. Similarly, MRP-II was well received and delivered quanitifable benefits in the manufacturing industry.

functionality for each of customer acquisition, contract management, service ordering, circuit provisioning, customer support, billing, trouble ticketing and other network operations. At least, it would not be economically justifiable to do so when it might be possible to find a specialist ("best-in-breed") OSS software vendor that has focused on one of these functions and implemented the functionality in software.

6.7 Service Provisioning for xDSL and IP-VPNs

The strong ties between operators and end users have evolved into more complex relationships. In the traditional telecommunications world, one type of access (PSTN) supported one service supplied by the service provider (telephony). Now a subscriber logs in to an ISP and can use services – such as e-mail, video and music download – that are independent of the ISP. While providing access can still be a profitable business, it is increasingly a commodity game. The important thing is to provide as many profitable services as possible over the network as long as that network is adequate for the services that are currently saleable. Service provisioning solutions assist in this by:

- simplifying installation and activation processes
- reducing the need for labour in configuration, testing and service update work.

6.7.1 Provisioning xDSL

Integrating ADSL technology has been an expensive exercise for service providers. In particular, the labour-intensive, time-consuming provisioning of xDSL lines has long lead times and hampered the notorious problem of "building out the last mile".

The most cost-intensive areas of xDSL deployment are probably the local exchange, the local loop and the subscriber premises. The main costs at the local exchange are for rejumpering the copper wires and configuring the DSLAM. Rejumpering in the Main Distribution Frame (MDF) is labour-intensive because each subscriber line must be located and rewired by a field technician. At the local loop, line testing and provisioning of the ADSL lines are necessary but labour-intensive. Work at subscriber premises is a major contributor to the cost of ADSL provision, because a field technician must visit each customer and install the ADSL modem. In the industry, a visit to a customer site is known as a "truck roll".

Since the main costs of xDSL deployment are incurred in the above three areas, provisioning solutions for ADSL target those areas. A product offered by Ericsson approaches the issues in the following way:

1. Local exchange: the need to rejumper the MDF and reconfigure the DSLAM is eliminated by allowing subscribers to be remotely connected to the DSLAM from the NOC.
2. Local loop: there is automatic testing and provisioning of the copper line for the ADSL subscription when the customer orders the service.

3. Subscriber premises: the subscriber is given a home installation kit so that a technician is required only if the subscriber has difficulties. The technician assists over the telephone so that a truck roll is only necessary for in exceptional cases.

Siemens' AttaneXpress product suite and the AccessIntegrator product also address the problems in xDSL service provisioning mostly by attacking areas 1 and 2.

Although not specifically targeting xDSL, some of this functionality is also provided by primarily hardware-based products such as Avaya's netSet remote provisioning system. The latter comprises switching devices called Automated Main Distributing Frames (AMDFs) and is paired with an Element Director centralised control system. This approach is only suitable as part of a long-term upgrading of network infrastructure because it requires replacement of the traditional MDFs. Of course, the general automation of the MDF removes a large component of field technician work, quite apart from xDSL provisioning.

A product resulting from the Ai Metrix, Portal, Cisco and Microsoft alliance offers service providers a DSL solution that goes beyond the pure provisioning stage. In particular, it offers an integrated customer account management, billing, service activation/assurance and inventory management solution for quickly deploying DSL networks. The Ai Metrix's NeuralStar was integrated with Portal's Infranet to provide end-to-end service delivery which includes order entry, service activation, billing and ongoing service assurance. Cisco was involved by validating the integrated solution on a DSL network which used Cisco equipment.

6.7.2 Voice-over DSL (VoDSL)

VoDSL merges the PSTN and data lines on a common DSL line. SMEs are accustomed to fast provisioning of fixed telephony lines and data subscriptions. Further, in a market with so many potential customers, manual provisioning is not a practical option because there are too many touch points to configure. Automated provisioning solutions for VoDSL (such as Ericsson's) automate the entire provisioning flow, from customer premises to local exchange:

- Integrated Access Device (IAD) allows configuration of uplink speeds, downlink speeds and other customer-specific attributes of the DSL line.
- DSLAM: an ATM connection is set up from the IAD to the backbone to the VoDSL gateway.
- ATM/IP Backbone: bandwidth resources are reserved in the backbone.
- VoDSL Gateway: a new subscriber can be defined with service attributes for each telephone connected to the IAD.
- Local Exchange: telephone numbers can be assigned along with services such as voice mail, call waiting and third-party calling.

6.7.3 IP-VPNs

In addition to an Internet connection, many corporations have leased lines connecting geographically disparate sites. The use of two different lines creates operational and maintenance problems for the service provider. Using an IP-VPN (Virtual

Private Network) a service provider can offer Internet access and dedicated leased lines with a single connectivity service.

Ericsson offers a management solution for IP-VPNs that allows service providers to offer a migration from dial-up and leased line connections to network IP-VPNs or IP-VPNs that are based in customer premises equipment (CPE). The service provider can charge more because the range of services is broadened and higher access speed can be offered. The Ericsson approach is to automate the provisioning of customer site connectivity and of differentiated services. Since a wider range of services can be offered, the service provider can potentially establish a closer relationship with corporate customers by becoming a "one-stop shop". This might happily reduce customer churn. Further, instead of treating the company as a static entry point for connectivity, a single connection can dynamically allocate different connections to the various communications needs of the corporation. Different departments can have different services such as the web, e-mail, voice and enterprise resource planning (ERP) such as SAP. Each service can be automatically provisioned, along with its required associated traffic-handling characteristics such as throughput, delay and jitter.[22]

Since a corporate customer typically uses 10% of the capacity of a dedicated leased line, a service provider can improve the utilisation level of a backbone by hosting several logical networks, carrying different services over a single network. Differentiated services can be provided using common network resources. A single edge node (such as Ericsson's AXI 540 router) can aggregate traffic from different customer sites without using dedicated circuits in the backbone.

6.8 Service Provisioning and Activation Products

We shall look at a number of service provisioning and activation products that are offered by OSS vendors.

6.8.1 NetBoss Suite from Harris

Harris's NetBoss suite includes a module for service activation called Activate.IT. This module extracts service design/delivery data (such as design layout records) from inventory management, converts this information into the NE non-machine language, communicates with the NEs and confirms the ability of the NEs to support service delivery.

Activate.IT is a functionality of the NetBoss's "Application Suite", which is a set of functionality of the NetBoss communications management platform. Activate.IT integrates the activation of the various services ordered and assigned by the service provider to the individual network elements. Activate.IT automates the tasks involved in service activation, including:

- receiving assigned service orders and design layout records
- reading service assignments and features
- applying process rules and scripts to the appropriate network element.

[22] Paradoxically, a closer business relationship can be established by reducing the visibility of the service provider, since customers want transparency from basic utility-like services such as telecommunications.

The module also provides automatic recording and reporting of all service transactions. To ensure interoperability with common network components, Activate.IT uses Harris' interfaces called Smart Agents in order to interface with various systems, software, and devices.

Features of Activate.IT as a service provisioning solution include:

- activates circuit-switched and digital network elements and ISP services
- automatically configures elements
- expands network management capabilities
- records transactions for auditing while providing status to order management
- enables growth with less than proportional increase in operating costs.

6.8.2 Orchestream Service Activator

Service Activator is a product from Orchestream that supports metropolitan Ethernet services, IPSec-based VPN services and customer self-provisioning.

Service Activator supports Layer 3 VPNs using multiprotocol label switching (MPLS), QoS and security management. The latest version also supports Layer 2 services using MPLS and virtual LANs (VLANs), Layer 3 services using IPSec and multi-network VPNs. This gives service providers a single platform on which to automate the management of a broad range of network features to deliver real-time services.

Service Activator has several web interfaces and is equipped with a Java toolkit that network operators can use to create web portals to allow customer self-provisioning. Orchestream has ensured that Service Activator can be integrated seamlessly with InfoVista to deliver one-touch activation and assurance of VPN and other services.

Many service providers embrace Ethernet as a technology that is suitable for metropolitan data services. Metropolitan (metro) Ethernet is cheaper to provide than traditional time division multiplexing (TDM) services. Importantly, the technology is familiar and robust, and is ideal for several common applications including the interconnection of Ethernet LANs to form metropolitan area networks (MANs).

The two main network functions that need to be managed or activated to deliver VPN services are VLAN configuration and Layer 2 MPLS VPN management. Service Activator supports configuration of these features on a variety of network equipment.[23] Service Activator also supports the MPLS management frameworks in recent IETF draft proposals.

IPSec VPNs use encrypted IP tunnels from site to site across an IP network. The addition of IPSec support lends more choice to service providers in which technology they will use to provide VPN services. Indeed, service providers can offer carrier-provisioned IPSec and MPLS VPNs from a single interface.

Large service providers often have several geographically disparate networks, each of which normally has its own autonomous routing area. Using the support for multi-network VPN management, all of these autonomous systems can be managed

[23] Some of the vendors whose equipment is supported are Cisco, Juniper and Riverstone.

from a single interface to deliver end-to-end services. Thus, if a service provider sells services to a particular enterprise in several countries using its network in different countries, the enterprise customer can still enjoy a consistent service level across national borders and communicate with a single reporting and billing centre.

Support for customer self-provisioning is becoming common, although whether that support is useful depends on whether the provisioning system has end-to-end interfaces across the network. Service Activator's web interface allows enterprise customers to order, view and modify services without any involvement by service provider staff. The interface can be customised for web-based customer self-management and call centres. The interface can also be customised for branding so that the customers of the service provider are constantly exposed to their service provider's brand. The service provider can modify the service settings so as to limit which features of which services can be altered by the customer. This is essential as it may not be appropriate for a customer to have complete visibility into, or control over, the network service parameters.

A software toolkit for Service Activator allows the service provider or implementor, such as a system integrator, to integrate with other OSS platforms. For example, a service provider may desire that billing, fault or performance measurement data received from other OSS applications be displayed in a single interface.

The latest device control features available are:

1. Support for a driver for Cisco Layer 2 devices.
2. Enhancements in the drivers to deliver new Layer 2 and 3 MPLS, VLAN and IPSec management functionality.
3. Customised management components can be created for management of specific network device features. Thus, Orchestream makes it easier to add automated support for new device features and for new devices. The ability to plug in or customise in order to support new technologies reduces the growth in the number of management systems that a carrier needs to control its network.

In managing VPNs, the product can configure the network so that appropriate SLA tests begin between specific sites on any given VPN. This information can be automatically shared with an InfoVista product for real-time performance reporting in Orchestream User Interface.[24]

6.9 OSS framework for Provisioning

Here we describe an approach to automated provisioning. The framework outlined here was devised and is used by AdventNet, an OSS vendor.

[24] Despite the push towards industry-wide standard interfaces, it is difficult for OSS vendors to resist bilateral partnerships to ensure optimal connectivity between their products and so avoid the possibility of headaches for customers who want the functionality offered by products from both vendors.

6.9.1 Simplifying Service Activation

Service activation, or provisioning, is an area that is widely recognised as ripe for improvement. In the past, provisioning was often cumbersome, with considerable delays due to legacy systems and manual processes. Many service providers see this as a major hurdle to growing their business and reducing costs.

OSS software can reduce the provisioning burden of the service provider by automating the service provisioning process. While equipment vendors make it easier to automate configuration and provisioning of their equipment, OSS software seeks to integrate the diverse equipment used in the network. Unfortunately, many OSS solutions require a great deal of costly system integration work in order for it to be installed and this is widely recognised as a big problem in the OSS industry. Indeed, this was specifically addressed in the keynote speech at the Telemanagement World Conference in Nice in 2002.

The provisioning framework presented here provides a framework and tools for building vendor-specific and multi-vendor provisioning applications. It meets the needs of many equipment vendors and other equipment providers for provisioning and configuration of their devices in a multi-vendor network. It enables service providers and system integrators rapidly to customise provisioning capabilities and add new support for multi-vendor networks. It supports a variety of interfaces to the network elements and element management systems via a multi-protocol configuration engine which supports the SNMP, TL1 and CLI/Telnet management protocols. It is extensible, allowing new support to be plugged in, including support for proprietary protocols. Support for CORBA, Remote Method Invocation (RMI) and XML interfaces to southbound element managers and devices was added in recent years. This allows connectivity with new element managers and network elements.

The open provisioning framework presented here simplifies the integration of provisioning applications. Many different applications are needed for provisioning, depending on the needs of the domain or application. For example, OSS products recently made support for customer self-provisioning available. It is a goal of the provisioning framework that it be easily extensible to include software such as customer self-provisioning interfaces and any module with new, specialist functionality.

The provisioning framework seeks to minimise the development and integration effort needed to develop a system which can provision multi-vendor networks. Two key aspects of the framework provide these benefits:

1. An XML service provisioning template makes building solutions much simpler, and the multi-protocol configuration engine means that devices supporting only one management protocol can be provisioned.
2. An extensible server enables application- and domain-specific extensions to be quickly developed for specific needs, so that these extensions derive all the power and benefits of the existing capabilities.

Using the AdventNet provisioning framework, many different provisioning operations can be supported with minimal effort, often merely by creating and editing XML templates. This extensible framework permits rapid development and ongoing improvement of provisioning applications without large capital investment.

6.9.2 Provisioning Framework Architecture

This section describes the architecture of the provisioning framework and the services upon which it is built (see Figure 6.4). The relational database sits at the core of the system and keeps all OSS, EMS and NMS data in a reliable, transactional, open data store. This core RDBMS can be a centralised or distributed system to meet the specific deployment needs of a service provider.

The RDBMS contains tables for all necessary OSS, NMS and EMS data. This can include customer, service, order, security, inventory, fault and other provisioning-related data that the system needs to maintain and function.

Particular services form the basis of the framework. These include Java Enterprise services, protocol services, the topology and network element database, discovery services and mapping services.

FCAPS services are used by the provisioning framework as needed to carry out provisioning functions. A key service is the configuration management toolkit. This allows direct communication with the network element(s). AdventNet is enhancing the configuration management toolkit to support CORBA and XML communication with a broader range of EMS. For example, the configuration server will be able to

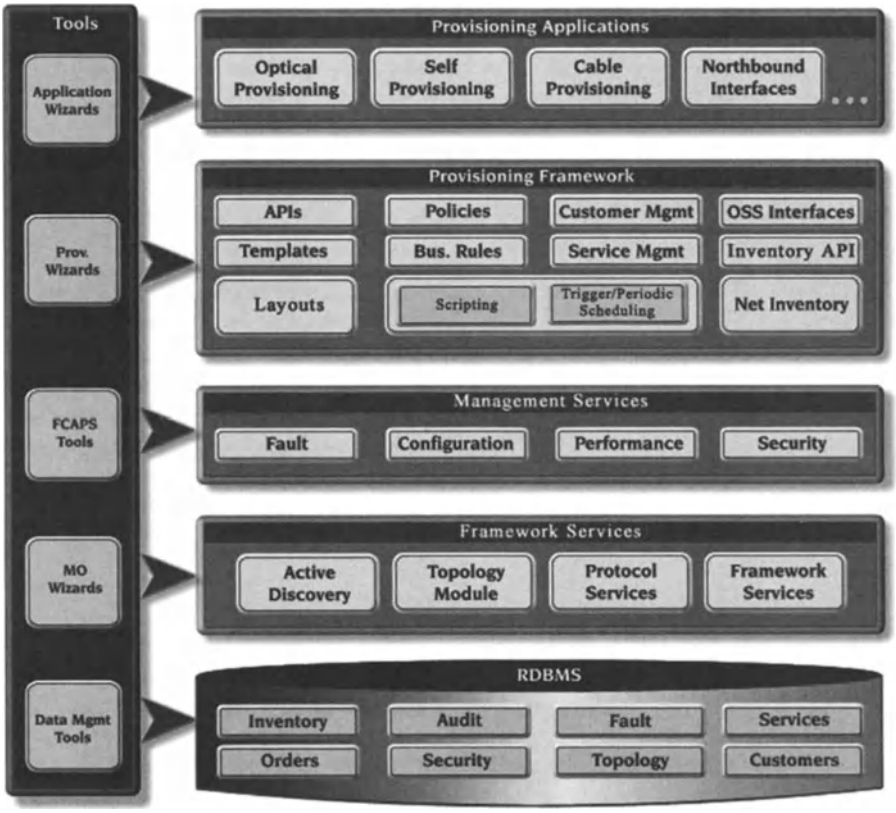


Figure 6.4 Provisioning framework architecture.

support the TMF's CORBA interfaces for provisioning by sending XML configuration commands to EMS systems.

The provisioning framework layer is built on top of the configuration management toolkit and supports the server provisioning functions. Important components of the provisioning framework architecture include:

- APIs for building applications and interacting with the module
- XML-based template scripting and rendering
- business rules for customising provisioning capabilities
- customer management information and order processing information
- service management for modelling and managing services
- network inventory database
- OSS interfaces to allow service activation and workflow automation
- OSS tools to interface with the provisioning framework and provide flow-through provisioning.

Multiple provisioning applications can be built using the framework and can co-exist within the framework. Examples are:

- domain-specific (such as optical) provisioning
- customer self-provisioning
- special purpose or application-specific northbound interfaces.

The provisioning framework is illustrated in Figure 6.5. The framework integrates many functions into a common module, which can easily be extended for domain- and vendor-specific solutions.

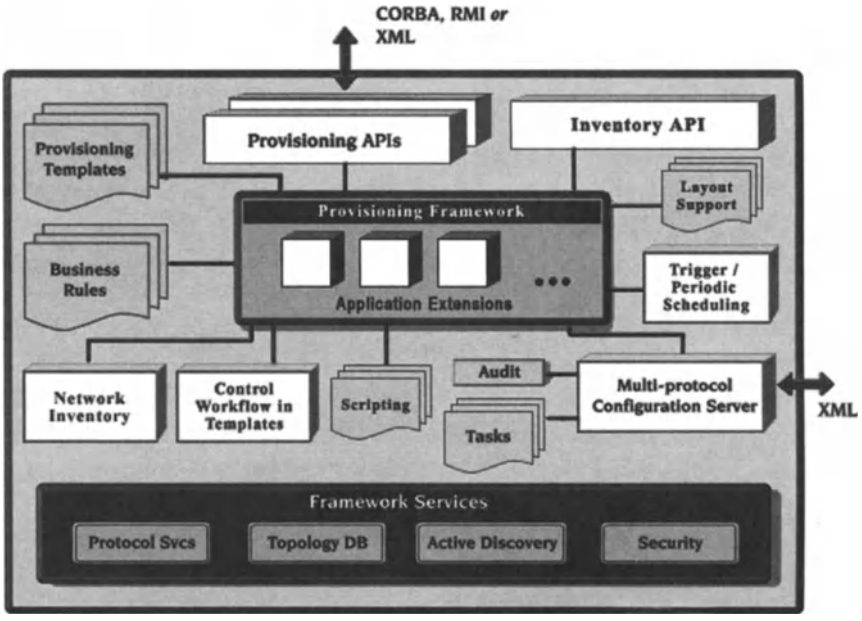


Figure 6.5 Provisioning framework.

A major goal of the provisioning framework is to allow network equipment vendors to build provisioning extensions for specific devices and applications. They can build provisioning module extensions and expose that functionality via provisioning APIs. The module provides a common provisioning API, but applications can supplement these APIs using one or more domain and vendor-specific provisioning APIs.

The provisioning templates are XML documents that have parameterised provisioning profiles for configuring multiple devices in the network. The operator provides the parameters while carrying out a provisioning operation. Provisioning templates also contain information for automatically rendering the forms that request these parameters from the operator.

The business rules are Java filters that operate on the templates before they are presented to the user or applied to the network. Rules can also be applied after completing a provisioning operation.

Some configurable filters that are driven by XML rule files need to be provided to minimise the coding to achieve simple tasks. Examples of such simple tasks are allowing particular provisioning templates only at off-peak times. Periodic and trigger-based scheduling helps in scheduling provisioning activities.

The OSS interfaces to the provisioning module are extensible and have a similar design to the provisioning APIs. The design supports the addition of CORBA and XML interfaces for specific interface description languages (IDLs) and document type definitions (DTDs). DTDs are used to specify XML document structure. In particular, a DTD defines the names and contents of elements that are permitted in an XML document.

The XML-based, multi-protocol configuration server is at the bottom of the provisioning framework. The configuration server supports TL1, SNMP and CLI/Telnet to the network element. The provisioning framework builds its templates on the XML configuration capability of the configuration server.

The design of the configuration engine is given in Figure 6.6. It supports user-specific tasks or configuration XML files that contain the detailed parameters that are needed for configuring network elements using one or more protocols. These tasks can be loaded, added or modified by users and applied to the network element using the configuration client.

The flow of configuration XML to be applied to a device begins with the client requesting the server to configure one or more network elements using the supplied XML. Before applying the configuration, the XML can be pre-processed using custom filters. Security and audit processing are completed and the configuration is applied. Once configuration is complete, the topology database is updated according to the XML configuration. Custom filters can be applied to directly update the database.

The provisioning framework builds upon this capability to provide template-based provisioning. It makes it possible to parameterise the templates so that many different service-provisioning requirements can be met. For example, service templates can be set up that require some specified user input. The template captures the detailed configuration requirements while specifying the rules governing the necessary inputs from the user and other sources of information.

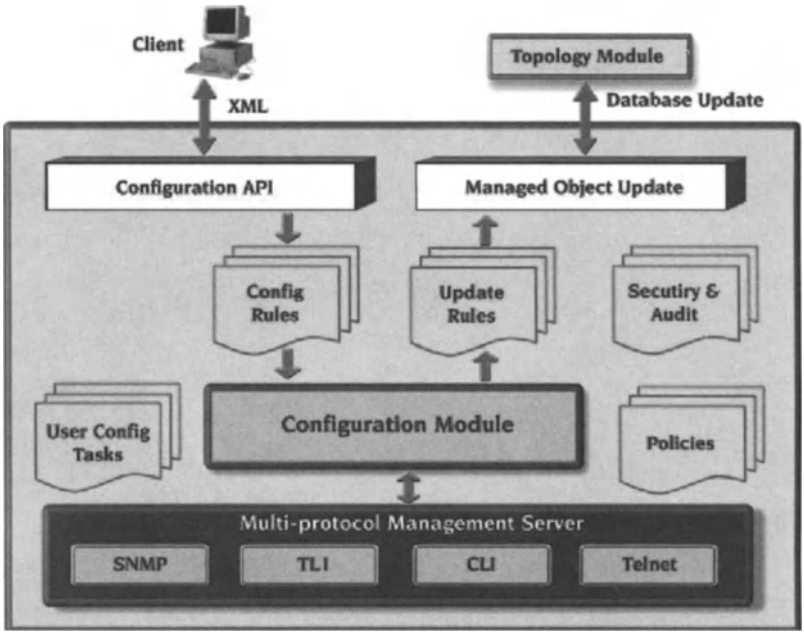


Figure 6.6 Configuration engine design.

6.9.3 Provisioning Framework Flow and Core Functions

The focus of the provisioning framework is to simplify service provisioning and allow rapid development of service provisioning applications. AdventNet uses a flexible template-driven approach, which minimises development for specific applications and allows easy customization of provisioning functions after deployment. This approach allows administrators employed at the end-customer to configure templates to suit their specific service, network and data needs.

The parameterised XML service provisioning templates may be created for domain-specific configuration tasks, such as configuring a frame relay interface on a router. These templates take parameters (for example, the router name) and are designed to collect data from the network and the inventory database based on these parameters. While provisioning activities are being carried out, the data are collected and forms are presented to the user based on XML definitions of forms in the templates. These forms can be customised to present only the relevant data to operators. The user input is collected and used in provisioning, along with the collected inventory and network data.

The XML-based templates are grouped as stages, which assists in part-provisioning. Stage-based template processing or part-provisioning refers to the ability to customise the workflow (the sequences of data gathering) in the template. This flow captures many provisioning scenarios, so that these functions can be performed simply by providing the XML and without custom development. In some cases it may be necessary to apply rules, or other custom functions, to the template data

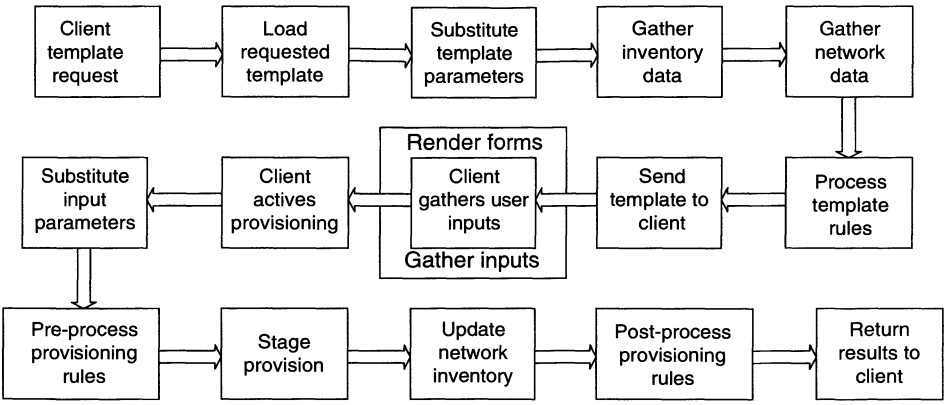


Figure 6.7 Template-based provisioning flow.

before they can be used in provisioning. The server provides several ways of adding custom rules and functions, so that the data can be massaged or rejected before they are presented to operators or before they are applied in actual provisioning. The flow of the provisioning framework is illustrated in Figure 6.7.

In words, the provisioning flow can be described as follows:

1. The client application requests the template from the server.
2. The server then loads the template requested.
3. Parameters provided by the client are inserted into the template per the XML configuration.
4. The server gathers inventory data as specified in the XML template, and template parameters may be used.
5. Network data are gathered, using the template parameters and inventory data as far as necessary, and are used to complete the XML template.
6. Template rules and filters (if any) are applied.
7. The template is sent to the client.
8. The client receives the template and renders it for the user. A client user-interface component (a forms wizard) allows client applications to embed their own code. This component takes the XML template as an argument and executes the rest.
9. After the user has filled in the forms, the modified template is returned to the server to activate provisioning.
10. Users inputs are substituted as necessary in the returned template.
11. If pre-provisioning rules and filters were set up for the template, then they are applied.
12. The XML code that represents configuration data is sent to the multi-protocol configuration module to execute provisioning of the network as specified in the XML.
13. If the provisioning is successful, then the inventory database is updated as specified in the template.

14. Post-provisioning rules and filters (if any) in the template are applied.
15. Final results are returned to the client.

Many cases of provisioning for SNMP, TL1, CLI/Telnet, CORBA and XML can be achieved using XML templates and these mechanisms. What is more, this is possible without writing custom filters and rules. In other cases, custom filters and rules may be needed to transform the data and perhaps gather sufficient data.

In more elaborate provisioning scenarios, this procedure will be repeated several times within a single client application. For example, bringing up a device may mean configuring each interface using a service template. We now describe the different components mentioned above.

6.9.4 Provisioning Templates

The provisioning template is a parameterised XML file and its DTD is an extension of the configuration DTDs so that the configuration module can be leveraged directly. The purpose of the extensions is to support the parameterised forms for inventory, network and user input, and flexible updates of the network inventory database.

The provisioning-specific tags are:

- **Template**: Wraps one or more configuration tasks in a template that includes tags to support user input and inventory updates. The goal is to allow almost any data in the task to be parameterised and displayed in forms for user input. Similarly, any of the data can be used to update the inventory database.
- **Stage**: There should be at least one stage tag in the template to be provisioned, otherwise an exception to the template is raised. The tag controls the workflow of data gathered during template processing.
- **Inventory Input**: This is to gather inventory data before initializing the template for the client.
- **Network Element Input**: This is to gather network data directly from the network elements before initializing the template for the client. The sub-tags correspond exactly to the configuration tasks, except those that are strictly for uploading data.
- **Form**: This tag allows the user to specify forms for user input parameters that will be gathered at runtime, including display information for automatic form generation on the client.
- **Inventory Update**: The inventory database updates should be allowed for each data element that is in the configuration task or from user input at runtime.
- **Initialization and Context**: The initialization tag includes the tags and parameters to connect to the specified database, from which inventory inputs are fetched or updated.
- **Script**: This tag provides the ability to embed Bean shell scripts directly in the templates to be rendered. This facilitates method calls, loops and other specific operations.
- **Goto**: This facilitates conditional invocation of forms.

The template XML contains parameters as described in the next section.

6.9.5 Template and Other Parameters

The templates are made useful by allowing parameters that can be substituted during each phase of the template-based provisioning. The different kinds of parameter are:

- **Template Parameters:** These are the initial parameters needed to scope the provisioning functions, such as the hostname of the device being provisioned, or the source and destination of a connection being provisioned.
- **Inventory Inputs:** The data gathered from the network inventory are substituted here.
- **Network and Network Element Inputs.**
- **User Inputs.**

6.9.6 Filters and Rules

To permit custom processing of configuration data at each stage, the provisioning framework supports custom Java filters to process the template data at various stages of the provisioning flow. The filters are Java code classes that are invoked during template processing. There is one instance of each configured filter for each template.

To allow the setting of rules and policies without writing Java code, it is wise to provide special-purpose filters. These filters read XML rules files to determine how to process specific templates. Initially, only a few simple filters would typically be provided – for example, to prevent the use of some of the templates on specified hosts, or at specified times of day.

There are three kinds of Java filter, based on when these filters are applied:

1. Template filters are applied during template initialization, after inventory and network inputs are collected.
2. Pre-provisioning filters are applied just before the network configuration actions begin.
3. Post-provisioning filters are invoked once network configuration is complete.

These filters are in separate configuration files, where the template name and complete class names are specified. In all cases, the template with the current configuration data is supplied to the filter and the returned value is used for the next stage of processing. Filters can be used to gather additional network and inventory data from any appropriate source. This allows complete extensibility for specific needs that are not met by the provisioning flow described earlier.

6.9.7 Application-specific Extension Modules

The ability to implement domain and vendor-specific extensions is an important feature for many service providers. This is accomplished by allowing OEMs to develop API extensions that leverage the provisioning APIs, the configuration module APIs and the topology module APIs.

The design of application extensibility is illustrated in Figure 6.8. One or more domain and application extensions can be added to the provisioning module, via an XML configuration file. Based on this configuration file, the application extensions

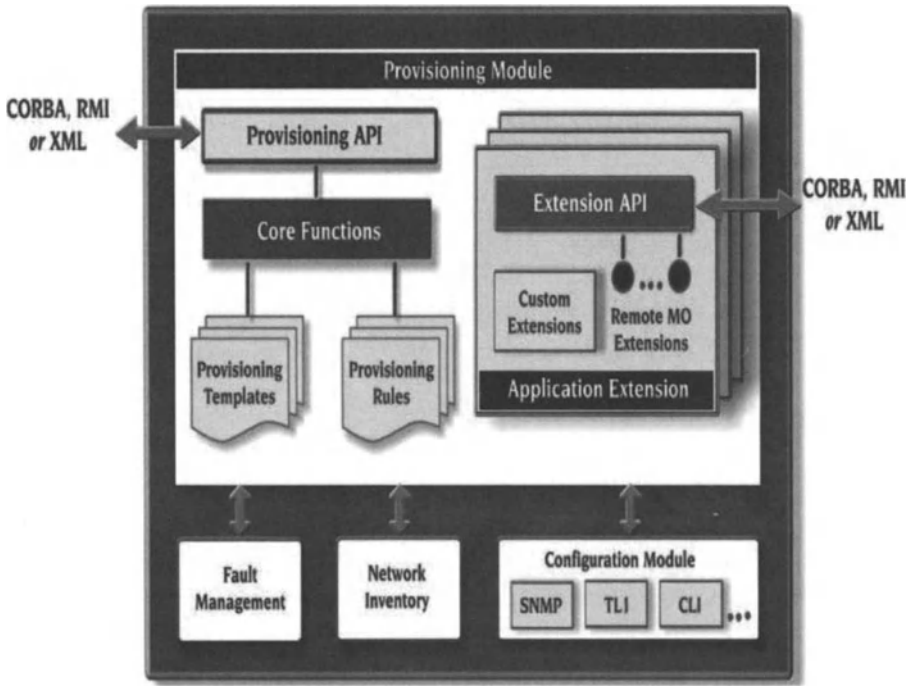


Figure 6.8 Application extension design.

will be initialised and made available. Each module is named and specific extensions can be accessed by name from the main provisioning API or from any of its extensions.

In addition to providing new API functions to provisioning clients, extensions can work with templates and extend the template processing capabilities in three ways:

1. Extensions can support special processing of template data as part of every client interaction: initializing and sending template to client, initiating the provisioning activities after user input and sending the results back to the client. This includes bypassing some or all of the template-based provisioning functions on the server.
2. Extensions can implement special parameters in the template, which will be processed by the application extensions.
3. Extensions can implement special configuration or other tags, so that these tags are processed by the application extensions at any stage of processing. For example, all network configuration can be done in the application extension, bypassing regular configuration and processing the new tags defined in the XML templates.

6.10 High-level Product Case Study

The TEMPo architecture follows TMN and supports FCAPS management functions. It has embedded device management functions for a single device and a high-level,

network-wide provisioning EMS providing end-to-end and integrated network management functions.

TEMPo, Tenor's Element Management Platform, is a distributed multiplatform client–server solution. A fully scalable and redundant architecture ensures system performance and resilience under heavy loads. Software updates and distribution to client applications are supported via push services that are provided by the server to minimise administration.

TEMPo represents and provisions services by creating "service contracts" that contain the customer information, order numbers, service endpoints, scheduled bandwidths, QoS constraints and associated SLA metrics. For example, a user might create a service contract for customer X specifying a 100 Mbps private line-equivalent service for video applications 9am–5pm on weekdays which drops to a best-effort service at other times, and also specifying that SLA reports are automatically generated daily and weekly detailing the service performance actually achieved, showing availability, latencies (delay), loss and utilization.

Services are created through a point and click, forms-based graphical interface, or through integration with existing OSS or third-party provisioning software.

In addition to graphical client interfaces, TEMPo provides integration interfaces using CORBA, RMI/Internet Inter-ORB Protocol (RMI/IIOP) and Enterprise Java Beans. Northbound interfaces allow integration of end-to-end management services with existing customer OSS to create automated provisioning systems that reduce the time to activate a service from weeks to minutes. This integration gives service providers the ability to offer a zero-touch provisioning model, allowing actual network provisioning operations to be performed automatically based on system orders.

When provisioning a new service, a service provider enters a contract with subscribers to ensure that the service meets a specified QoS. The contract is the SLA which includes metrics such as availability, latency and loss. To validate these and other important service parameters, the equipment and management system collects, stores and presents statistics for reporting on SLA compliance. These data must be collected from each managed element and distributed to reporting applications and billing systems. Tenor's TN250G performs traffic monitoring and measurement to support the creation of SLA reports. TEMPo collects, correlates and archives these measurements to provide historical and real-time performance reports which can be sent to third-party tools and applications.

AdventNet and Tenor share the understanding that service provisioning can be complex and time intensive for a service provider owing to its involving many departments, skills and systems. By addressing this, OSS vendors can help service providers reduce their costs. With the introduction of optical transport technologies to access, metro and long-haul networks, the ability to increase and control bandwidth has been improved by software provisioning techniques. The challenge ahead for service providers is rapid provisioning for the services layer. This will allow the combination of optical transport technology and packet-based traffic to deliver services and earn revenue.

TEMPo enables service providers to create and provision enhanced services that are focused on transforming optical capacity into services. The product has service shaping and monitoring capabilities, and business interfaces so that service

providers can create and deliver differentiated services. TEMPo draws on the service foundation classes of TN250G. Support for legacy service models and new service models eases service migration.

6.10.1 Product Overview

TEMPo is offered in two versions:

1. TEMPo Network Center Edition is an application for device configuration, network topology management, service creation and provisioning, fault management and real-time statistics reporting.
2. TEMPo Operator Edition is a network management solution for smaller network environments.

TEMPo Network Center Edition is a distributed, multi-user solution. It has a platform-independent management interface which provides GUI-based:

- device configuration
- service provisioning
- alarm management
- performance management
- SLA reporting.

The distributed architecture utilises an RDBMS engine with a distributed client application.

TEMPo Network Center Edition has historical data reporting and multiple concurrent client support, which is essential for global service provider networks. The product has APIs, including CORBA and RMI, as northbound interfaces. These allow integration of end-to-end management services with existing customer OSS.

Any network management and provisioning system involves the three architectural elements infrastructure and interfaces, domain and element management, and service creation and enhancement.

6.10.2 Infrastructure and Interfaces

Database support is necessary, as the database contains the information about the network and the provisioned services. A database must be chosen to achieve performance, scalability and seamless integration with the rest of a service provider's OSS environment. TEMPo allows customers to utilise any other JDBC database, but has been optimised for Oracle environments.[25] As TEMPo is written in Java, it can run on any hardware and software platform that supports a Java Virtual Machine.

6.10.3 Domain and Element Management

Service networks require continuous uptime. This means that device and network monitoring and management are key. Achieving service flexibility and a high level

[25] Pehaps because of Oracle's longer history with high-end RDBMS applications, Oracle has a greater presence in the OSS space than the competing Microsoft RDBMS technology.

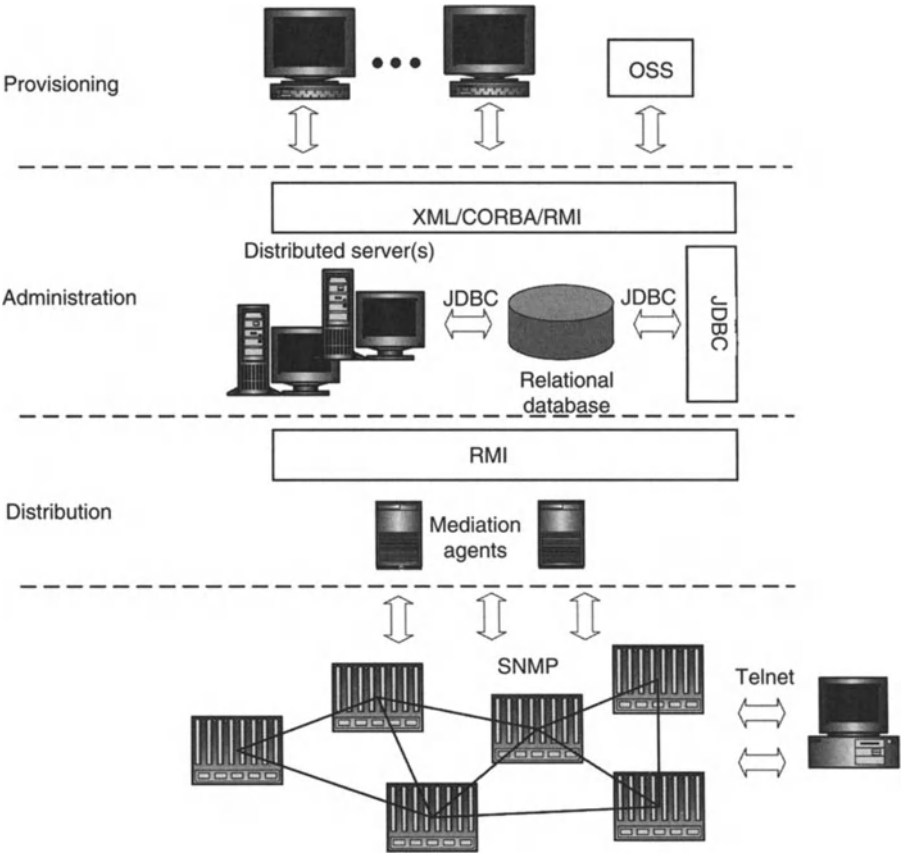


Figure 6.9 TEMPo's distributed architecture permits flow-through service provisioning in a scalable environment.

of network availability begins with a focus on element and domain management. With increasing bandwidth and port density and more sophisticated services comes the burden of device configuration. Networking elements commonly support many hundreds of interfaces, each with a specific service requirement. Of course, a command line interface is totally inappropriate for this type of environment. Service provisioning solutions must offer GUI-based point-and-click functionality, and TEMPo does offer this.

6.10.4 Service Provisioning and Enhancement

TEMPo was designed to enable service providers rapidly to define and provision enhanced service offerings. It is the first service provisioning product to allow a new class of service creation and provisioning through a graphical template which Tenor calls "ServiceExpress". The interface can be customised by service providers to represent standard and custom service offerings. From the interface, services can

quickly be defined and provisioned. From a ServiceExpress form, services are provisioned and then monitored to ensure that they are delivered.

ServiceExpress increases provisioning quality by masking the complexities and potential inaccuracies in provisioning through a CLI. To be fair, such functionality is no longer unusual. It would be a brave OSS vendor who lumped a service provider with nothing but a command line interface with which to provision new services.

6.11 CallGate Service Provisioning Functionality

CallGate is a mediation device and multi-filtering tool. CallGate filters switch-related data to and from GSM and analogue switches. Fulfilling a typical mediation role, it acts as an interface between switches and data destinations such as billing systems. CallGate functionality includes service provisioning of network resources such as switches. Figure 6.10 describes the general setup in which CallGate is usually installed.

Service provisioning commands are received by the CallGate server from the customer care layer. The commands are converted into the format(s) required by the switches and devices which must receive provisioning commands in order that the service be activated. In the case of service provisioning towards a switch, CallGate typically uses MTP/CMISE. The protocols that are recognised by network devices are an ongoing difficulty faced by OSS vendors, since there can be various old and exotic equipment in any network and such equipment may only recognise some non-standard proprietary protocol. If the software of a particular vendor does not support that protocol, then often the OSS software product will have to be extended by the vendor in order to support the protocol.

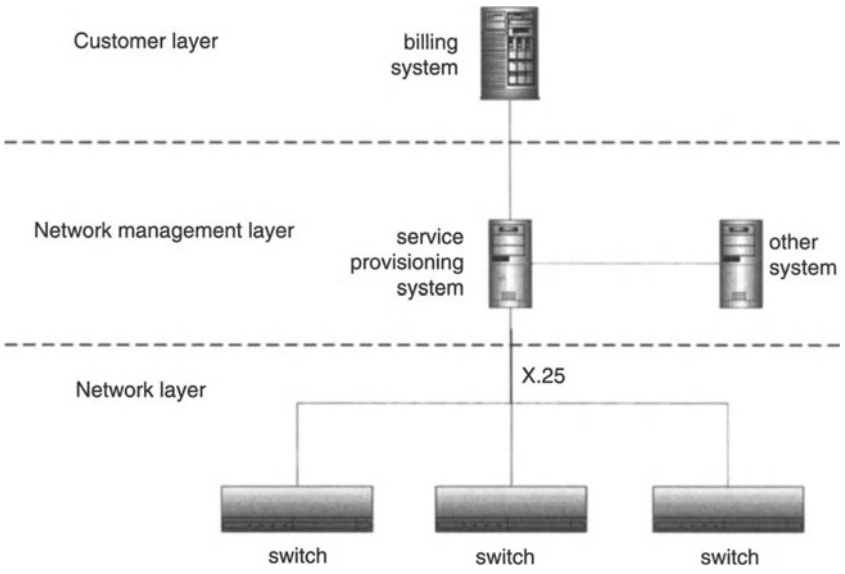


Figure 6.10 CallGate general setup.

Service provisioning information is received by the centralised server (information from customer care or billing system). If necessary, the information is split and forwarded via the LAN/WAN to each CallGate satellite. Each CallGate satellite updates the switch connected to it while the central CallGate server deals with the switch directly connected to it. Each satellite receives service provisioning verification information from the updated switches via the direct connections, and similarly with the central CallGate server and the updated switch connected to it. The verification information is forwarded from the satellites to the centralised server via a LAN/WAN. Verification information is summarised (for example, "updates in all switches are OK") in the centralised CallGate server and forwarded to the billing or CRM system, via a LAN/WAN.

Besides executing service provisioning commands by sending instructions to network elements, CallGate can read results or status information from the switches and forward it to a customer care system. If there is more than one (satellite) customer care system, then the commands are split at the central server and distributed.

Examples of typical Service Provisioning commands are:

- adding new subscribers to the switch (in series or individually)
- disconnecting and reconnecting subscribers (in series or individually)
- adding and removing value added services (in series or individually)
- bar all outgoing or incoming calls
- restrict calls within a close user group.

6.12 Service Provisioning OSS Solution

We base a detailed description of a service provisioning OSS solution on the Clarity Service Manager OSS product. The aim of service management products is to streamline provisioning and activation of telecom services. A service management OSS product should allow convergent service provisioning for products such as leased line, fixed, mobile, freecall, broadband and IP-based services. Service manager provides a single platform for convergent service activation. Users specify design rules for:

- shortest path through the network
- smallest number of equipment nodes to utilise
- least cost routing
- protected paths
- traffic distribution.

6.12.1 Service Orders

A service order is created to "order" work that is associated with providing a service to be done on a circuit. Such work may include assigning tasks, creating a circuit, designing a circuit, creating/modifying circuit work orders, creating new service order revisions and viewing the service order delivery report.

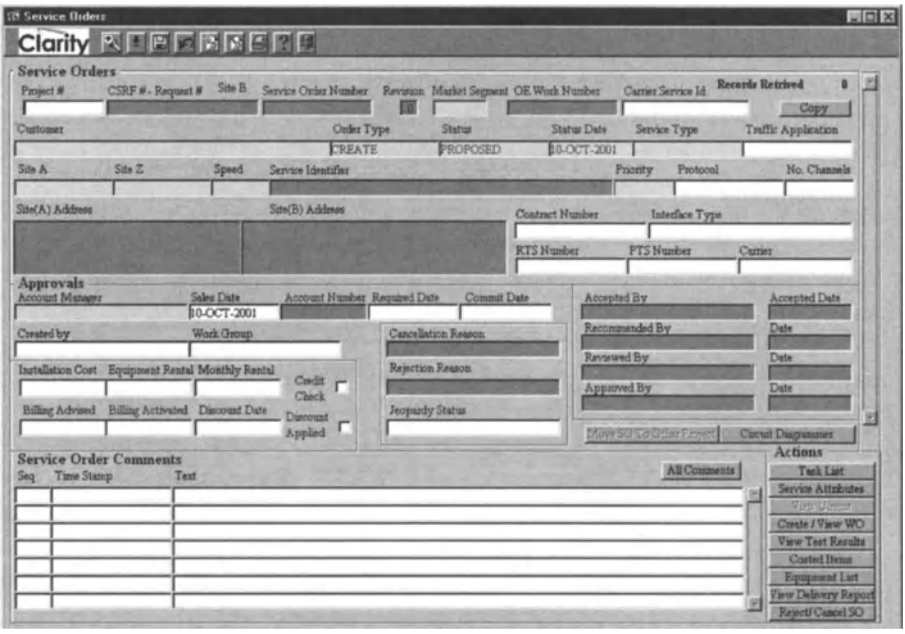


Figure 6.11 Service orders screen in Clarity Service Manager.

A service order screen contains the information that one expects a paper-based service order to have, such as shown in the Figure 6.11.

Information involved in raising a service order is given in Table 6.1.

Every service provider has an approvals process. Service orders are not executed automatically without question, since doing work costs money. A service provisioning or service management OSS implementation will have to account for the approvals process. This will follow the usual outline of any approvals process. For example, an account manager will be included as the person who requested the service, who can then be seen as having requested the service order on behalf of the customer who ordered the associated service. This is different from the user who actually entered the service order – which typically will be automatically populated with the username of the user who is logged in and creating the record. If all is going as it should be, then this user should be acting under the instruction(s) of the Account Manager. The Work Group of that user will also be automatically recorded.

There will be a sales date, which is the date on which the service was sold or on which the service was ordered. The customer's account number will be included and a "Provide by" date will be included, which is the date by which the customer wants the service to be available. There will then a "Commit date" which is the date by which the Account Manager has agreed that they can have the service available. The sale will have a contract number associated. The service provider may have to run a credit check on a customer, and whether or not this has been done is recorded along with its result. A record must be kept of whether a discount was given to the customer and the date on which the discount was agreed.

Table 6.1 Service order.

Information	Description
Service group number	Owing to the association between service orders and service groups, a unique number is generated to associate the service order with a service group
Product domain number	The description that applies to the service group number also applies to the product domain number
Service order number	This is a unique service order identification number
Revision number	
Market segment	The customer's market category is specified here. There are many ways to specify market category, e.g. corporate, wholesale
Carrier name	
Service number (carrier specified)	A number which specifies the service that the carrier will provide that is associated with the service order. Of course, the carrier may have its own name for the service that the customer has specified, and this allows a mapping between the service order and the carrier's terminology
Customer	The customer that has requested the service for which the service order is being raised
Order type	This is where the type of work that the service order requires to be done is specified
Status	Status options depend on the particular network management centre. Possible statuses are Proposed/Recommended/Reviewed/Approved/Cancelled/Closed
Status date	The date at which the current status was attained
Service type	The type of service that the customer has ordered (e.g. PSTN – Analogue)
Traffic application	This is the purpose of the service, e.g. Access, exchange-based, point of presence (POP)-based
Site A	The location where the service begins
Site A address	This is automatically populated
Site B	The location where the service ends
Site B address	Similar to site A address
Speed	The speed at which the circuit operates
Circuit name	One way of ensuring that circuit names are unique is by appending site A with site B with speed and then appending a 3-digit number to the result
Priority	The level of urgency of the service order
Protocol	The communication protocol to be used, such as TCP/IP
Number of channels	The number of channels that are allocated, in the case that the service is bandwidth-based
Interface type	This is the type of connection, such as V35, RS232, RS422
RTS number	"Request for Technical Study" identification number
PTS number	"Preliminary Technical Study" identification number
Installation cost	This is the cost of installing the service
Equipment rental cost	Monthly charge for renting the equipment
Monthly rental	Monthly charge for the service, not including the equipment cost

The person who accepts the work outlined in the service order has their name recorded along with the date on which they accepted the work.

Whether or not the work specified in the service order is necessary or appropriate may not always be obvious. In particular, it may not be clear to the Account Manager or to the manager whose responsibility it is to approve or reject the service order. It is possible that none of these people is familiar with the technical ins and outs of how a particular service is made available to a customer after it is sold. Hence the necessity of the work in a service order being recommended by a knowledgeable employee. The name of the person making the recommendation is given along with the date on which the recommendation is made.

The name of the person responsible for reviewing the service order is recorded along with the date. The name of the person responsible for approving or rejecting the service order is recorded with the date of their decision. If they rejected and hence cancelled the service order, then the reason for cancellation must be recorded.

A service order will typically have work involved for which there is a charge, and so the Billing Group must be advised. Of course, the service itself will certainly be charged for. The date on which the Billing Group is advised is recorded. The date on which the billing is activated is also recorded. This will be done following an advice from the Billing Group.

The service order feature of a service management OSS application should provide an item list for the service order together with costing for each item. The service order feature should also allow service attributes to be specified. A link to the appropriate tools may be provided from the service order screen. This will allow software-based work that is specified in the service order to be conveniently carried out immediately. For example, the Clarity Service Manager allows the Circuit Diagrammer application to be opened from the service order screen.

If the project to which a service order applies changes or if the service order is no longer needed in a particular project but happens to be just right for another project, then it can be moved to other projects with a single click. The ability to copy service orders would be desirable and is not difficult to add to a service management OSS.

A service order can be directly linked to a product domain record or the service order can be linked to a service group record which is then associated with a product domain record. This association has the effect of creating a template, with the entry of implementation tasks and other service order characteristics.

The work involved in satisfying a service order is given in the implementation task list. As the screen in Figure 6.13 shows, the first implementation task in completing a service order is the approval of the order, which was discussed above, and the final task is closure of the service order. A service order is closed when all implementation tasks have been completed so that the service order has been implemented and the service is active.

Each implementation task aside from approving and closing a service order generally requires a Work Order since the raising of a Work Order is the means by which the workforce of a service provider is given instructions. Work Orders are the link between the raising of a service order resulting from a customer purchase and the work that the service provider must carry out in order to deliver to the customer what they have purchased.

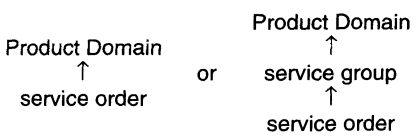


Figure 6.12 Association between product domain and other entities.

6.12.2 Work Orders

Details about Work Orders do not fit within the scope of this book. The Work Order features of Clarity's Service Manager essentially implements Work Orders as they are generally understood.

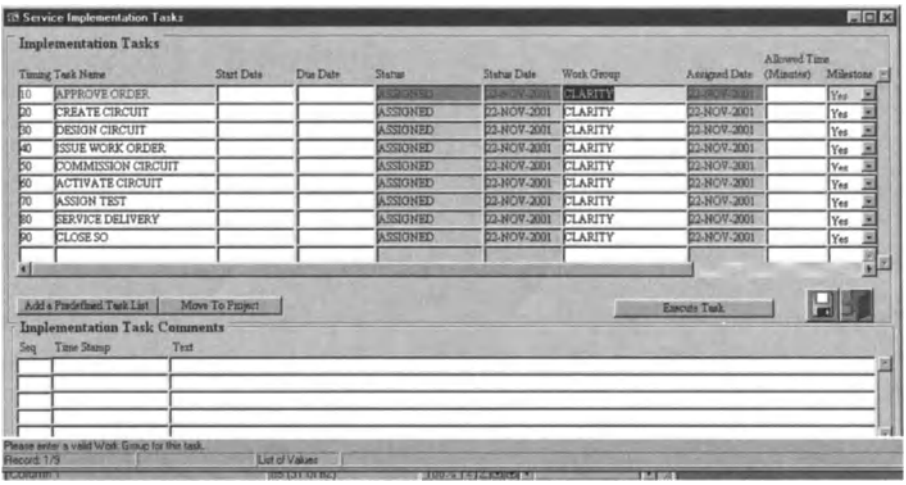


Figure 6.13 Service implementation tasks for a service order.

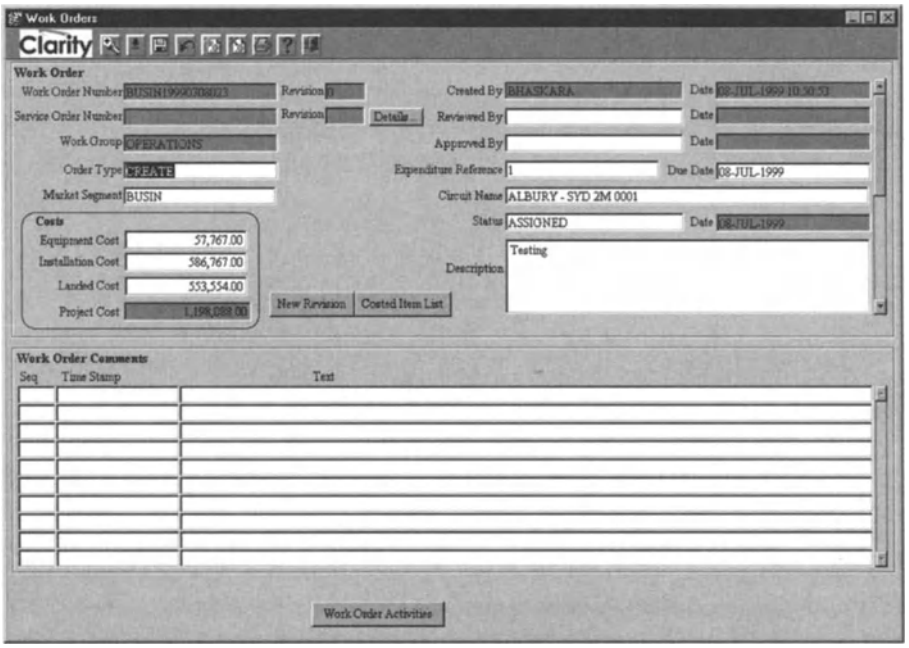


Figure 6.14 Work Orders screen in Clarity Service Manager.

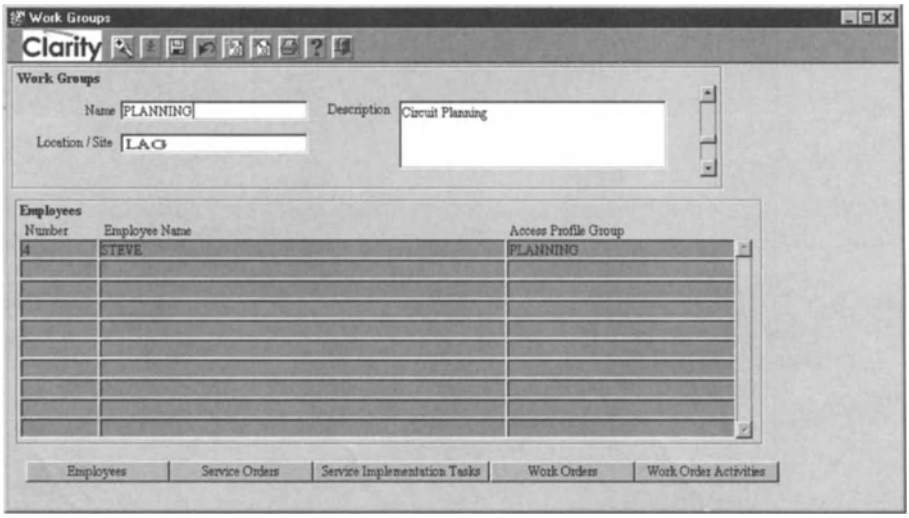


Figure 6.15 Work Groups screen in Clarity Service Manager.

The Work Groups in the Work Order screen are an interpretation of the organisation of staff which perform the work in a carrier. For example, the screen in Figure 6.15 illustrates the beginnings of the Work Group record for the Work Group called "Planning", which could be charged with any type of planning. In a given carrier, the word "planning" might have a very particular meaning in an operational sense. In other carriers, a more specific name would be. This is a good example of how an OSS should be flexible enough to fit with a carrier's particular processes. If an OSS requires a carrier or service provider to modify their business processes to fit with how an OSS vendor thinks a telco should be run, then that OSS vendor may not sell much software. Although there are similarities between all telcos, there are also key differences resulting from their particular history and evolutionary path and from particulars of the market which they address.

At industry conferences and trade shows, OSS vendors are repeatedly told that no two telcos are alike and successful OSS vendors have taken this message to heart, ensuring that their offering is not inadvertently designed in such a way that would serve to lock a telco into some particular business process design. In the past, the biggest competitors of OSS vendors have been in-house development teams of telcos themselves. A carrier or service provider knows its own needs better than anyone else knows them, and often has the resources to build its own OSS from the ground up. In the not-too-distant past, *all* carriers wrote their own OSS software. Now, the benefits of buying from independent and specialist OSS vendors are undeniable and widely known, so that carriers and service providers are increasingly

[26] The same is happening in the network equipment space. For example, Wipro says in "Packet over SONET interface: a design strategy", a December 2001 whitepaper: "There is a definite shift in product engineering today, away from 'developing everything in house'. Network equipment vendors are increasingly looking out for components, so that they could concentrate on fine-tuning system architecture."

choosing to shun in-house development.[26] However, the background of being able to write an OSS precisely how *they* want it written remains a high-order issue for carriers, even as they become customers of OSS vendors. Indeed, customisation – including the rewriting of large slabs of code – is something with which OSS vendors, unfortunately, are very familiar. This operates as a powerful incentive to OSS vendors to build generality, programmability and well-defined interfaces into their OSS software offering so that it can be implemented in unique environments with minimal hard coding and, as far as possible, avoiding long and complex implementation projects. It is true that regularly engaging in wholesale rewriting of code to please the particular desires of a telco offers a revenue stream for OSS vendors. However, in the long run, an OSS vendor can gain much more market share by purely selling software licences. To be able to do this, programmability and customisability must be built in to the software.

By this point, the reader may have noticed that the Clarity Service Manager is doing nothing but implementing in software each step of the operations of a telecom service provider. In other words, OSS implement in software business processes. To the extent that OSS software does this, it automates business processes.

A cynic might ask, "So how much value does an OSS actually add, if it simply transfers to software what the service provider already did without software?" This brings us to a crux of the business case for OSS software.

The answer is no different from the answer to the same question about word processing software: "How much value does MS Word add when all it does is enable us to write, when we were able to write long before Wordstar or other ancient word processing software was marketed?" Although we might mostly choose to do with software precisely what we did without it, we can do it much more efficiently, we can automate certain tedious aspects of tasks, we can share information more easily, we can avoid errors, we can integrate separate processes by transferring information electronically, and so on. The question of whether spreadsheet software has made accountants' lives easier is readily answered. Using spreadsheet software, it is simple to perform tasks that accountants did before PCs came along. However, spreadsheet software also allows accountants to do things that were previously nearly impossible, such as "What if" scenarios, which, before spreadsheet software, would only have been able to be performed by people who would have been regarded as accounting geniuses. Because such a task requires rare skills in humans, it would certainly not have been systematically carried out except by the largest companies which had the resources to invest in armies of data processing staff.

What do the obvious benefits of software tell us about OSS in telecom service providers? In the context of Work Orders, electronic entry of Work Orders in the Clarity system can immediately inform reviewers and approvers (see Figure 6.15) of the existence of a Work Order. Errors in the selection of the relevant circuit (if any) are avoided by referring to the table of circuits. Costs can be automatically calculated. Non-completion of the Work Order by the due date can at least be informed to the appropriate manager to ensure that the Work Order is raised to an urgent status. It is easy for any manager to check on all involved in the Work Order for the purpose of performance reviews or to understand what went wrong if something did go wrong in executing the Work Order. The phrase "any manager" in the previous

Table 6.2 Attributes of a product domain.

Attribute	Description
Domain number	A unique number for the product domain that is specified by the user
Customer ID	The identifier of the customer connected to the product domain
Customer name	This will be automatically populated based on the customer ID
Account manager	This is the account manager of the selected customer
Request number	These will be added as requests are made for this product domain. That is, there is a one-to-many relationship between product domains and product domain requests
Requested delivery date	This is the date that the initial product domain was raised
Product type	This comprises the various product types which makes up the product domain. There will be a many-to-many relationship between product domains and product types
Market segment	This is the business category into which the product domain fits
Assigned workgroup	This is the service group management workgroup to which the product domain is assigned
Request status	Each service order that is related to the product domain is given a status
Date	This is the date on which the order status was last changed
Created by	This is the user ID of the person who created the product domain and entered its details
Date created	This is the date that the product domain was entered into the system
Created workgroup	The workgroup of the user who created the product domain at the time they created the product domain

sentence is important because when the application is distributed on a network, there is no need for administrative staff to chase up a piece of paper stored in some filing cabinet. So, although Work Orders are almost as old as work itself, implementing Work Order functionality in OSS software is a great value-add for a telecom service provider. Similar things can be said about other OSS functionality, such as service orders.

Many other features of Clarity's Service Manager, which we are using for illustrative purposes, are examples of the "What if" scenarios of spreadsheets: they simply would not be possible without software. For example, to what extent were product domains or service groups used before OSS software came around? Circuit diagrams and graphical information system (GIS) functionality are functionalities which even more obviously have relatively lame parallels in the pen-and-paper world.

6.12.3 Product Domains

The product domain is a high-level tying together of customer order, service group and service orders. They are related in the following way:

- A product domain contains one or more service groups and must be created before entering service group details.
- A service group may contain more than one service order and must be created before entering service order details.
- A service order can exist with or without a product domain and a product domain–service group. However, if a service order has a service group then it must also have a product domain.

A product domain comprises the attributes given in Table 6.2. The main screen for product domains is shown in Figure 6.16.

6.12.4 Service Groups

A service group is a service category label or a group of services. Over many years of implementing OSS software for telecom carriers, Clarity found that carriers wanted to

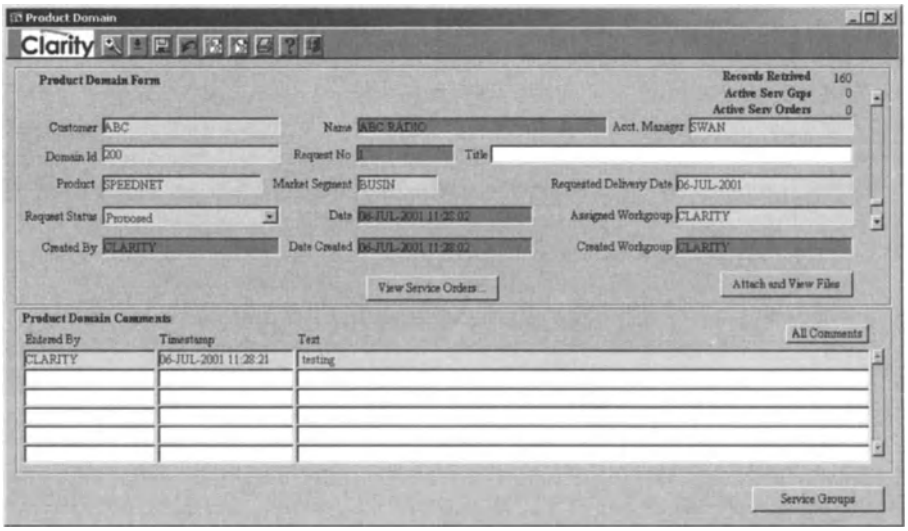


Figure 6.16 Product domain screen in Clarity Service Manager.

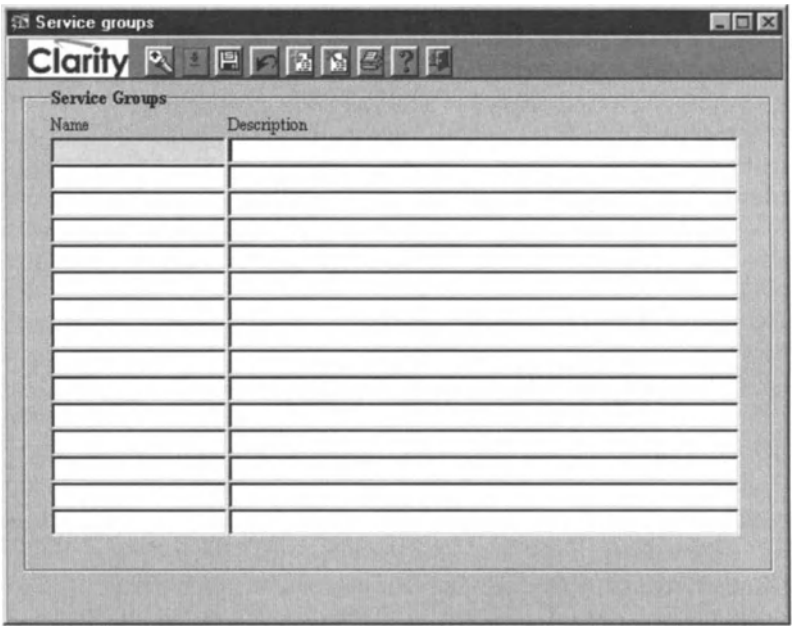


Figure 6.17 Service groups in Clarity Service Manager.

be able to categorise their services. Service groups allow this. For example, a carrier may have a service group called "Broadband". Service orders for, say, 500 kbps, 1 Mbps xDSL and Tx connections would be attached to this service group. Alternatively, a service provider might like to create "xDSL residential", "xDSL business" and "Tx business" as service groups.

A service group is associated with a product domain. Business rules apply when updating or creating service group records, none of which are unexpected (as they generally simply respect the usual requirements for when a record is related to another):

- A service group cannot exist without a corresponding product domain, and so the product domain corresponding to a service group must be created before creating that service group.
- Service group records cannot be updated if there are open service orders attached to the service group.
- All service orders associated with a service group must be closed before the service group can be closed.
- A service group cannot be deleted if it has an associated service order, since the service group associated with a service order is an essential feature of the service order.
- The Entered By and Timestamp fields are automatically populated when comments are entered for a service group.

Aside from categorisation of service orders for creating a connection between service orders and marketing (or other)-oriented service classification, service

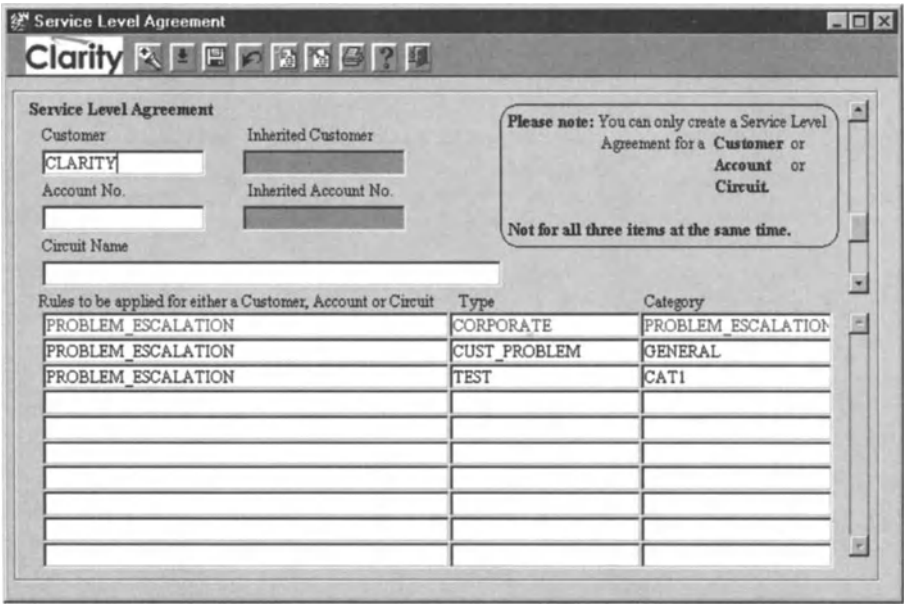


Figure 6.18 Service Level Agreements in Clarity Service Manager.

groups speed the creation of service orders. In particular, an implementation task list can be defined for service groups. When a service order is raised, there is no need to think through the necessary tasks to complete the service order because the implementation task list in the associated service group provides a template. In the Clarity implementation, those tasks from the service group list that are wanted in a service order must be selected manually. However, if a service provider desired, this could be modified so that all implementation tasks are added to a service order attached to a service group and then negative decisions are made on which of these tasks should be *removed* from any given service order.

6.12.5 Service Level Agreements (SLAs)

The service level agreement screen provides a list of rules that are to be adhered to under an SLA for a customer, account or circuit.

SLAs are indirectly a part of service provisioning since the resources that are necessary for a service to comply with SLA rules must be made available at the time of provisioning. Strict SLA rules may demand that redundant capacity is made available in the event of a failure, whereas weaker or non-existent SLA rules may make it sufficient that an alternative path be found from available capacity and be activated only if and when a failure occurs in the provisioned capacity.

6.13 Conclusion

The functions of service provisioning and activation lie between services and configuration management. Provisioning a service involves sending commands to network devices so that certain network functionality is invoked. Activation involves sending commands such that the appropriate network elements begin to act in concert making a service available to a customer. The many different products touched upon in this chapter hint at the wide variety of approaches and emphases in carrying out the basic steps involved in service provisioning. Because the services that are available are so various, there has been plenty of room for OSS vendors to differentiate themselves by appealing to particular needs and problems of telecom service providers. OSS vendors are also educated by their customers, who are service providers. Different service providers have different "must haves" and so the functionality of different OSS products is seen to show different emphases.

As services become more complex while lifespan decreases, service provisioning OSS must become richer in functionality. This is because service provisioning and activation software will become important in ensuring that service providers are not limited in implementing the precise service specifications that are needed to generate revenue. Northbound interfaces will also become more important as marketing, sales and business support software will be plugged in for maximum automation in the entire product lifecycle. At the same time, vendors of service provisioning and activation software will have their hands full in ensuring that their *southbound* interfaces cater to the increasing functionality and complexity of network devices, although this will be aided by the increasing intelligence in network elements.

7 Implementing Service Level Management

7.1 Introduction

Service level management (SLM) is the managing of various phases in the entire lifecycle of a telecom service. SLM is also sometimes referred to as *total* service level management.

The lifecycle phases can be reduced to the following five:

1. Service design and development.
   (a) Model a new service.
   (b) Deploy the OSS infrastructure. Preferably, the service provider's existing OSS infrastructure would be sufficiently flexible that "deployment" of OSS infrastructure for a new service would merely involve reprogramming, installing plug-in OSS modules or extending OSS modules using APIs.
   (c) Service import/export.
   (d) Define service level objectives.
2. Negotiation with prospective customers and sales, including definition of SLA for customer(s).
3. Execution, including service level monitoring.
4. Re-assessment.
5. Repeat the previous four steps.

Implementing SLM involves more than integrating a technology solution with existing OSS systems. It also involves creating business processes. We will identify aspects of SLM that need to be considered. In addition, we will provide some pointers on how to ensure a successful implementation with reference to several industry initiatives.

7.2 The SLA universe

There are three areas associated with the application of SLAs to an operator's business. Internal SLAs formalise day-to-day matters such as how long people at the NOC expect a field technician to take to reconfigure a switch. As shown in Figure 7.2, internal SLAs are most often focused on managing components of the service delivery

K. Misra et al. (eds.), *OSS for Telecom Networks*

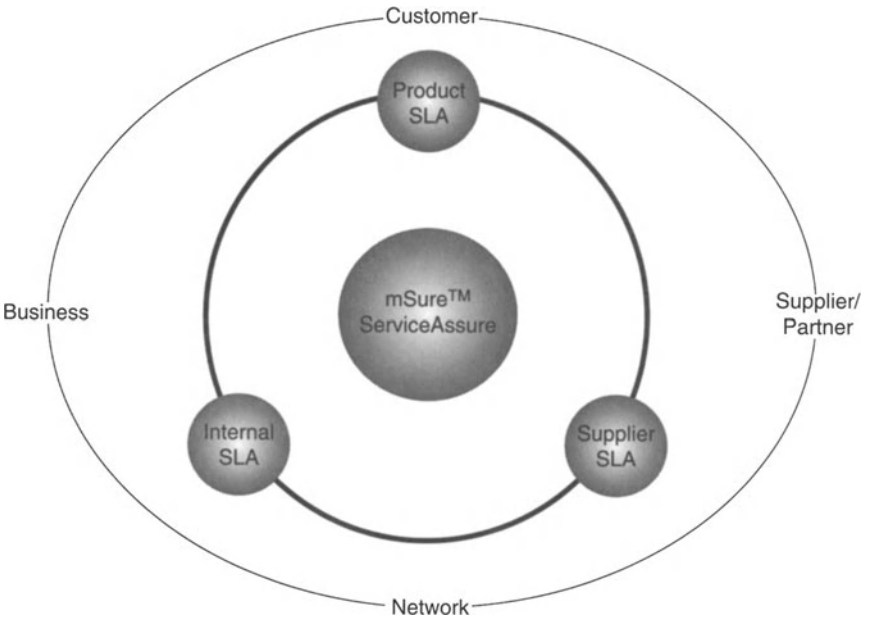


Figure 7.1 The SLA universe.

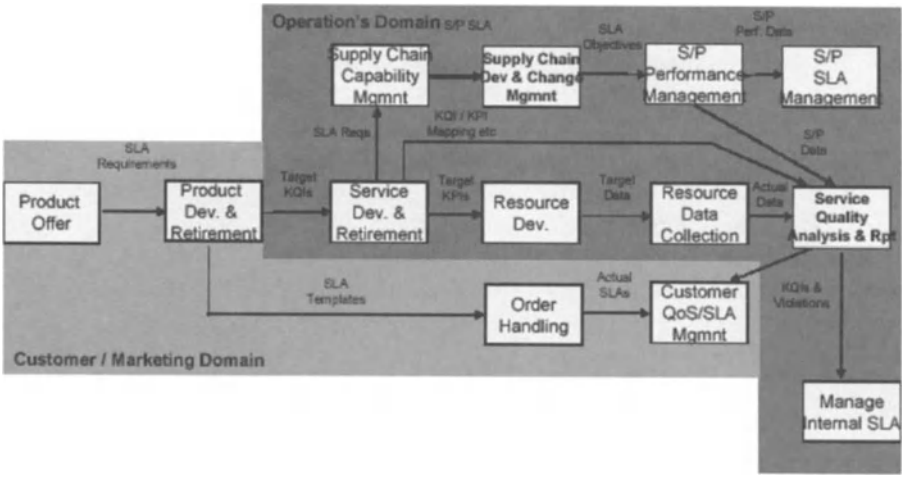


Figure 7.2 eTOM process flow.

chain. The components are aggregated to form the measurable corresponding to the end-to-end service. By their nature, these SLAs are often not truly customer-focused as they are not written in terms that can be understood by the end-customer. The key users of these SLAs are the functions responsible for managing the service

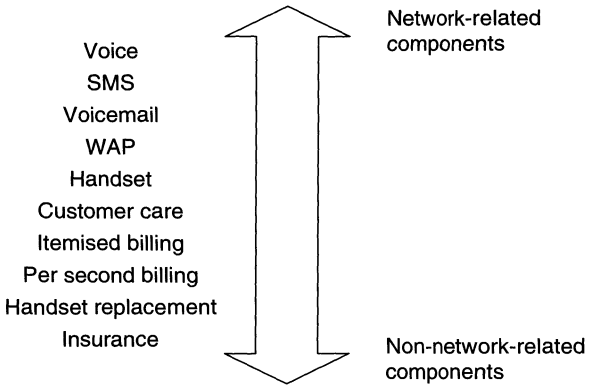


Figure 7.3 Product components.

components and for managing those components against agreed quality objectives. Other parts of the operator's business rely on internal SLAs:

- to drive improved efficiency
- for understanding the service delivery performance within the functions of the business.

SLM is not restricted to network-based service components. Indeed, SLM is also be applied to non-network services, such as billing.

As operators become more reliant on third parties to deliver services and content, it becomes important to implement SLAs with those suppliers and partners. The implementation of SLAs in this area will often improve the quality of content delivery. It will also enable the operator to pass some of the financial risk of service degradation on to their third parties. These SLAs are similar to internal SLAs and form part of the end-to-end SLA. This may, however, have an impact on the accounting processes for paying for the content services.

The final area that needs to be considered is the end-customer or external SLA. This is the type of SLA with which people are most familiar. In considering this area, the nature of the products sold to the customer must be taken into account. Although internally the focus may be on individual services, the offering to the customer tends to be a product comprising a number of services, both network and non-network. An example of a non-network service is maintenance or helpdesk support.

Figure 7.3 shows the components of a typical product offering for a mobile operator. Some components are network-derived, but several have little or no dependence on network resources. However, the contract with the customer is likely to encompass all aspects of the product including the non-network aspects. Therefore, the external SLA is likely to consist of a wide-ranging set of parameters defined in terms that are easily understood by the customer. Deriving this set of SLA criteria requires a clear understanding of what is important to the end-user, and is therefore best achieved by discussion and negotiation with the intended users.

To validate SLA service parameters, such as availability, latency and loss, the equipment and management system must collect, store and present statistics for

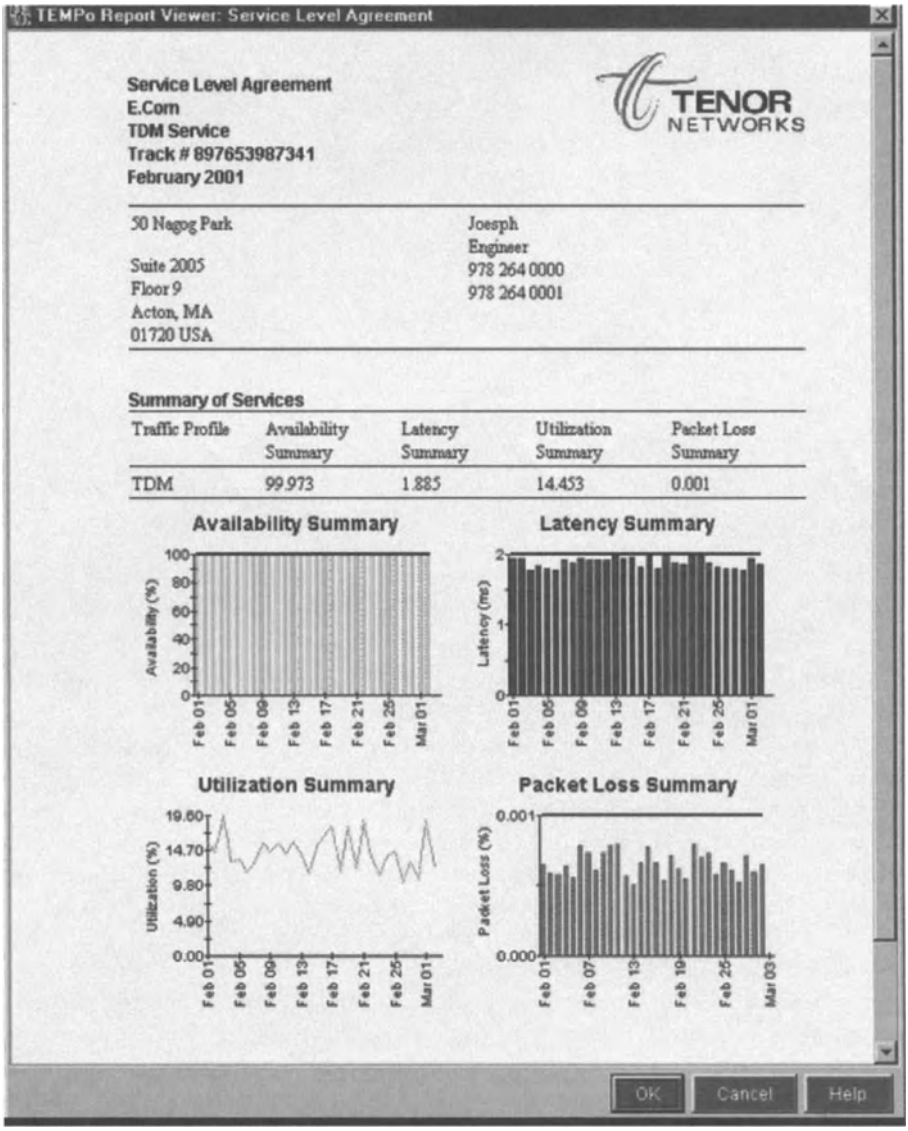


Figure 7.4 TEMPo's data collection and reporting module generates SLA reports for each service created.

conformance reporting. An example of how this might be experienced at the NOC is given by Tenor's TEMPo product. This product performs traffic monitoring and measurement to support the creation of SLA reports. TEMPo collects, correlates and archives these measurements to provide historical and real-time performance reports. These data can be presented in customizable record formats for easy integration into third-party tools and applications.

TEMPo provides web-based graphical access to SLA conformance information and reports can be saved in PDF format. TEMPo's SLA reports are automatically

generated daily and weekly. They detail the service performance achieved, showing availability, latencies, loss and utilization. These reports are created by either the point-and-click or the forms-based GUI method, or through integration with other software, such as the existing OSS systems.

Figure 7.2 shows the simplified eTOM process flows involved in defining the types of SLA discussed above. It also attempts to identify where those processes fit in relation to the domains of customer/marketing and operations. Some of the terms which appear in Figure 7.2, such as key performance indicators (KPIs) and key quality indicators (KQIs), are discussed later in the chapter.

7.3 End-to-end QoS

Driving a motor car and driving a telecom network can be compared in order to help in understanding the tasks involved in managing a telecom network. The way in which telecom networks have been managed in the past focused on short-term and immediate problems, rather than on the issues that ultimately matter most to the business of the telecom service provider.

Every day, thousands of operators travel by car to work. Unsurprisingly, these drivers spend more than 95% of the time looking out of the front windscreen. Of the remaining 5%, some time will be spent checking the rear view mirror, regulating the temperature, adjusting the volume on the CD player or radio, listening to traffic reports, etc. The drivers' focus is on reaching their destination safely, comfortably and on time. Less than 1% of time will be spent checking the vehicle instrumentation for problems.

Once that same operator arrives at work, they will spend 95% of the time checking for alarms on the network's "engine management system". They will even spend considerably more time drinking coffee than "looking out of the window" to see what their customers are experiencing.

An analysis carried out as part of the TMF's SQM Catalyst project showed that of the 4000 or so possible alarms that the targeted network components were capable of generating, less than 0.1% gave any indication of the impact on the customer. Thinking in this way leads to the obvious question of why operators manage their networks primarily on network events. Even performance management data give more information on the performance of an individual network component than they do about the performance of the end-to-end service delivery. Of course, no one would suggest that an operator should ignore the alarms generated by network components any more than a driver should ignore the oil warning light. But surely the emphasis must, metaphorically, be to look out of the window to ensure some depth of awareness of customer experience.

The findings of the TMF's SQM catalyst project highlighted the need to find more effective ways of managing service quality. It found that this could be achieved by collecting data from multiple sources and aggregating them to provide an overall quality measurement. The project then proved the concept by aggregating alarms, performance statistics, Call Detail Records (CDRs) and application event logs.

Figure 7.5 presents an example of how end-to-end QoS may be measured for a service. This may be done by breaking a service down into service components, each

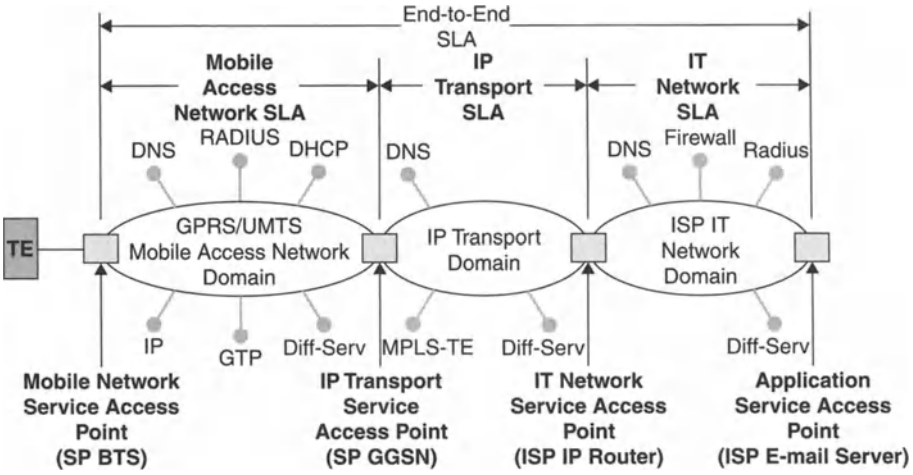


Figure 7.5 End-to-end QoS.

of which may have one or several internal SLAs associated with it. The combination of the SLAs that are internal to an end-to-end process flow provides a measurement of the totality of a service, and hence gives an indication of customer experience.

The diagram presents a granular approach that is not always appreciated in SLM discussions. This level of granularity is not even offered in all SLM systems. It may be possible to measure service quality through the use of passive and intrusive probes that provide a full end-to-end measurement. However, this approach provides little information that will enable the operator to arrest or prevent service degradation. By breaking the service down into components, and making that information available to the user, it is possible to provide root cause information that will allow faster corrective action.

7.4 Industry Initiatives

A number of activities within the industry are focused on SLM. Once again, the TMF has a number of projects focused on managing service quality. The TMF SLA Handbook[28] gives details of the structure and application of SLAs. The TMF Wireless Services Measurements Team (WSMT) were, at the time of writing, defining a set of measurables required for service quality from UMTS radio access network (RAN) resources. These new sets of measurements were submitted to the third-generation mobile partnership project ("3GPP") for inclusion in the Release 5 specifications. The WSMT produced a handbook[29] that defines the methodology for defining the SLA parameters. The work of this team includes mapping the processes defined in the TMF eTOM[30] on to the process flow for defining SLAs.

[28] SLA and QoS Handbook, TMF Version 1.0, November 2000.
[29] TMF Wireless Services Team Handbook, GB923.
[30] TMF eTOM Business Process Framework, GB921 version 2.5.

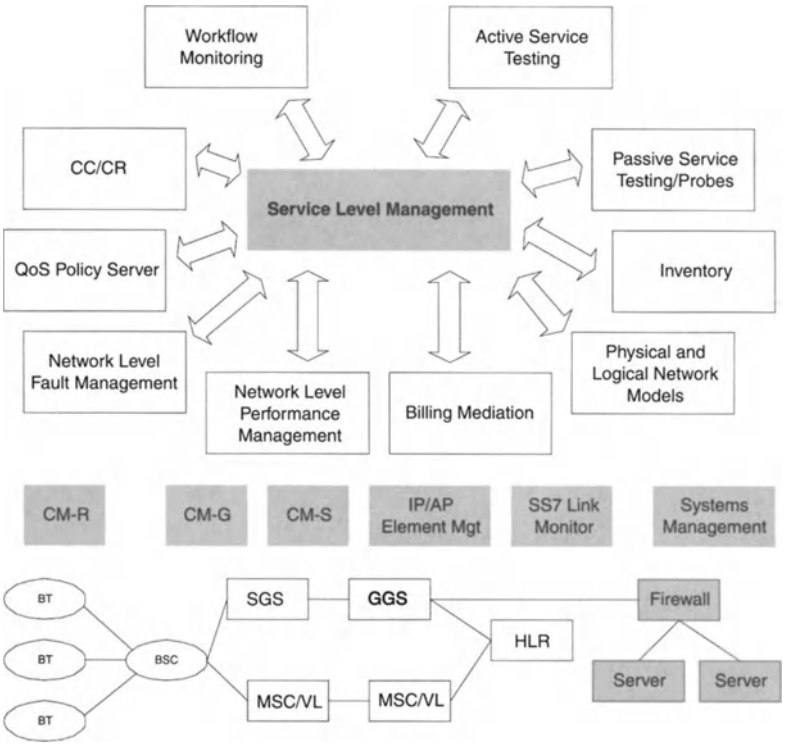


Figure 7.6 SLM architecture.

7.5 Technology

There are several offerings from vendors that claim to provide the technology for SLM. Unfortunately, some of these solutions do not provide complete coverage of all that is required for effective SLM.

Some products are based primarily on the collection of fault data and are built on top of existing fault management solutions. However, as discussed above, in many cases there is only a restricted capability to measure service quality when using fault management data alone. This type of approach also tends toward reactive rather than proactive management.

The SLM architecture shown in Figure 7.6 indicates that a number of different data sources are required to support effective SLM. As discussed above, these data sources are not just network focused but also need to extract data from other *business* systems.

A solution based on aggregating data from multiple sources raises questions regarding how the complexity of such a solution can be implemented and managed. There are two keys to achieving this kind of implementation.

First, a hierarchical definition of services is used in such a way that allows the inheritance of generic key quality indicators. Figure 7.7 uses the definition of a

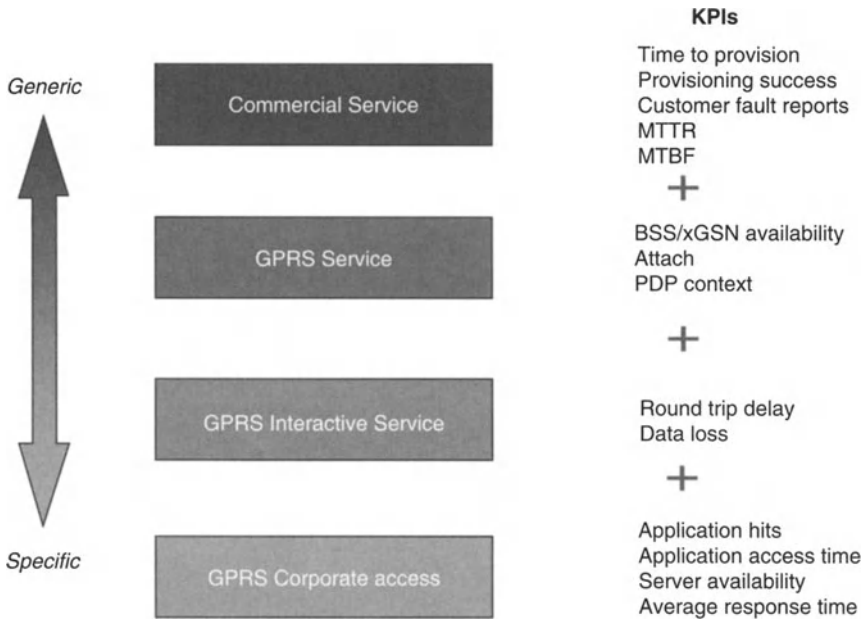


Figure 7.7 KPIs with a range of granularities.

GPRS service to show the inheritance approach for defining SLA parameters. A number of measurables that are common to all services are defined. In addition to these, other parameters may be identified that are applicable to all GPRS services. The approach then extends to defining additional parameters that are required for managing an SLA against a specific GPRS service. This hierarchical approach reduces the time and effort to implement SLAs for new services that are based on existing services.

Second, the solution must be capable of supporting the use of hierarchical measurements. There are a few solutions that successfully support this granular, hierarchical approach to SLM. One such is ServiceAssure from Comnitel Technologies.

The inheritance approach enables the operator to implement service management in a structured way and to deliver some real benefits to their business quickly. A fully top-down approach to defining SLA may be time consuming, and in early implementations can appear daunting. On the other hand, the definition of generic services tends to be quicker, easier to understand and easier to implement.

7.6 Example: Key Quality Indicators for Wireless Services Measurement

Service Providers have historically reported the performance of their networks against a set of business agreed key performance indicators (KPIs). These KPIs are,

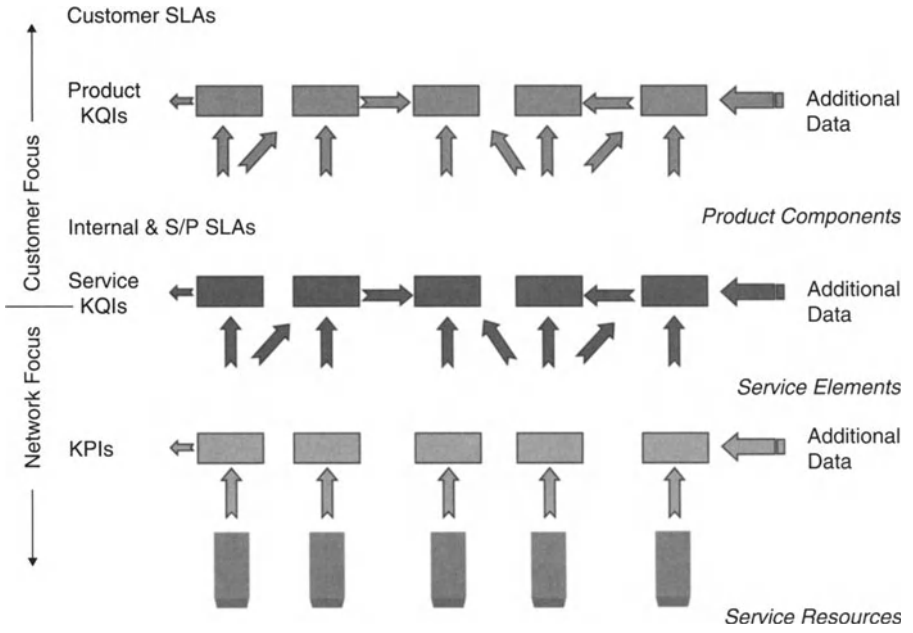


Figure 7.8 Hierarchy of key indicators.

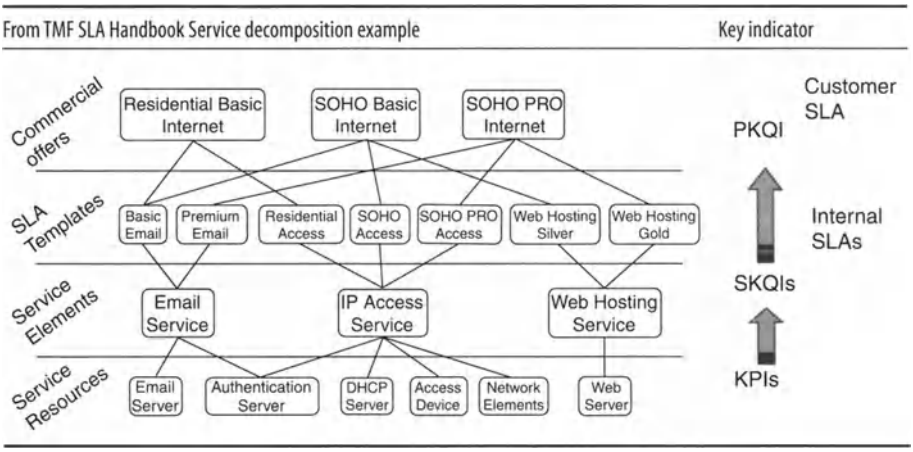
by their nature, network focused and provide little direct indication of the end-to-end service delivery that the network supports. Nevertheless, KPIs are an important measurement for network operations and will continue to be so for the foreseeable future as they indicate performance of individual service components. KPIs are usually derived from a number of data sources, and not just limited to network performance statistical data. For example, network fault data are also often used in calculating availability KPIs.

The move towards service-focused management leads to a requirement for a "new breed" of indicators that are focused on service quality rather than on network performance. These are key quality indicators (KQIs) and each measures a specific aspect of product performance, product components/services or service elements, and they are calculated using data from a number of sources including the KPIs.

At the highest level, a KQI, or a group of KQIs, is required to monitor the quality of the product offered to the end-user. These KQIs often form part of the SLA between the provider and the customer. Product-specific KQIs derive some of their data from the lower level service KQIs, with service KQIs focused on monitoring the performance of individual product components/services. In its simplest form, a KQI may have a single KPI as its data source. Usually, a service KQI will aggregate several KPIs to calculate the service element quality, and the product KQIs will aggregate multiple service KQIs. Figure 7.8 describes the key indicator hierarchy.

The most obvious use for KQIs is to calculate SLA compliance. Usually, multiple KQIs will be used in the SLA management processes. Service KQIs typically support

Table 7.1 Key indicators versus SLA handbook.



multiple internal or supplier/partner SLAs and product KQIs typically support multiple *external* or customer SLAs.

A service element may depend on a number of network resources and so a service KQI may call upon multiple KPIs in the aggregation process.

The nature of services that may be included in products offered by service providers is likely to result in variants of a basic service. Therefore, it is important that any given KQI can utilise other KQIs as part of its aggregation algorithm. Additional contextual information is used to complete the aggregation process. For example, it should be possible to add weighting factors to reflect the relative importance of each KQI parameter.

To summarise, starting from network elements, the network performance data are aggregated to provide KPIs that indicate service resource performance. Decomposition of the KPI provides the information that is necessary to identify the individual service resource that causes degradation in service quality. The KPIs are used to produce the service KQIs and these in turn are the key indicators of the performance of the various elements of any particular service.

Service KQIs are the primary input for management of internal or supplier/partner SLAs that calculate actual service delivery quality against design targets. Service KQIs are also used in supplier/partner contractual agreements. Service KQIs provide the main source of data for the product KQIs. The purpose of the latter is to manage product quality and to provide data for monitoring SLA compliance.

Table 7.1 describes the mapping of the key indicators onto the service decomposition described in the TMF's SLA Handbook.

Considering the top-down view, a service degradation is identified at the KQI level. It is then possible to drill down logically through the components of the KQI to the particular KPI/KQI which is associated with the cause of the degradation. Further decomposition of the violating KPI/KQI provides information on the root cause of the degradation. This enables the user to identify the particular service resource which caused the service quality degradation.

7.7 Conclusion

Implementing effective SLM requires deployment of an SLA management system, but there are several other issues of the project that must be considered:

- impact on existing business processes
- understanding the mapping of services to service resources
- defining the KQIs
- integrating with existing OSS for the collection of KPIs from multiple sources of differing data types in real time.

Although at first the task may seem to be very complex, a hierarchical approach to defining services and measurables provides a shorter time to implementation. It also ensures that tangible benefits are delivered to the business.

We thank Comnitel Technologies for allowing us to use their material in this chapter.

8 Telemanagement Forum: TOM

8.1 Introduction

The Telemanagement Forum (TMF) is a telecommunications industry organisation which works to develop OSS standards and to catalyse development of OSS technologies for the telecom industry. Members of the TMF include telecommunications service providers, infrastructure owners and managers, network equipment vendors, systems integrators and network management OSS vendors.

The TMF has developed the Telecom Operations Map (TOM) to act as a guideline for processes in the management of telecommunications network infrastructures, services and other aspects of the business of telecommunications. More recently, the TMF developed the Extended Telecom Operations Map™ (eTOM), which seeks to define processes across the entire telecom service provider business, and includes extensions that are necessary for a telecom service provider to function as an e-business. Amongst other things, eTOM encompasses human resources management, financial management, marketing and links between telecom service providers and other stakeholders such as suppliers and customers.

TMF has issued documents on TOM, eTOM and the applications of both. Information on these documents can be found at www.tmforum.org. This chapter introduces the TOM.

The aim of the TMF in developing the TOM is to describe a high-level identification of the primary end-to-end processes of fulfillment, assurance and billing and their sub-processes.

TMF's development of the TOM has proved useful for the industry because it:

- uses a high-level and generic approach which can be applied to almost any particular setting
- has been developed in consultation with the industry and embodies a wide range of views on network operations
- is based on real network architectures
- is based on how service providers actually manage their network operations.

8.2 What is the Telecom Operations Map?

The TOM uses the layers of the Telecommunication Management Network (TMN) model of the International Telecommunications Union (ITU-T) to organise core

K. Misra et al. (eds.), *OSS for Telecom Networks*

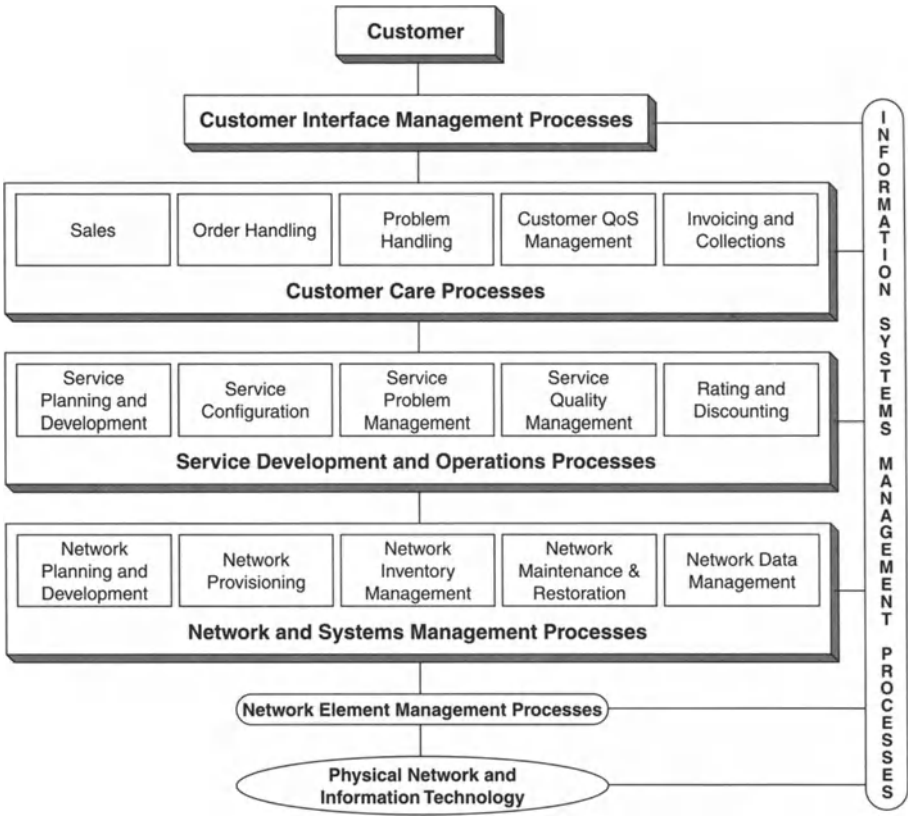


Figure 8.1 The Telecom Operations Map.

business processes. At the same time, TOM divides the Service Management layer into:

1. Customer Care processes
2. Service Development and Operations processes.

The Customer Care layer focuses on processes triggered by individual customer needs, such as new orders, billing and fault-handling. The Service Development and Operations Processes layer focuses on processes triggered by some group of customers subscribed to a service or family of services. This division of layers also reflects accountability for direct customer contact handling in Customer Care Processes and the need to focus on integration and automation of Customer Care Processes.

Customer Interface Management may be managed from within the individual Customer Care processes or across several Customer Care processes. As web-based interfaces become more common, the integrated Customer Care interface is a service differentiator. Thus, Customer Interface Management is distinguished as a separate entity rather than being included as just another customer care process. Customer Interface Management represents the actual interface between service provider and

customers. Interfacing functions are carried out as integrated customer contact management, voice response units and web interfaces. In turn, these are supported by Customer Care processes. As the TOM process framework is independent of organization, technology and service, it has sufficient generality to describe future as well as current processes.

The TMF has long recognised that the drive towards automation of telecommunications processes is economically compelling. Owing to increased competition and widespread automation in telecom operations, an operator cannot begin profitably to offer services without implementing a large degree of automation. Hence, "the leading focus of the TMF's mission is to enable end-to-end process automation of telecommunications and data services operations processes." The TOM has become the framework for achieving the end-to-end process automation envisioned by the TMF.

The TOM establishes a vision for the industry to compete through process-driven approaches to operations management. Part of this is to ensure integration among essential OSS involved in service delivery and support. TOM:

- defines the business processes used by service providers
- defines the linkages between these processes
- identifies interfaces
- defines the use by multiple processes of information relating to customers, services and network infrastructure.

Through the TOM, the TMF has sought to continue its progress in establishing:

1. An industry-owned common business process model.
2. Common definitions to describe the processes of any service provider.
3. Agreement on the basic information required to perform each process, sub-process and process activity. This includes sufficient high-level information to serve as the starting point for business requirements and information model development, and the satisfaction of those requirements through industry agreement and products.
4. A framework for identifying which processes and interfaces are most in need of integration and automation, and which are most dependent on industry agreement.

Besides the business process framework, the dimensions of operations management that are addressed in the TOM are:

- business management
- end-to-end process flows
- service or technology, e.g. broadband, IP, mobile/wireless
- business management sub-processes
- sub-processes or functions
- information exchange
- business relationships with suppliers and/or other providers
- systems/application and data architecture that supports the business processes.

Business process management is the design and definition of a company's processes and information, including the supporting data and systems structure. Business process management starts with a business process *architecture* designed

to accomplish a specific company's objectives upon which is defined the detail of end-to-end processes to deliver services to customers and to meet business requirements.

8.3 TMN and a Business Reference Model are Foundations of the TOM

The TOM takes the Telecommunications Management Network (TMN) model of the ITU-T as its basic foundation. TMN is built around the schematic pyramid shown in Figure 8.2.

The TOM defines processes at each of the TMN's five layers. The TOM also defines information flow between processes at the same layer, horizontal flow and information flow between processes at different layers, vertical flow. The TMN framework applies to the infrastructure of a particular service provider, though there may be many companies *contributing* at different layers of the pyramid for any particular service provider.

In addition to being based on TMN, the TOM is also based on a business reference model which provides a framework for interaction between the business of a service provider on one the hand and the businesses that do not fall directly in the TMN pyramid on the other, such as vendors of third-party software applications. This business reference model is shown in Figure 8.3.

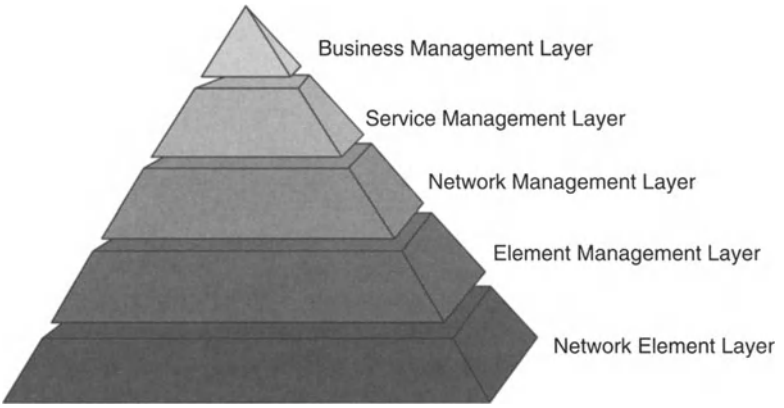


Figure 8.2 Telecommunications Management Network pyramid.

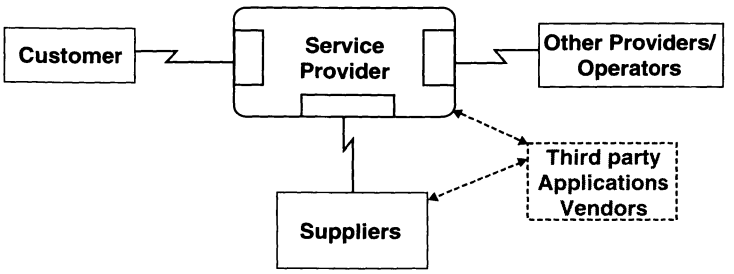


Figure 8.3 TOM business reference model.

The role of the TMN framework in the TOM processes is more explicit than the business reference model in that the layers of the TOM (see Figure 8.1) correspond to the layers of the TMN. Indeed, the basic pyramidal structure of TMN can immediately be seen in the TOM. We give a little more background for the business reference model since its role is not so explicit in the TOM.

The interfaces to customers, suppliers, third-party application vendors and other service providers are external to the service provider in question. It is usually an aim of effective OSS deployment that these interfaces be integrated into the delivery and support processes of the service provider. In order to achieve this integration, which largely means nothing more than effective management of the value chain, standardised processes are required to support management capabilities. As far as possible, these interfaces should be implemented in software, but this is not always possible or practical. In those cases, the substitute is rigorous procedures for staff to follow, with appropriate leeway for decision-making and for handling unforeseeable events.

A central purpose of the TMF, and certainly of the TOM, is to promote the standardization of management processes across the telecom industry. If standards are followed in developing OSS components and in developing interfaces, then certain information can be leveraged across the enterprise of the service provider. Indeed, relevant information can be leveraged across the entire value chain to include customers, suppliers, other providers, partners of the service provider in question and the third-party application vendors. For example, suppose the output of an alarm correlation OSS component is in a standard format. Then that information could electronically be made available at different places where it might be valuable:

- Field technicians for locating and repairing the fault.
- Customer service staff, who may have to answer complaints about a sudden service fault.
- Suppliers who may have to provide materials or equipment for repair of the fault or replacement of the relevant piece of infrastructure.
- Developers of third-party applications, which take correlated alarm information and process it in different ways for advanced alarm monitoring, SLA management or other purposes.
- Enterprise customers who wish to have real-time information on the network for whose capacity they have paid.
- The services offered by a partner service provider may be impacted by a network fault and so they might benefit from access to the alarm information.
- Partner service providers with whom there is an agreement in place such that a network fault automatically activates capacity in the partner's network on a temporary basis.

Unless information is generated in a standardised format and, perhaps more importantly, interfaces are standardised, most of the above information sharing would not be possible. Rather, information will have to be absorbed by staff, telephone calls will have to be made, deals done, notes taken on pieces of paper and so on. Eliminating this kind of inefficiency is a key goal of the TMF. An important first step to achieving this is standardizing — or at least achieving transparency in — processes of service providers.

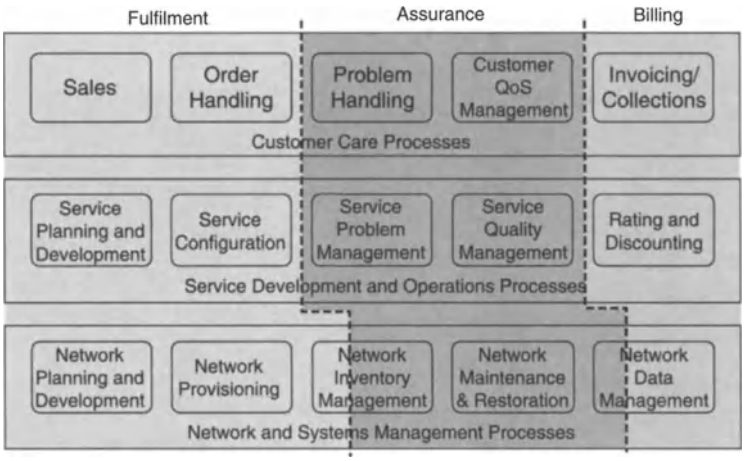


Figure 8.4 Fulfilment, Assurance and Billing (FAB) end-to-end process breakdown.

8.4 End-to-end Process Flow

The objective of any service provider is to automate its processes to deliver value to its customers. Therefore, design and definition of the end-to-end process flows for providing services to customers are important. The TOM breaks these process flows into three basic end-to-end processes that are common to any service-oriented business:

1. Service fulfilment is the timely and correct provisioning of what the customer ordered.
2. Service assurance is maintaining the service, requiring timely response and resolution of customer or network triggered problems, tracking, reporting, managing and taking action to improve performance for all aspects of a service.
3. Service billing is providing timely and accurate bills and providing billing query support that allows adjustment handling and payment collections.

The three customer-focused activities of FAB cut across the TOM chart, and this is shown in Figure 8.4. Figure 8.4 shows the predominant processes which need to be integrated and automated to support each of fulfilment, assurance and billing as end-to-end processes.

The network inventory management and network data management boxes have each been split across two different end-to-end processes, fulfilment and assurance, and assurance and billing. Hence there is not merely either an input or output function played by either network inventory management or network data management. For example, network inventory management includes physical implementation activities in the network, and is important in both the fulfilment and assurance processes for final installation of a service and for the restoration and repair of a service.

We have been talking about FAB as three process *flows*. Figure 8.5 gives an idea of how the FAB process breaks down into dynamic end-to-end process flows. The

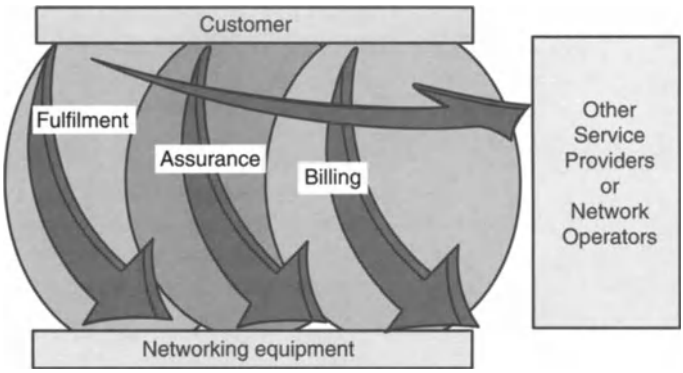


Figure 8.5 FAB end-to-end and flow-through process flows.

essential flows are:

- between the customer interface and support in a network element
- from sale through billing
- Between other providers and network operators.

The curved vertical arrows represent the process interactions between the customer interface and the network elements, i.e. process flow-through. The overlapping balloons show that fulfilment, assurance and billing include specific processes from the framework. However, all three end-to-end processes have interfaces among many processes across the map. The direction of the three arrows show end-to-end flow. As Figure 8.5 suggests, it is mostly the customer who initiates a fulfilment process. However, the assurance process can also be triggered by the customer or by network elements. Billing predominantly starts with data collection in the network and proceeds through to bills being presented to the customer. The curved horizontal arrow shows process flow interfaces required with other providers and network operators. None of the three aspects of process flow can be ignored in achieving effective integration and automation.

8.5 Telecom Operations Map in 3-D

Certain activities and sub-processes either:

- are directly required for business management processes or
- are functional sub-processes that must support or be supported by the fulfilment, assurance and billing processes.

Business management sub-processes and functional sub-processes, or sets of process activities, can be represented in a three-dimensional overlay diagram. Overlay diagrams show that the TOM framework processes support these business sub-processes. An example is shown Figure 8.6 for the management of SLAs. Figure 8.6 should be read in conjunction with Figures 8.4 and 8.5.

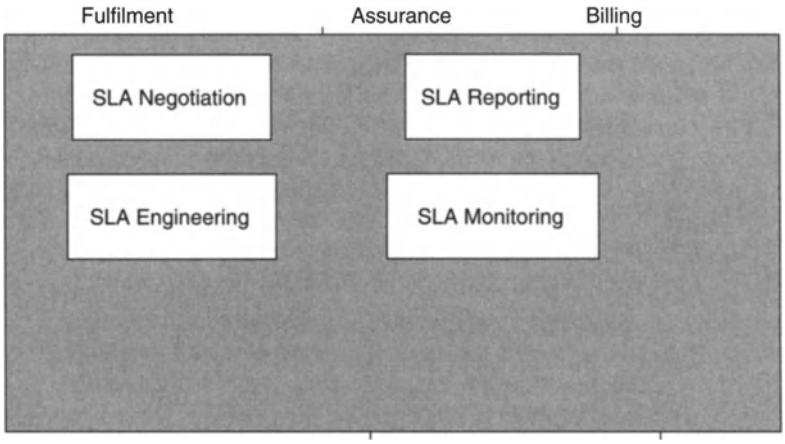


Figure 8.6 TOM overlay for mapping SLA management.

The service provider must meet market demand for service quality and procedures for assuring contracted service quality. The SLA is an agreement between the service provider and the customer which specifies the contracted service quality levels. In managing an SLA, the service provider must consider a range of its operational sub-processes, including identifying and defining SLA parameters, release management and configuring operational processes and systems to support and report on the life-cycle management of the SLA.

The SLA is part of the service offering and is therefore developed in the service development process, as shown by the SLA Engineering overlay box. In some cases, a customer sale negotiation may result in a new SLA requirement. In such a case, the service development process is used to establish the new requirement, if appropriate for the business.

By overlaying Figure 8.6 on Figure 8.4, the SLA Engineering box indicates that the service configuration process may need to be involved in support of pending SLA offers. Once the specific SLA offer is available, the fulfilment process triggers the assurance process to manage SLA performance through monitoring and reporting. An SLA may include measures for performance, pricing, installation intervals, billing timeliness, time to repair, call handling and other metrics or requirements.

This overlay approach can also be used to look at functional sub-processes or capabilities required across the map. Test management is an example of a functional sub-process that must support several framework processes, for example, service configuration during fulfilment, problem handling in assurance, rating and discounting as part of billing, and network maintenance and restoration as part of assurance or billing. Support for test management capability is managed either within the specific process or as a competency/capability function available to any process.

Common processes across services and technologies are essential to:

- deliver to process cost targets and process performance improvement
- provide a consistent customer service experience

- enable cost-effective management of people that ensures a fulfilling work life for staff
- enable cost-effective process automation.

Service providers can differentiate their services from those of competitors by adopting a unique approach to process activities and the information they use to support any specific service. Although the TOM at the framework level is technology- and service-independent, it is intended to be applied to real-life services and technologies. The processes, flows and basic information will be consistent across two different technologies or services, but the detailed information will have differences. Certain activities or sub-processes may be required for one service or technology and not another. A service provider's business rules may dictate a different sequence of activities owing to the nature of specific service or customer requirements.

New services are essential for growth in demand for telecom services. This has been borne out by the regular introduction of new services and technologies in the telecom and data services industries over the last few years. New services often have new process requirements or at least require a new perspective on existing process requirements. In mobile services:

- New process requirements have been introduced by offering mobile customers the ability to "roam", where customers of a "home" service provider may use the infrastructure of another service provider.
- Fraud management has been extended to address the new requirements of roaming.

To manage roaming, the home mobile service provider and the serving service provider enter a roaming agreement. This is done directly or through a clearing house. Development and implementation of the roaming agreement come under the Service Planning and Development Processes of the TOM, and include interfaces across all layers of the map. Processes to detect and prevent fraud are common to most networks. Mobility and roaming add to the complexity of detecting and preventing fraud.

8.6 Business Relationships with Suppliers/Partners

Joint service arrangements have arisen in support of interconnection between service providers, referred to above as "partnering". This arose from the growth seen in the 1990s in partnerships, alliances, mergers, acquisitions, new market entrants and competitive access providers. To provide a service, one service provider must interface with one or more other service providers or network operators. When providing a joint service or subcontracting services, service providers may interface with different functions at multiple layers of the TOM (see Figure 8.1). The interface points vary depending on:

- the degree of process automation of the service providers involved
- the service or technology being supported
- whether a service provider is acting as a wholesaler or retailer

- whether a service provider is using their own network infrastructure or is merely acting in support of a specific customer service instance.

With the advent of e-business, a service provider's processes must increasingly be viewed as part of an overall value chain. Hence the interfaces with suppliers, other providers and network operators are important. Since many processes that were once entirely internal to the service provider may now have external aspects, it is critical to understand the relationships with other service providers of a service provider's internal processes.

8.7 Service Provider TOM Application

The service and network management structure and the process design of a service provider depend on their corporate mission, target markets and strategies. The basic operational functions identified within the TOM are process blocks that can be applied to various service providers' infrastructures and organizations.

The specific process boundaries shown in the model do *not* correspond to organizational boundaries. In one service provider, a single work group may have responsibility for taking orders and for receiving notification of problems from a large customer. Another work group might be responsible for tracking the progress of a work order or trouble ticket and reporting on its completion. On the other hand, customer contact might have been outsourced to another company with its group that conducts all customer contact, giving input and instructions to all other processes. In each case, the basic function being executed is similar.

A service provider must incorporate critical aspects from the views discussed above and effectively translate business mission, goals and strategies into business process management architecture along with an organizational structure. An integrated and automated systems/application and data architecture, and an operational environment that executes these processes with high quality, must support the process architecture.

The TOM describes the business process framework and identified aspects that support business process management. The TMF recommends that each service provider evaluate the TOM processes and identify how they perform each process internally, including the business rules and policies to which each business process applies.

8.8 Example: Network-detected Fault/QoS Problem

Figure 8.7 shows a possible sequence of activities in response to a *network-detected* problem. The problem may not affect services as a result of "self-healing" capabilities in the underlying network and information technology infrastructure. For example, SONET/SDH networks have some instant redirection capabilities. The service provider's policy could be to decide on how to repair the problem at the network layer and, subject to "no break in service", may not even inform the service layer of the event.

The figure shows two ways in which a potential service-affecting problem may be identified: first, by raising an alarm and, second, by a synthesis of network data

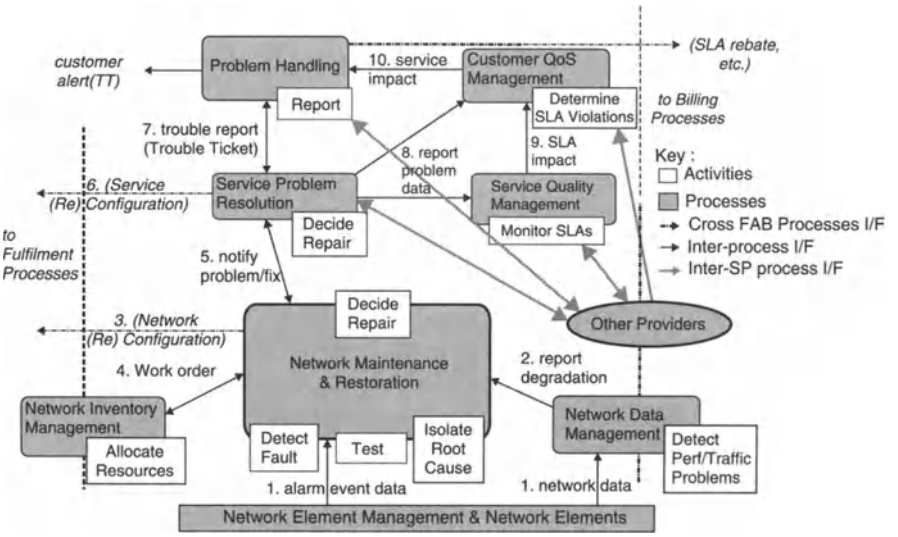


Figure 8.7 Segment of assurance process flow.

using network data collection and correlation. Neither method is exclusive, because alarm data may be used for performance reporting. In addition, network data management collects and processes both performance and traffic data, in addition to usage data. The usage data are used in the billing process. This illustrates overlap between a number of different processes.

Most service providers are driving their service assurance processes to become mostly proactive, meaning triggered by automation rather than triggered by the customer. Until recently, customer care processes have been reactive. However, the pressure to reduce cost, customer demand for more control, and customer demand for being proactive, customer-enabled service support are driving a major shift to interactive support through automation. With the advent of Internet access, interactive customer-enabled service includes giving the customer visibility into and control over service performance and also performance of the network on which they are reliant. Service providers who are highly competent in customer service have recognised that there is value in collecting information from the customer at each customer interface point. The information so captured can be used in various ways across the enterprise, such as to customise special offers and to relate better to the customer in future interactions.

If a service is provided jointly with other providers or operators, then the primary service provider must have interfaces with the other service providers or network operators to monitor and support the service.

8.9 Conclusion

The TOM has provided a reference framework for thinking about telecom operations in toto. Many OSS vendors find the TOM invaluable in navigating telecom

operations. Thinking about OSS involves thinking about telecommunications at an abstracted, or "meta", level. Although it is sometimes necessary to think in concrete terms of switches and cables, this can sometimes cloud the real issues involved in operational management of a carrier.

An analogy is that in thinking about inventory management for a frozen food supplier, it is often superfluous to think in terms of peas or beans. One might be better served by abstracting to frozen produce with a certain lifespan which is carried in packages with certain properties of weight and density, and so on. In other words, one must abstract upwards to what is relevant, and that is what the TOM does for telecom operations.

9 Telemanagement Forum: eTOM

9.1 Introduction

The Extended Telecom Operations Map™ (eTOM) is a business process framework which places the service provider's enterprise within its overall business context. In addition to the operating processes involved in delivering telecom services, the wider enterprise of the service provider and its business interactions along with its relationships to other organizations are represented. Moreover, the eTOM describes the processes and interfaces that are required to insert the service provider's business into an e-business or e-commerce setting.

The Telecom Operations Map (TOM) sought to provide a general framework for the automation of the operations side of carrying on the business of a telecom service provider. Of course, this included the interfaces between the various components, all of which were internal to the enterprise concerned. In an e-business setting, automation goes beyond operations to include many more aspects of the service provider's enterprise. This requires systematic process descriptions for all aspects of the service provider's enterprise, the definition of interfaces between those components and the definition of interfaces between those components and the TOM. In an e-business setting, automation extends to the connections between customers and suppliers. Thus, eTOM provides a generic framework for interfaces at the edge of enterprises that hook up inter-enterprise processes.

The eTOM can be viewed as having three major classes of process (see Figure 9.1):

1. strategy, infrastructure and product which cover planning and lifecycle management
2. operations, which covers the core of operational management
3. enterprise management, which covers corporate or business support management.

We now introduce the eTOM business process framework, and explain its structure and the significance of each process area. We also show how the high-level eTOM structure is decomposed to lower level processes. Because of the level of abstraction of the eTOM, much of this chapter may seem arcane. However, the abstract approach of the eTOM is its strength because it allows the eTOM to be applied by most any service provider in structuring its operations. Further, the eTOM seeks to support the e-business model while e-business is a relatively new idea and is itself rather abstract.

K. Misra et al. (eds.), *OSS for Telecom Networks*

The conceptual structure is an overall context that differentiates strategy and lifecycle processes from operational processes in two large process areas, represented by the two largest boxes in the figure. Beneath these are the third large area of enterprise management processes, in grey. The structure differentiates functional areas in four horizontal layers across the two upper process areas. Figure 9.1 also shows the internal and external entities that interact with the enterprise.

Operations are at the heart of eTOM. This area includes processes that support customer operations and management, and processes that allow direct interactions with the customer. These include day-to-day operations support and readiness processes. The eTOM view of operations includes sales management and supplier/partner relationship management.

Strategy, infrastructure and product include processes that develop strategy, commit to the enterprise, build infrastructure, develop and manage products and develop and manage the supply chain. In eTOM, infrastructure includes that which is required to support functional processes such as CRM. These processes direct and enable the operations processes.

Enterprise management includes the basic business processes that are required to run any business, with a focus on the enterprise level. Enterprise management processes have interfaces with almost every other process in the enterprise and are

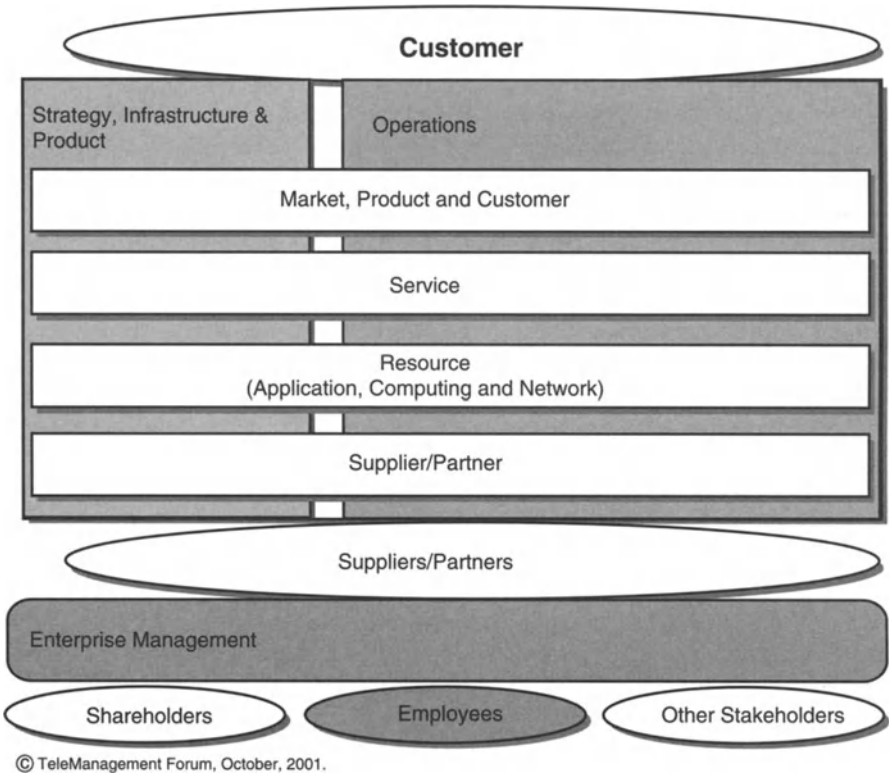


Figure 9.1 eTOM conceptual structure.

sometimes called "corporate" processes, such as with financial management and human resources management.

In addition to the major process areas, the conceptual structure of the eTOM addresses the supporting functional process areas, the horizontal layers. The supporting functional areas reflects the major expertise and focus needed to pursue the business:

1. Market, product and customer processes include those dealing with sales and channel management, marketing management, product and offer management, CRM and ordering, problem handling, SLA management and billing.
2. Service processes deal with service development and configuration, service problem management, quality analysis and rating.
3. Resource processes include those dealing with development and management of the enterprise's infrastructure, whether related to products and services or to supporting the enterprise itself.
4. Supplier/partner processes, predictably, include those dealing with the enterprise's interaction with its suppliers and partners. This involves processes that *manage* the supply chain and processes that *support* the operations interface with suppliers and partners.

Figure 9.1 also shows the major entities with which the enterprise interacts: customers, suppliers, partners, employees, shareholders and other stakeholders.

9.2 Top-level View of the eTOM

Below the conceptual level, the eTOM comprises process groupings which provide a first level of detail at which the entire enterprise can be viewed. These processes are considered to be at the "CEO level".

The eTOM is defined generically to ensure independence from organization, technology and service. To reflect the way businesses look at their processes, the eTOM supports two perspectives on the grouping of process elements:

1. *Vertical* process groupings represent end-to-end processes within the business, such as those involved in billing flows to customers.
2. *Horizontal* process groupings represent functionally related processes within the business, such as those involved in managing the supply chain.

The operations and strategy, infrastructure and product process areas include this two-dimensional structure.

The integration of all these processes provides the enterprise-level process framework for a service provider. This is the Level 0 view of the enterprise with the vertical and horizontal process groupings shown in Figure 9.1. Next, Level 1 gives process detail that refines the Level 0 view and Figure 9.2 shows how Levels 0 and 1 relate to one another. Process decomposition proceeds with each level being refined into a set of constituent processes at the next level.

The Level 0 enterprise view decomposes into seven vertical or end-to-end Level 1 process groupings and also eight horizontal or functional Level 1 process groupings in four layers. These vertical and horizontal process groupings represent alternative

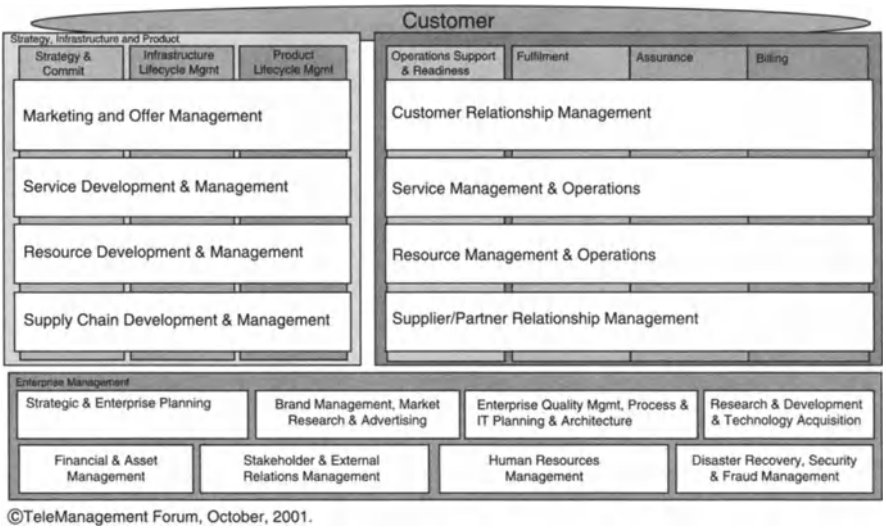


Figure 9.2 eTOM Level 0 view of Level 1 process groupings.

views on how processes can be associated. These alternatives have been selected to yield a common view of the Level 2 processes and so do not represent a divergence in the model. In addition, there are eight additional enabling and support Level 1 process groupings within enterprise management.

9.3 eTOM Operations Processes

Our discussion of eTOM operations processes will be divided into the vertical and horizontal groupings.

9.3.1 OPS Vertical Process Groupings

The Operations (OPS) process area contains the direct operations vertical process groupings of fulfilment, assurance and billing, together with the operations support and readiness process grouping (see Figure 9.3) The FAB processes are sometimes referred to as customer operations processes.

The TOM focuses on the customer processes directly represented by FAB. However, FAB processes are not explicitly included in the TOM framework map although they could be shown as an overlay. In an e-business world, the enterprise must focus on enabling and supporting the FAB processes. Hence the eTOM integrates FAB into the overall framework. In this chapter, we shall elucidate the components of FAB, which are:

1. fulfilment
2. assurance
3. billing
4. operations.

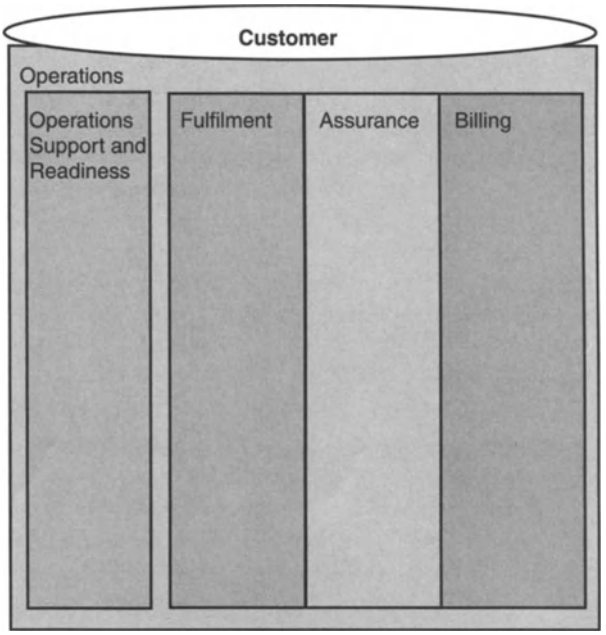


Figure 9.3 eTOM OPS vertical process groupings.

Fulfilment is responsible for providing customers with their requested products. It translates the customer's business or personal need into a solution which can be delivered using the specific products in the portfolio of the enterprise. This process informs the customers of the status of their purchase order and ensures timely delivery.

Assurance is responsible for the execution of proactive and reactive maintenance activities to ensure that services provided to customers are continuously available and adhere to applicable SLAs. Assurance processes perform continuous resource status and performance monitoring to detect proactively possible failures. These processes collect and analyse performance data to identify potential problems and resolve them without impacting the customer. Assurance processes manage the SLAs and report service performance to the customer. Assurance processes absorb trouble reports from customers, keep customers informed of the status of reported problems and ensure restoration and repair.

Billing is responsible for the production of timely and accurate bills, for providing pre-bill use information and billing to customers, for processing their payments and for performing payment collections. It also handles customer inquiries about bills, provides billing inquiry status and is responsible for resolving billing problems to the customer's satisfaction. This process also supports prepayment for services.

In addition to these FAB process groupings, the OPS process area of the eTOM includes the fourth vertical process grouping of operations support and readiness – see Figure 9.3. This grouping supports the FAB processes. In general, the operations support and readiness processes are concerned with activities that are "more"

asynchronous than those in FAB. These processes are less concerned with individual customers and services while being more concerned with groups. They reflect a need in some enterprises to divide their processes between customer-facing operations and real-time FAB operations. In addition, other operations processes which act as a "second line" in carrying out the operational tasks. Not all enterprises will choose to employ this split or to position the division in the same place.

In applying the eTOM in particular scenarios, the operations support and readiness processes may be merged with the FAB processes for day-to-day operation. The separation remains acknowledged to reflect a real-world division that is present or emerging in many enterprises. Operations support and readiness processes can be important for e-business opportunities and for customer self-management. The benefits of some aspects of customer self-management are discussed in Chapter 6.

9.3.2 OPS Horizontal Process Groupings

The OPS process area of the eTOM has four functional process groupings that support the operations processes discussed above and manage operations to support customer, service, resource and supplier/partner interactions (see Figure 9.4). The TOM uses the ITU-T TMN logical business, service and network layers to organise

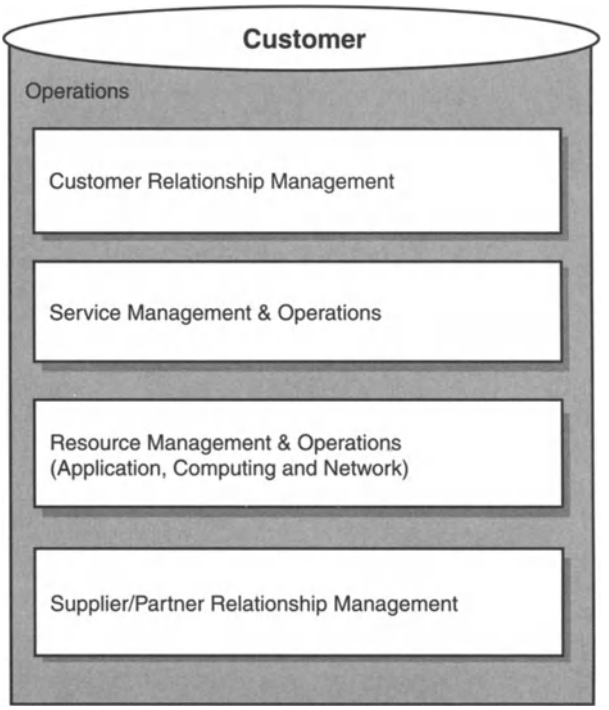


Figure 9.4 eTOM OPS functional process groupings.

the core business processes. This facilitated mapping the management functions defined in TMN to the TOM processes. Since eTOM is an evolution of TOM, and since the TMN layering approach does not cease to be relevant simply because we have placed the service provider in an e-business context, the TMN logical layers are loosely coupled to the functional process groupings of eTOM.

The Customer Relationship Management (CRM) process grouping considers the basic knowledge of customer needs and includes functionalities needed to acquire, maintain and enhance a relationship with a customer. CRM is about customer service and support, covering storefront, telephone, the web or field service. It is also about minimising churn, cross-selling, up-selling and direct marketing. Finally, CRM includes the collection of customer information and using that information to personalise, customise and integrate the delivery of service to a customer. That information is also used to identify opportunities for increasing the value of a customer.

CRM applies to retail and wholesale customer interactions. An example of the latter is where an infrastructure owner wholesales to a "virtual" service provider, such as Virgin mobile services, which in turn markets and sells directly to end-consumers.

The introduction of CRM is a key feature of the evolution from TOM to eTOM. At the highest and most general level, TOM included two process groupings to manage relations with customers: customer interface management and customer care. In TOM, it is explicitly mentioned that customer interface management may be:

- a distinct process within customer care or
- performed as part of the lower level customer care processes.

eTOM advances the TOM in several ways:

1. It expands customer care to CRM, which has broader concerns than customer care or customer interface management. CRM is the integration of customer acquisition, enhancement and retention through managing the customer relationship over time. For eTOM, CRM also represents the integration of sales and service processes and ensures a consistent customer interface across all CRM functional processes.
2. eTOM integrates customer interface management for FAB across all CRM functional processes and with customer processes. Customer interface management covers all kinds of contact with the customer, such as telephone, e-mail and personal interaction. It aims at integrating and coordinating these different interface types to provide a consistent interface and highlights the requirement for customer process control and customer self-management. eTOM also encourages the design of solutions so that systems interface used within the enterprise are the same as those used by customers. The eTOM CRM processes include an expansion of TOM customer care processes to:
   (a) focus on customer retention
   (b) improve enterprise process to include exceptional customer response
   (c) integrate marketing fulfilment execution
   (d) represent the billing function at the customer level and revenue assurance.

Service management and operations focus on knowledge of services, including matters such as access, connectivity and content. It also includes the functionality needed to manage and to run the operations, communications and information services. The focus is on service delivery and management as opposed to the management of the underlying network and information technology. Some of the functions involve short-term service capacity planning, the application of a service design to specific customers or managing service improvement initiatives. These functions are closely connected with the day-to-day customer experience.

These processes are accountable to the business management layer function of product management, the profit and loss accountability, to meet a minimum target set for service quality, including process performance and customer satisfaction at a service level, in addition to service cost.

eTOM differentiates day-to-day operations and support from planning and development, and other strategy and product lifecycle processes. In the TOM, these service layer processes either were not differentiated or were not addressed at all. The eTOM structure better depicts the structure of an enterprise, especially suited for an e-business environment.

Resource Management and Operations (RMO) maintains knowledge of resources such as application, computing and network infrastructures. It is also responsible for managing all these resources such as networks, IT systems, servers and routers utilised to deliver and support services required by or proposed to customers. RMO includes functionalities responsible for the direct management of all such resources, such as network elements, computers and servers utilised within the enterprise. These processes are responsible for ensuring that the network and information technologies infrastructure supports the end-to-end delivery of the required services. The job of these processes is to ensure that infrastructure runs smoothly, is accessible to services and employees, is well maintained and is responsive to the needs of the services, the customers and the employees. For example, faults in desktop hardware should be addressed promptly and the internal LAN should be reliable.

RMO has the basic function of assembling information about resources, that is, from network elements and/or element management systems. From there, RMO integrates, correlates and summarises data in order to pass them on in an appropriate form to service management systems or to take some automatic action in an appropriate network resource.

In the TOM, the "network and systems management" processes were included at the highest, most general level. This is not adequate in an e-business setting, as application and computing management are as important as network management. Moreover, network, computing and applications resources must increasingly be managed in a joint and integrated fashion. Thus, eTOM introduces the resource management and operations process grouping, along with the corresponding resource development and management grouping within the strategy, infrastructure and product management (SIP) processes. This provides integrated management across the applications, computing and network sets of resources. These areas also combine the network element management processes of the TOM, since these processes are inherent in any resource management process, as opposed to a

separate process layer. The RMO processes thus manage the complete service provider network, sub-network and information technology architectures.

eTOM differentiates day-to-day operations and support from planning and development and other strategy and lifecycle processes. In the TOM, these resource layer processes were not differentiated or were not addressed.

Supplier/Partner Relationship Management (SPRM) supports the core operational processes, both the customer instance processes of FAB and the functional operations processes. SPRM processes align closely with a supplier or partner's CRM processes. The inclusion of SPRM in eTOM is one of the key ways in which eTOM differentiates itself from the vertically integrated enterprise framework of the TOM. SPRM processes define a direct interface with the appropriate lifecycle, end-to-end customer operations or functional processes with suppliers and partners. The processes include issuing RFPs as part of the buy process, issuing purchase orders and tracking them through to delivering, handling problems, validating billing and authorizing payment, and also quality management of suppliers and partners. When the enterprise sells its products to a partner or supplier, it does so via the enterprise CRM processes, which act on behalf of the supplier/enterprise.

The TOM addressed other providers merely by showing "Other Providers" as providing inputs or receiving outputs.

9.4 eTOM Strategy Infrastructure and Product Processes

Referring back to Figure 9.2, we will first discuss vertical groupings of the SIP processes and then the horizontal groupings.

9.4.1 SIP Vertical Process Groupings

The strategy and commit processes along with the two-lifecycle management process groupings, are shown as three vertical end-to-end process groupings (see Figure 9.5). The strategy and commit processes provide the focus within the enterprise for generating specific business strategy and gaining staff commitment within the business for the new strategy. The infrastructure lifecycle management and product lifecycle management processes drive and support the provision of products to customers. Their focus is on meeting customer expectations whether through

- product offerings
- the infrastructure that supports the operations functions and products
- the suppliers and partners involved in the enterprise's offering to customers.

In the SIP process area of the eTOM framework, there are three SIP vertical process groupings:

1. The Strategy and Commit process encompasses the generation of strategies to support the infrastructure and product lifecycle processes. It also encompasses business commitment within the enterprise to support these strategies. This embraces all levels of operation from market, customer and products, through

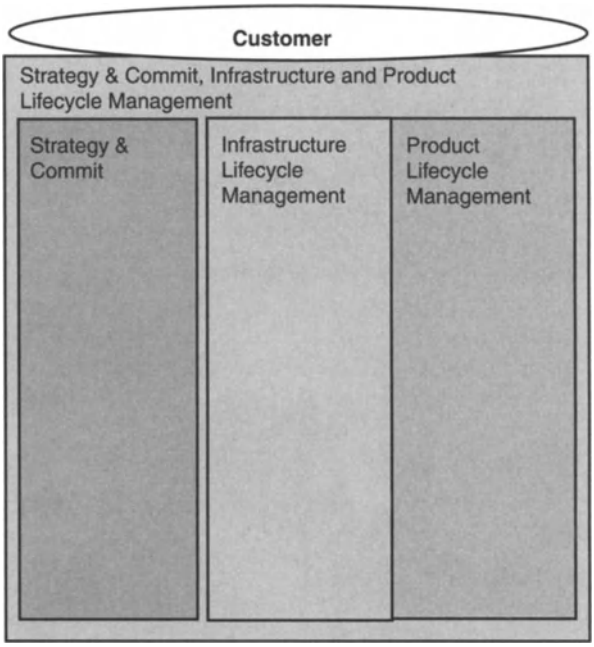


Figure 9.5 eTOM SIP vertical process groupings.

the services and the resources on which these depend, to the involvement of suppliers and partners in meeting these needs. Strategy and commit processes focus on analysis and commitment management. These processes provide the focus within the enterprise for generating specific business strategy and gaining buy-in within the business to implement this strategy. Strategy and commit processes also track the success of strategies and make adjustments as required.

2. Product Lifecycle Management processes drive and enable core operations and customer processes to meet market demand and customer expectations. Performance of lifecycle processes is viewed at the highest levels of the enterprise, owing to their impact on customer retention and competitiveness. There are two end-to-end lifecycle management processes in the eTOM: infrastructure and product. Both have a development and deployment nature, in that both are concerned with developing and/or introducing new infrastructure or new product. Product lifecycle management deals with introducing new products, in the form of services delivered to customers, and assessing and taking action on product performance.
3. Infrastructure lifecycle management deals with development and deployment of new infrastructure, assessing performance of the infrastructure, and taking action to meet performance commitments.

The eTOM decouples the lifecycle management processes from day-to-day operations processes represented by the operations processes, operations support and readiness, FAB. The TOM integrated some of these processes in the core operations framework and this sometimes resulted in some confusion and lack of understanding.

Lifecycle management processes have different business cycle times and different types of enterprise-level objective and are closer to being *business* processes as against *operations* processes. Mixing these processes with the customer priority processes diminishes focus on the lifecycle management processes and diminishes focus on enabling and supporting customer operations processes.

Processes in an e-business environment must all look to how they are enabling and supporting interaction with the customer. In addition, lifecycle management processes must be designed to meet cycle time and other performance characteristics that are critical to the success of the enterprise, for example, time to market of new products and infrastructure unit cost. Another key characteristic of the lifecycle management processes is that they interact with almost every other process of the enterprise. Lifecycle management processes also interact with each other. Product lifecycle management processes drives the majority of the direction for the infrastructure lifecycle management processes either directly or indirectly. These processes prepare the customer and functional operations processes to support customer interaction for products, providing the infrastructure for the products to be delivered and providing the supplier and partner interface structure with the enterprise. To enable and support customer and functional operations, these processes must often be synchronised for timely and quality delivery.

The infrastructure lifecycle management process grouping is responsible for defining, planning and implementing application, computing and network infrastructures, and also other support infrastructures and business capabilities such as operations centres and architectures. This applies in connection with the resource layer or any other functional layer, such as CRM voice response units that are required to provide information and communications products to the customer and to support the business. These processes identify new requirements and new capabilities and design and develop new or enhanced infrastructure to support products. Infrastructure lifecycle management processes respond to needs of the product lifecycle management processes including unit cost reductions, product quality improvements and new products.

The product lifecycle management process grouping is responsible for the definition, planning, design and implementation of all products in the enterprise's portfolio. The product lifecycle management processes:

- manage product development to target profit and loss margins, customer satisfaction and quality commitments
- manage the delivery of new products to market.

In order to design and manage products which will succeed in their specific markets, the lifecycle processes must take into account:

- the market across all key functional areas
- the business environment
- customer requirements
- competitive offerings.

Product management processes and the product development process are distinct process types. Product development is characterised as a predominantly

project-oriented process that develops and delivers new products to customers. It also develops new features and enhancements for existing products and services.

9.4.2 SIP Horizontal Process Groupings

Corresponding to the operations functional process groupings, the strategy infrastructure and product domain also has four functional process groupings (see Figure 9.6). These support the SIP processes described above and the management of operations to support marketing and the offer, service, resource and supply chain interactions.

In the SIP process area of the eTOM framework, there are four SIP functional process groupings that support the SIP management processes (Figure 9.6).

The marketing and offer management process grouping focuses on the knowledge of running and developing the core business of an information and communications service provider. This process grouping includes functionalities necessary for defining strategies, developing new products, managing existing products and implementing marketing and offering strategies that are especially suitable for information and communications products and services.

Marketing and offer management are well-known business processes, especially in more technology-oriented companies, where the rate of innovation and brand recognition determine success. Although most companies deploy all these functions, depending on the size of the company they are combined in different ways. Marketing and offer management are enabling processes, but are also the key

Figure 9.6 eTOM SIP functional process groupings.

processes accountable for commitment to the enterprise for revenue, overall product performance and profit and loss. Marketing and offer management deal with product, markets and channels. They also manage market and product strategies such as pricing, sales, channels, new product development and retirement and marketing communications.

The service development and management process grouping focuses on planning, developing and delivering services to the operations domain. It includes functionalities necessary for defining the strategies for service creation and design, managing and assessing the performance of existing services and ensuring that capabilities are in place to meet future service demand.

The resource development and management process grouping focuses on planning, developing and delivering to the operations domain the resources that are needed to support services and products. It includes functionalities necessary for defining the strategies for the development of the network and other physical and non-physical resources, introduction of new technologies and interworking with existing ones, managing and assessing the performance of existing resources and ensuring that capabilities are in place to meet future service needs.

Supply chain development and management focus on the interactions required by the enterprise with suppliers and partners, who are involved in maintaining the supply chain. The supply chain is a complex network of relationships that a service provider manages so that it may source and deliver products. In the e-business world, almost by definition, companies increasingly work with suppliers and partners in order to broaden the products they offer and improve their productivity.[31] Supply chain development and management processes ensure that the best suppliers and partners are chosen as part of the enterprise supply chain. They help support sourcing decisions made by the enterprise and ensure that the capabilities are in place for interaction between the enterprise and its suppliers and partners. They ensure that the contribution of suppliers and partners to the supply chain is timely and delivers the required support, and that their overall performance and contribution are as good as or better than those of vertically integrated enterprises. These processes include establishing and maintaining all the information flows and financial flows between the provider and supplier.

9.5 eTOM Enterprise Management Processes

Enterprise Management involves the knowledge of enterprise-level actions and needs and encompasses all business management processes that are necessary to support the rest of the enterprise. These processes are necessary in any business because they are needed to run the business at the enterprise level, to direct the business and to support the direct and indirect customer processes. Enterprise management processes include corporate-level processes such as those for financial, legal, regulatory and public relations management. This area sets corporate strategies and

[31] Terms that have been used to describe these relationships include synergistic clusters, coalitions and business ecosystems.

directions and provides guidelines and targets for the rest of the business. These are sometimes considered as the "corporate" functions and/or processes. Enterprise management also includes strategic planning for the enterprise in addition to information systems strategy development and management. Enterprise management processes in general do not have a customised aspect for information and communications service providers.

The enterprise management process groupings are:

- strategic and enterprise planning
- brand management, market research and advertising
- financial and asset management
- human resources management
- stakeholder and external relations management
- research and development, and technology acquisition
- enterprise quality management, design of business processes, and planning and architecture of IT systems
- disaster recovery, security management and fraud management.

A basic process modeling methodology was used to start the process modeling work of the eTOM. A top-down approach was adopted in the development phase of the eTOM. This permitted the definition of the business process framework at the enterprise level in a series of Level 1 process groupings. These Level 1 processes are split into vertical or end-to-end and horizontal or functional groupings, with the dependent Level 2 processes positioned within the vertical and the horizontal grouping appropriate to the process concerned. As described in the process methodology, eTOM uses hierarchical decomposition to structure the business processes.

Through hierarchical decomposition, complex entities can be structured and understood by means of the formalization of their components. Hierarchical decomposition enables detail to be defined in a structured way. Hierarchical decomposition also allows the framework to be adopted at some levels as appropriate but not at others.

For the eTOM, each process element has a detailed description that includes the purpose of the process, its basic inputs and outputs, its interfaces, high-level information requirements and business rules. The eTOM process model also depicts process flow in a way that drives end-to-end process and process flow-through between the customer and the supporting services, resources and suppliers/partners.

Based on this process modeling approach, the eTOM process work starts at the Enterprise level, called Level 0, and shows the Level 1 processes (see Figure 9.2). Each Level 1 process is decomposed into its Level 2 component processes. Decomposition is continued until the detail level of the process elements has reached the stage where it is appropriate to define a process flow.

9.6 Conclusion

The eTOM is an enterprise process framework for service providers. The processes of the enterprise fall into four major categories with 12 enterprise level process groupings.

The main strengths of the eTOM framework are that:

- It is consistent with, and enhances, the well-established TOM.
- While addressing operations and maintenance, it also covers all significant enterprise process areas.
- It is e-business oriented, introducing concepts such as retention and loyalty, a new business relationship context model and supplier/partner relationship management.
- It not only covers network management, but also enlarges its scope to application and computing management and management integration while defining the relationships between these areas and network management.
- It decouples lifecycle management, including development processes, from operations and day-to-day processes.
- It can represent both the framework, static, and the process flow, dynamic, views of processes, including high-level information requirements and business rules for strong linkage to automation solutions.
- It provides a process framework that reflects the most current thinking in designing and documenting processes.

It is clear that the eTOM is of immediate benefit in the billing and customer care side of telecom operations, because clear value is derived from linking those areas to the marketing, sales and financial control systems of the wider organisation. At the lower, network management level, bandwidth swapping makes the connections with suppliers and partners very valuable by smoothing capacity utilization. The eTOM formalises this. The eTOM has only begun to seep into the thinking of network management OSS vendors, but it will inevitably grow in importance with automated interlinking of telecom carriers.

10 Network Inventory Management

10.1 Introduction

Telecom network inventory is slightly different from inventory in the "bricks-and-mortar" world. When one thinks about physical inventory, one generally considers that the inventory itself has intrinsic value that can be realised by its sale. For example, if a Parisian patisserie has 1000 croissants in inventory, then by selling those croissants their value is realised. A service provider does not realise the value in their network by selling the network itself. Rather, the service provider sells capacity in the network's traffic-carrying capability. Thus, physical network inventory needs to be managed primarily for maintenance and for usage accounting purposes. Usage accounting is necessary for billing. On the other hand, a bakery must manage its physical inventory to realise optimally the value in its capital, which includes raw materials, baking oven and other requirements. In short, the physical inventory in a telecom network is a capital good that is not offered for sale as part of the service provider's day-to-day business operations. Only the service-carrying capacity of the physical network is offered for sale in day-to-day business.

The role of a network infrastructure manager in network inventory will largely be to pull together the inventory information provided by a number of systems. That is, network equipment usually either comes with a network management system that is equipped with inventory management functionality for that network equipment, or the company that installed the network equipment will have purchased a management system that includes network inventory management functionality. The point will be that any given network will have network equipment from a multiplicity of vendors installed over a long time period. Hence there will be multiple NMS. Therefore, a network manager's difficulty will often not begin with inventory management per se. Rather, it will be in managing network inventory management systems!

Service providers who own and manage their own network infrastructure often use "swivel chair integration" which means that network managers literally sit on a swivel chair allowing them to switch quickly between the various inventory management systems that are being used for the network. The humans do the pulling together of the information. Centralised information stores may exist in the form of spreadsheets, word processing documents, notepads and so on. It is easy to see that this is inefficient and expensive, and ultimately an inferior way of carrying out

K. Misra et al. (eds.), *OSS for Telecom Networks*

inventory management. Many companies now sell software designed to integrate multiple network inventory management systems and to provide centralised network inventory management. The benefits of this are now outlined.

10.2 The Business Case for Inventory Management

This section is based on the approach taken by BoldTech to network inventory management. BoldTech is an OSS software vendor and service provider.

The provisioning process must be efficient in order for telecommunications service providers to be able to deliver service to their customers quickly and at low cost. Inventory management is central to the provisioning process. The ultimate aim is to have a provisioning environment where orders can flow from inception to activation on the network with minimal manual intervention, or "zero touch provisioning". There are several issues with many providers that prevent this from happening. Among these, inventory inaccuracy most increases the chance that a provisioning cycle will be unsuccessful. Since multiple workgroups may be involved in provisioning, either independently or interdependently, real-time accuracy of inventory is a worthwhile goal.

An ROI-based business case exists for network inventory for incumbent and next-generation service providers. This works by quantifying the costs and benefits associated with a network inventory system, including economic valuation. The economic valuation includes:

- payback
- cash flow analysis
- benefit-to-cost ratio
- net present value
- internal rate of return
- payback period
- sensitivity analysis.

The business case focuses on:

1. revenue recognition through acceleration of customer in-service dates through provisioning processes improvement
2. savings in operational expenses by increasing the productivity of provisioning personnel
3. capital expenditure offsets achieved by recovery of existing network asset inventory, and the correct and appropriate classification of the existing inventory state.

BoldTech issued a whitepaper entitled "Network Inventory: a Critical Piece for Solving the Elusive Flow-through Provisioning Problem" that was helpful in quantifying the cost–benefit trade-offs of implementing a network inventory solution.

While the business case presents an enterprise-based focus on costs and benefits, and identifies critical areas for improvement, more detail is needed to drill down to provide a network inventory solution. A network inventory solution should provide an integrated view of the entire network, bringing together inside and outside plant

for the physical and logical network. This consolidated approach provides:

- productivity gains in processes that are used to activate customer services
- reduced time for realization of service revenues
- better asset utilization through accurate representation of deployed capital assets.

An inventory solution will ideally integrate into the OSS enterprise using standard interfaces, providing open access to spatial, physical and logical models. This requires network and communications business knowledge and technical skills in order to deliver network inventory solutions that enable a carrier to:

- manage the increasing demand for capacity
- improve time-to-market for new services
- increase operational efficiencies.

An inventory OSS solution should allow the overall solution to be provided faster, by leveraging "best-in-breed" products from companies that have solved similar issues for other service providers.

10.3 Network Inventory Domain Object Models

A significant obstacle to an efficient and accurate provisioning process is the presence of disparate representations of inventory across organizational departments and OSS. This disparate representation requires both human resources and applications either to make inventory changes in several places or to work through translation rules to come to a representation of data that can easily be understood. This additional effort can lead to errors in representing the data correctly and will cause orders to fall out. This is called "redlining" in the provisioning process.

A Domain Object Model (DOM) provides a common representation of inventory across the enterprise. The DOM is necessary for avoiding order fall out and redline situations, because it allows the provisioning process to be completed rapidly and with fewer errors. Telecom consultants such as BoldTech have spent much time in building network inventory DOMs. These DOMs are the starting point for development efforts in which OSS integrate accurate inventory to add further automation to the provisioning process. Some DOM features include:

- support across all telecom technology layers fibre, SONET, ATM and IP
- historical, present and future views of inventory
- extensibility for adding new types of inventory and their attribution
- generic applicability regardless of how individual OSS represent inventory
- logical, physical and spatial schematic views.

10.4 Inventory Load and Reconciliation

Without a consistent and accurate view of network inventory between the network and OSS, the current state of the inventory and its percentage utilization cannot be

maintained accurately. The result is a higher capital expense on incremental network augments.

Integrated loading and reconciling of inventory data from network management systems can be done using common data objects. This approach limits the network-specific formats and communication protocols, and allows users and other applications to perform the following functions:

- inventory loading
- data retrieval
- reporting
- discrepancy checking
- reconciliation
- synchronization.

10.5 Inventory Service Architecture

The traditional method of using Enterprise Application Integration (EAI) tools to synchronise inventory between inherited silo OSS does not generally provide a complete solution.[32] Segmented inventory stores must be consolidated before a carrier is likely to see value from its inventory management system. The value is apparent though automation of the provisioning process, requiring less human intervention and, thus, faster cycle times and more immediate revenue recognition. Inventory is an enterprise-wide resource and must be logically centralised to be available in a consistent manner to every person and OSS within the organization.

A favoured approach to enterprise-wide inventory architecture is to use logically centralised services. Features of this inventory service architecture include:

- separation of functionality into distinct layers/tiers
- use of distributed object technologies to provide location, platform and programming language independence
- generic, flexible interfaces for retrieval and update of network inventory
- full extensibility through the use of metadata and business rules.

10.6 Network Inventory OSS Implementation and Integration

Integrated inventory enables carriers accurately to provision new services faster because there is always clear knowledge of the underlying network. With accurate network inventory, network operators are more likely to be able to determine:

- which service ports are available
- whether or not there is enough bandwidth
- if there is physical connectivity to the end-customer
- the planned capacity for a link.

[32] The term "inherited" has replaced the older, and now politically incorrect, term "legacy".

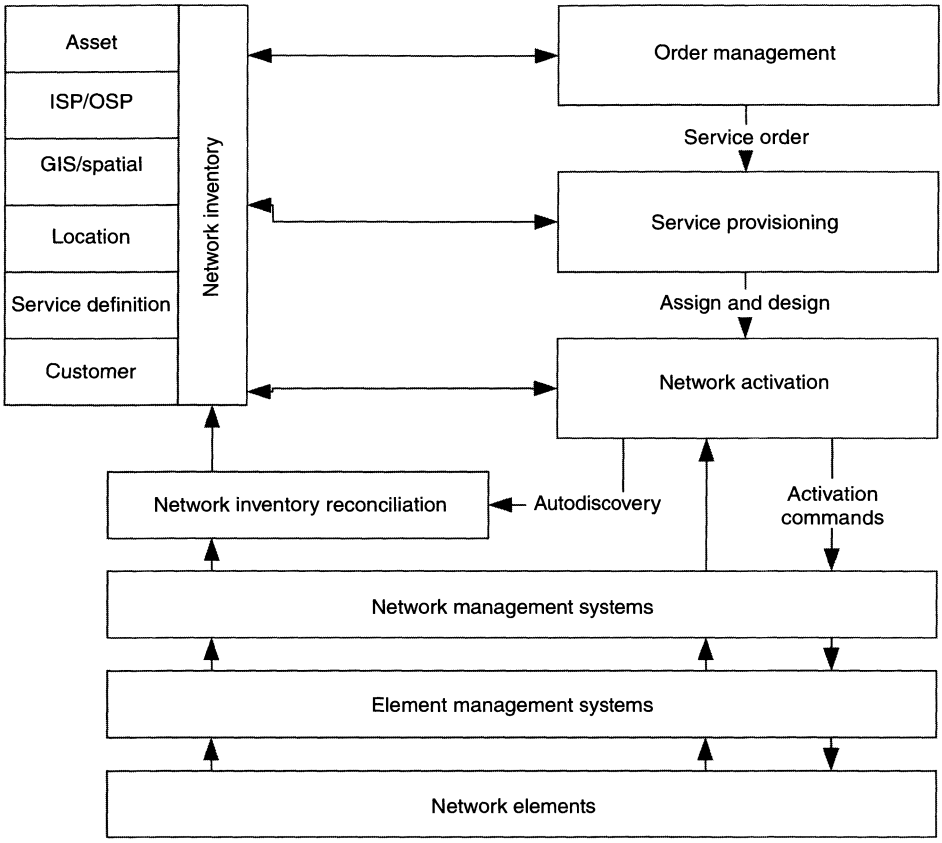


Figure 10.1 Network inventory in the context of network operations.

It is worthwhile for an operator to seek to solve the problem of inventory management because it potentially delivers a high business ROI. Figure 10.1 shows how network inventory could potentially be part of an effective OSS solution.

10.7 TMF Network Inventory Process Outline

The inventory management business process in the TOM encompasses anything related to physical network and information technology equipment and the administration of this equipment. The process is responsible for:

- installation and acceptance of equipment
- the physical configuration of the network
- managing the spare parts and the part return/repair sub-processes
- software upgrades.

Many service providers outsource to suppliers or require as a service from equipment suppliers some or many of the sub-processes, process activities or functions of

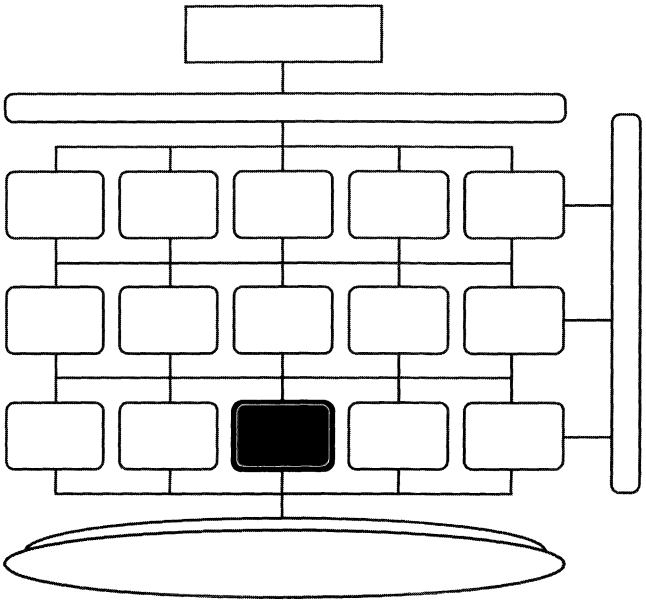


Figure 10.2 Network inventory box in the TOM.

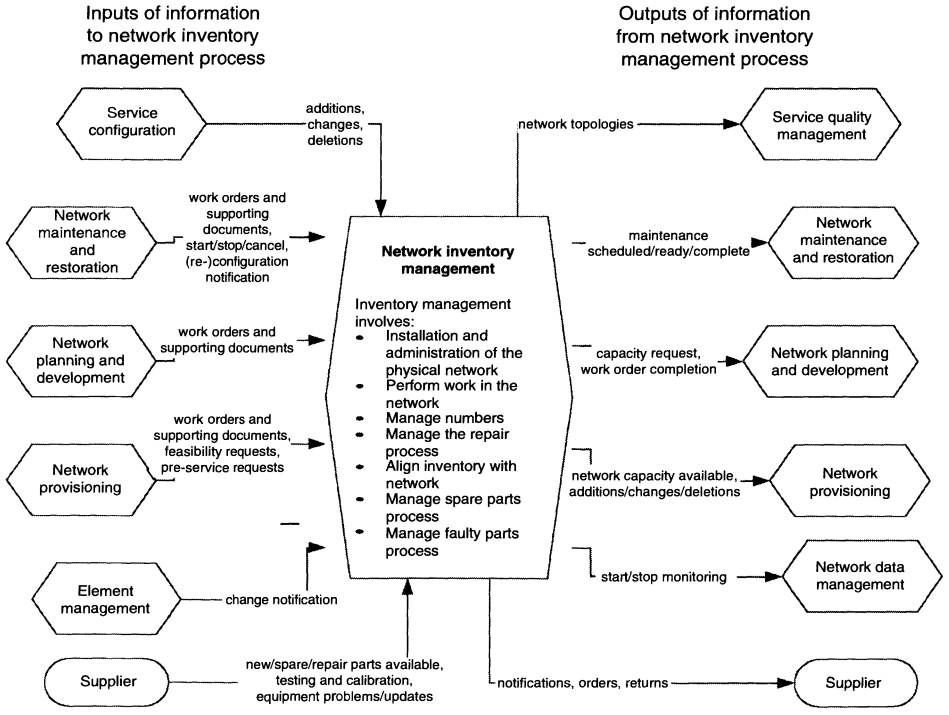


Figure 10.3 Connections between network inventory and other components of the TOM.

this process. Components of this process include site work, repair, software upgrades, part, equipment and software inventory. The aim of this process is to install and administer the physical infrastructure. The process starts with a work order request for installation or implementation required by a network problem. It can also start as a result of administration of the sub-processes corresponding to the repair or replacement of faulty parts or installation of spare parts. It ends with the successful installation or implementation in the network.

10.8 Overview of Inventory Management Products

10.8.1 Telcordia Inventory Management

It is not unusual for up to 20% of a company's inventory assets to be sitting idle in a warehouse or field location. A significant portion of a company's capital network investment may be unused, damaged but repairable, obsolete or simply forgotten. Telcordia Inventory Manager targets this issue by helping to ensure new services or repairs. The telecommunication companies stock extra inventory for switching, transport and other network functions at field and at the intermediate warehouse locations.

Companies have a duty to shareholders to keep constant watch over expenses while simultaneously allowing for the fact that the amount of inventory needed to support provisioning processes grows. Telcordia's Inventory Manager product seeks to balance these, largely by helping companies reduce the costs of excess equipment.

The nature of the telecommunications industry poses a unique inventory challenge. Unlike the consumption, build or sell inventory model used in other industries, which was touched on at the beginning of this chapter, the telecom industry uses a reuse model. A reuse model is one in which inventory is re-used, requiring continuous management. To respond to new service requests, spare inventory is dispersed throughout various field locations, central offices, mobile switching centres, cell sites, bay stations, maintenance garages, large customer equipment closets, trucks and so on. This requires inventory to be managed locally and interactively.

Telcordia's Plug-in Inventory Control System and Detailed Continuing Property Record are designed to be used by the network management and engineering departments of a carrier. It enables an organization to track the inventory and investment of plug-ins, capital tools, test sets and frame-mounted or "hardwired" Central Office Equipment (COE) in addition to any non-COE items that conform to COE-like accounting rules.

10.8.2 Visionael Inventory Management

Visionael provides a good example of an inventory-centric OSS solution. Visionael is also widely regarded as one of the leaders in inventory management OSS.[33]

[33] The author was fortunate to be sent a large pack of materials from Visionael and also to be a regular visitor to their stand at the Telemanagement World Conferences in Nice.

Inventory of network equipment, facilities and circuits sits at the centre of the Visionael solution. The Visionael Repository is an inventory configuration and change management solution. The device library of the Visionael Repository is one of the most complete available and provides ready-to-use models of equipment. The device library provides the basis for Visionael's engineering design capabilities and its ability to present detailed physical and logical views of the network in historical, "as-built" or future states.

Visionael can establish engineering rules and constraints, and also audit the network inventory against the current actual state of the network. Several data collection methods can be used to populate the database initially, or to perform real-time validation of the "as-built" state of the network as recorded in the Visionael Repository.

Visionael provides a network inventory, configuration, circuit and change management solution that can be integrated with other business and network management applications. Such a broad suite of features is not uncommon in the OSS industry. It can also be seen in the offerings of Clarity International, Cramer Systems, Dimension Data and TTI-Telecom.[34]

Visionael's network inventory management solution links the network planning, provisioning, customer service, operations and engineering staff to a single, up-to-date knowledge base of network connectivity and configuration.

Visionael announced its next-generation Visionael 6.0 product, which is based on the company's three-tier application architecture. Visionael 6.0 addresses the needs of OSS and Next Generation OSS (NGOSS) and migration to NGOSS by including change management functionality. The Visionael network inventory application is designed for telecom, DSL and web-hosting service providers, long-haul fibre service providers and network outsourcing companies. Visionael aims to support the lifecycle of communications networks with auto-discovery of SNMP-managed network devices, network design utilities, circuit provisioning, comprehensive reporting and the mapping of logical to physical views of the network down to the port-to-port level. Visionael's API strategy has some similarities to that of AdventNet, and is based on CORBA, Java, C++, COM and XML languages, facilitating one-to-one integration with OSS components and data exchange through a middle layer messaging bus system. The APIs are mostly intended to be used at the time of implementation, so that systems integrators and customers can set up interfaces to exchange data in the Repository with other OSS applications such as service order management, activation and fault management tools.

Visionael's three-tier architecture enables easier integration with other OSS components, such as workforce management and CRM applications. Other factors allowing flexibility and scalability include:

- Support for hundreds of concurrent users and very large databases.
- APIs at both the client and server layers, which AdventNet also provides in its fault and configuration management products.

[34] The author was fortunate to be co-presenter with a senior member of TTI-Telecom, Bruno Mayersohn, at the 2002 Telemanagement World Conference in Nice. The presentation given was on WINMAN, which is an initiative for unified management of IP-over-DWDM networks.

- Distributed processes across multiple middle-tier servers, minimising the bandwidth required for communications with client workstations, which improves performance.
- Administrators can introduce changes at the application and data layers without having to upgrade client workstations.

Visionael 6.0 allows freeform sketching for creating intelligent diagrams for schematic network representation. Intelligent diagrams comprise logical and physical network models in a single environment.

Visionael data collection has network discovery capabilities that allow network staff to obtain data from multiple repositories and from active network elements. This allows field discovery over remote networks, and holding data in a temporary database for later importing these data into the Repository for analysis.

A Device Intelligence Module (DIM) is "awake" to the peculiarities of particular network devices, and can be configured to read MIBs embedded in specific vendor devices.

Visionael's network inventory platform also supports migration to new DWDM network backbones and layered edge services.

10.9 Partnering Between Consultants and OSS Vendors for Inventory Solutions

Service affiliate programs are offered by inventory management vendors to present the benefits of their products and to sell their products to the clients of companies which provide consulting, implementation and other services to telecom carriers.

10.9.1 Visionael and AFO

Visionael has established strategic partnerships with system integrators (SIs) to enhance the installation and support of its network inventory management solution. Visionael's implementation partners include Logica, Getronics, Norsistemas, Global Management Systems, Inc. (GMSI), ADC, NetAxis, GuideComm and Windward Consulting.

Advance Fibre Optics (AFO) is a provider of network management software and services for fibre-optic networks. AFO joined the Visionael Service Affiliate Program in order to help bring network inventory management solutions to CLECs, cable companies and utilities offering network services. Fibre networks enhanced the ability of companies to process transactions across networks. AFO provides OSP InSight fibre network documentation and mapping software to manage telecom networks. The alliance was intended to help AFO broaden its offering so that its customers would be able to manage every aspect of their network, including inside plant, ultimately increasing their ability to handle increased network transactions without growing pains.

This particular partnership had borne fruit by late 2001. AFO and Visionael developed an interface between the Visionael 6.0 and OSP InSight. The latter allows

providers to archive network data, link it geographically to create a digital model of the network and then retrieve it to support mission-critical operations. The integrated solution gives network engineers access to inside and outside plant network details, where the network is based on fibre-optic, coaxial and/or copper cabling technology.

Unification of inventory management is a common theme in OSS systems, where network inventory is often stored in disparate systems/media. It is the mission of many OSS vendors in developing inventory management software to unify the inventory management aspect of telecom operations. In the case of the Visionael – AFO partnership, an understanding of the need to combine inside and outside plant applications was an important driver of the project.

10.9.2 DMR and NetCracker

DMR TELCOM360 is the telecom division of DMR Consulting. NetCracker Technology is a vendor of network inventory management and provisioning solutions. The two companies partnered in order to allow DMR to provide NetCracker's inventory management and provisioning web-based products as a solution to clients' OSS requirements.

The NetCracker suite of products enables service providers to automate and streamline network planning, design and inventory management and provisioning processes. NetCracker solutions integrate with network software including customer care, order management, trouble ticketing and activation. NetCracker's solutions give customers text and graphical views of logical and physical inventory. Information is provided to allow networks to be scaled quickly and cost-effectively, to speed the bringing of new services to market and to handle growing user demands.

DMR's business and technology solutions framework, DMRframe360, automates telecom business processes. The framework integrates process-specific applications with business process automation to provide efficient and flexible systems for critical business methods. DMR's solutions include business architecture, CRM, billing and revenue management, network operations and assurance and enterprise technology.

10.10 Inventory Management for IP VPNs

A recent inventory management solution for IP VPNs was presented by Dimension Data and several telecommunications companies participating in a TMF Catalyst project.[35] The aim of the IP VPN Management Catalyst project was to bring predictability to IP services over shared infrastructures and providing customer self-configuration of complex VPNs with evolving requirements. More complex situations were catered for by using a multi-layer, multi-vendor and multi-service environment. In particular, the project used plug-and-play to accommodate diverse

[35] More specifically, the second phase was demonstrated at the Telemanagement World Conference in Nice, May 2001.

inter-service provider business models with associated changes of information exchanged in a business-to-business environment. The leader of the project opined that it was challenging to offer multi-layer, multi-carrier VPN services and this was largely because there were few or not any OSS which could deal with this environment without swivel-chair integration and telephone calls or e-mails between different sites of different carriers. While consuming resources, this also limited the opportunities for growth and differentiation.

The project sought to deliver to service providers a more complete utilization of network inventory management. The project used TM Forum NGOSS concepts for the development of integrated OSS. The project also quantified the use of the TMF's NGOSS plug-and-play framework to reduce time to market for new services in a realistic environment and addressed the additional issues, including security and scalability.

10.10.1 Netscient

Netscient's N-Center is an object-oriented network planning and inventory management solution. It has modules for equipment, circuit and transmission management. N-Visage performs traffic modelling and route optimization at the logical layer and can plug into N-Center as an integrated solution.

Netscient incorporates ESRI's mapping technology across all its planning, design and inventory management products within its product portfolio using MapObjects.[36] Netscient's integrated inventory management solutions supports the full array of ESRI's advanced mapping technology for site planning and spatial data management of a telecom company's physical assets from trenches, ducts and cable routes.

10.10.2 Fujitsu

FLEXR Plus is Fujitsu's element manager/subnetwork controller that simplifies the management and administration of network elements. FLEXR Plus has a graphical format that allows provisioning, maintenance, inventory management and monitoring of advanced SDH equipment such as the Fujitsu Lightwave Cross-connect and Digital Microwave Radio.

FLEXR Plus accesses the entire network through an X.25 interface to the packet switched network or an interface to a local communications network/Ethernet. Once connected to a single Fujitsu network element, FLEXR Plus gives the user access to all other Fujitsu SDH elements in the network via the SDH data communications channel. This allows provisioning, performance and alarm status monitoring of up to 1000 network elements.

The FLEXR Plus displays a network map with equipment and network icons to assist in isolation of alarms and retrieval of status information. Provisioning can be done using point-and-click. To simplify the creation of work performance records, FLEXR Plus can store log files. We shall see details of how log files are used in AdventNet OSS products.

[36] ESRI is a geographical information systems (GIS) company that was established in 1969.

Some OSS vendors include inventory management under the head of configuration management. For example, Clarity's Configuration Manager product manages telephone number creation and allocation, and also supports logical and physical configuration of telecom networks. The product also integrates network elements and network element management systems.

10.11 Functionality of Inventory Management Product

An OSS product with inventory management functionality such as Clarity's Configuration Manager can interface with other OSS such as materials management and geographical information systems, and can track physical network inventory data such as:

- sites, customer, carrier, common and interconnect
- manholes, network access points, network elements (NEs)
- wireline and wireless transmission equipment.

Clarity's approach to network inventory OSS supports the network inventory items listed in Table 10.1.

We discuss some of these features in more detail in Chapter 11. However, many of the features are also relevant to inventory management.

To define an assembly circuit, one must give a name to the proposed assembly, assign a status – proposed, active or inactive – and give a description of the assembly. In creating or modifying an assembly, one must define assembly endpoints and define internal assembly links.

The ability to bulk load and automatically create the physical connectivity between network elements, frames and equipment adds to staff productivity at the Network Operations Centre (NOC). Assemblies will be available to network operations staff as abstract capacity during the circuit design and provisioning process.

Provisioning staff will be able to select the entry point to an assembly and have the system automatically assign/allocate capacity through the assembly.

One must specify the parent connections for an assembly under consideration. The details of a parent connection typically comprise:

- sequence number for the parent connection appearance on the screen
- cross connection

Table 10.1 Network inventory items.

Category	Action
Assemblies	Model an assembly of equipment, frames and physical connectivity
Circuits	Create circuits, modify existing circuits and query existing circuits
Network elements	Create, modify and query network elements and apply port templates
Port templates	Create port templates that can be selected to enter network element details
Rings	Create SDH rings that can be used to create path diversity in a network
Reference screens	Set up core network configuration details
Reports	Display and print specific network configuration information.

- location of source network element
- type of source network element
- instance of source network element
- terminating card port name of the source and destination network elements and card identifiers – there may be more than one of each
- destination network element
- type of destination network element
- instance of destination network element.

One must also specify path connections. Specifying a path connection involves the same type of information as for a parent connection.

In defining an assembly, one must define the assembly endpoints. This involves specifying the source network element, which is the network element where the connection begins, and the start and end locations of the source network element. The same must be specified for the destination network element. Finally, one must specify the type of link. Defining an internal assembly link also involves defining a source and destination network element, their start and end locations and the type of link.

Transmission or transport circuits are a central component of network inventory. Thus, an inventory management product should provide the ability to query circuits that are in the inventory. Circuits are queried on the basis of key features. Some are listed in Table 10.2.

When querying, a user may want to specify filters with one's query to speed the process of finding the records for the relevant circuits for which one is searching. Possible filter options are given in Table 10.3.

The results of a query will display a variety of information depending on the particular OSS inventory solution. Typically, something like the information in Table 12.4 might be displayed.

Circuit capacity details (see Tables 10.5 and 10.6) are just as important as the presence of circuits themselves, the services they provide and the customers to

Table 10.2 Circuit attributes.

Query type	Description
Service ID	Searches for service IDs
Circuit name	Directly search circuit names
Customer	Searches for circuits and returns customer circuits
Services affected	A circuit name is specified and the query returns child circuits associated with the circuit specified provided that those circuits are leased
All service affected	A circuit name is specified, and the query returns a list of carriers and the child circuits of the specified circuit that are leased by each carrier
All bearers	A hierarchical list is given of circuits associated with the specified circuit
Network element	A list of circuits that use the specified type of network element is returned. A location can also be specified so that a circuit is returned only if it uses the specified network element and is in the specified location
Locations	All circuits which run between the specified sites are returned
Alternative name	Circuits may be assigned an alternative name. The user can query for all circuits which carry a specified alternative name

Table 10.3 Filters for circuit queries.

Filter	Description
Proposed/reviewed/confirmed	Provides circuits with a particular status. One might be able to suggest greater than or equal to proposed, which would mean proposed, reviewed or confirmed
In service	This would include circuits with the status of Commissioned or In Service
Out of service	Includes circuits with the status of Out of Service, Cancelled, Pending or Delete
Show bearers	Includes the parents (or first-level bearers) of each circuit returned in a query
Check for alarms on displayed circuit	Current alarm status is given (if available) on each circuit returned in a query.

Table 10.4 Circuit records showing circuit details.

Information item	Description
Circuit name	Circuit name
Bearer	Bearers on the circuit
Tributary	Tributary number
Customer	The customer to which the circuit provides service
Service ID	Service identification for the circuit
Status	Proposed/Reviewed/Confirmed/etc.
Alarm	Whether or not the circuit has recently been affected by an alarm

Table 10.5 Circuit capacity – bearers and ports.

Circuit detail	Description
Spare bearer capacity	The unused bandwidth that is available for a selected circuit. This is the difference between the speed (in Mbps or Gbps) specified in the bearer record and the total speed of that bearer's tributaries
Spare configured ports	These are the ports that are free for use on the terminating network element equipment of the circuit

Table 10.6 Circuit capacity details.

Information item	Description
Circuit	Circuit name
Service ID	Service order identification number
Speed (Mbps)	Speed of the nominated circuit/bearer
Speed of tributaries (Mbps)	Total sum of the speed of all child circuits
Over 80% capacity	A true or false value, indicating whether or not the circuit is operating at 80% of its capacity or not (i.e. is the sum of the capacities of the tributaries 80% of the circuit's capacity?)
Tributary	Circuit name of the child
Tributary number	Sequence number of the child in the bearer
Speed	Circuit speed of the child
Terminating equipment (A end)	The network element on which the A end of the circuit terminates
Terminating equipment (B end)	The network element on which the B end of the circuit terminates
Port name (A end)	The port name at the A end of the circuit
Port name (B end)	The port name at the B end of the circuit
Card name (A end)	The card name at the A end of the circuit
Card name (B end)	The card name at the B end of the circuit

which they provide services. Indeed, capacity is critical to inventory in any context. Therefore, an inventory management OSS system should make it easy to determine the capacity of circuits.

Details of network inventory are dependent upon network configuration – for example, the parent/child hierarchy of circuits – and are central to service provisioning. The availability of network inventory, such as circuit capacity, affects how network resources are to be provisioned to enable a service. Network inventory overlaps with network configuration particularly for *logical* network resources, as opposed to physical resources, since the logical structure of a network is mostly coincident with what is configurable.

The author thanks Clarity International for assistance with this section.[37]

10.12 Asset Location

It is one thing to know what one's network assets are and another to know *where* these assets are located. Although inventory management OSS applications do account for asset location, if only by providing a text field in which to enter it, the issue of *how* the location is discovered is sometimes not mentioned. The answer is often prosaic: field staff must go and find the location of the asset so that the location can be keyed into the inventory management OSS application. Of course, it would be preferable if the labour involved could be reduced or eliminated.

In this section, we discuss one proposal for automating asset location. This section is based on the paper "JLocator – Web-based asset location system" by Luca Deri of the University of Pisa, Italy. We give some background on the need for automated asset location and a brief introduction to the approach taken by JLocator. While introducing a cutting edge proposal, the reader will also see many of the issues that are involved in asset location. Unfortunately, discovering asset location or "localising assets" has not been systematically carried out in the past.[38]

Physically locating assets is made complex by network growth and mobile computing. Existing network management tools are usually not suitable for managing dynamically moving assets and provide few facilities for asset localisation. In addition, asset management products delegate to human operators the task of locating physical assets.

JLocator is a Java-based system that allows assets to be dynamically localised. Users can locate assets through a web interface, and external applications such as asset management systems can take advantage of asset location information provided by JLocator. JLocator's distributed architecture makes it scalable and platform-independent.

Mobile computing, teleworking and shorter computer lifespan have complicated system administration and service provisioning. In particular, IPv4 provides little support for mobility forcing a mobile computer to change its IP address as it changes location. The result is that management applications are usually unable physically

[37] Clarity is now a subsidiary of Powerlan Ltd.

[38] We shall use the term "localise" as meaning "determine the location of".

to locate a network resource and cannot access services provided by mobile computers when their IP addresses change.

Asset management systems allow network administrators to track company assets and provide them with information such as asset manufacturer, installed applications, inventory ID and services provided. Typically, asset management systems assist administrators in purchasing new equipment, detecting computing resources and allocating maintenance/asset cost for each group or division of the organisation. Commercial asset management systems provide few facilities for automatically locating assets and keeping track of their movements. Operators are responsible for updating the information in the asset database concerning asset location whenever assets change position. Owing to this half-automatic asset management procedure, the asset database frequently has a random mixture of fresh and outdated location information.

Computer-based telephony applications are slowly substituting conventional telephony equipment. The basic idea is that each mobile user has a computer to which incoming calls are diverted. Network servers are responsible for transparently diverting calls to the position of the mobile host. Hence it is increasingly important that tools are available to keep track of the physical host position in order to:

- Locate malfunctioning computers that are connected to a large network.
- Allow some applications such as computer-based telephony applications to function.
- Find how to contact the user of a mobile computer that is malfunctioning at any given moment. For instance, an asset tracking system should provide the number of the telephone that is closest to such a user.

An effective asset localisation system would be a prime example of software that has high ROI and good connectivity.

The SNMP community attempted to address the problem of asset localisation by specifying the *sysLocation* variable in MIB-II. The variable *sysLocation* contains the location of the asset in a unspecified form such as building, floor and room. In order to use this solution:

- An SNMP agent must run on each asset.
- In order to retrieve asset positions, assets should be active because the SNMP agent running on the assets must reply to queries.
- Network administrators must update *sysLocation* whenever the asset changes position.

This update process is not automated: it relies on human operators and so it is possible that *sysLocation* has outdated values. Hence this approach does not constitute a solution to the asset localisation problem.

Asset location generally can only be performed if some support for the task is provided by the underlying network. For example, consider a LAN based on bus Ethernet (10-100Base/2) or token ring. In that case, it is not possible, from a programming viewpoint, to discover whether two hosts are physically adjacent. On the other hand, point-to-point networks such as ATM or 10Base/T Ethernet *do* allow localisation of hosts. Once the physical location of network connectors is known

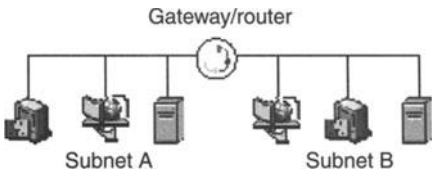


Figure 10.4 Subnets attached to a gateway/router.

and stored in a database, it is possible to calculate which host is attached to a given connector. This, in turn, localises the host. If the underlying network is point-to-point, hosts can be located independently of the network protocol used by the hosts.

JLocator currently works in the IP world only, as IP is the most widely used protocol. The algorithm used to locate hosts relies on the media access control (MAC) address. The MAC address is the hardware address of a device attached to a network. Although it differs for various physical media, network board manufacturers guarantee its uniqueness and this allows a computer to be uniquely identified independently of its IP address. Some hosts implement single link multi-homing, which allows multiple IP addresses to be associated with the same hardware interface.

In the Internet world, the Address Resolution Protocol (ARP) is used to map dynamically and correlate MAC and IP addresses. The IETF's RFC 1157 defines the object type *atTable* that allows, through SNMP, cached ARP tables to be retrieved. Each *atTable* entry associates an IP address with its corresponding MAC address. Considering Figure 10.4, when two or more (sub)networks are interconnected, packet routing requirements demand that a gateway/router sits between them. The gateway is informed of the MAC addresses of both subnets, and so the union of all the gateways/routers of a network contains the association IP/MAC address for every host in the entire enterprise network.

Drawbacks arise in using ARP because:

- Networked hosts that have never communicated with other peers are not present in the cache.
- Cached entries expire and so a host that has not communicated for some time may not have an entry in the ARP table.

The cache can be refreshed to ensure that it has an entry for each active host. Before searching the ARP table, a host pings the hosts that belong to the known networks. This operation "wakes up" the hosts and hence propagates their MAC addresses, so refreshing the ARP table. JLocator also keeps a persistent MAC address cache that is periodically refreshed. When the associated MAC/IP address is known, the MAC address must be associated with the physical location of the network element that it identifies. Once the position of a cable coming from a specified port is known, it is possible to localise a host at any given moment.

This is broadly how JLocator finds the location of assets. This section has outlined some of the problems in existing approaches to asset localisation, explained the challenges in asset localisation and described an asset localisation implementation that is superior to many of the other solutions currently available.

11 Configuration Management

In this chapter, two of the latest configuration management implementations in the market are detailed. While describing front-end features, some implementation details will be given to help understand the back-end functionality of an OSS implementation. The first implementation to be discussed has an excellent API implementation, and some aspects of this feature are described in detail. Much more space will be spent on the first implementation, in order to flesh out the details of how it is used. Central functionality will be focused upon in the second implementation.

11.1 Introduction

Configuration management OSS functionality refers to the management of information and other forms of assistance and support for configuring network devices. Configuration management functionality varies widely between OSS products.

The AdventNet Configuration Management Server (CMS) product is advanced in addressing the nuts and bolts functionality of network devices or Network Elements (NEs). CMS focuses on providing the network administrator with as much ability as possible to learn about the configuration state of NEs and to control their configuration state. On the other hand, the Clarity Configuration Manager (CCM) product is strong in functionalities such as circuit design and displaying circuit topology. CCM also provides information on features that are specific to particular telecom network technologies, such as optical transport. For example, CCM makes it easy to manage synchronous digital hierarchy (SDH) rings and has explicit and hard-coded functionality to support SDH rings. This is a result of Clarity's feature-set being driven by the needs of the specific customers that Clarity has served over many years. By contrast, CMS seeks to provide maximal extensibility through Application Programming Interfaces (APIs) so that the third-party software developers, or in-house developers, at service providers can extend or customise the product for specific needs. Both approaches have their strengths, which are reflected in the success of both AdventNet and Clarity's products.

K. Misra et al. (eds.), *OSS for Telecom Networks*

11.2 Configuration Management Implementation

AdventNet describes its CMS as a customizable, web- and Java-based, cross platform, Internet infrastructure management OSS software application. The design of CMS leverages the ubiquity of the internet and the convenience of the web for configuration management.

CSM provides foundations on which:

- Network equipment vendors can build element and network management systems.
- Telecom carriers can build custom network management solutions.
- Independent Software Vendors (ISVs), System Integrators (SIs) and enterprises can build application, system and network management solutions.
- Service providers can build OSS.

CMS allows solution providers to leverage technologies such as Java beans, XML, JDBC, CORBA and SNMP to deliver web-based solutions for providing access to management information.

More complex and disparate devices are being integrated into the network infrastructure. A greater diversity of applications and services such as ERP and video conferencing are being provided over the same network infrastructure. Maintaining availability, reliability and responsiveness of these services requires coordinated management and configuration of many devices such as routers and applications from several vendors and manufacturers. Devices and applications from different vendors often have their own configuration interfaces and rules. Unified management of these devices and applications adds great value to the service provider, or at least greatly reduces cost.

Most enterprises and service providers have manual or custom software for managing and configuring their network infrastructure. However, as the network infrastructure becomes large and complex, the frequency and the complexity of the configuration tasks rise rapidly. For example, a slight change in policy might mean reconfiguring hundreds of devices. Even if there is unified management at the NMS of any class of device, most networks are multivendor networks and so the change must be repetitively translated for, and should be an input to, a large number of management applications. This is wasteful and so OSS vendors such as AdventNet and Clarity have built successful businesses by providing automated configuration management software that supports configuration and rollback of hundreds of different types of devices and applications. These systems provide options that allow integration into existing network management systems for end-to-end management of the infrastructure.

AdventNet CMS, CCM and several other products seek to address these requirements with a multi-protocol configuration of management frameworks. Because all service providers' technical and business needs are different, each of these solutions can generally be used as a standalone system or integrated into other NMS/OSS/EMS/Provisioning systems/Legacy Network Management systems. Some of the main features of CMS are:

- **Support for variety of management protocols:** Support for multiple protocols such as SNMP, TL1, Telnet and Trivial File Transfer Protocol (TFTP) allows configuration of a variety of devices.

- **Integration with OSS/NMS/EMS/provisioning systems:** Multiple northbound interfaces, such as RMI, CORBA and Java Management Extensions (JMX), allow integration with existing legacy NMS, third-party OSS or provisioning software. This allows management information to be made available centrally.
- **Configuration management features:**
  —Task-based configuration over multiple devices in a single operation.
  —Ability to store configuration profiles that can be applied over multiple devices.
  —Support for "multi-tasks" which configure multiple devices in a specified sequence in a single operation.
  —Template-based tasks can be created whose values can be obtained from network inventory, from the user or from the device itself.
  —Configurable rollback policy for each task.
  —Support for updating the inventory database with the latest configuration changes.
- **Security and auditing features:** User-based authentication and fine-grained authorization provide controlled access to various configuration operations. Auditing options maintain an audit trail of network changes.

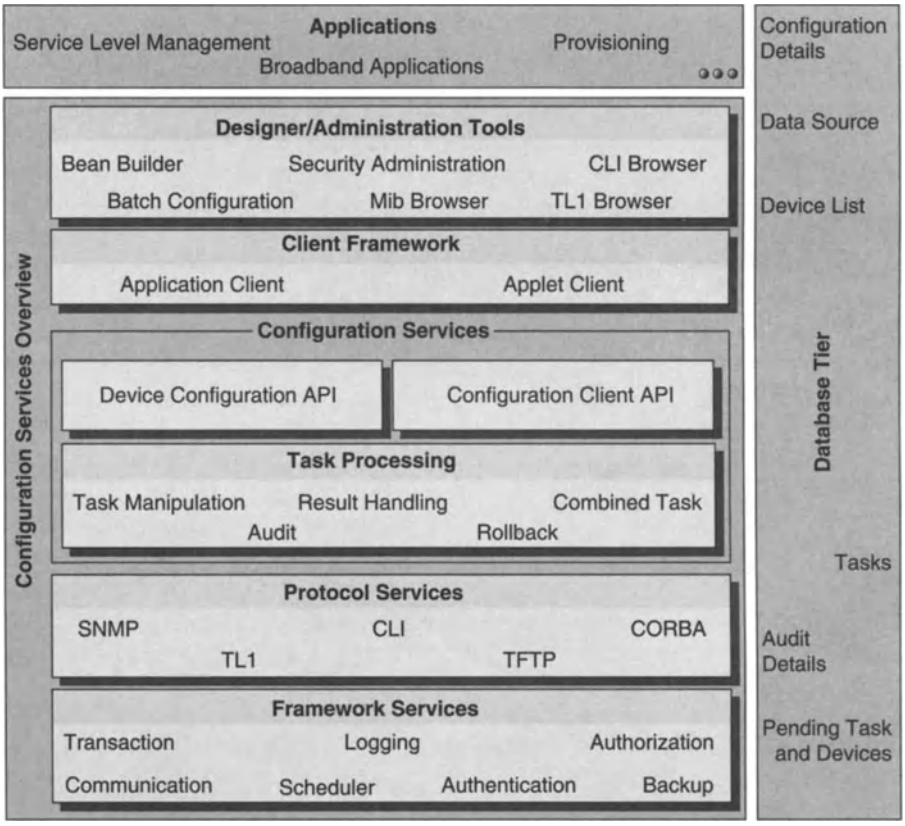


Figure 11.1 Overview of AdventNet configuration management server.

CMS is customizable for specialised requirements and increasing diversity of the infrastructure. CMS is future-proof because its functionality can be extended by the connection of other products via software interfaces. Customization and extensibility features include:

- **Standards-based framework:** Built over a scalable server framework that is based on standard technologies such as XML, EJB, JMS, JFC, JDBC, LDAP and HTTP.
- **Customizable client:** The clients are also developed over a generic AdventNet Client Framework that allows customization of the client software.
- **Integration of new and proprietary protocols.**
- **Developer tools:** Developer tools provide support through the entire lifecycle, from customizing and testing through to deployment and support.

Figure 11.1 provides an overview of the AdventNet CMS.

On the deployment architecture, if a configuration management solution requires management of less than 10 000 elements, and only modest frequency and complexity of the configuration tasks, then the single server architecture is most suitable (see Figure 11.2).

In the single server architecture, the middle tier of the management solution, the servers, built on the CMS, is deployed to run in one Java Virtual Machine (JVM) process. This could be deployed on a Java Runtime Environment (JRE) or in a J2EE server environment, such as WebLogic or JBoss.

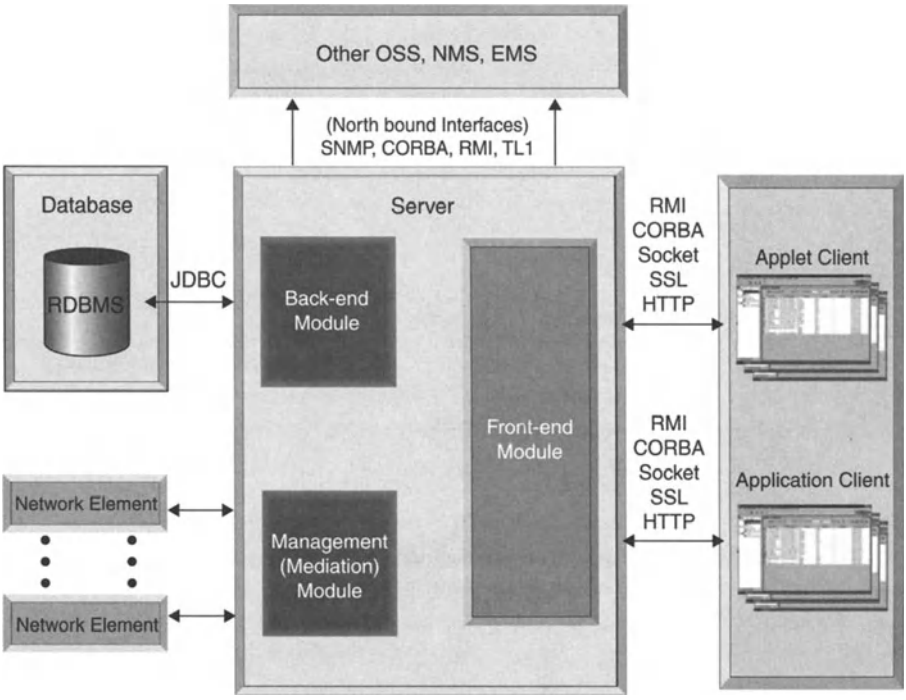


Figure 11.2 Single server architecture.

A highly scalable CMS is one in which:

- a large number of clients are accessing management data from the NOC
- a large number of elements are being managed
- there is high frequency and complexity of configuration tasks
- there is tight integration – and much communication – with northbound NMS/OSS/provisioning systems.

In this case, a Distributed Server Architecture is most suitable (see Figure 11.3).

CMS seeks to help developers to leverage many technologies. CMS supports protocols such as TL1, XML and SNMP out-of-the-box. It also has an interface that facilitates the integration of other management protocols. Java and associated technologies are emerging as important means for building network management solutions for server- and client-side applications. Reasons for this trend include ease-of-use and the shortening of development cycles. More tools are becoming available to network management developers as the industry grows.

Table 11.1 gives an overview of the technologies on which CMS is based.

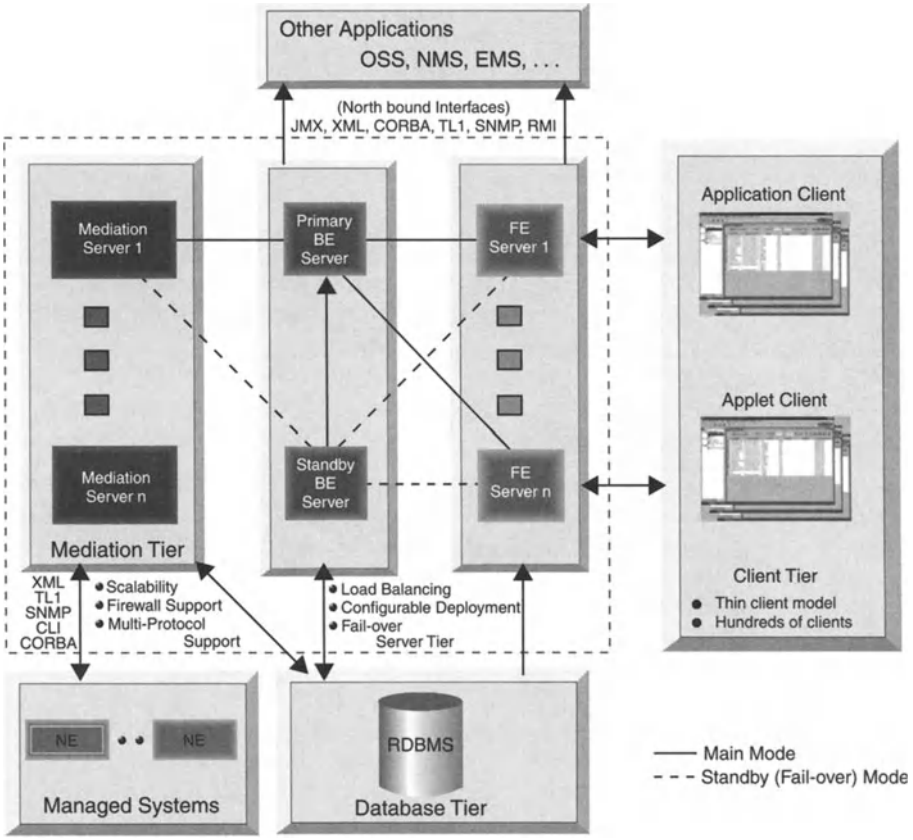


Figure 11.3 Distributed server architecture.

Table 11.1 Technologies supported by CMS.

Standard technology	Version(s) supported	Key benefits
EJB	1.1	EJB simplifies the development of scalable and transactional middleware components. The EJB facilitates building deployable components where the components of an application can be independently deployed and managed
XML	1.0	XML facilitates data exchange between application components and between applications
JAXP	1.1	JAXP provides a standard API to process XML documents using DOM, SAX and XSLT. This makes it easier for applications to choose an XML parser implementation based on application needs, for example, high-performance vs low-memory footprint parsers
JNDI	1.2	Java Naming and Directory Interface (JNDI) is a unified interface for multiple naming and directory services, facilitating location transparency. JNDI allows applications to locate and use at runtime the services on which they are built. Without JNDI, an application would be forced to assume the availability of services in predetermined locations
JDBC	1.1 and 2.0	JDBC provides database technology-independent, RDBMS and ODBMS, and platform neutral APIs to access data
Java RMI	1.0	Remote Method Invocation (RMI) aids distributed computing which assists in scalability and performance
CORBA	2.3	CORBA provides a programming language-independent distributed object-computing environment for building scalable applications
HTTP	1.0 and 1.1	HTTP facilitates a standard communication mechanism between Web Browser Client and the Web Server. HTTP is also used as communication mechanism between other clients and servers
JFC	JDK 1.2 and JDK 1.3	JFC is a collection of widgets which facilitate building GUIs that are consistent across platforms
JSP	1.1	Facilitates dynamic content in web pages by allowing Java code to be embedded in HTML pages. Presentation and dynamic content can be separated with Java beans and JSP tags
Servlets	2.2	Servlets enable the running of classes in the server based on client inputs
TL1	1.0	Transaction Language 1 is an ASCII-based management protocol widely used in telecommunications for managing the telecom infrastructure. TL1 is ASCII based and so it can be used as a human-to-machine and a machine-to-machine language
SNMP	v1, v2c, v3	SNMP is the most widely used management protocol, defining the communication mechanism between the managed entities and the management station, the information model and the security and access control mechanisms for sharing management information

EJB, Enterprise Java Beans; JAXP,?; JNDI, Java Naming and Directory Interface; JFC, Java Foundation Classes; JSP, Java Server Pages; SAX,?; XSLT,?; ODBMS,?

11.2.1 Product Installation Cycle

CMS offers a development environment to encourage the use of CMS as a *platform* on which a management solution can be built. AdventNet uses the concept of the "Configuration Management Experience" to explain how the complete product lifecycle needs of a management solution can be realised using the product. These lifecycle needs are captured in five steps (see Figure 11.4):

1. rebranding the management solution
2. customizing mediation services
3. customizing management services
4. integration and testing
5. packaging and deployment.

1. **Rebranding:** The application can be rebranded to display the name of the user's company, the name of its product and its logos.

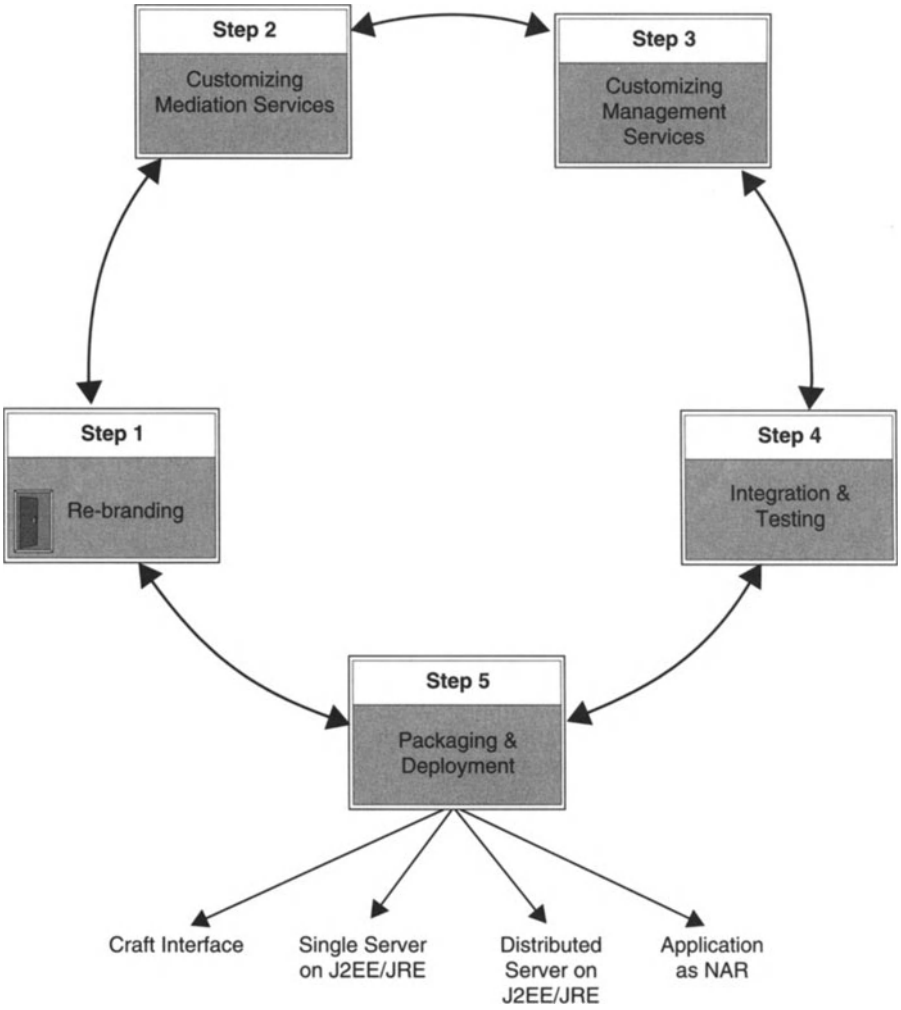


Figure 11.4 Five steps in the "CMS experience".

2. **Customizing mediation services:** Southbound protocol services enable support for a number of southbound management protocols in the configuration management system. While CMS supports standard management protocols out-of-the-box, developers can add a custom protocol provider in order to support any other protocol. Such support is often added during the implementation project, which is carried out in conjunction with an SI, for example, when it is discovered that the network critically relies on some old piece of network equipment that does not understand SNMP or any widely known management protocol. When this occurs, it usually does not pose a problem for an experienced OSS vendor or SI with a track record of implementing OSSs.

3. **Customizing management services:** The configuration service is provided to modify system configuration for optimizing its performance and the network's performance. Through the element configuration user interface, tasks can be preconfigured and the systems which need to be configured can be chosen. The task, saved as an XML file, can be modified and applied to other such systems.

Software distribution, upload and download, services assist the network administrator in managing systems inventory. A task can be configured through the GUI for uploading and downloading the software to a TFTP enabled system. The software may be a new application or an operating system upgrade patch.

The authentication service supports the management of user authentication. Through the user administration tool, users can be added and deleted and passwords assigned and modified. Based on the authentication, users can be allowed to create or execute particular tasks.

Using the access control/authorization service, a network administrator can define user privileges for secure management of the systems.

11.2.2 Product Architecture

Management applications must get smarter as network infrastructure increases in size, heterogeneity and complexity. End-to-end management and scalability are the order of the day for most OSS functionalities. Configuration management is one such functionality as it is so central to service provisioning and inventory management.

The following are some of the key configuration management challenges that must be addressed to achieve end-to-end, scalable management:

- **Support for multiple management interfaces:** Although there may be great diversity in network elements, the management application should be able to communicate with all devices.
- **Provide a scalable enterprise configuration:** When a network reaches a certain size, automation is necessary because it becomes impractical to manage and maintain the configuration information on an element-by-element basis. Automation also reduces the staff costs and ensures consistency of configuration throughout the network.
- **Audit configuration changes:** In a large network, configuration changes must be logged.

AdventNet has designed CMS as a multi-protocol configuration management solution with the above features. With regard to scalability and reusability, standards such as XML allow integration with third-party NMS.

Configuration management has a task-based architecture with the following features:

- The administrator can configure devices using multiple management interfaces. Multiple tasks can be configured over multiple sets of devices.
- Configuration profiles applied to a device are stored so that the same configuration can later be applied to another set of devices.
- Tasks can be changed before being applied to the network by the server.

- Information such as user and time of configuration change allow task auditing.
- User authorization can be required before creating or executing a task.
- Inventory database is updated with configuration changes made to NEs.

A diagram of the architecture of CMS is given in Figure 11.5.

CMS is the heart of the AdventNet configuration management solution, and is the component which actually does the configuring of network elements. The server

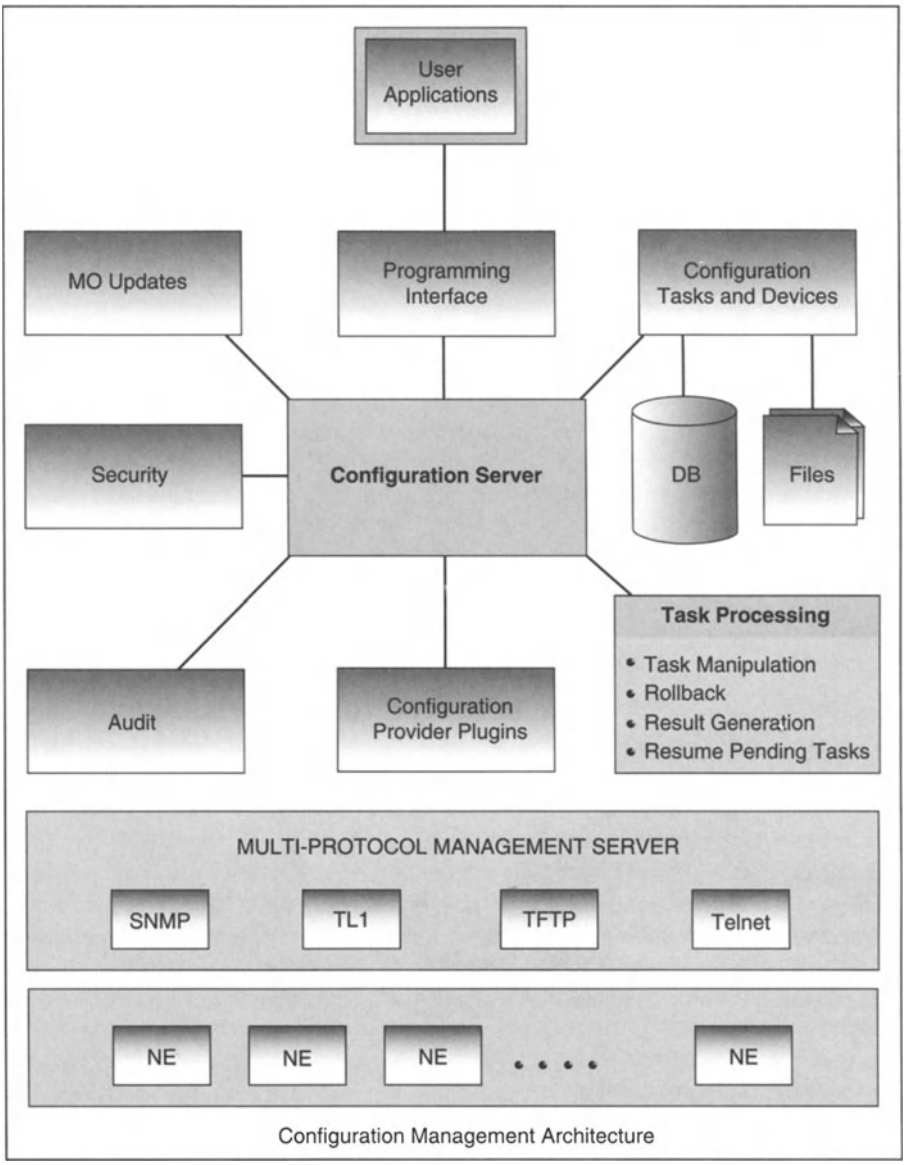


Figure 11.5 Configuration management architecture.

also ensures that each NE commences operation in the desired state, carries the required parameter values and has the desired relationship with other NEs. Thus, it interacts with components such as the task processing, configuration provider, security, audit and inventory update systems.

Configuration attributes are represented as "tasks". Task processing involves identifying attributes and devices, performing authorization checks, storing attributes and devices separately, task manipulation, task execution and result generation. The storage of attributes and devices in a persistent data store allows reuse of the information for other configuration operations. Before a task is applied to a device, task manipulation rules can be applied so that the task is modified. It is possible to plug in user-specific task manipulation rules. Finally, result processing is equipped with a rollback feature.

To communicate with NEs, providers of protocol-specific configuration commands can be associated with the configuration server. The configuration server integrates multiple configuration providers.

The CMS makes it easier to configure NEs. To address the security threats that could arise, authorization checks are carried out by AdventNet security management services before the task is configured.

When very large networks are involved, it is necessary to log configuration details. The configuration management solution has a customizable audit facility. Configuration changes trigger updates in the inventory database.

To sum up, CMS helps network management staff achieve some defined functionality by configuring network devices as necessary. Its APIs can be used to develop user applications and to maintain network resources and network configuration changes.

11.2.3 Configuration Server Database Schema

Audit-related information is stored in the RDBMS and can be accessed via APIs. The back-end server maintains and updates the information model in persistent storage. It is instructive to see the various tables that are implemented in the RDBMS:

- **Task to Device List Map:** When a task is defined and devices are associated, the mapping between the tasks and device lists are stored here.
- **Task Audit:** This table is used to store task-level audit details. It contains information such as task name, time submitted, device list and data source.
- **Attribute Audit:** This table contains audit details at the attribute level and contains information such as the number of retries and ending time of execution.
- **Device Audit:** This table stores device-level audit details, including information such as device name, task name, starting time of execution and ending time of execution. This also contains the configuration status, that is, success or failure.

11.2.4 Tasks

A "task" is an entity containing configuration attributes and the data used to configure the devices to achieve a desired functionality. Attributes, on the other hand, comprise the command to be executed and the set of values needed to execute the

command. For example, in the case of SNMP, the attributes are the OIDs. In the case of Telnet, an attribute is the command to be executed in a network element.

While creating a task, the task level and attribute level information must be mentioned. The task-level information is set using methods of the *TaskGenerator* class. This is the basic class used to define and generate configuration details in XML format. This class provides methods for defining task-level information such as the task name, whether or not rollback is needed, and whether or not an existing task should be overwritten.

To define task attributes, the classes *ProtocolAttribute*, *SNMPAttribute* and *TelnetAttribute* are used depending on the protocol used for device configuration.

Creating a task involves:

- getting the ConfigClient instance
- defining the attribute-level information
- specifying the task-level information
- handling the result of configuration.

An "attribute" is an entity comprising the command to be executed and the values passed to execute the command. The attributes for the task can be set using the methods in the classes *ProtocolAttribute*. If a task is defined by a derivation from a task template, then device attribute values can be set using a "data source". In that case, the data source would form the main data repository/source from which the task receives its input. A data source is an entity which holds a set of values for the place holders configured in a template. The place holders, in turn, are configured when setting up the task template. More than one data source can be associated with any given task template. The use of a data source can be prohibited via the task's Input Type parameter. More information is given on data sources below, when we discuss task execution in the context of task templates.

Detailed task-level information including task name, protocol to be used for executing the tasks, rollback type and rollback document are specified. The task is then created using the methods provided in the *TaskGenerator* class.

When a task is executed over roughly 10 devices, the result of the configuration is only available after all the devices are configured. An option is available to ensure that the result of device configuration is sent as soon as the device is configured.

Recall that a task is an entity that contains the configuration attributes along with the data used to configure a device to achieve a desired functionality. To configure SNMP-enabled devices, an SNMP task must be created.

As discussed earlier, the methods in the *TaskGenerator* and *SNMPAttribute* class are used to set the task and attribute information for the task. To provide SNMP-specific details, the SNMP-specific class *SNMPTaskGenerator* can be used. This class is an extension of the TaskGenerator class and defines specific methods for SNMP-related functions. The ConfigClient instance is obtained and the attributes set. Creating SNMPv3 tasks involves special considerations due to particular features of SNMPv3, especially security features.

Once task information and attributes are set, the task is converted to XML format and saved. Once the task is saved, the result of the saved operation can be handled so that the ConfigClient instance receives the desired result.

Creation of Telnet, TFTP and TL1 tasks is similar to the creation of SNMP tasks.

11.2.4.1 Executing Tasks

Having created a task, it will at some point be necessary to execute the task. Once the task has been defined by setting the task and attribute level information, the task is stored in a particular directory. To execute a task, the task should be associated with a set of devices which are given in a device list. It is over these devices that the task will be executed. A method of the ConfigClient API class is used to execute tasks.

In associating devices with a task, the devices attributes can be set using the methods in the *Device* class. For example, to execute an SNMP task or TFTP task, the devices are defined as shown below. The Device *method* takes as string inputs the host on which the task must be executed and the port through which the configuration must be accomplished. For example, for an SNMP task, devices are defined as shown in Figure 11.6.

Once device properties are defined, the task is associated with devices. Next, the task XML code is generated and passed as an argument to the ConfigClient API's method that will execute the task.

The result of task execution can be obtained using the audit trail. The methods in the classes *DeviceAudit*, *AttributeAudit* and *TaskAudit* can be used to discover the device, attribute and the task level details.

If the task is created and executed in a single command in some program, then, depending on the Boolean values that have been assigned to, or are returned by, the methods setPersistence, setNewTask and setOverwrite, the task may or may not be stored in the database. For example, the persistence property will be in effect only if the method setNewTask is set to "true".

11.2.4.2 Sets of Tasks

When a set of tasks is executed from the client, a session is established between the CMS and the device under configuration. Once the task is executed, the session is terminated. If the user wishes to execute, say, 10 tasks over the same device, then rather than closing and reestablishing the session for the next task, the user can reuse the session already established.

A combined task is a combination of sub-tasks where each sub-task is defined in some particular protocol. It is useful to create a combined task when a sequence of operations must be carried out on a device, and each operation is carried out by a task. For example, if a user wishes to configure a device with two different tasks, such as to allocate capacity and send a confirmation message to the NOC, then a

```
Device d1=new Device("localhost", "161");
Device d2=new Device("device1", "161");
```

Figure 11.6 Code for defining devices.

single task can be created with these as sub-tasks. In order to create a multiple task, the tasks to become the sub-tasks must already be created.

Device lists are those which have a set of devices, grouped by certain properties. To execute a task over a set of devices, those devices can be specified as a device list and stored. When a task is executed, the task can be associated with a device list, so that the task is executed over all devices given in the device list. Once the device list is stored, tasks can be associated at the time of execution.

Before creating a combined task, the individual tasks must be created. If there already exist individual tasks called t1 and t2, then the combined task may be created as shown in Figure 11.7.

11.2.4.3 Configuration Upload and Download

"Configuration upload" is the process of obtaining configuration data from a network device. Conversely, "Configuration download" is the configuring of a device with some supplied set of parameters.

The *DeviceConfiguration* API is used for uploading and downloading configuration information. This functionality accommodates situations in which configuration details, present in a particular device, are set over a number of other devices. Configuration details can be uploaded from a device and downloaded to other devices.

```
TaskGenerator tg=new TaskGenerator();
tg.setTaskName("combinedtask1");
tg.setNewTask(true);
tg.setOverwrite(true);

/* t1 and t2 are existing tasks */
 Vector subTask=new Vector(1);
 subTask.addElement(t1);
 subTask.addElement(t2);

/* d1 and d2 are existing devices */
 Vector subDevices-new Vector(1);
 subDevices.addElement(d1);
 subDevices.addElement(d2);

tg.setDevices(subTask, subDevices);
String taskXML=tg.getTask();
configclientAPI.executeTask(taskXML);
```

Figure 11.7 Code for creating combined task.[39]

[39] For readers who are not familiar with object-oriented programming, "tg" is assigned as the name of a new instance of the *TaskGenerator* class. The invokations of the methods of *TaskGenerator* should be self-explanatory, as their names are intuitive. The reason why we have given this example is simply to give the reader some idea of what is involved in making use of APIs with which OSS software is equipped.

The information such as attributes and the devices from which attributes must be fetched are specified in the task XML code. The getDeviceConfiguration method of the *DeviceConfiguration* API can be used for configuration upload, which gets the task XML as its argument. Invoking this method fetches values for the attributes from the devices that are specified in the task XML.

The executeTask method is used to download the collected data. This method also gets the task XML obtained using the getDeviceConfiguration method and applies the task to a set of devices.

11.2.4.4 Task DTD

An XML document can optionally have its grammar specified by a Document Type Definition (DTD). A document that is required to be of some particular type is then validated against the DTD corresponding to that type. Accordingly, there is a "Task" DTD that specifies the format of XML code which is used to define any task so that any task that generated using an API or from a client conforms to this DTD. The task DTD has several tags of which <ConfigTask> tag is at the highest level. Table 11.2 explains the parameters of this tag.

11.2.4.5 Creating Task Templates

Assuming that a task has been defined to change the system property of a device, when defining the task, the required value for the system property is specified. Once

Table 11.2 Parameters for ConfigTask DTD tag.

Tag name	Description
TaskName	The name of the task to which the record applies
IsTemplate	Specifies whether or not the task is a template task
IsNewTask	Specifies whether or not the task is new. If "true", the task will be created, and if "false", the existing task will be overwritten
IsOverWrite	Specifies whether the existing task should be overwritten if there is a task with the same name
IsSequential	Specifies whether the devices should be configured in a sequence or whether order does not matter
RollbackDocument	The name of the rollback task is specified here, so that this task will be executed if the rollback type is set to rollbackDocument
RollbackNeeded	This parameter is used to specify whether or not rollback is needed. If rollback is set to true, either the current configuration or rollbackDocument is executed
CombinedTask	This parameter is used to specify whether or not the task is a combined task. In other words, the tag specifies whether the task contains sub-tasks
Persistence	This parameter is used to specify whether or not the created task must be stored. By default, this is set to "true" and all of the tasks will be stored. If it is desirable to execute the task on the fly without storing it, then this is set to "false"
Version	The task version
Description	A description of the task
RollbackLevel	The rollback level set for a particular task. It can be a task level, device level, etc.
DeviceResult	Specifies whether or not the result of configuration of a sub-task of a combined task must immediately be sent to the clients. If "false", then the result is sent only upon completion of all sub-tasks of the combined task
InterPacketDelay	Sets delay between successive packets, when the task is submitted for execution
Encoding	The encoding which will be used to parse the task XML, such as ISO-8859-1, UTF-8, UTF-16, etc.

a task is created with particular parameters, the same task cannot be reused for another value. Template-based configuration allows for task reuse and allows variation of only the system property value.

The template takes input from the data source which forms the repository of input data. When the configuration template is executed, the values to be filled in the template are taken from the data source.

The setTemplate method of the *TaskGenerator* API class specifies whether a configuration task or template must be created. If so, then the setTaskName method of *TaskGenerator* is used to set the name of the configuration template. The getTask method of *TaskGenerator* is used to create the task XML code and the task is then stored in this format.

11.2.4.6 Source of Data for Completing Tasks

When a device is to be configured by a task, the parameter values to be assigned to the attributes must be supplied to the task before that task is executed. In the case of a template-based task, these data will be inputs to the parameterised placeholders of the template. The types of parameter that forms the data input to the templates are:

- **Inventory Inputs:** comprises data gathered from the network inventory database.
- **NE Inputs:** comprises data obtained by querying network elements for the values to be used.
- **User Inputs:** comprises values input from the user during task execution.

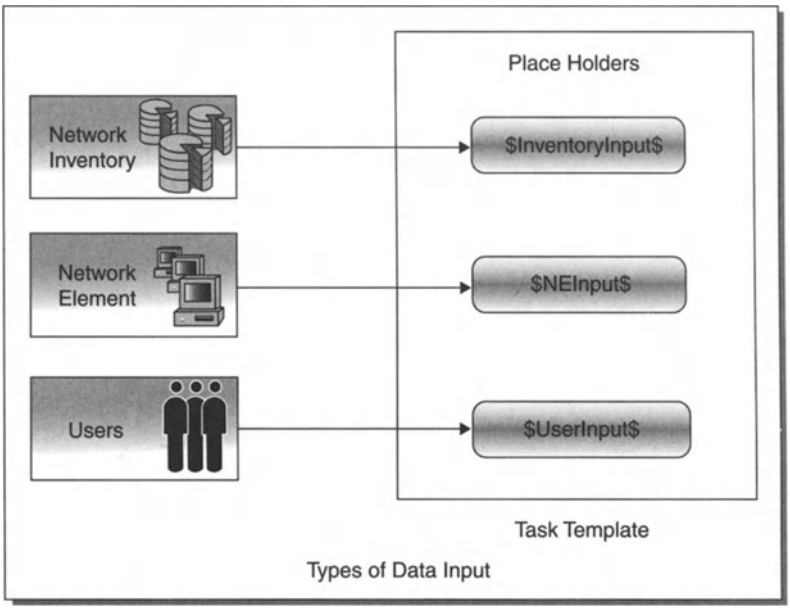


Figure 11.8 Types of input to a task template.

The input type is chosen depending on the requirements of the particular situation. For example, to achieve an automated environment, it would not make sense to require user input.

"Data source" forms the repository that supplies data to a task template. The data source defines the nature of input given to the template, inventory input, NE input or user input.

The class *DataSourceGenerator* provides methods which can be used to create the data source. The setDataSourceName and setDescription methods set the name of the data source and description. The setAssociatedTasks method sets the tasks associated with the data source. Only the tasks specified using this method can use the data source.

Once the details regarding the data source are specified, the various types of input must be added – inventory input, NE input or user input – and each type has its own method. After specifying the details, the data source XML must be created and stored.

Once the template and associated data source are created, the template is associated with the data source and the set of devices over which it must be executed. This enables the template to take the values from the data source and configure the devices. Once the data source is associated with the task template, the task can be executed.

Template-based configuration allows task reuse. A data source can be defined which will hold the values for the attributes and supply them to the template. Once a template is populated with the values provided by the data source, it is used to configure the devices in the network.

Flexibility is provided to modify the values passed from the data source to the template before it is filled. The *InventoryHandler* API is used to manage the inventory input to the template. The *getInventory* method modifies the values supplied by the data source and populates the template with the modified values. This method will return the modified inventory input to the template.

The *InventoryHandler* implementation can be invoked for implementation of any task, data source including implementation at the attribute level.

Flexibility has been provided in CMS to use a particular *InventoryHandler* instantiation merely for a specific attribute. This can be achieved by writing a class and passing that class as the parameter for the addInventoryInput method of *DataSourceGenerator*. When this is done, while being filled by the data source, values are assigned to the attributes by the task that is passed by the *InventoryHandler* instance to the addInventoryInput method.

11.2.4.7 Task Auditing

Auditing allows administrators to know about the tasks that have been executed, the devices over which the configurations have been executed, the attributes changed during the configurations and when the device configurations were executed.

It is possible to define the tasks that can be performed over any given set of devices. When the device list is selected and the task executed, each configuration that has been performed over a device is logged in the database.

There are three types of auditing:

1. Task-level auditing: shows which tasks have been performed.
2. Device-level auditing: shows which tasks have been carried out at the device level.
3. Attribute-level auditing: records details of the attributes that have been configured.

11.2.4.7.1 Audit Details

Audit details are stored in tables corresponding to each of the three types of auditing. For example, the task level audit table contains the information shown in Table 11.3.

The audit level is a measure of the type of auditing needed for administering the configuration carried out over a device:

- Level 0 means to *suppress* auditing, so that there will be no auditing of the configuration details.
- Level 1 means to audit only device and task details, so that entries are made in the TaskAudit and DeviceAudit tables only.
- Level 2 means to audit device, task and attribute details, so that entries are made in all three tables.

When a device is configured with the auditing option set, the audit details are stored in the database. These details can be accessed from the database using API methods.

When a device is configured, audit details such as the name of the task, starting time of task execution, time of finish and the user who executed the task are logged in the database for audit purposes. This auditing information may accumulate in the database so that, eventually, older information need no longer be retained in the database. The "clean audit interval" parameter deletes information older than a specified time period, that is, it sets how often we start with a clean audit database. The default value of the "clean audit interval" is 1, so that audit details older than one day are deleted from the database.

11.2.4.7.2 Auditing: Custom Views

Custom views are user-defined views that satisfy certain match criteria. Custom views can be used to limit or otherwise customise the information that is presented

Table 11.3 Task level audit table.

Value	Description
Execution ID	This is the number which will be incremented sequentially for every task submission
Username	This stores the name of the user who has executed the task
Taskname	The name of the task which was executed is stored here
Device list	This is used to store the DeviceList names over which the task was executed. The DeviceList contains a set of devices which can be stored like a Task. The list of devices can be given as comma-separated values
Data source	This is used to store the data source for the task template, i.e. the template can get its input from Inventory, network element, etc.
Execution time	This is the time at which the task was submitted for execution

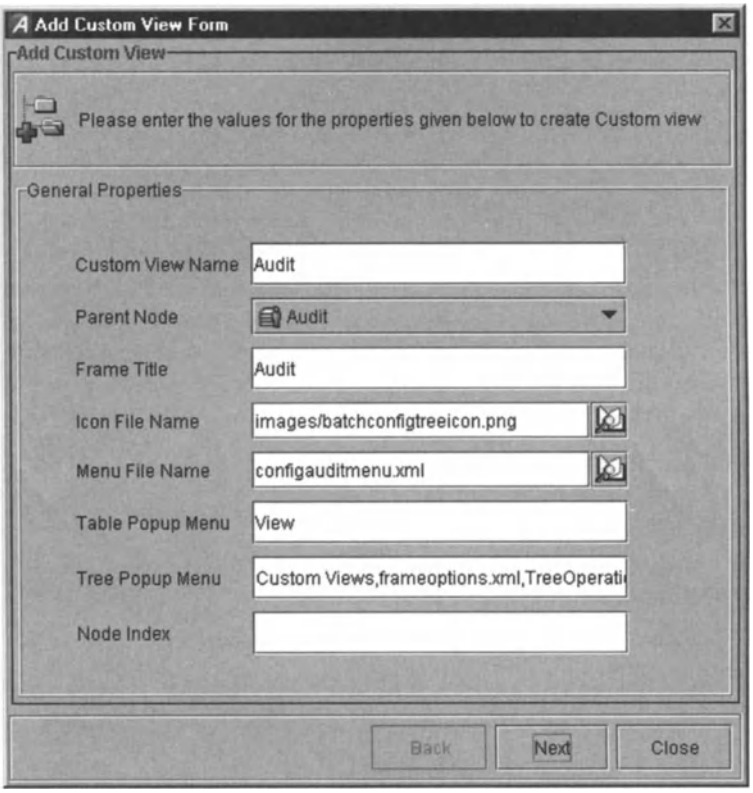


Figure 11.9 Defining a new custom view in CSM.

to the user. AdventNet's configuration management software supports custom views while viewing the auditing details. Custom view options can be selected from a drop-down menu, just as would be done for customisation of features in MS Word. The Custom Views are divided into five categories: add, remove, modify, save and rename. A new custom view is set up as shown in Figure 11.9.

Custom views are important because staff productivity can be greatly increased by customizing views in a way that is appropriate to the particular business process of a service provider.

In the form for defining custom views, details such as the name of the custom view, the *tree node* where the custom view must be created,[40] the icon to be associated for the custom view, the table and other prosaic details such as tree pop-up menu can be customised.

In the screen shown in Figure 11.10, the match criteria can be specified, so that all the objects that satisfy the match criteria fall into a category and form the custom view.

The screen in Figure 11.11 is used to select the columns that must be displayed for the newly created custom view. The user can select the required columns by

[40] Tree nodes were introduced in Chapter 4.

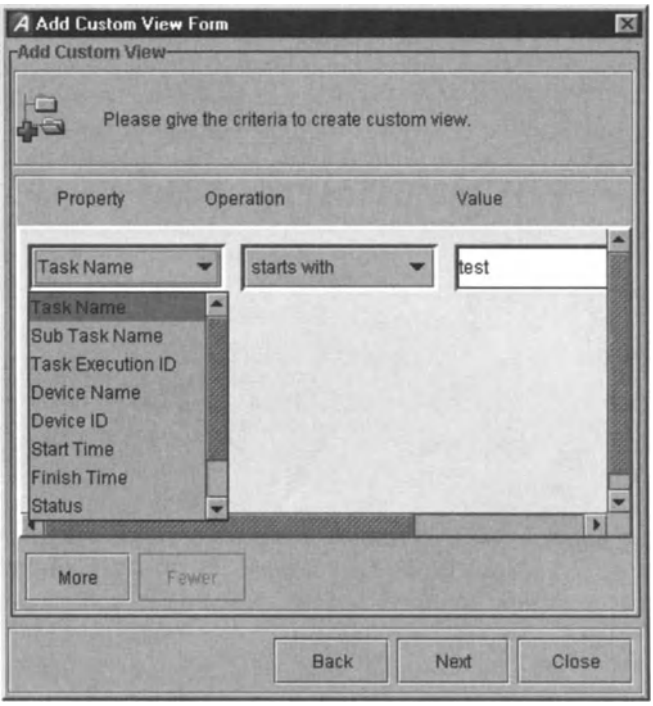


Figure 11.10 Setting up attributes for a custom view in CSM.

selecting the appropriate check boxes and the display names can be customised by entering the names in the text boxes.

Audit details can be searched from all tasks that have been executed, as shown in Figure 11.12. Specifying criteria is straightforward, with the selection of a parameter and the entering of criteria relating to that parameter which must be satisfied.

After running a query, the audit browser window is displayed, from which a task can be selected for attribute-level audit details. Attribute-level audit gives the configuration status of every attribute. The attribute details form shows the name of the Attribute, Finish Time of the task, the number of retries and the status (success or failure) of the configuration task (see Figure 11.13).

11.2.4.8 Task Manipulation

CMS allows tasks to be defined and executed over a *set* of devices. Once a task is defined and applied to the devices, a condition may arise which requires task manipulation. It is possible to modify the task before the CMS applies it to the network. For example, suppose that whenever the "sysName" attribute of a particular device is changed, it is effected by appending an identifying string, which may be the name of the company. A user's particular updates can be "plugged in" before the task is applied to the device.

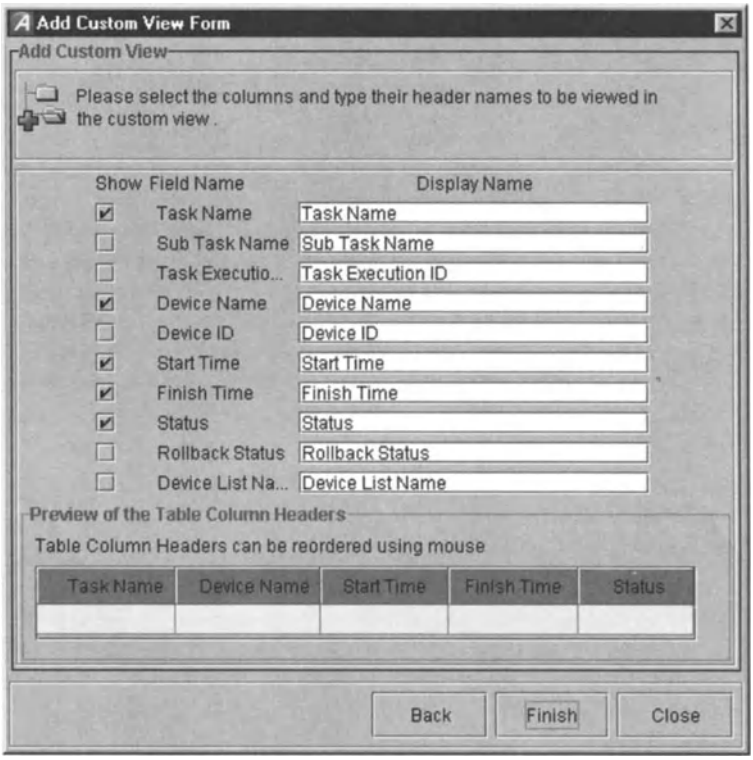


Figure 11.11 Specifying details for a custom view in CSM.

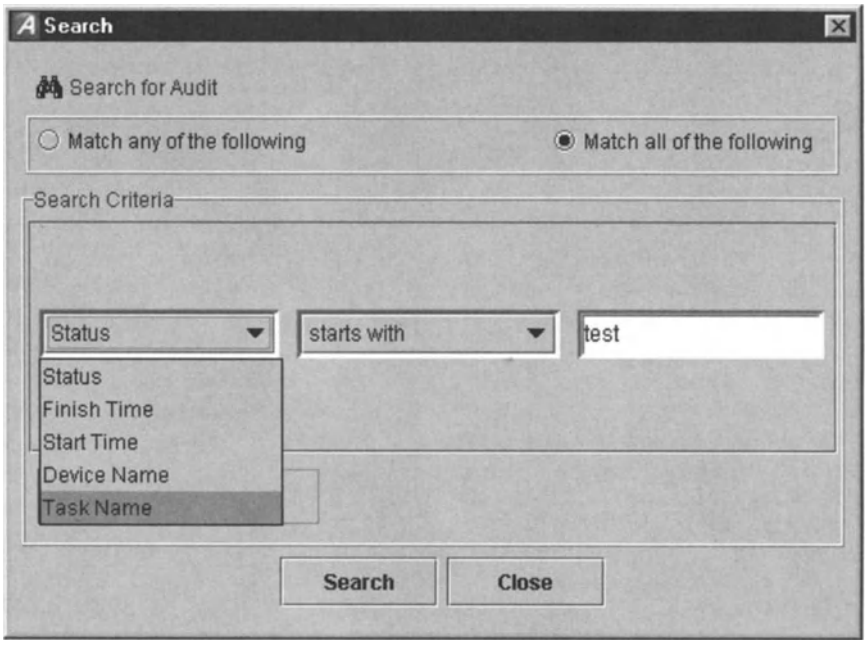


Figure 11.12 Searching for audits in CSM.

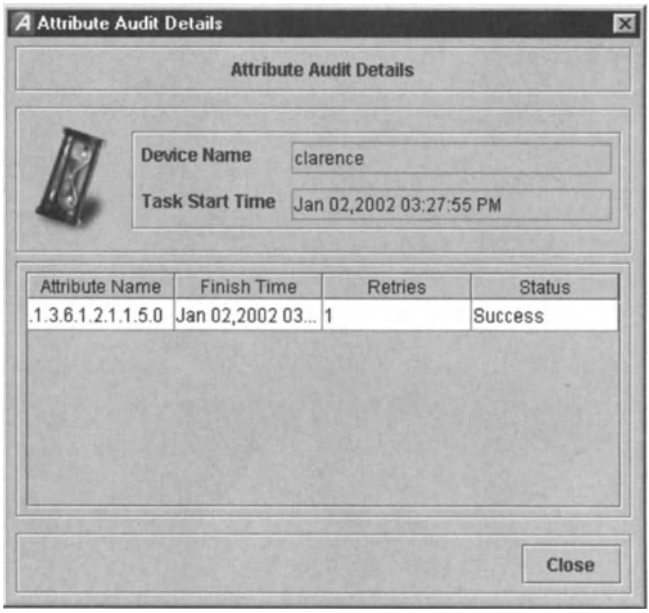


Figure 11.13 Attribute audit details in CSM.

11.2.5 Using the ConfigClient

The AdventNet implementation is fundamentally a client-server model. The CMS is the server product which complements the configuration management clients, "ConfigClients".

When the application requests a task-based operation, the ConfigClient communicates with the CMS through RMI or TCP and sends the XML code for the task. The CMS sends the result of the operation to the application through the ConfigClient after processing the task.

To perform a task-based operation, the instance of ConfigClient must be obtained. The user application must register itself with the ConfigClient to get the result of operation performed by the task. When the client is registered, the ConfigClient returns a unique ID which is used to obtain the result of the operation.

The ConfigClient class allows an applet or user application to communicate with CMS. Before that communication can begin, a particular method must be used to get the ConfigClient instance.

The *ConfigClient* class is extended by the *NMSConfigClient* and *defaultConfigClient* classes. The *NMSConfigClient* uses the *NMSSession* and the Back End–Front End (BE–FE) Client Architecture. The *defaultConfigClient* opens a new session by establishing a new socket connection between the client and CMS.

To use the *NMSConfigClient* as the implementation class, it must be run in the configuration client JVM. It is also possible to run code in a different JVM, and there is a particular class that is to be used in order to do this.

When a combined task is executed, it may be desirable to inform some users of the result of task execution. Alternatively, suppose the combined task comprises three interdependent sub-tasks, then it may then be desirable that the second task should not be executed if the first task fails, and so on. An API is provided to allow such operations.

When the sub-tasks of a combined task are executed, the result of the task is passed to a method. From there, some user-defined code can be given which is called whenever the task is executed, under certain conditions.

11.2.5.1 Handling Configuration Updates

If multiple clients are connected to the CMS, then changes made by any one client should be informed to all other clients connected to the server. Such changes include saving, deleting and executing tasks, saving and deleting device lists and saving and deleting data sources. A method is called whenever these operations are performed. Methods are available to obtain the necessary details of an update.

There is a method which obtains the update ID which returns a ConfigConstant value such as a task, device list or data source. Another method is used to obtain finer details such as whether an entity has been added, deleted or executed. Clients can be notified of whether task execution has been started or completed. When using the Save operation of a task, device list or data source, then the corresponding XML code for any of those three can also be obtained.

Suppose that two clients are logged in as the same user. If client A attempts to delete the task when client B is associating a device list to the same task, then upon deletion of the task, a message will be given to client B saying, "The task for which you are associating the device list has been deleted from the server."

11.2.5.2 Plugging in Proprietary Protocols

In a large network, many classes of device will understand different protocols. Indeed, vendors often build their own EMS with proprietary protocols. In this case, it is necessary to integrate different protocols. CMS has a facility that allows the plugging in of proprietary protocols. The protocols SNMP, TL1, CLI and TFTP are pre-integrated with CMS.

The provider class written for a specific protocol should extend a particular interface. After extending the interface, a handful of methods must be implemented and implementation details given for the protocol that is to be integrated.

11.2.5.3 Handling Client Requests

The question arises of what should be done in order to handle *proprietary* requests submitted by a client. CMS is equipped with an interface that defines and handles user-defined requests that are sent between a client and its server. A method *specifies* asynchronous requests and a different method *sends* asynchronous requests from the client to the server.

11.2.5.4 Integrating Topology Modules

The role of a topology module is to discover network resources and store them in a centralised database. Proprietary topology modules can be integrated with CM. This is sometimes necessary so that the system is aware of the devices present in the network so that devices can be configured from clients.

A configuration client must list the devices present in the network so that the devices required can be selected from the client and a device list can be made from it, in order to allow a configuration task to be applied. The configuration client should be aware of the implementation so that it can search through the topology database and list the devices for configuration. Interaction with the topology database is made possible by a class that is provided for the purpose.

11.2.6 Security Implementation

It is often not appropriate for any user to be able to make a change to network device configuration. A facility for user authentication is provided in precisely this vein. CMS is equipped with an API to allow the implementation of a custom authentication model. Even details such as the encryption mode for the username and password bytes can be customised.

CMS uses an API to perform an authorization check for saving, deleting and executing tasks, saving and deleting device lists and saving and deleting data sources. An implementation must be provided for a checkPermission method in order to provide the necessary authorization checks. If the authorization fails, the method throws an exception.

The *ConfigPermission* class provides methods which can be used for authorization, depending on the nature of the authorization scheme for any particular user. There are also classes that extend *ConfigPermission* which have been defined so that more specific methods can be accessed while checking the permission for task, device list and data source.[41]

11.2.7 Rollback

The rollback functionality allows any user-specific operation to be performed, such as restoration of older configuration data, when any single device cannot be configured. Rollback can occur for a device either by using current configuration or by using a rollback document.

Suppose a task is executed over 10 devices and the task is intended to change the location of those devices. If the task is executed, then all 10 devices will be configured so that they have the modified location value. While executing the task, if a device cannot be configured or accessed, then it may be desirable to restore the older values to all the devices. This is a case where current configuration is used.

"Current configuration" is where the server collects and stores older configuration data from the devices before configuring them. The task is then applied to the

[41] Such a class is known as a sub-class.

set of devices, either sequentially or in parallel, as specified. If configuration of any one of the devices fails, then the CMS will attempt rollback to restore the older configuration values to those devices which have already been configured. After this, the configuration of other devices in the set will not be attempted.

It may be desirable to execute a task if any of the devices refuses configuration. In this case, an existing task can be set as a "rollback document". In this scenario, CMS starts the configuration and if any device's configuration fails, further configuration ceases and the rollback document will be applied to the already configured devices.

When configuring a large number of devices, if one device fails then the rollback will be applied to all the devices. In some cases, it might be desirable only to execute the rollback for failed devices and to continue configuring other devices. For such options there are four types of rollback level: task level, device level, sub-task level and combined task level.

In a task-level rollback, if a device cannot be configured, then rollback will be carried out for all previously configured devices, including the device for which configuration failed, thus restoring the previous state. Configuration will not be attempted for the unconfigured devices.

In a device-level rollback, if a device is not configured, then rollback will only be applied to that device and configuration of the other devices will continue.

In the case of combined tasks, a task will contain many sub-tasks. When one sub-task fails, then rollback can be applied to devices in that sub-task only – and this is done using sub-task-level rollback.

Suppose that there are three sub-tasks, S1, S2 and S3, in a combined task, and after S1 is successful a device fails in sub-task S2, then the combined task level rolls back all devices in S2 in addition to S1. Sub-task S3 will not be attempted for configuration.

11.2.8 Synchronizing the Inventory Database

As in Chapter 10, we are again faced with the overlap between configuration management and inventory management, at least at the "logical" level of inventory. CMS can be used to configure a set of devices with a particular task. Once these tasks are executed over a set of devices, the information about the devices that have been configured will be stored. After configuring a device, it may be necessary to synchronise the inventory database so that it is updated with the new details.

For example, when changing the "name" property of a device, it may be necessary to update the database with this information in order to change the device properties to synchronise them with the new values. An interface has been provided for this purpose which can be used to update the managed object properties in the topology database, updating the inventory database (if any), and so on.

Once configuration or rollback has been carried out, the attribute passed is checked for the presence of the managed object name. If present, then that managed object will be updated with the current configuration details. If the attribute has no managed object name, then the managed object name of the device will be taken and updated to synchronise the topology database.

User-implemented updates can be plugged in to the database by implementing a particular interface.

11.2.9 Debugging

CMS is equipped with debugging facilities, which give details such as the username, the task executed and the time of operation. Debugging levels can be defined such that if "1" is given, then only the CMS-related debug will be switched on. If the input is "2", then more detailed debugging information will be switched on.

11.2.10 Trivial File Transfer Protocol – TFTP

TFTP is a forerunner protocol of File Transfer Protocol (FTP). The TFTP protocol is used to transfer information in both directions between a client and its server. TFTP is intended to be used when bootstrapping diskless systems are used. Devices without memory, such as routers and switches, also use TFTP in order to obtain bootstrap information from their servers.

TFTP protocol uses UDP for transferring files between server and the client, and by default uses port "69" to transfer data. When TFTP is used, the file that is being transferred is split into packets, each containing 512 bytes of data. The completion of the transfer is intimated to the receiver by sending a packet that has 0–511 bytes. The mode is one of the ASCII strings *netascii* or *octet*, in any combination of uppercase or lowercase, terminated by a byte of 0.

The denotation "*netascii*" means that the data are lines of ASCII text with each line terminated by the two-character sequence of a carriage return followed by a linefeed, called CR/LF. Both ends must convert data from this format to the format used by the local host. An "*octet*" transfer treats the data as bytes without any visibility into, or interpretation of, the content of the data. In addition, CMS has file transfer APIs that conform to the IETF's RFC 1350.

11.2.11 Server Framework

The framework services forms the nucleus of the CMS as various network management modules and applications can be built upon them (Table 11.4).

High availability is a key component of carrier-class systems and so the server framework provides fail-over capability to a hot standby server.[42]

CMS has two BE servers, one of which is designated primary and the other standby. The standby server takes over the workload when the primary server fails or is brought down by an attack or excessive load. The primary server may be brought down for reasons such as scheduled maintenance, system failure, power outage or the crash of a JVM. Fail-over is where the standby server assumes the workload of the primary server. When the primary server is running again, the workload is moved back to the primary server by a process called "failback".

In the distributed architecture, fail-over support is provided when one BE server is down and the other takes over and continues to provide the service. The fail-over

[42] Hot standby means that load can be transferred to the standby server without manual intervention and with almost no disruption to services. The reader may have heard the phrase "hot-swappable" in relation to network cards and other peripherals, similarly meaning that load can be transferred without disruption to services: at the instant that one line card ceases to function, the other comes into operation.

Table 11.4 Server framework services.

Framework service	Description
Scheduler	Used to run tasks in the context of threads at a specified time. Using scheduler services, the maximum number of threads to run the tasks can be controlled
Logging	Used by modules and applications to print log and error messages. The printing of log messages to module-specific log files, the log filename, maximum number of lines to be printed in a file and logging levels can be configured using the configuration file and APIs
Persistence	Helps in moving Java objects from memory to the database and in controlling the number of Java objects in memory
Backup	Backup service can be used to backup and restore the CMS database. It can also be configured to get executed at specified intervals. The backup and restoration process is supported for different databases, for example the backup data from Oracle can be restored to mySQL
Module	Module service provides support for adding custom modules or user-specific modules, in addition to supporting the starting of the modules in different configurations.
Communication	Communication service facilitates the communication between the Front-end (FE) and Back-end (BE) servers and also the communication between the client and the FE server. Custom modules can make use of this service via the API
Transaction API	This comprises the ability to define multiple object updates in a single operation with the ability to roll back. This feature is basic to any large scale system in order to guarantee data integrity. User applications involving a sequence of multiple object updates can access the Transaction API to perform operations such as commit, rollback and the transaction timeout
UserStorage API	User-defined objects or custom objects which do not implement *DBInterface* or extend *ManagedObject* or any of its derived classes can be stored to, and retrieved from, the product database using the *UserStorage* API

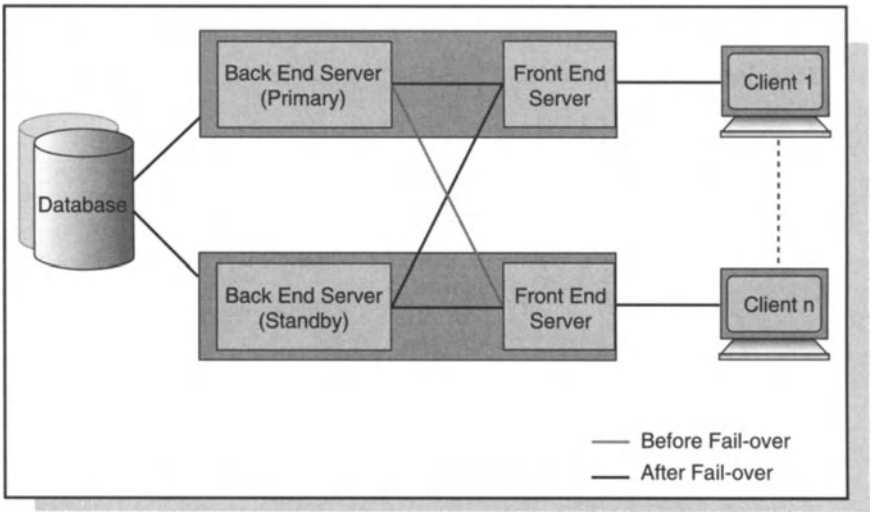


Figure 11.14 CMS framework fail-over.

process does not affect the clients connected to the FE servers, except during the brief fail-over interval when they may receive timeouts. Clients are not required to restart when fail-over occurs. An e-mail notification regarding the fail-over can be sent to a specified address, if the system is configured for this.

One of the two BE servers which is running as the primary server periodically updates the database with the current status of "alive" or "failed". The periodic interval in which the primary server must update the database regarding its presence is known as the "heart beat interval".

The standby BE server checks the database at regular intervals to ascertain whether or not the primary server is running. The periodic interval in which the standby server checks the database is the "fail-over interval". When the primary server fails, the standby server detects it and double-checks whether or not the primary BE server has failed. If the primary server fails, then the standby server takes over the role of primary server and re-starts the management services. At this moment, the FE servers switch their connection over to the standby server. Clients connected to the FE servers continue to receive service without knowledge of the fail-over.

The FE server acts as a link between the BE and the client. Whenever the server is started, one FE starts along with the BE and standalone FEs can also be started. When the client makes a "read only" request, the FE gains direct access to the database. When the client's request involves a write operation, the request is routed via the BE server.

If the FE server to which a client is connected fails, then the client will lose its connection and must reconnect to another FE server. The FE server fail-over mechanism is designed to overcome this problem by enabling the client to be connected automatically to some other FE server to ensure continuity of operation.

The FE server fail-over has been designed such that a list of FEs along with their properties is maintained in the BE and during startup all clients receive this list. The FE server properties that are maintained in the list include host name, mode of connection between the FE and client, web server port of FE, secondary port of FE and secondary port directory of FE. This information is important because there may be preferences as to which FE server the client fails-over.

The BE server maintains information about the FE server to which any given client is connected and also maintains a list of other FE servers. When the list changes, the BE notifies all clients of the change. There are two ways in which communication between an FE server and client may be disrupted, and they are dealt with in different ways:

- When a client is *disconnected* from the FE server, the BE notifies the client that it has been disconnected and provides it with a list of the other FEs which it maintains.
- When an FE server *fails*, the clients to which it is connected detects a communication failure. The client is automatically connected to some other FE server from the list whose mode of communication with the client is the same as that of the now-failed FE server.

11.2.11.1 Scheduling Services

Scheduling services allow automated network configuration processes to be scheduled. Scheduling allows services to be implemented at a desired time without

manual intervention. Of course, this can improve the efficiency of operations. Periodic scheduling invokes a particular task periodically, such as every 10 seconds. In other cases, the time periods might be much longer. For example, subscribers in the UK and other football/soccer fans around the world might sign up for a service that informs of the scoring of goals in real-time. During the normal football season, this would require the reservation of bandwidth at particular times on a weekly basis. During the World Cup, the period would be for particular hours of each day.[43]

11.2.11.2 Logging Services

Logging is used to keep track of the actions taking place in the server. When the server is started, any action occurring in the server, such as initialization, port on which the server was started, list of processes started or error messages, are recorded or "logged" in log files. The default logging parameters in the logging configuration file are:

- create the default logs directory
- direct log messages to *configout* and *configerr* files
- distinguish the file to which output messages are sent from the file to which error messages are sent
- set to "3" the default logging level for the various modules of the product.

We give some of the more important logging parameters to indicate the extent to which logging can be customised. It is only possible to report upon information that is logged which puts logging and its customization near the centre of network management. For example, the dependence of reporting on logging means that any data that are relevant to SLA management must be logged for future reference. Indeed, it is arguable that any data that are materially relevant to network performance – and so which might impact upon customer experience – should be logged for potential future reporting, in order that the carrier can correlate such things as negative customer experiences with network events. Thus, customizability of logging is important in limiting or enabling the ability of a carrier to monitor the standard of their services.

Logging parameters are as follows:

- **Filename:** Name of the file in which the log messages are to be written.
- **MaxLines:** Limits the number of lines in the log files to a desired value. When there are additional lines to be written, a new file is created, for example, configerr.txt, configerr1.txt, or configerr2.txt.
- **MaxLinesCached:** Limits the number of times the log messages are written in the specified text file. The default value for the parameter "MaxlinesCached" is 0 in the logging parameters configuration file, which means that messages are not

[43] In the second phase of the WINMAN project (to develop an integrated management system for IP-over-DWDM networks), implementing scheduling for periodic automated provisioning was a priority for just such as an application as informing fans of the scoring of football goals in real-time.

cached at all but are immediately entered into the appropriate log file. If the value were set to 100, then the first 100 messages would be collected in the cache and be logged in the corresponding text-based log file at the same time. For example, when the user seeks to print 50 messages at a time, once the cache is full with the first 50 messages then it will print or write the 50 messages in the log file and then continue to store the next 50 messages in the cache. Specifying some cache file allowance avoids frequent writing to the log file which can improve application performance and also reduce the incidence of conflict when the log file is being written to from several sources.

- **FileCount:** This is the maximum number of log files that can be written at any time.
- **LogLevel:** When a particular level is specified for messages to be recorded, the messages with levels lower than or equal to the given value will be recorded. The LogLevel parameter allows the user to choose how much detail in the messages is to be logged, and module-specific log levels can be set. The levels are as follows:
  —LogLevel = 1 indicates summary or important messages
  —LogLevel = 2 indicates intermediate or frequently generated log messages
  —LogLevel = 3 indicates verbose or error messages.
  The levels cascade, so that if a level of 3 is given, all the messages with levels 1, 2 and 3 are written.
- **LogsDirectory:** This is the name of the directory in which both system and server related log files are to be stored. The user can create sub-directories specific to a particular module under the logs directory.
- **useTimeStamp:** Setting this to true appends to the log message a timestamp of when the action or event occurred.
- **Logging:** This parameter enables/disables the logging of messages. That is, messages are recorded if and only if Logging is set to true.
- **Key name:** The Key name differentiates modules and logs module-specific messages. The Key name is also used to identify the type of message, such as output message or error message.
- **DisplayName:** The display name is a module-specific name that is appended as a prefix to the log message.

11.2.11.3 Persistence Services

The RDBMS resource pool is a store for all the connections and queries. Queries are called "prepared statements" and "statements" and these will be explained below. A central resource pool conserves system resources and provides more efficient usage of the database connections.

Connection pooling is the ability to manage the database connections from a central pool rather than leaving the connection management to each of the modules. Connection pooling enables product modules or user applications to obtain a connection from a pool of connections. This allows load balancing on the connections, rather than overloading a connection while leaving the others free. Once a connection has been created and placed in a pool, product modules or user applications

can call the getConnection method without performing the complete connection process, thus providing ready access to connections that are already open.

Any database operation performed in the product is made via the pool of connections. No connection is dedicated to a particular module. User applications which use connection pooling can call the appropriate methods available in the *ConnectionPool* class to perform specific database operations. The connection pooling architecture distributes the connections among the various processes.

A "prepared statement" is an object that represents a precompiled SQL statement. An SQL statement is pre-compiled and stored in a *PreparedStatement* object. This object can be used to execute the statement multiple times. In the transaction and non-transaction modes, prepared statements are applied to all available connections. This ensures that a connection request that is made at any time to perform some database operation has the corresponding prepared statement associated with it.

Every prepared statement has a unique ID. Each connection in a connection pool has a vector or tuple whose elements are "wrappers" over a prepared statement. Each wrapper has the prepared statement ID, the SQL string and the prepared statement itself. This allows a prepared statement to support transactions. Working with vectors of such wrappers allows any particular connection to have more than one prepared statement.

Prepared statements are created only when a request is made using a fetch method. This avoids performance degradation problems in Oracle databases, since preparing all prepared statements in advance would degrade performance.[44] In addition, prepared statements can be cached to enhance performance. The number of prepared statements maintained in the cache is configurable. Caching ensures that only some prepared statements are prepared over the connections while the rest are prepared only on an as-needed basis.

A statement represents an SQL query. Unlike a prepared statement, a statement is compiled and executed whenever it is used. Every statement applies over all connections, just as with prepared statements. The caching mechanism is also supported for statements and the cache size is configurable.

An instance of the *ConnectionPool* class is created to perform any database operation. This instance can be accessed to store any object in the database. Remote applications which need to access *ConnectionPool* must explicitly create an instance of it.

Connection pooling is supported in both transaction and non-transaction mode. The *ConnectionPool* class has a single method, and that is to get a connection.

In the transaction mode, a connection is granted when a transaction begins. Connection pooling in the transaction mode can lock and unlock connections at the appropriate times when a transaction is started and completed over a connection. Connection pooling will also lock the connection based on the thread involved in the transaction. In particular, multiple threads involved in transaction processing might request connections, wherein the connection must be allocated/de-allocated for each transaction based on availability of connections.

[44] This performance degradation is a result of "open cursor" problems.

In non-transaction mode, there is no concept of connection locking. However, prepared statements are locked when used in order to avoid conflicts which would arise from two applications simultaneously using the same prepared statement. Owing to this locking, it is important that an application does not hold a prepared statement for a long time. This is done using a configurable prepared statement timeout.

When a prepared statement is requested from the RDBMS resource pool in transaction mode, it is ascertained whether it is being called by a thread that is itself still in transaction mode. If so, then the connection pooling returns only the prepared statement that was prepared in the same connection as that used in the transaction. If a request for a prepared statement comes from a thread which is *not* in the transaction mode, then the RDBMS resource pool returns the respective prepared statement from any one of the non-transactional connections where the prepared statement is free. If none are free, then the RDBMS resource pool waits until one is returned. The prepared statement is returned either by the application once the operation is complete or it is automatically returned to the pool on timeout.

It is important that a statement requested from a thread involved in a transaction returns the statement from the same connection that is used in the transaction.

11.2.11.4 Module Services and Communication Services

User-written modules can be written and started as a separate thread from the same JVM as the server. This feature allows the user to write their own server modules and plug them into the server framework. User-written server modules can use the framework API and provide useful management information to the user-written clients.

There is communication both on the server side (between the BE and FE servers) and on the client side (between the FE servers and the Java Client). This communication takes place through a common connection which uses TCP socket or remote method invocation (RMI). User-written modules can also use this common connection through the API.

The BE server performs "core tasks" such as receiving and processing traps. The FE server deals with clients and performs "client tasks" such as servicing client requests and maintaining the state of the client with respect to the data being displayed. A BE server provides services to one or more FE servers and each FE server provides services to one or more clients.

A user-defined FE server can act as either server or client. To distinguish these, we refer to it as ServerFE when it acts as a server and as ClientFE when it acts as a client.

The user-defined client module connects to the framework client. This client in turn connects to the FE server. The FE server then creates a session dedicated to this client connection and passes the session to the user-defined *ServerFE* class. The *ServerFE* class uses the session reference to create a user-defined *sessionFE* class instance which processes the user's client requests. If requests to the client are not processed in the user-defined *sessionFE*, then they are forwarded to the BE server.

The request transferred by the user-defined *sessionFE* module, *ClientFE*, is routed to a user-defined *sessionBE* for processing. When the ClientFE connects to ServerBE,

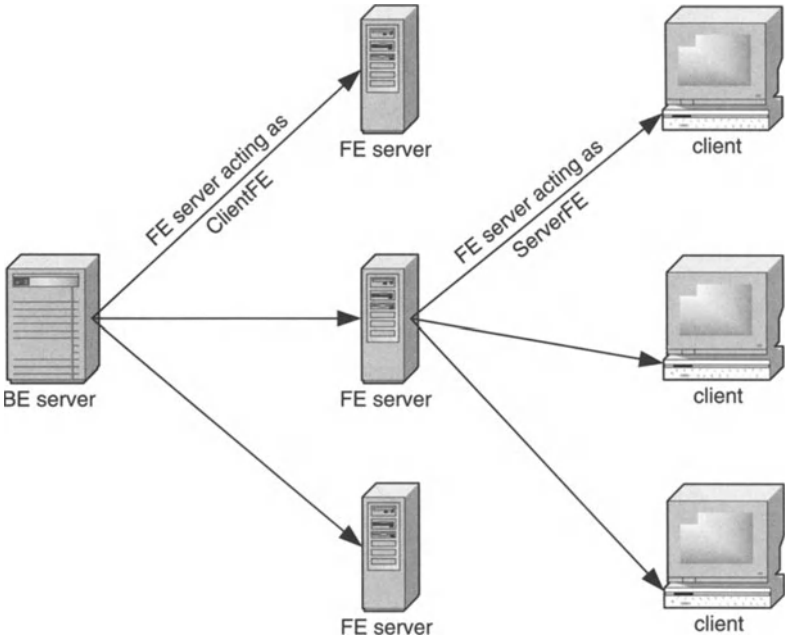


Figure 11.15 Services provided by BE and FE servers.

the *serverBE* creates a session dedicated to this ClientFE connection and passes the instance of the session to the user-defined *ServerBE* class. The user-defined *ServerBE* class using the session reference and creates a user-defined *sessionBE*, which processes the user's requests which are specified by *clientFE* (see Figure 11.16).

The transport provider interface allows users to use their own protocol implementation for communication between the client and the FE server. Thus, the communication between the client and the FE server is configurable. By implementing the following interfaces and editing the *transportProvider* configuration file, users can define their own means of transport between the client and the FE server.

The *sessionTransportProvider* API provides the basic input–output operations that can be expected by any transport protocol to transfer the data between the client and the FE server. The *TransportProvider* API creates a transport session between the client and the server, which is responsible for all aspects of communication between the client and the FE server.

A "transaction" is a single unit of work and comprises a set of operations. All the operations of a transaction are completed or none are completed. CMS provides a *Transaction* API in which application-level transactions can be provided to define multiple object updates in a single transaction. Transaction support requires updating multiple objects in one atomic operation with the ability to roll back in the event of failure of some subset of the operations in the transaction.

The CMS defaults to performing all database operations in transaction mode, although transaction mode can be disabled. Transaction support ensures that the database operations involving a sequence of operations are considered as a single

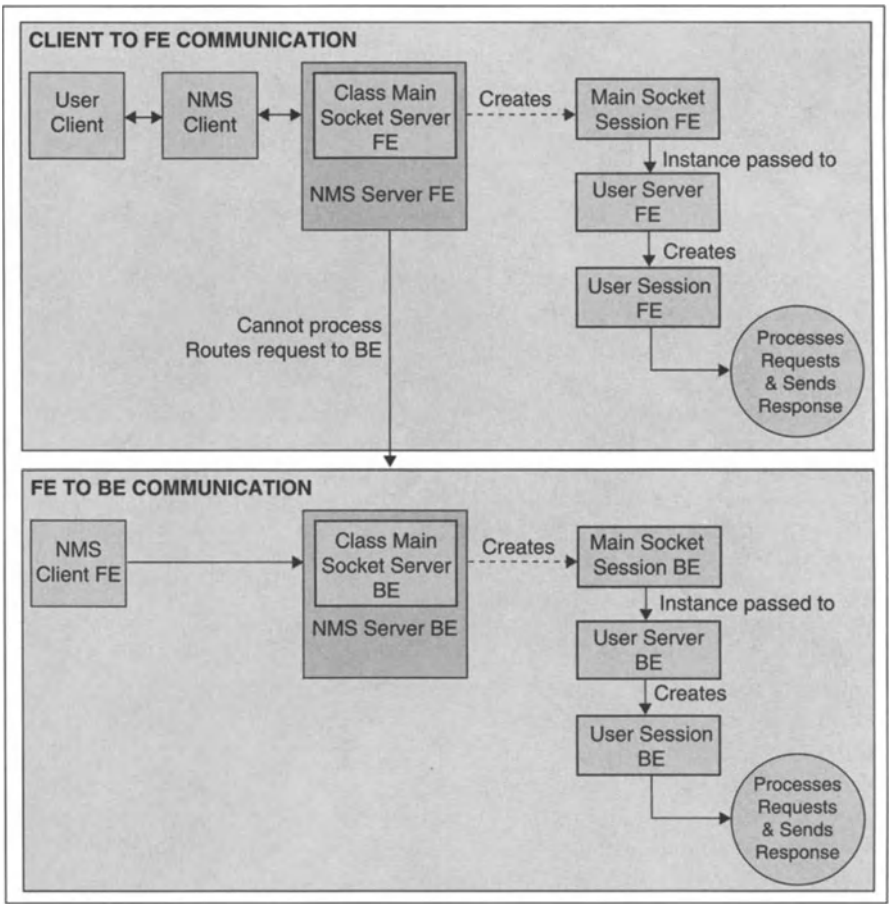


Figure 11.16 Communication between client, FE server and BE server.

unit of work. For example, consider a network which has a set of nodes and sub-networks, which in turn also have sets of nodes. In a network discovery transaction, objects are added to the database only if every network discovery operation succeeds.

Nesting a transaction makes it an operation of another transaction. This makes transactions more complex, but also allows for more possibilities in configuration management. The developer may initiate several transactions from within a transaction and demand that the transactions be carried out in a particular sequence. Indeed, some of the nested transactions may be executed concurrently. Further, transactions invoked from within a transaction may fail independently of their invoking transaction and independently of one another. This allows alternative transactions to be used in the event that one of the alternatives fail. Thus nested transactions are the basis for a general-purpose reliable programming environment in which transactions are freely composable modules.

In a nested transaction, an inner transaction can be committed only by the operation which initiated it and then the outermost commit takes effect. If an exception or timeout occurs in the inner levels of a transaction, then the entire transaction is rolled back. When an exception occurs at any point, it must be handled such that it is propagated to the highest level transaction.

User-defined objects or custom objects can be stored to, or retrieved from, the database even if they do not implement the main database API and do not extend the *ManagedObject* class. The *UserStorage* API can be used to add, delete, update and get *UserObjects* to and from the database. Since the methods are built around the database operations, this enhances the usability of the API. When the server is started, the *UserStorage* API can be accessed remotely via RMI or from the same JVM.

11.2.11.5 Severity Levels

Severity is the measure or metric of criticality used in the product to handle network management occurrences. CMS makes heavy use of severities in most of its primary management functionalities. To give an idea of the pervasiveness of severities, they are used to assign criticality to an event or alert and to show the status of a managed object or map symbol.

Given the nature and importance of severities, it is natural to allow developers to configure and customise the severities that are handled by CMS. Indeed, severities are configurable and customizable. By default, the severity level configuration file "SeverityInfo.conf" defines seven severity levels: Critical, Major, Minor, Warning, Clear, Info and Unknown.

The configuration file must contain at least four severity levels, which must include Clear, Info and Unknown. These three severities are used internally by CMS and are mandatory for its functioning. Any number of severity levels, also called severity nodes, can be added to the configuration file and each should have a unique name.

The severities defined in the configuration file are logically ordered to form a tree structure, the "severity tree". The "Clear" severity level is the root of the tree. The criticality of each severity defined is measured by its distance from the Clear node of the severity tree. The greater the distance from the Clear node, the more critical is the severity. "Critical" is the highest severity. Severities can be defined either above or below the "Clear" node but not on either side.

The *Severity* API is used to access properties of the severities from user-written code.

11.2.11.6 Front-end (FE) Server

Whenever CMS is started, one FE server is started along with the BE server. Apart from this, it is possible also to run standalone FE server(s), when there is a need for more scalability – in particular, when there is a need to connect more clients.

The FE server can be started and run in two ways, either in the same JVM as that of the BE or in a different JVM. Predictably, the flow of control between the BE and the FE is different in each case.

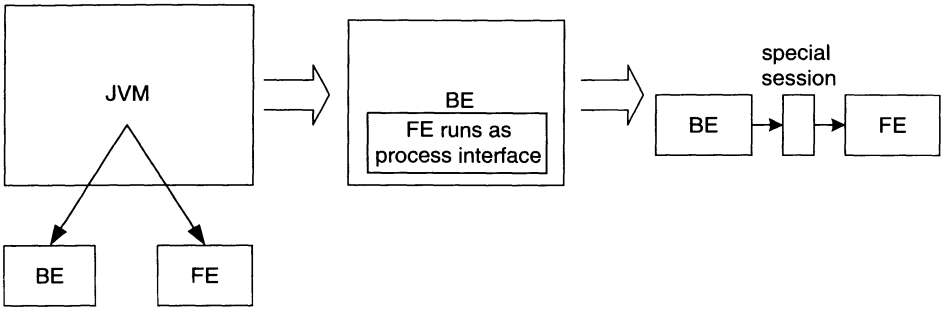


Figure 11.17 Starting BE and FE in the same JVM.

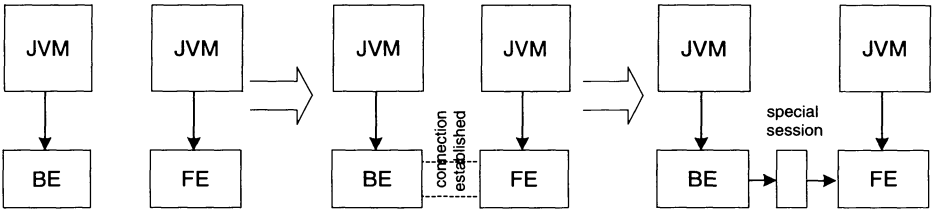


Figure 11.18 Starting BE and FE in different JVMs.

Starting the BE and FE servers in the same JVM means starting the CMS, that is the BE–FE combination, and a particular relationship is then set up between the BE and FE servers. When CMS is started, FE is started as a process within the BE so that the BE initialises the FE as a running process interface. When FE is initialised, BE creates a special session for the FE so that all communication from the FE passes to the BE through this special session.

Starting BE and FE in different JVMs means starting BE first and then connecting the FE to the BE. They can be in the same or different machines. When the FE establishes a connection with the BE, then the BE creates a session and the connection from FE is transferred to this session. Henceforth, all the communication from FE to BE takes place through this session.

All client requests are communicated only through the socket sessions to the FE server. FE categorises requests as read or write requests. All read requests are directly handled by the FE server and data are fetched from the database. The write request is passed to the BE and then the database is updated.

11.2.11.7 EJB Session Beans

Enterprise Java beans (EJB) technology is a distributed component model that enables developers to focus on solving business problems while relying on the J2EE platform to handle system-level issues. The FE server uses stateless session beans which do not maintain any information about the client session. Thus minimum server resources are used when clients are connected.

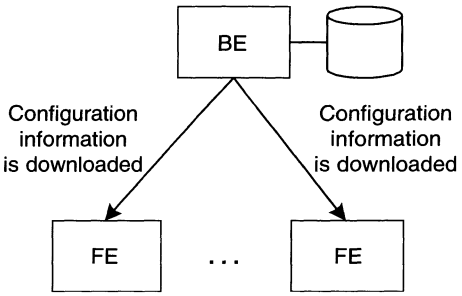


Figure 11.19 Configuration information is downloaded from the BE server to FE servers.

As the FE server's session beans comply with J2EE standards, they can be deployed in a J2EE application server such as WebLogic, JBoss or WebSphere.

The configuration management FE server can be configured to use the session beans managed by the EJB Container. When the FE server uses the session beans from the EJB container, the server is said to be running in "EJB mode". In this mode, the session beans are deployed in the EJB container of any of the application servers and the EJB client looks up these beans during the normal functioning of the EJB client.

This client is referred to as an EJB client that looks up the session beans from the application server using the Java Naming and Directory Interface (JNDI) name. The operator starts the FE server in EJB mode and connects the EJB clients.

11.2.11.8 Dynamic Downloading of Configuration Files

In a distributed environment, where we have a single BE server and multiple FE servers, it may be necessary to change the configuration of a number of FE servers. Rather than reconfiguring FE servers one-by-one, it will be easier to make the changes in a centralised location and for the change to be propagated across all FE servers and their clients. To allow this, the configuration information of the FE server and its client is held only in the BE server. This information is downloaded to the FE server when the FE server is connected to the BE server and when the client is connected to the FE server.

11.2.11.9 Proxy APIs

Proxy APIs are the Remote Method Invocation (RMI) APIs bound in the FE server with the implementation for the same interfaces as that in BE server's RMI registry. Proxy APIs serve three main purposes:

1. To overcome the security restriction of the applet used by the client. The Java-implemented security policy may restrict a client applet's RMI look-up capabilities to a host other than the applet's code base. In that case, the applet in the client that is downloaded from the FE server would be unable to look up the APIs in the BE server if the BE server is running in a different machine from the FE server. The applet can get handles for RMI APIs in the FE server. It is then useful

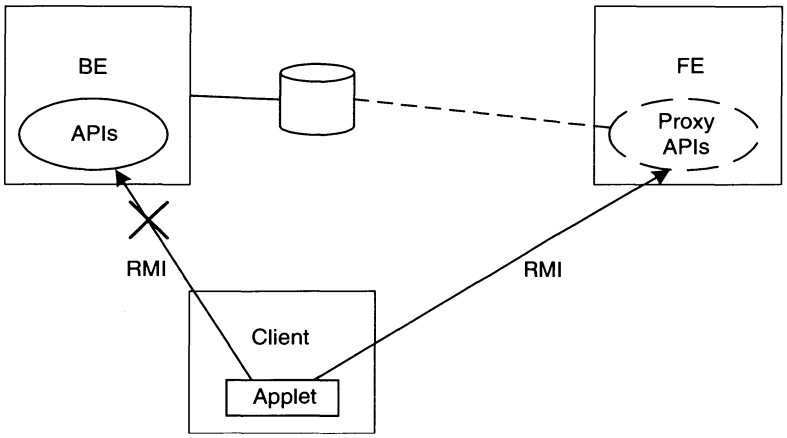


Figure 11.20 Client applet invokes methods from proxy APIs.

for proxy APIs with methods that are equivalent to the APIs in BE to be provided in the FE server. The client applet may then invoke methods in the FE and achieve the desired functions.

2. For any read operation, the client's request to the proxy API is sent directly to the database. This reduces the load on the BE server.
3. Since all proxy APIs implement the same interface as that in the BE, it is irrelevant to applications using these APIs whether the APIs are from the BE or FE. Therefore, such an application can safely use the same interface for holding the reference irrespective of the server to which it is connected.

11.2.11.10 Utility APIs Used in FE Server

The client to communicate with the FE server uses some utility APIs provided in the FE server. The categories of utility APIs are as follows:

- APIs bound in RMI extend the remote interface and are bound in the RMI registry of the FE server.
- The JDBC API has methods for getting database-related information such as column names of table and aliases of property names used in tables.
- The Log API can be used to log some informatory messages/errors in the FE server's log files from an application running in a separate JVM. This will be helpful in centralizing logging services and managing log messages. If some information about errors that occurred in clients are logged in the FE server, then it will be easy to track the problem.
- The generic FE API has some general methods managing the FE server. For example, the getActiveUsers method returns a list of users who are currently logged into that FE server.
- Thin APIs or lightweight APIs have methods which use Java objects rather than objects of CMS. These APIs are provided for different modules and are used by corresponding session beans for their functioning.

11.2.12 Client Framework

The Java client framework provides a customizable infrastructure that can be used to develop and integrate user-defined applications. The Java client runs both in a client browser and as a standalone application. The GUI framework of the client is driven by a set of configuration files (written in XML) on a per-user basis. Applications can extend the client with new screens and panels for specific information needs.

The client framework makes it easy to customise client parameters, such as fonts, icons, status bar, the client tree and the panel settings by configuring the respective configuration files. The client framework also supports client-side internationalization so that clients that are customised in different locales can be connected to a single server.

11.2.12.1 Client Configuration Files

Various facilities such as user configuration of servers and client operations, configuration of the mode of data storage and web server administration are accomplished through configuration files.

The icon objects, menus, both frame-specific and panel-specific, toolbars, fonts, internal frames, MIB parameters and so on can be configured by modifying the appropriate configuration file.

For example, a configuration file is used to specify the MIBs to be loaded by the server. The list of MIBs specified is displayed in the "load MIBs" list box of the respective user interface. The user can choose the MIB from the listbox and load it such that a tree view of that MIB is created under the MIB manager in the Java client. If nothing is manually selected then, by default, the IETF's RFC1213-MIB is displayed in the load MIB list box of the MIB manager.

11.2.12.2 Customizing Default Builder Projects

In a Java client, the following are integrated:

- the bean builder management application development framework
- the configuration management user interfaces
- security administration
- runtime administration.

These applications have been built and integrated into the Java client using the Network Management System Archive (NAR) mechanism.

The batch configuration user interface is invoked by selecting the "Batch Configuration" node under the "Configuration Management" node in the client tree. Batch configuration helps configure devices to requirements by defining tasks. All user interfaces in the batch configuration module are specified using the bean builder. Hence the user can customise user interfaces by loading them in the builder.

Batch configuration deals with configuring network devices at runtime by defining tasks and subsequently executing them on the required devices. Requests between client and server are handled by a separate session called *ConfigSession*. This session

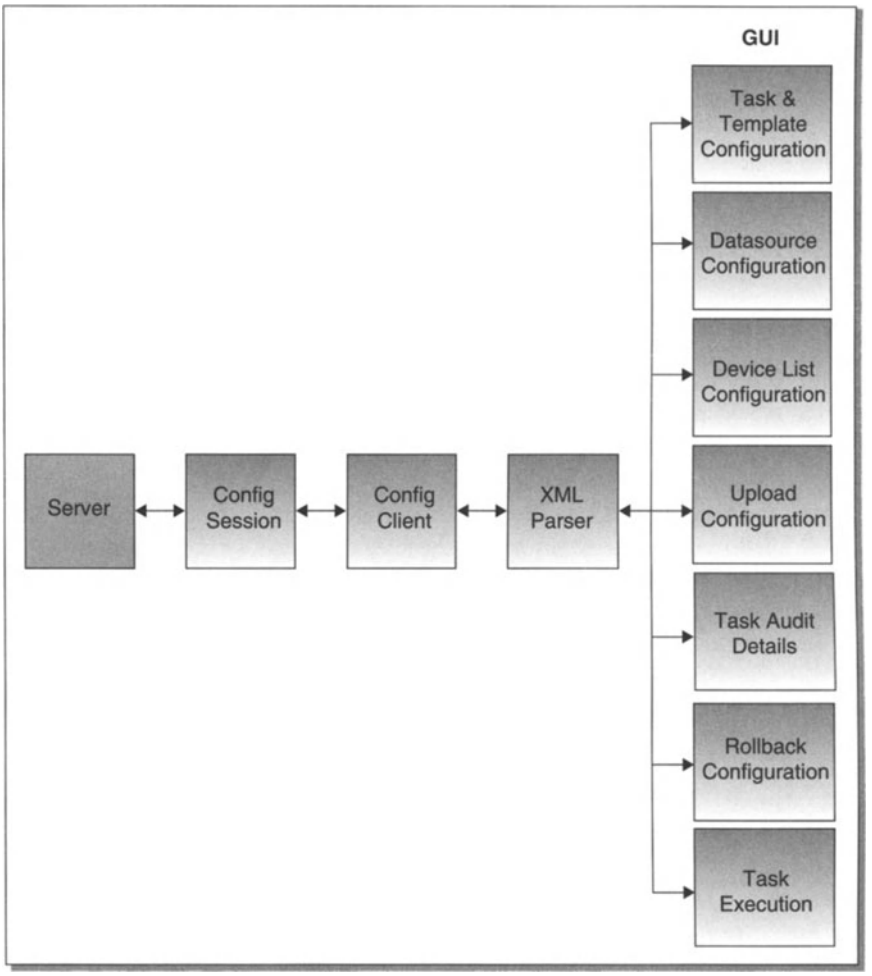


Figure 11.21 Batch configuration architecture.

is responsible for communication with the CMS and is started at the same time as the server. The tasks created are stored in the database. All tasks currently configured in the server are displayed when the user selects the "Batch Configuration" node in the client tree. The user can perform task-related operations using the "Configuration Operations" menu.

Communication with the server is handled by *ConfigSession*, which is initiated when the client appears. All classes which need to communicate with the CMS implement a particular common interface.

The batch configuration module is built on the model/view/controller (MVC) architecture. Figure 11.21 illustrates the batch configuration architecture.

The batch configuration user interface is available as a customizable and pluggable application. The module can be customised using a bean builder IDE.

11.2.12.3 Framework API

The NMS user interface API class provides APIs for handling custom views and client tree functionalities from within the Java client framework. These API methods can be accessed only within the same JVM as that of the Java client. The NMS user interface API provides many static methods which can be used to handle various custom view functionalities and perform tree operations.

The custom view APIs can be used for creating custom views under a specified parent node of any panel. Operations such as adding, modifying, removing and renaming custom views can be performed using methods of this API. There are also methods for creating temporary custom views.

Client tree node operations can be performed through the methods of the NMS user interface API. Operations include adding, modifying, moving and removing nodes in the client tree.

The NMS user interface API also provides other useful user interface-related utility methods that can be used by application programs within the client JVM. Utility methods are related to internal frame handling, cursor handling and panel handling. In addition to creating custom views through the Java client user interface and through static entries in the *tree* XML file, it is possible to create and handle custom views. This way of handling custom views has advantages over others in that the custom view operation can be invoked from any user interface component (menu, toolbar, etc.) of the client. The custom view that is added is stored in the database so providing a view for all clients connected to the same server.

It was mentioned above that temporary custom views could be added. These provide the flexibility to create and view custom views for the desired client without displaying them for *all* connected clients. Temporary custom views can be created which are viewable in the client without a corresponding node being added in the client tree.

The client can be customised. Indeed, the user can build their own client while still being able to use the product session's framework on the FE server side. The *ClientFramework* API is used to create connections and instantiate sessions, with the main functionalities of authenticating the user, and creating the main socket connection and instantiating the sessions.

The Java client uses the utility API to create a connection to the server framework. Sessions will then be instantiated through main socket connection. The *ClientFramework* API initialises the common and generic socket connections. The constructors of this class are used to initialise the socket client. When using a custom client, the *ClientFramework* API acts as an initialiser for client–server communication. It authenticates by checking username and password and then initialises the common socket connection and the GenericSocket connection.

Communication between the server side (FE server versus BE server) and the client side (FE server versus Java client) occurs through a common connection which uses either TCP socket or RMI. User-written modules can also use the common connection.

11.2.13 NAR Packager

Any application developed in AdventNet's Management Builder development environment can be integrated with CMS using the (NAR) file. The NAR file contains the

resources and details that are needed for integration with CMS. It is beyond the scope of this book to explain how this works.

11.2.14 MIB Browser

CMS comes equipped with an SNMP MIB browser module. This module permits loading of MIBs, MIB browsing, walking through the MIB tree, searching MIBs and performing all other SNMP-related functions. It is possible to view and manipulate the data available through an SNMP agent in a managed device through the MIB browser. More details on the MIB browser are given in Chapter 4.

11.2.15 Working with Batch Configuration

Task information includes details such as task name, description, protocol used by the task and task attributes. Task details are viewed in the Batch Configuration panel as shown in Figure 11.22.

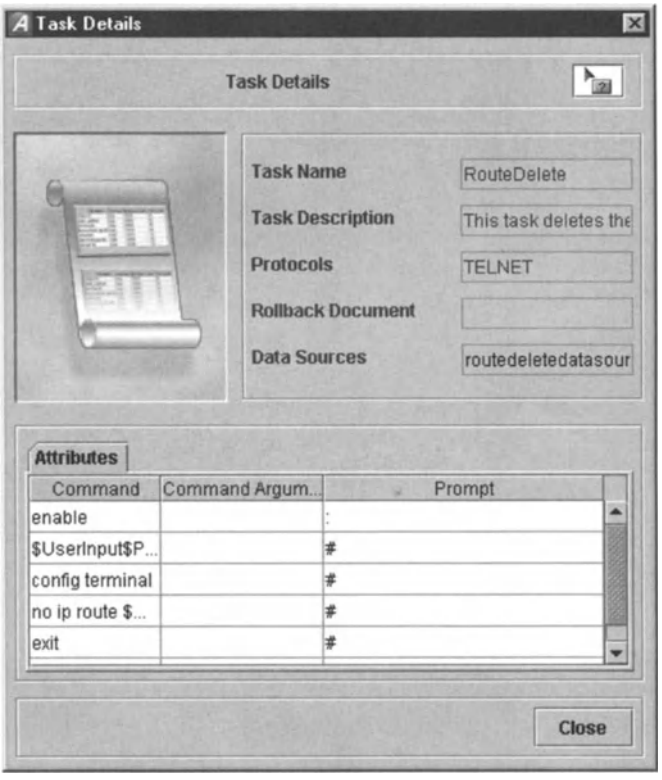


Figure 11.22 Defining details of a new task in CSM.

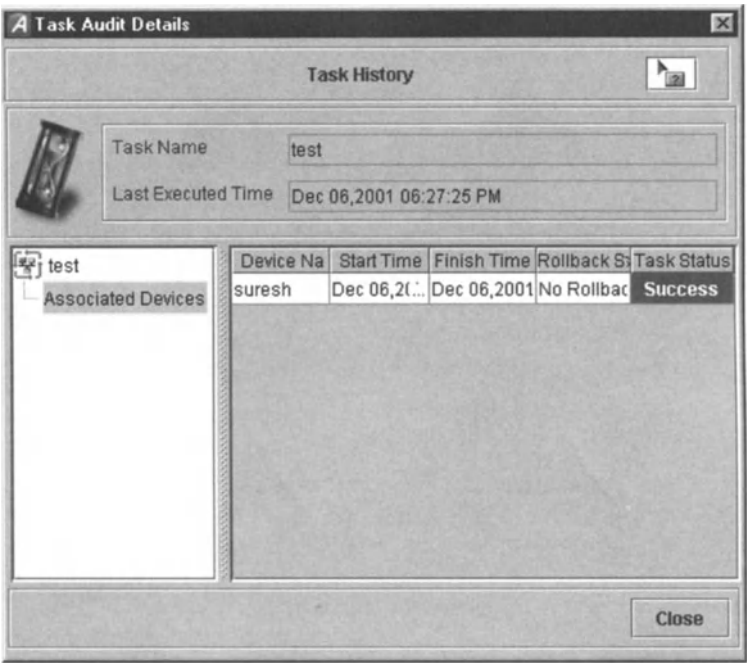


Figure 11.23 Task audit details or task history.

11.2.15.1 Viewing Task History

We can differentiate between a task defined in the abstract and an instance of a task that was actually executed. The history of an executed task includes time of execution, the set of devices over which it was executed, its rollback status and other details which are generally regarded as audit details. In CMS, the audit details of selected tasks are as in Figure 11.23.

The screen in Figure 11.23 exhibits a one-to-many relationship between a defined task and the set of instances of executions of that task. Start and finish times are shown, success or failure of the task execution on each relevant device, and so on. In the example shown, there is only one device called "suresh" associated with the task "test". The task execution instance shown in the *last* or most recent execution of the task "test".

11.2.15.2 Executing a Single Task

Once a task is defined with a set of attributes and values assigned to them, it can be executed on one or more devices. The devices to be configured are defined by a device list but can also be selected individually before executing the task. Whether the latter is possible depends upon how the task was defined.

Device lists comprise devices which are selected based on some criteria. It is necessary that device attributes must be configured before a task can be executed since

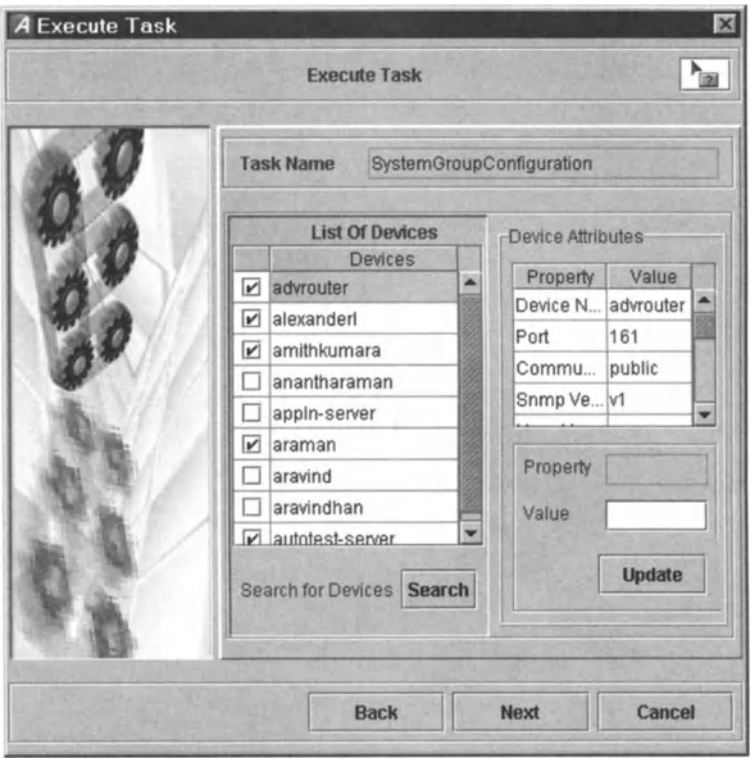


Figure 11.24 Specifying devices on which a task will act.

devices are the locus of task execution. Selection of devices on which a task can be executed is shown in Figure 11.24. The attributes of a selected device are shown in the right pane and device attributes can be added and modified in the lower right pane.

11.2.15.3 Executing Tasks as Templates

As with "normal tasks", the devices that are to be configured by template tasks can also be defined under a device list or be selected individually.

For template tasks, values are assigned for the place holders when the data source is configured. Data sources are created by assigning a set of values to the place holders in the task template. Only one data source can be associated at the time of execution of a template. Existing data sources are displayed in the task execution wizard. Wizards improve usability, especially for first-time users, but are not provided in all current OSS software. The data sources are displayed in the left column of the screen and each data source's defined place holder on the right side of the screen. Values for configured place holders can be entered as data source parameters in a screen such as that in Figure 11.25.

Figure 11.25 Configuring data source parameters.

11.2.15.4 Executing Combined Tasks

When two or more configured tasks, possibly based on different protocols, are to be executed at a particular time, then they can be combined together to form "combined tasks". Every configured task which is listed as a sub-task under a combined task can be executed on separate devices. There is a wizard for executing combined tasks (see Figure 11.26).

As seen in Figure 11.26, the sub-tasks that are present in a combined task are displayed under their respective protocols in the tree.

11.3 Configuration Management Implementation

We now turn to configuration management as implemented by Clarity's Configuration Manager product. Clarity Configuration Manager (CCM) manages telephone number creation and allocation, in addition to supporting the logical and physical configuration of telecommunication networks. The product integrates NEs and network element management systems (NEMSs). With appropriate interfaces, the product can communicate with other support systems such as materials man-

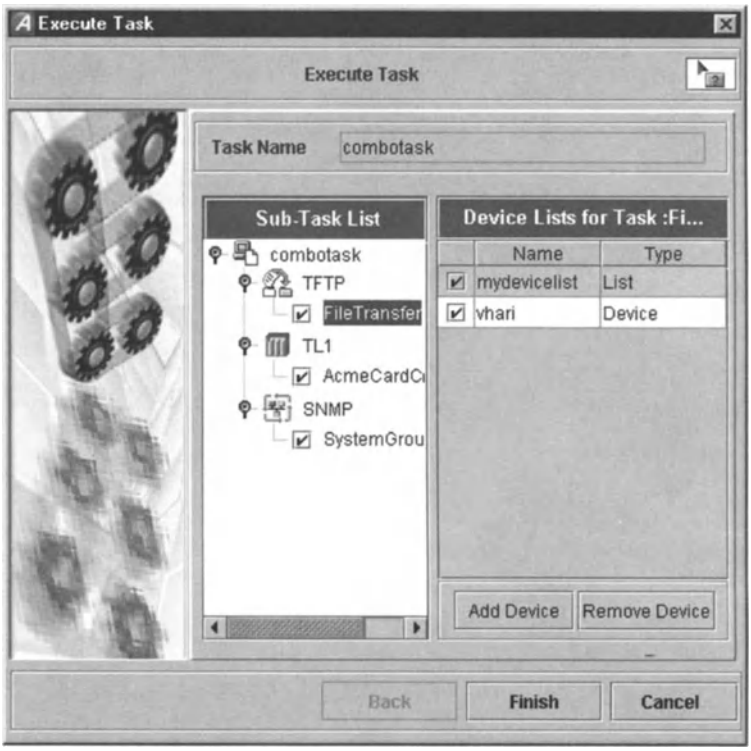


Figure 11.26 Task execution wizard in CSM.

agement, GIS and physical inventory tracking systems. Such physical inventory includes customer, carrier, common and interconnect sites, manholes and network access points and wireline and wireless transmission infrastructure.

CCM is used to create and modify circuits and the circuit details that are used in a network. The text-based version of the circuit editing screen is shown in Figure 11.27.

The circuit information includes the following:

- a unique ITU-T compliant circuit name
- the service order number concatenated with the service order revision number
- the service type, that is, the name of the service that is supplied on the selected circuit
- the status of the circuit
- the customer who owns or is leasing the circuit
- the customer account number to which the circuit is billed
- the date the circuit was put in service or is expected to be put in service
- the guaranteed speed of operation for the service
- the circuit type, child or bearer
- the person who designed or created the circuit including the NE interconnections
- whether alarms are to be checked as part of a query
- the A and B rings if the circuit is to be used to connect two rings
- comments that can be entered in a free text field.

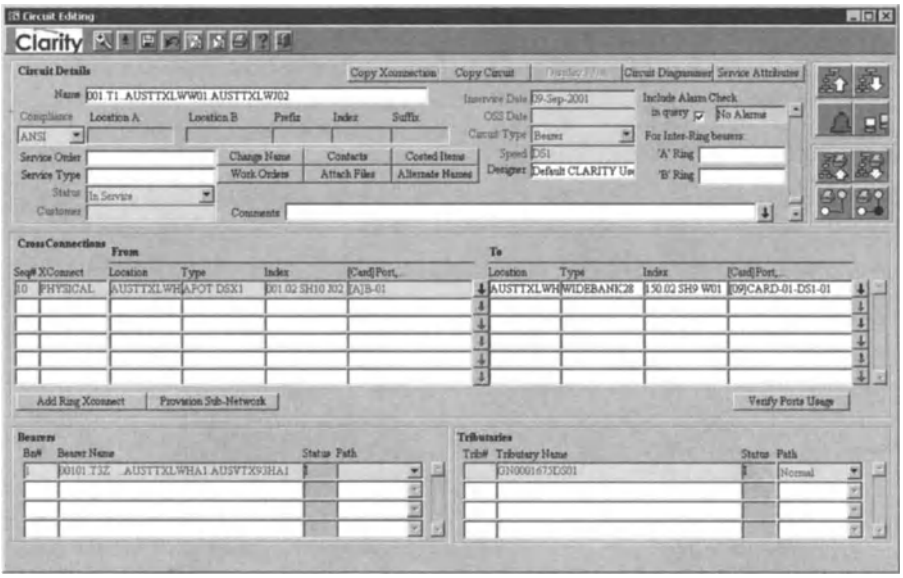


Figure 11.27 Circuit editing.

The cross connections subform displays the cross connections for the circuit whose record is displayed. The subform lists essential information about the cross connection such as the position of the cross connection in sequence (sequence number), the cross connection type, such as drop insert or patch, the source and destination NE's location, type, instance and the port name of the terminating card.

From the circuit editing screen, the user can move directly to the screen for adding a ring cross connect, provision a sub-network or provision a broadband service.

The bearers section of the screen shows all the next level bearers that carry the selected circuit. Information about the bearer includes the position of the bearer in sequence – "sequence number" – the name of the bearer, the status of the bearer (planning stage, in-service or out-of-service) and the path for which the bearer is used. The tributaries section is similar.

The circuit editing screen details assembly models. Assembly details include its unique identifying name, its status (proposed, active or inactive) and a description. The assembly endpoints and internal assembly links must be selected for an assembly, although the assembly links must already have been created in the system. Among the important features of the assemblies functionality is the ability to bulk load and automatically create the physical connectivity between NEs – frames and equipment. Assemblies become available as abstract capacity during the circuit design and provisioning process. Provisioning staff can select an entry point to an assembly and then order the system to assign capacity automatically throughout the assembly.

Parent connections must be defined for each assembly. Defining a parent connection requires the sequence number of the parent connection and the relevant the

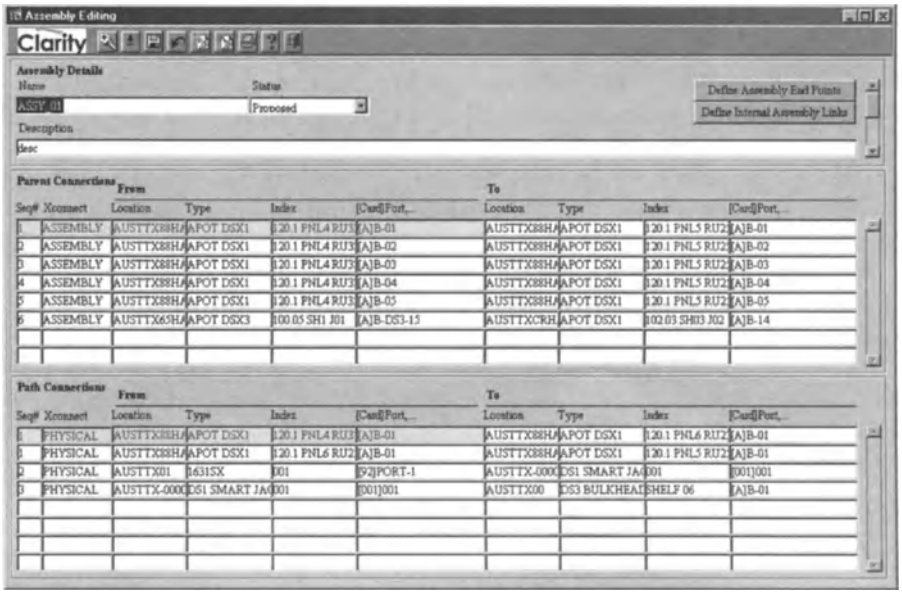


Figure 11.28 Circuit assembly editing.

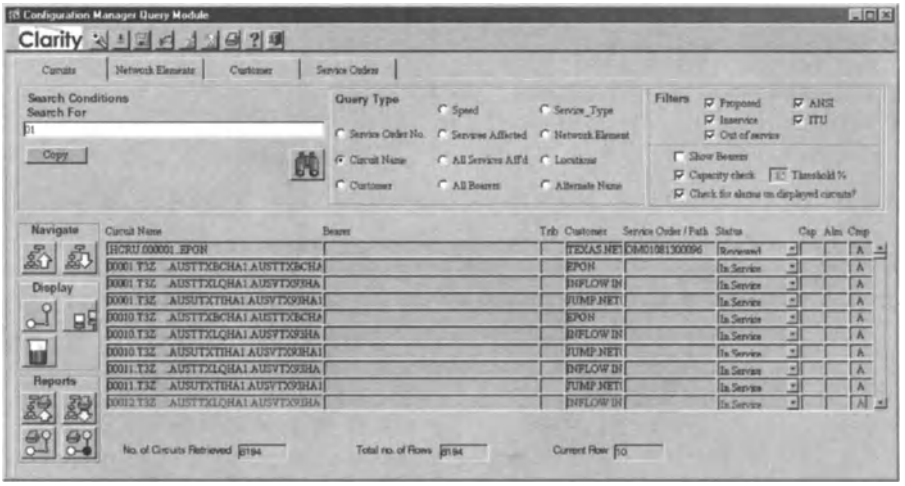


Figure 11.29 Circuit query.

cross connection. Next, the source and destination of the parent connection must be defined. This involves defining, for each end, the location, type, instance (given by an index number) and the terminating card port name of the network element. Path connections are also given for any selected assembly and these are detailed in the same way as the parent connections. The assembly editing screen is given in Figure 11.28.

The circuit query screen is given in Figure 11.29.

Circuits can be queried by service ID for any service that the circuit supports, circuit name, customers which the circuit services, any services that are *affected* by – or are dependent upon – the circuit, bearers, NE and/or location. To narrow a search for circuits, additional criteria that can be specified such as the status of the circuit – proposed, reviewed, confirmed, whether the circuit has been commissioned – which would mean that the circuit is "in service", bearers for the circuit, and alarms that are affecting the displayed circuit. A circuit query will return the name of the circuit, bearers on the circuit, the circuit's tributary number – if it is a tributary, the customer to which the circuit provides service, the service identification of the circuit, the status of the circuit and, finally, whether or not the circuit has recently been affected by an alarm.

Circuit capacity details the spare bearer capacity, and spare configured ports can be searched. The cards and slots, parent ports and child ports can be viewed for each spare configured port for any given circuit. The circuit capacity screen provides the following information:

- circuit name
- service order identification number
- speed
- sum of the speeds of its child circuits – in other words, the total speed of its tributaries
- whether or not the speed of the tributaries is more than 80% of the bearer speed – indicating whether the bearer is close to being overloaded
- circuit name of any child circuits or tributaries
- sequence number of each child in the bearer
- terminating equipment at the A and B ends of the circuit
- port and card names at the A and B ends of the circuit.

It is also possible to zoom up to bearer/parent circuits and to zoom down to tributary/child circuits.

From a circuit query, it is possible to begin editing the circuit – refer to the above circuit editing screen – or to open details on an NE in the circuit (see Figure 11.30).

Ring cross connections are an essential component of circuits, as explained in Chapter 2. CCM is equipped with a text-based screen for defining cross connections (see Figure 11.31).

Defining a ring cross connection requires the location at which the ring is to be entered, the type of NE at which the ring is to be entered, the starting point of the cross connection, the index/instance of the NE at which the ring is to be entered, the starting point of the cross connection, the port which indicates the starting point of the cross connection and the cross connection type. The same information must again be given for the endpoint of the cross connection. The direction that will be followed around the ring must also be specified.

In editing a circuit, it may be necessary to define a route for the circuit through a sub-network. A sub-network is a collection of circuits that are connected by appropriate NEs. It is, literally, a network that is a subset of some entire network of a carrier. It is possible to provision a route through a sub-network so that the route forms part of a circuit (see Figure 11.32).

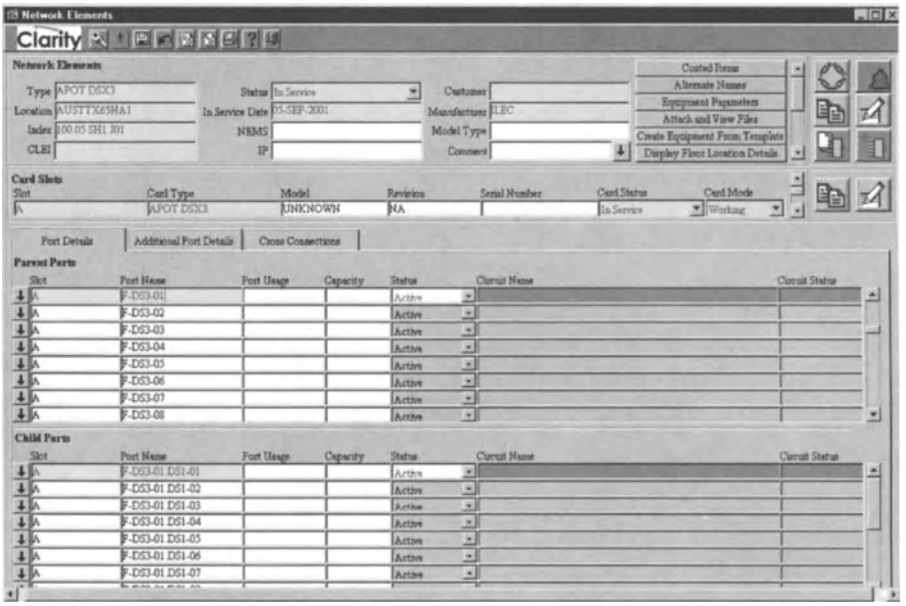


Figure 11.30 Network elements.

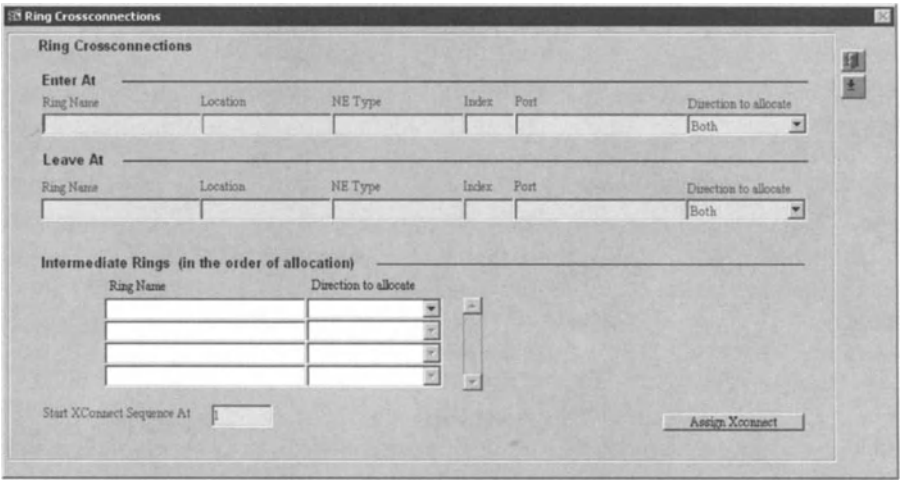


Figure 11.31 Ring cross connections.

The preferred route through the sub-network must be named and the point at which the sub-network entered, also called the A end. The A end is uniquely specified by providing location, NE type, index/instance of the NE, the port line number on the NE, the starting timeslot, the number of timeslots, the logical name of the A end port and the data link connection identifier name. The corresponding information must be specified for B end, where the traffic will exit the sub-network.

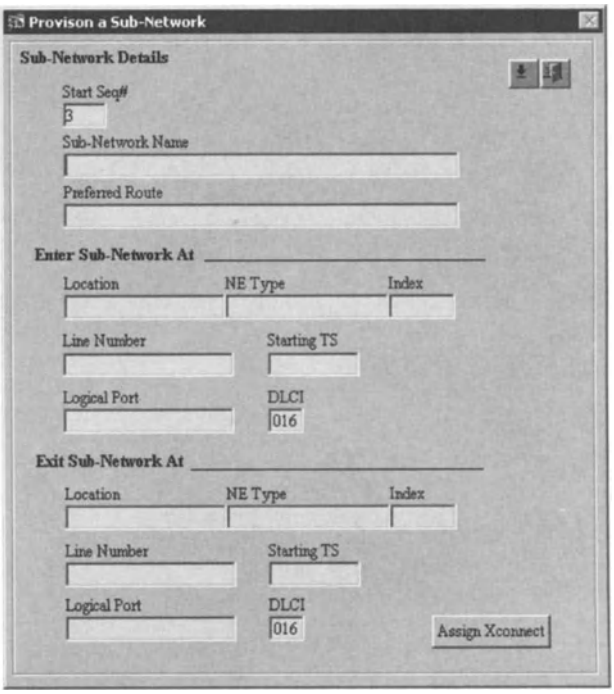


Figure 11.32 Provisioning a sub-network.

It may be necessary for any circuit to set up automatic generation of customer drops. A customer drop is a connection from a customer site to the telecom network.[45] Thus, the system should be instructed to allocate the next available connection route through the network based on preset design rules. The service technology is specified in order to dictate the connection rules. CCM forces the selection of video drop, telephony drop or data drop, although these can be customised at implementation either by Clarity technicians or the system integrator in charge of the implementation. It would also be easy to give a carrier the ability to enter entirely new service technologies. The usual details must be given for the start and endpoints of any customer drop.

Service attributes can also be associated with a circuit. The cross connection details include the name of the circuit that owns the currently selected cross connection, the cross connection sequence number and the network element type. CCM allows the creation of templates so that preset cross connection values can be loaded.

The port parameters screen is used to define port attributes. It is also possible to load predefined values from a technology template. A port parameter includes an attribute name, a data type (alphabetic, character, numeric), the value of the service attribute, the person that defined the attribute (customer, engineer or system) and the optionality of the attribute (whether it *is* optional or mandatory).

[45] Specifically, a customer drop connects the customer's terminal to the distribution box.

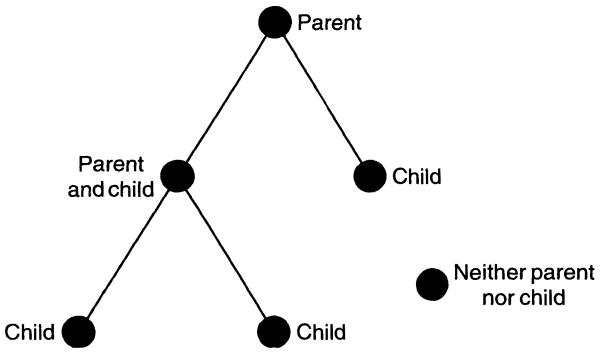


Figure 11.33 Port inheritance graph.

The NE screen is shown in Figure 11.30. With NEs, CCM allows:

- Querying: including querying of port templates and alarms, while the latter shows alarms which affect the current NE with alarm type, port number, data and time of alarm report, alarm priority and alarm severity.
- Creation of records: network element records, card records and port records, requiring a slot, card type, card status, and all ports and child ports; it is mandatory to enter the NE type, index, status, in-service date, customer and manufacturer.
- Modifying and deleting records
- Creating and applying port templates.

Troubleshooting is a large part of the role of any staff at the NOC. Effective troubleshooting is first a matter of drilling down to understand precisely where a problem might be arising. CCM provides this drilldown ability from the NE screen. In particular, the following additional screens can be opened in the context of the NE whose record is currently displayed:

- alarms which affect an NE
- port templates
- adding ports from a template, which requires the name of the port breakdown template, the parent port for the ports in the breakdown template, the port slot number for the card slot on the NE and the child port prefix for the ports that will be created
- cross connections, which include the circuit name and card port/slots included in the cross connection
- associated SDH rings.

In CCM, the tree structure of ports allows the possibility of parent only, parent and child, child only, or neither parent nor child (a lone node). When defining ports, it is essential to have a mental picture of the tree or hierarchical structure that is appropriate to one's purpose (see Figure 11.33).

Whereas the AdventNet product provides the concept of templates for configuration tasks, the Clarity product provides templates for defining NEs. In particular, in CCM, a port template can be used to create ports quickly and even automatically.

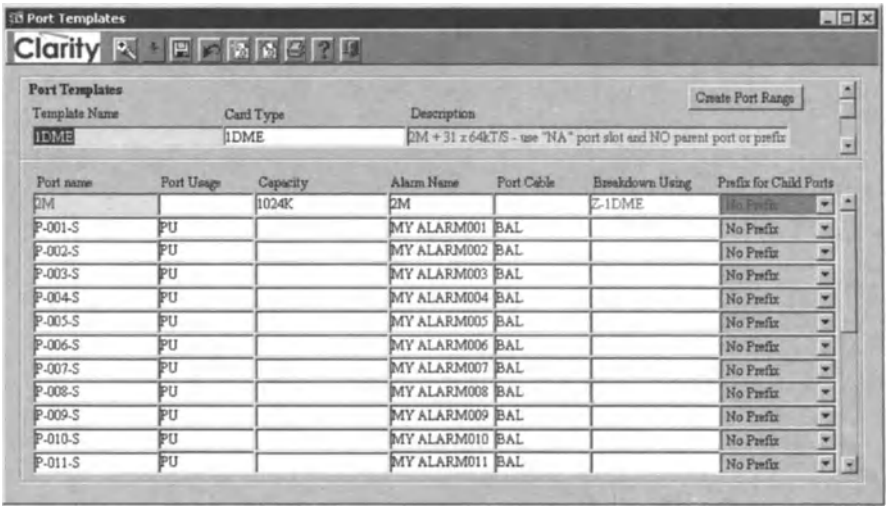


Figure 11.34 Creating port templates.

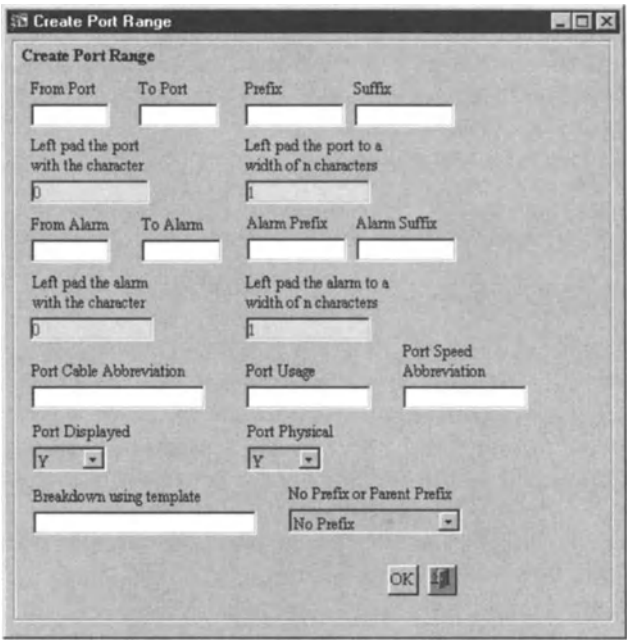


Figure 11.35 Specifying a port range for port templates.

A port template gives skeleton information in a port or port type so that often-recurring details need not be re-entered whenever such a port is created. A port template has a unique template name, a description of the template, a generic port name, the main purpose of the port – alarms, the name that is used for the port by

the NEMS, and the name of the cable that connects to the port. The range of ports for a port template can also be specified.

In practice, port templates are automatically applied to network elements in order to avoid manually creating parent and child ports within the network elements screen. A port *breakdown* must exist before a port template can be applied, so that the port has a hierarchical position. A port breakdown template can be applied in the following ways: apply top-level ports, create child ports with a prefix and create child ports with the parent port as a prefix. Although using a port template, it is at the same time possible to add ports manually from the NE screen. That is, in creating a port hierarchy, one is not confined to using only port templates or only manually entered ports. Templates are built up from the lowest level of the parent–child hierarchy, so that if child ports exist then they must be set up in a port template before the parent ports.

Specifying a port range is a fast method for creating port templates that contain many records. This involves entering a range of sequential entries for a port template, rather than entering all port numbers individually (see Figure 11.35).

Finally, CCM allows the building of SDH rings so that path diversity can be built in the network.

11.4 Conclusion

No attempt is made to weigh the relative merits of the products CSM and CCM. The most important lesson from the approach taken by CSM is that it is possible to customise and automate configuration management. APIs allow extension of a base product ad infinitum. We have sought to remove the mystery for extensibility via APIs by providing a whirlwind tour through some of the classes and methods of the CSM API.

We have sought to show what is involved in configuration management without requiring an understanding of particular network technologies such as Code Division Multiple Access (CDMA) or GSM for wireless and GigE, SDH or ATM for optical. Discussing details of network technologies in the context of configuration management only makes sense if we are to talk about the particular SNMP or TL1 commands that are to be sent to NEs. Such commands differ between network technologies and between network technology implementations.[46]

Configuration management goes to the heart of network management OSS for telecom networks because it is about the collection and manipulation of data on network state and it is about the modification of network state. As NEs become more intelligent, configuration management will become increasingly central to telecom operations. In a sense, configuration management is about the management of automated NEs, since once NEs are appropriately configured, then they are able on their own to do the job that is required of them. In this sense, configuration management is perhaps the most important aspect of network management.

[46] At the risk of labouring the point on what we have decided to omit, such technology-specific implementation details are more appropriate for a user manual on how to administer or use network management OSS if one is to begin work in an NOC sometime next week. Once one is familiar with the basic framework of configuration management, it is not hard to check for the SNMP commands or the MIB figure for the technology with which one is concerned, such as ATM or ADSL.

12 Fault Management

12.1 Introduction

No system is infallible. Detection of recovery from and damage limitation of failures are needed for every mission-critical business function. The network is a mission-critical component of the telecom business as it is critical to the revenue and reputation of the service provider. Understanding, monitoring and managing individual NEs may not be hard, but as the network increases in size and complexity, understanding the relationship between the various elements, monitoring and managing their interrelationship and the problems that arise due to their interrelationship can become challenging. Service impact of faults also becomes more complex as the network and the services offered become more complex. Owing to cost and the possibility of human error, it is not practical to deploy an army of personnel to administer a large network. Fault management OSS systems are currently the only economical way to achieve network-wide fault monitoring.

Products such as Fault Management Server (FMS) from AdventNet seek to allow the rapid design and deployment of comprehensive fault management services in order to enhance first availability and second performance. These products correlate and manage notifications and present critical information in a way that is useful to managers. Notifications are modeled as abstract objects which are better suited to tracking and management by software. FMS can be thought of as a framework that is both flexible and extensible, allowing domain/technology-specific filters and rules to be plugged in. This chapter is based on the approach to fault management that is taken in FMS. As in Chapter 11, we will give some insights into how FMS can extended through its APIs.

12.2 Events

The "Event" is the basic unit of management information that is complete in itself and relates to an occurrence such as the discovery of an element, status update of an element or failure in an element. The set of all Events comprises a repository of all system occurrences. Alarm results may be correlated with other events to represent faults in NEs that may need immediate attention.

K. Misra et al. (eds.), *OSS for Telecom Networks*

Event and alert objects define the basic attributes and actions needed for fault management. Users can model their own objects by extending the basic units to plug in domain-specific properties. It is possible to add additional attributes to these basic units or their derivatives at run time.

Events may be basic notifications from those SNMP agents which are external to the fault management system or they may be status poll events and threshold events which are generated when FMS is integrated with other network management services. Event generation can be induced through customised means using APIs. FMS models the failure information as event objects. Each event object pertains to a single failure in an element and has unique properties. There can be occurrences of multiple failure in an element and hence there will be a flow of events, each representing a unique failure.

The generated Events are parsed to make the notifications readable to the user. They are then processed by event filters which apply automatic actions to an event with the help of Event Observer, which is a user-written class. Other applications can be notified when events are added, deleted or modified.

Events can be extended if additional properties are needed beyond those provided in the raw event object.

12.2.1 Event Generation

Events can be generated in either of the following ways:

- generated from traps or autonomous messages received
- received from other applications via event APIs.

Events will be generated when SNMP traps or TL1 autonomous messages are received by FMS. First, the raw traps will be converted into useful event objects, using filters and parsers provided by FMS. Second, the event objects will then be added to the Event Manager and stored in the event database.

New event objects can be added using API methods. These events are added directly into the Event Manager and processed by event parsers and event filters before being stored in the database. The methods of the *EventAPI* class can be exported via RMI for remote invocation of the API methods.

12.2.2 Events and Information Flow

Network notifications are messages that are sent by the agents of managed objects without any manual intervention or request from the server. These messages are sent when failures occur in the nodes. Notifications are of different types depending upon the protocol which the managed object uses to communicate with the server. Each protocol has a unique way of sending notifications. For example, the agent implementing the TL1 protocol notifies the server through "autonomous messages", which are internally processed so that events are generated accordingly. FMS is an example of a "protocol neutral" product because it supports many of the common network management protocols.

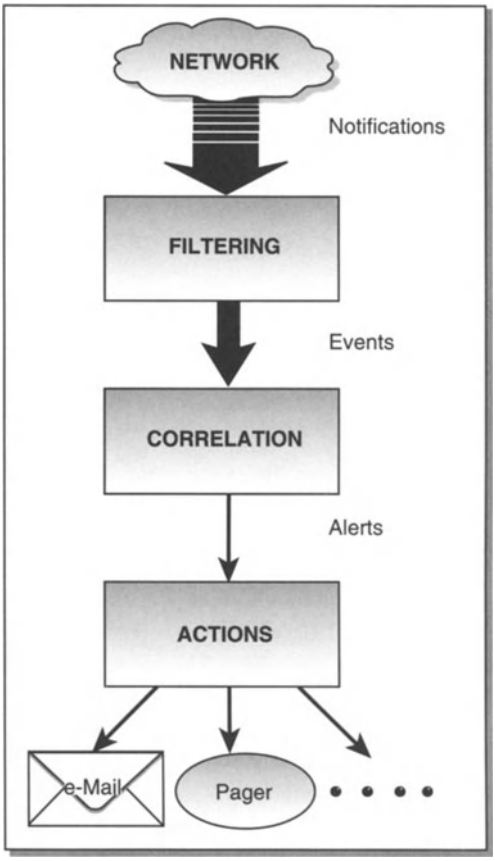


Figure 12.1 Event flow.

Notifications are received as raw data from the network and must be converted to a format which can be understood by console operators. Information such as whom to call, whom to page and procedures to fix the problem are made readily available to track down the source of the problem and to solve it. It is wasteful for staff to spend time on finding out how to resolve a problem *after* the problem has arisen because, at that point, all efforts should be directed towards fixing the problem.

The architecture of the fault management system addresses these issues by allowing filters and parsers to be plugged in easily. Filters help validate notifications received from the various managed elements and also apply domain-specific rules to notifications. Rules can be structured to validate the incoming events based on time, thresholds or number of events. For example, transient or temporary and self-correcting system failures occur as a normal part of network operation. Rules can be useful in these circumstances for filtering out transient network failures. This helps reduce thousands of events to 10–20 real problems that need to be addressed. Parsers disseminate the raw data so that concerned parties quickly receive a meaningful interpretation.

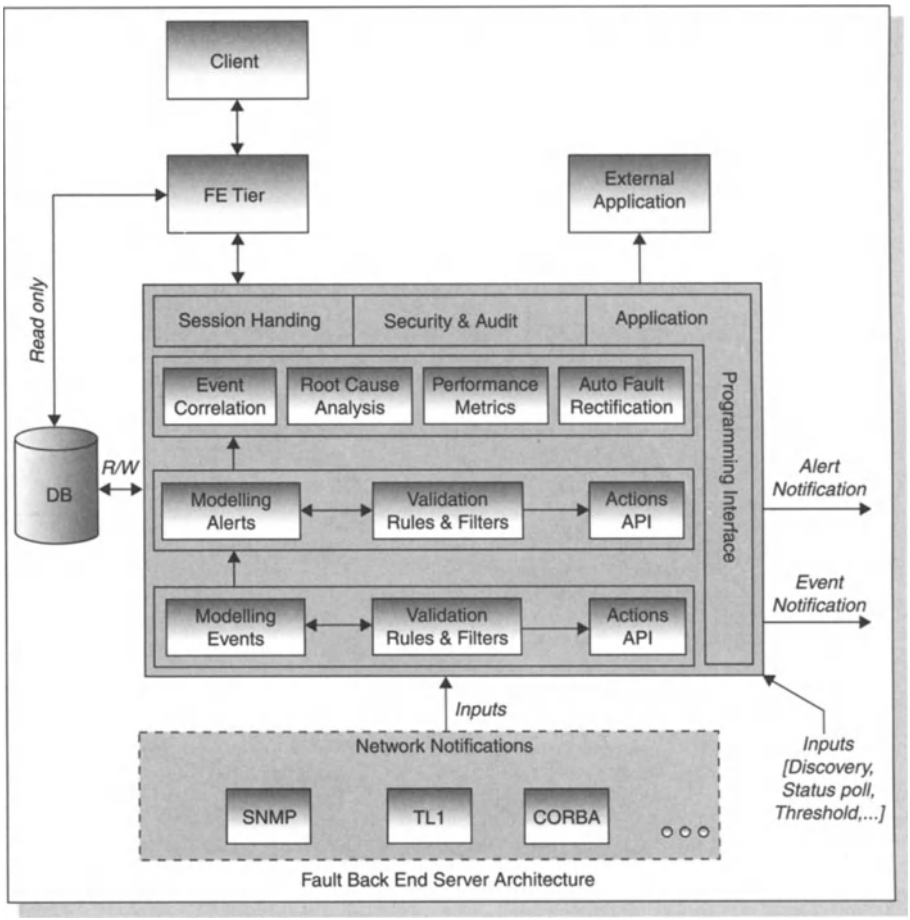


Figure 12.2 Fault management information flow.

FMS allows automated action to be taken at various levels of the event flow. The actions supported out-of-the-box[47] include e-mail, paging, command execution and custom actions. Apart from the automated actions, FMS has a formal notification mechanism. In particular, external applications can register with FMS for notifications. Registered applications are notified during the various stages of event flow.

An RDBMS is used as the persistence layer which facilitates achieving the high level goals of availability, scalability, performance, concurrence and atomicity of all the operations.[48] The persistent information is used to generate reports and for auditing. An internal caching mechanism helps to speed up transactions.

[47] "Out-of-the-box" is a software industry term meaning "without need for extension or expert configuration".

[48] Persistence refers to data that is stored – i.e. data that "persists".

The services offered by FMS are exposed through APIs. AdventNet intends that these be used to develop scalable and secure applications on top of the core functionality of FMS.

Some consider fault management to be the most vital part of any management system OSS from a network administrator's point of view. Once a service has been designed, provisioned and activated, then ensuring continuing availability of that service to customers is largely a matter of managing faults. Effective fault management should enable the administrator to identify failure in the networked environment with minimal effort.

12.3 Traps in SNMP Devices

Notifications, also known as event reports, are examples of events in a managed system. Notifications are sent to a list of managers configured to receive events for that managed system. In the case of SNMP, traps provide the value of one or more instances of managed object information.

Any SNMP-enabled device generates traps that are defined in the MIB which is implemented by the SNMP agent. The MIB contains trap details such as the name and a short description of the trap and device-related information. The trap definition varies with the SNMP version. The major difference between the message formats is in identifying the events. For SNMPv1 traps:

- The trap is represented as trap-type in the MIB.
- Generic type, specific type and the enterprise object ID together uniquely identify a fault and these are available as separate fields.
- Information is carried as values that are assigned to variables.

For SNMPv2 traps:

- The trap is represented as notification type in the MIB.
- Trap object ID identifies the type of fault.
- Information is carried as objects.

Any trap contains the information on:

- Device that has encountered a problem.
- Type of fault.
- Both the device and the agent sending the notification, if they are different.
- Time of fault.
- Possible variables that might have caused the problem and their values at the time the problem occurred, although this is not always possible depending upon the agent implementation.

12.4 TL1 Notifications

The notifications received from TL1 devices are in the form of TL1 autonomous messages. Autonomous messages are received from TL1 devices without any command being input from the fault management system.

The TL1 connection processes specified in the configuration file are started when the fault management system is started. This process reads the file which lists the devices to which the connection is to be established. Accordingly, TL1 fault management systems register to receive autonomous messages from the specified devices. The event process receives the autonomous message sent by the TL1 device and converts it into a fault event. This event is then passed through the event parser and emerges possibly after some modifications have been made by the event parser. The events leaving the parser are then processed by the event filters. Events which survive the filtration are stored in the event database.

12.5 Configuring Notification Reception

FMS can be configured to receive SNMP notifications (traps) at one or more ports. The port(s) that are listening for traps are specified by entries in the trap port configuration file.

When traps are received by FMS, other applications may be notified using the Trap Observer mechanism. Each application to be notified registers with the event manager as an Observer for Traps. Notification of traps are made in SNMP PDU form and contain Trap PDU information.

When the management system is configured to listen for traps at a particular port, then no other application can listen for traps at that port. The Common Trap Receiver mechanism allows more than one application simultaneously to listen for traps at a port and allows any application to register for multiple ports. Registration is carried out through API methods.

12.6 TL1 Autonomous Messages

Using a fault management OSS such as FMS, the user can establish a connection to TL1-enabled devices and listen to autonomous messages sent by those devices.

A TL1 fault management module will receive an autonomous message, parse it, construct the TL1 event object and pass it through the event parsers and filters. The event parsers and filters can be customised to modify/convert the event before it is stored in the database.

A *TL1 connection process* is an FMS process that handles TL1-related functions such as establishing connections to the TL1 devices and listening for autonomous messages. A configuration file lists the TL1 devices to be managed. This file can be used to specify the list of TL1 devices, the various login commands and the initialization commands for the devices. IP address, TL1 port and connection handler entries must be given for each device.

The TL1 port is the port in which the TL1 agent is running. The TL1 port must be specified since the TL1 agent is not running on any specific port. Multiple ports can also be specified using comma-separated values.

The name of each device group configured is specified by the device group name. This name is used to allow TL1 devices to be added at runtime. Once all necessary

parameters have been configured at the device group level and a device group name has been specified, the user can use the runtime Add Node facility from the Fault Client, a client application of FMS, to discover and add any TL1 device not already listed in the configuration file. All parameters specified for a device group are used to discover, connect and add the TL1 managed objects to the topology database.

The dictionary is the XML file with the syntax for input and output messages for any particular type of TL1 device. FMS is equipped with a *Connection Handler* class to establish a TL1 session with a TL1 device. The IP address of the TL1 device to be discovered must be specified.

The TL1 device needs a login command to communicate. This command generally corresponds to a particular user and password. FMS uses this command to open a session with the TL1 device as part of connection establishment and uses this session for all subsequent communication with the device.

Other commands can also be specified as part of connection establishment. A maximum of two commands for specific initialization of the TL1 device can be specified. The initialization command ("*init*") is for clearing the memory of the device, device error counts, dealing with pre-existing data in the device and so on. The *login* and *init* commands are sent to the device whenever a TL1 session is opened by the fault management system.

12.7 Filtering and Processing Notifications

A trap may need processing before it is converted into an event. Trap filters can be used to modify the properties of the trap before converting it into an event. The following are characteristics of the functioning of trap filters:

- Process and modify the SNMP PDU and return the SNMP PDU passed to it. After successful modification of the properties, if the PDU properties match any filter criteria, then an attempt will be made to match the PDU returned with the remaining trap parsers and filters and event parsers and filters.
- Construct an event object from the trap PDU that was passed to it and return the event object. The event object will be passed through the event parsers and filters, but not through trap parsers.
- Return extended event objects.
- Drop the trap by returning "*null*". This restricts the trap from further processing by FMS.

A custom trap filter can be made to carry out any of these functions.

Figure 12.3 illustrates where trap filters are configured, with the evident functions of:

- adding a trap filter
- modifying a trap filter's details
- deleting a trap filter
- enabling/disabling a trap filter
- viewing trap filter details
- ordering trap filters
- saving configuration.

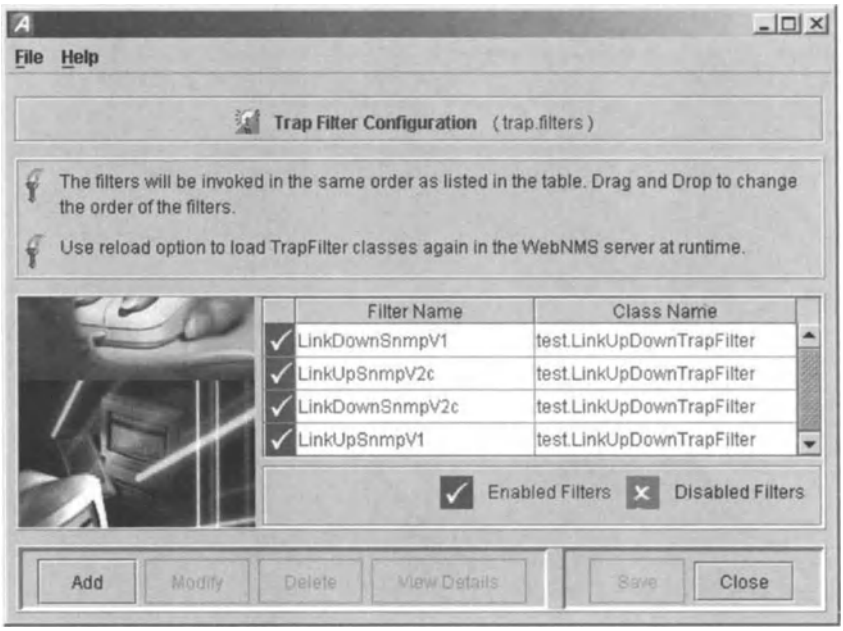


Figure 12.3 Trap filter configuration.

12.7.1 Trap Parser Configuration

If a trap filter returns a trap PDU, then the traps may be parsed to generate meaningful events. A tool is provided for configuring trap parsers.

Options available with the Trap Parser Configuration are Add Trap Parser, Modify Trap Parser, Delete Trap Parser, Save To File, Load From MIB and Load From File. Rather than specifying the match criteria manually, an MIB with defined traps can be translated to trap parsers by loading from an MIB or from some other file.

The Configuration of Trap Parser involves setting trap parser parameters and match criteria, defining the parameters for the output event object and configuring the trap parser list. Table 12.1 lists the trap parser configurable parameters.

The output of the trap parser is an event object whose attributes are defined by the trap parsers. Selecting appropriate values for important attributes, such as failure object/affected entity, source and message text, allows quick identification of a failure.

While specifying the output values for the event object generated, information present in the incoming trap PDU can be used. The information in the trap PDU can be accessed using the tokens which represent values of the fields in a trap.

Incoming traps are parsed by configuring a list of trap parsers. Only one trap parser can be applied to a given trap. The match criteria in the parser determine whether a specific trap matches the parser or not. Once a matching trap parser is found, an event will be generated by applying the trap parser and no other trap parsers will subsequently be applied to that trap. Users can enable or disable any particular parser.

Figure 12.4 Trap parser configuration.

Table 12.1 Trap parser input fields.

Parameter	Description
Nodes	Nodes to which a trap parser is to be applied are specified by the Agents' IP addresses
Groups	If managed objects can be grouped, then it makes sense to allow specification of groups to which a trap parser is to be applied
Enterprise	This field is used to specify the enterprise OID of the SNMP v1 trap. The parser will be applied only if the trap enterprise field starts with the value specified
Generic Type	Each SNMPv1 trap has a Generic Type number. Users can specify the Generic Type number in this field so that the trap parsers will be applied only to corresponding traps
Specific Type	Each SNMPv1 trap has a Specific Type number and this can be used as a match criterion for incoming traps
Trap OID	Trap OID is an identification number which comes along with the Trap PDU. This is used only for SNMPv2c and v3 traps

12.7.2 Trap API

Trap parsers can be configured using the trap API. A set of APIs is provided to handle dynamically the features of trap filters and parsers. Using API methods, it is possible to automate the addition and deletion of parsers and filters and also automate how their properties are set. Trap API methods are published with RMI handle/trap API on the server. Code that is in the same JVM as the FMS can use the *TrapAPI* handle.

12.8 Customizing Event Processing

FMS allows users and developers to customise event processing based on the type of events generated for a network or device that is to be managed. For example, to manage a network of bridges, knowledge of the events generated can be used to configure parsers and filters that make it easier to track problems with these particular bridges.

User- and developer-configured parsers and filters form an important part of the flow of an event through a system, which is shown in Figure 12.5.

When traps are received by the event manager, they are passed through trap filters. Users can configure matching criteria for trap filters. As with any trap filter, a user-written trap filter could:

- generate a fault management event as its output
- generate the received trap without change
- drop the trap by returning null.

The next step is notification to Trap Observers. If the received trap has not been dropped by any trap filter, then registered trap observers will be informed of the trap. The next step is trap parsing, which should be configured to generate useful and appropriate event information from the trap received. For example, a trap parser could be configured to set the message of Event generated simply to be the same as that of the received trap message. Whereas the output of a trap parser is an event object, events can be generated by inputs other than traps, such as events generated during discovery, status polling, data collection when FMS is integrated with other NMSs or OSSs and events generated by users via the event API.

Events are then passed through a series of event parsers, which may be configured by users for further refinement of input events. An important role of event parsers is in identifying the affected element or failure object. For example, a "link down" trap may have a router and its interface as the failure object. Using the trap and event parsers appropriately, this can be converted to an event where the failure object is narrowed down to either the router *or* its interface.

Event filters allow still further processing of the event and generating actions, such as e-mailing a human operator. Filter actions might also include suppression of multiple events in a given interval, the running of shell commands on a server system, sending e-mails or traps and running custom Java code to filter events. Custom Java code may be needed where additional data must be retrieved or specific code applied to process the event.

Processed events are stored in RDBMS tables or logged into flat files and made available to the event browsers in a client. In modern fault management OSSs,

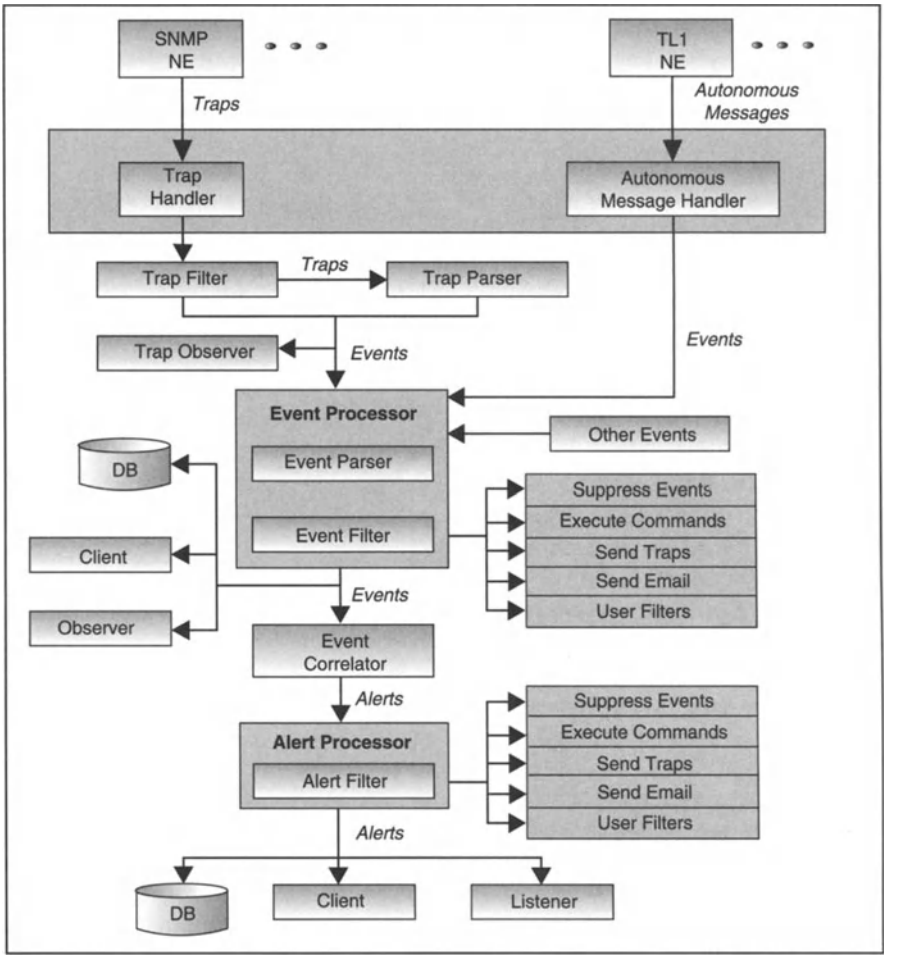


Figure 12.5 Detailed event flow.

the new information is generally available to event browsers immediately upon the processing of an event, limited only by the network over which the OSS system is running. The processed events are then sent to the fault manager for higher-level event processing. The primary element used by the fault manager is the failure object identified in event processing. Using this as a key to the fault, the events are processed to create or update alerts as appropriate. For all the events with the same failure object, only one alert object with the same failure object is created/updated. The alert object reflects the status of the latest event with the same failure object.

Alert filtering can be applied for processing alert and executing appropriate actions. Alert filter actions available are generally the same as event filter actions. The processed alerts are stored in RDBMS tables or flat files and users notified. Finally, Alert Observers, analogous to Event Observers, will be notified of the alert.

The server component of event management is the Event Manager, which includes the event and alert servers. These, in turn, are interfaces to the event browsers and alert viewers. Traps are received and processed by the Event Manager. The client components are the Event Browser and Alert Viewer. These are either Java applets that run in any Java-enabled web browser or ordinary HTML-based clients that are accessible from a web browser.

12.8.1 Extending Events

Users can add properties to events for their convenience or for the specific needs of a particular carrier. The trade-off is that a greater number of user properties may reduce performance, as the time taken to execute database queries increases with the number of properties to be searched. Adding user properties to an event object does not extend the event object. If it is necessary to add a large number of properties, then it is better to extend the event object.

Based on the device types and the need for additional properties, the decision as to whether or not to extend the events is taken. If the networked environment includes specific devices which need more properties to represent an event object, then it is better to extend event objects.

A custom class must be written to extend the *Event* class. Extended Events will have all the properties inherited from the Event object and the additional properties specified by the user. The extended Event object is stored in the database and becomes part of the working functionality of the fault management system.

12.8.2 Filtering and Processing Events

To identify the failure object corresponding to an event, the traps and input events should be parsed and the necessary details given. A trap parser makes notifications readable to the user. An event parser refines the message conveyed by the events. Although both the trap and event parsers seem to perform the same job, in fact an event parser converts other types of notifications – such as TL1 messages and user-generated event notifications – into a readable format before filtering events.

Event parsers are configured as shown in Figure 12.6.

Managing event parsers is the usual operations of adding, modifying, deleting, saving and loading event parsers and re-ordering the sequence of application of the event parsers. Configuring an event parser involves:

- setting parameters
- setting match criteria
- using tokeniser strings
- defining parameters for the output event object
- setting up the configured event parser list.

The Event Parser Configuration window has several parameters for configuration of an event parser. The first parameter is the "Name" field, which helps identify a parser when managing a list of parsers. The remaining fields of the event parser correspond to event object attributes.

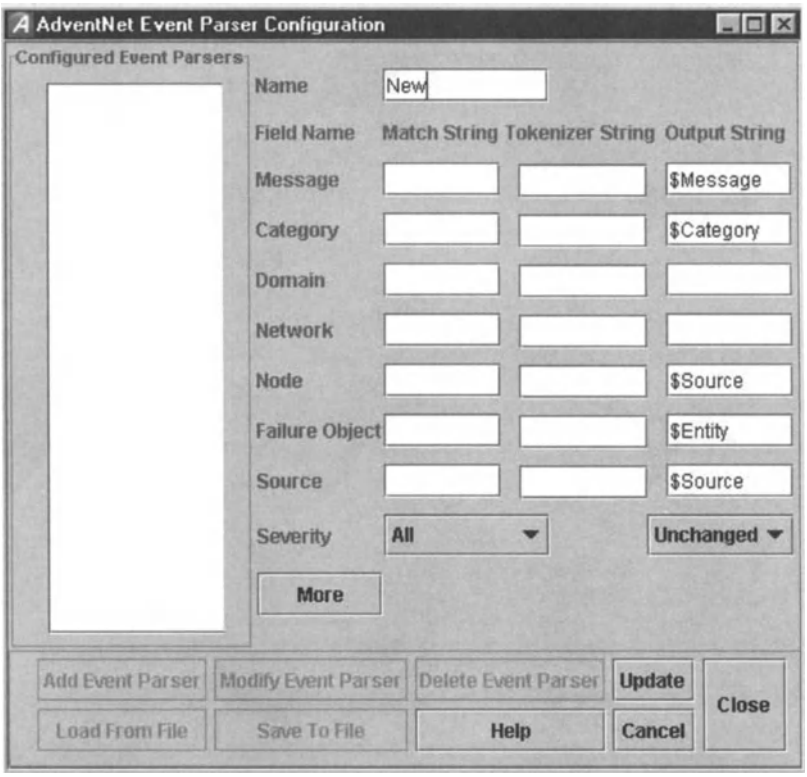


Figure 12.6 Event parser configuration.

The output of any event parser is an event object, which is the modified instance of the incoming event based on the definitions configured in the event parser. The match criteria determine whether the event will be parsed by the given event parser or not. Tokeniser strings allow the user to break up the input field into a series of tokens that can then be used in the output event object.

If an event is generated by a trap, then the associated trap PDU reference is maintained in the incoming event object. If the event object contains the trap PDU reference, then the properties of the trap can be used in the event parsers.

When an event arrives in the fault management system, the list of event parsers is checked to see whether the incoming event satisfies the match criteria of the event parser. If an event parser does match, then the event is passed through that parser. The outgoing event from the parser is then matched with the remaining parsers (if any) in sequence. This process continues until the parsed event matches no more parsers.

A set of parsers might be available in the event parsers list. Among these, a user can select a parser and disable it.

When an event is raised, certain actions may be executed aside from parsing. Event filters can perform automatic actions when an event occurs, such as sending an e-mail, suppressing the event and generating traps. Custom classes can also be executed upon generation of an event.

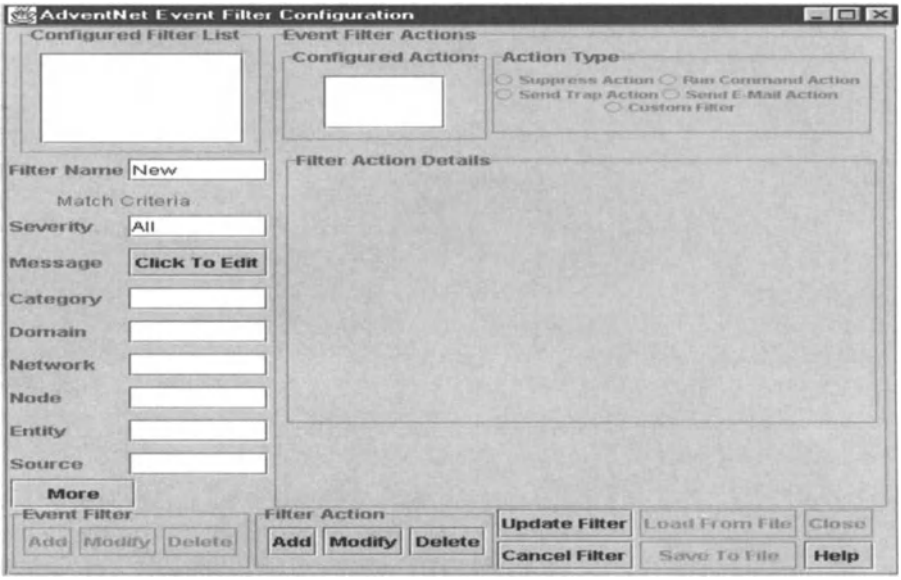


Figure 12.7 Event filter configuration.

Figure 12.7 shows the event filter configuration screen. There are many ways in which events can be filtered. Filter parameters include:

- **Filter name.**
- **Severity.**
- **Message:** Message is matched with the message of the incoming event such as interface failure, status poll failed.
- **Category:** Category to which the event belongs.
- **Domain:** Domain-specific information, which might be based upon physical location, functional or logical categorization of the source of the event.
- **Network:** Network to which the source of the event belongs.
- **Source:** Source of the event.
- **Node:** Additional information about the source of the event.
- **Entity:** Exact device in which the problem has occurred.

These parameters can be used to configure match criteria which determine whether or not an incoming event should be filtered.

12.8.3 Event API

Event-handling APIs enable independent application functionality for performing tasks which relate to accessing event objects, adding new event objects, registering as Trap and Event Observers for notification, among other things. The *EventAPI* methods can be accessed from within the same JVM as that of FMS or from a separate JVM using RMI.

A set of API methods allow the developer to create functionality for dynamic adding, deleting and accessing the properties of the event filter at runtime. There is also a set of API methods for dynamic handling of event parsers.

Event filters can be used to perform specific automatic actions or to use user-defined classes to process the event. Events generated are first passed through event filters based on whether they satisfy matching criteria. An event that satisfies these matching criteria will trigger – or be subject to execution of – an action specified in the event. Once again, applications can register as observers using the Event Observer API.

There are five actions given by FMS for event filters:

1. **Suppress action:** Suppresses the generation of events for a specified period of time as configured by the user.
2. **Run command:** Executes a program such as opening a notepad editor whenever an event of the matching criteria is received.
3. **Send trap action:** Sends a trap with specified properties whenever an event is received with the matching criteria.
4. **Send e-mail action:** Some specified persons can be notified of event received via e-mail.
5. **Custom filter:** When an event with some specific properties is received, the user-defined filter classes can be used to process that event.

Alert filters are used to manipulate incoming alerts and events per user requirements, subject to matching criteria. Alert filters can perform specific automatic actions such as e-mails and user-defined classes. The above five actions can also be specified for Alert Filters.

A filter action class can be used to group alerts. Figure 12.8 shows the alerts being grouped under a single group called "MyGroup".

Figure 12.9 shows a log file with messages being printed when alerts are generated.

12.9 Client Framework

The Java client framework provides a customizable infrastructure for developing and integrating third-party applications. The Java client can be run in a client browser or as a standalone application. The client's GUI framework is driven by XML configuration files on a user-to-user basis. Applications can extend the client with new screens and panels for specific information or branding requirements.

The client framework seeks to make it easy to customise the various client parameters, such as fonts, icons and status bar, the client tree and the panel settings by configuring the respective configuration files. It also supports client-side internationalization so that clients customised in different locales can be connected to a single server.

12.10 JMX Agent

The Java Management Extensions (JMX) technology is central to the fault management client framework supported by FMS. We shall explain how the JMX agent

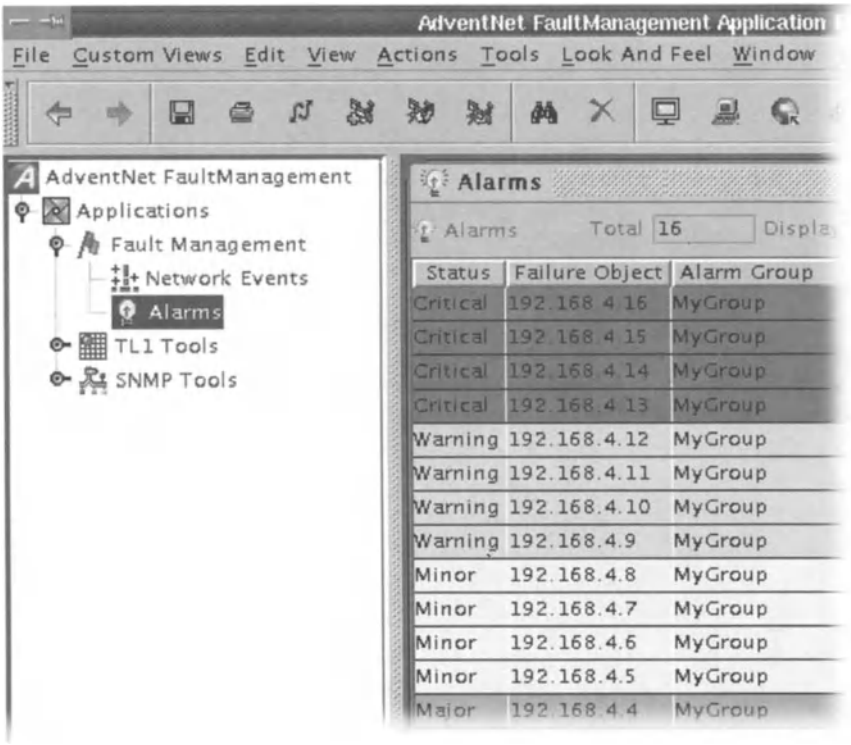


Figure 12.8 Display of alarms and alarm groups.

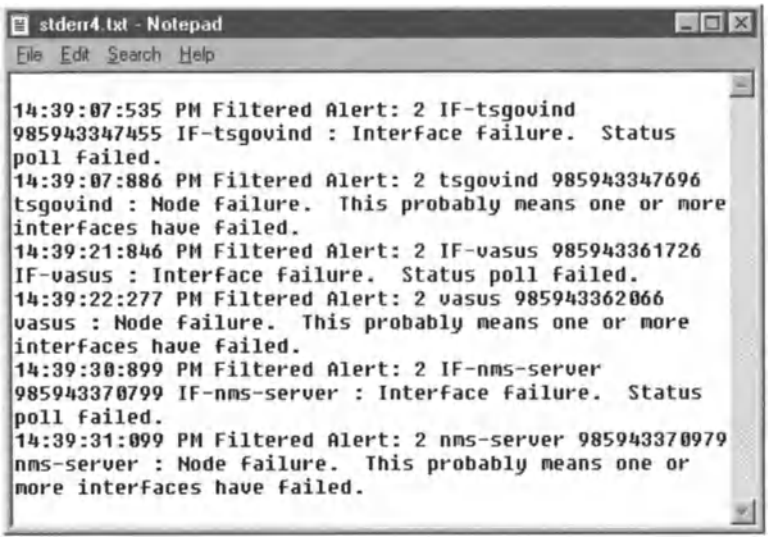


Figure 12.9 Log file generated is similar to plain text log files generated in other fault and alarm management software.

works and how it contributes to network management OSS more generally, and how it particularly assists with fault management.

We shall first review the role of the element management system and its relationship with higher level management systems. A network management OSS monitors the managed objects whereas a fault management system draws on this functionality to achieve its own aims.

In the case of AdventNet's offering, the Element Management System product makes it possible to monitor specific elements in the network. The main functions performed by an NMS are to:

- discover objects
- recognise topology and condition of various devices
- detect network or device faults
- automatically send messages on encountering an Event or Alert
- obtain configuration information and gather performance information of various devices
- confirm the connectivity and reachability between the Manager and the devices being managed.

The EMS is used to manage a specific set of devices or a number of devices connected in a network. The EMS is also used to manage different nodes such as SNMP or TL1 nodes. The EMS communicates with network devices directly using network management protocols. The functions usually expected from an EMS are to:

- provide common information about the NE, such as system up-time and system name
- monitor the operational state of the NE, such as startup and shutdown
- monitor the occurrence of NE malfunctions
- gather and processes NE performance information
- enable configuration of the NE
- carry out remote operations on the NE.

The need for both NMS and EMS in management is appreciated when many nodes, networks and NEs must be managed. The EMS can be used to manage a specific group which can be separated logically. Groups may be assigned to an EMS so that it manages a specific node, network or a number of specific devices.

The advantages of using an EMS to manage devices are:

- management of specific devices connected to different networks
- management of all devices in a particular network
- reduce the load on the NMS through sharing and balancing.

Hence the involvement of both NMS and EMS in the management hierarchy helps to manage the managed objects more effectively.

12.10.1 Interface Between NMS and EMS

The NMS and EMS are used for effectively managing a large number of NEs. A single NMS can manage a number of EMS. This hierarchy can be extended so that a number of higher level managers are used to manage a number of low-level managers.

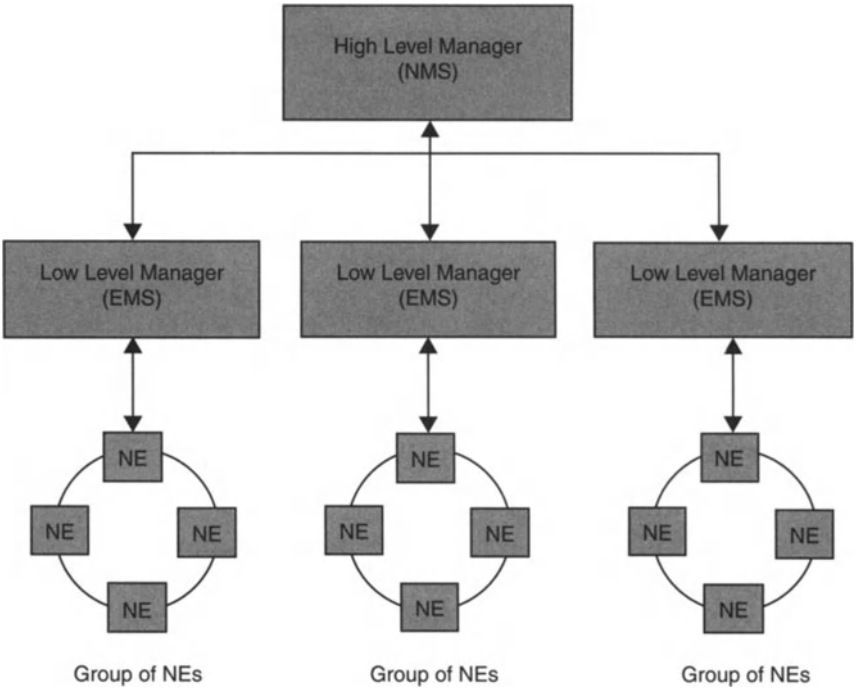


Figure 12.10 Relationship between NEs, EMS and NMS.

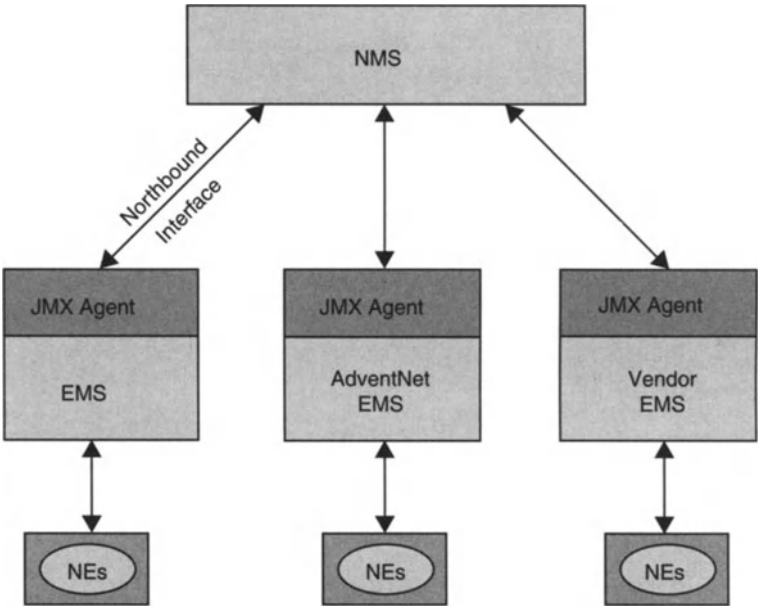


Figure 12.11 Position of AdventNet EMS and JMX agents relative to NEs and NMS.

A management structure benefits from a defined communication channel between the high-level and low-level managers. Communication between an EMS and its NMS is called northbound or upward communication. This communication is carried out using protocols such as SNMP, HTTP/HTML, RMI, CORBA and TL1, and through a northbound interface.

An interface between the NMS and the EMS should be able to:

- send independent notifications to the NMS when a managed object is added to or deleted from the management domain, that is, the managed network
- send autonomous notifications to the NMS when an event or alert is generated
- forward notification to specific managers
- provide details of the managed objects that are being added to or deleted from the management domain
- provide details of the networks being added or deleted
- provide details pertaining to fault management
- provide performance details of the NE to the NMS.

As mentioned above, as a client interface, FMS uses the JMX Agent. JMX specifications can be used to manage Java-based environments. Thus, the NMS can obtain details of managed objects from the EMS through the JMX Agent.

12.10.2 JMX Agent Architecture

The JMX Agent is the interface which is used between the NMS and the EMS. The JMX Agent is built based on the JMX specifications. JMX specifications advocate protocol independence. At time of writing, the JMX Agent supports the SNMP, CORBA, RMI, HTTP/HTML and TL1 protocols.

A JMX Agent comprises an MBean server, a set of Managed Beans (MBeans) representing managed resources and at least one or more protocol adaptors or connectors. The MBeans are Java objects which implement the resources and their instrumentation.

The MBean server is a registry for MBeans in the JMX Agent. The MBean server is the component which provides the services allowing manipulation of MBeans. All management operations performed on the MBeans are carried out through Java technology-based interfaces on the MBean server.

Protocol adaptors and connectors allow management applications to access a JMX agent and manipulate the MBeans contained in the MBean Server. Protocol adaptors give a representation of the MBeans directly in another protocol, such as HTML or SNMP.

The JMX Agent can respond to the queries sent by the higher level manager. It can also send independent notifications to the management system regarding the status of the application being managed.

The JMX Agent assists in monitoring managed objects. In general, the JMX Agent:

- Can send autonomous notifications to the higher level manager regarding the NE.
- Enables configuring of the persistence of an autonomous notification.

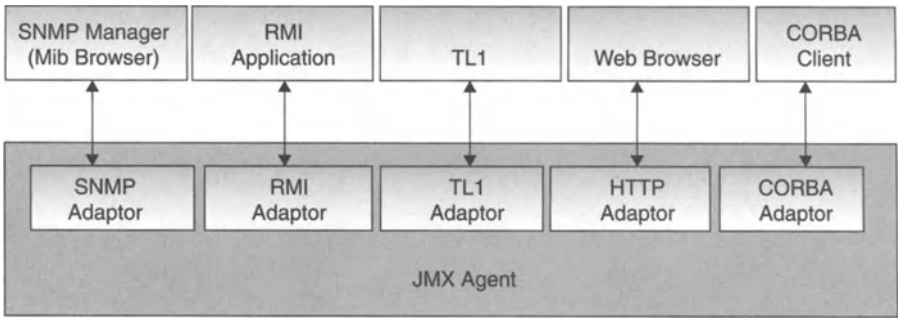


Figure 12.12 Breakdown of JMX Agent by protocol.

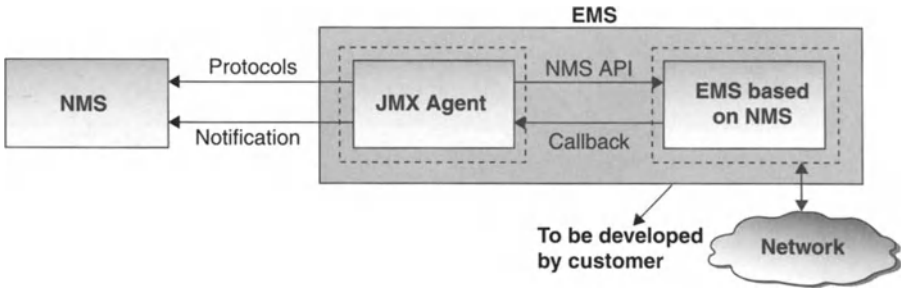


Figure 12.13 Representation of calls between EMS, NMS is a generic scenario.

- Provides details such as EMS host name, total memory and free memory. These details are obtained by the NMS by querying the EMS through the JMX Agent.
- Provides access to the data stored in the database.
- Can upload or download files using TFTP.
- Can register a number of agents with itself as sub-agents.
- Enables configuration at runtime.

The EMS responds to the queries of the NMS through the JMX Agent. The JMX agent can either perform the query on the NE and get the details or check the database of the particular NE and provide responses. The protocol involved in sending a response to the NMS through the JMX Agent is the same as the protocol of the query received.

12.10.3 JMX Specifications

The JMX specifications define the architecture, design patterns, APIs and services for application and network management in Java. The major benefit of this specification is that it provides a way to use common instrumentation on the agent and support multiple protocols for management. Since JMX is independent of the management infrastructure, the developer of the agent is freed from the details of each management protocol.

The JMX architecture is divided into the instrumentation level, agent level and distributed services level.

The instrumentation level defines how to use resources in the Java programming language so that they can be managed. The resources developed are called Java Manageable Resources. MBeans follows the design patterns and interfaces defined in this part of the specification, ensuring that all MBeans provide the instrumentation of managed resources in a standardised way.

The JMX Agent uses Model MBeans, which are dynamic MBeans that are configurable and self-described at runtime. They provide a generic MBean class with default behaviour for dynamic instrumentation of resources.

12.10.4 Management Information

When an NMS manages a number of EMS, different functions are expected from the JMX Agent. Autonomous notifications are sent by the EMS to the NMS through the JMX Agent. The functions provided by this feature are:

- Autonomous notification when a managed object has been discovered in a management domain. Notifications are also sent when a managed object has been deleted from the domain or when there is a change in the attribute of the managed object.
- Notification that alerts of some severity has been raised in a particular NE.
- Notification when an alert of any severity has been cleared.

Notifications sent by the EMS can be configured to be forwarded to one or more higher level manager. The higher level manager can be configured to reject notifications from a particular EMS. Notification forwarding can be configured at runtime.

The NMS can obtain details about the EMS by sending queries to the various EMS through the JMX Agent. Details can be obtained such as EMS host name, version number and threads being used.

The details exposed to the NMS for querying are the total number of Alerts generated, the total number of Events generated and the details associated with each alert or event generated, such as severity, owner name and configuration of trap filters. The management information provided by the JMX Agent to the NMS is the same across different protocols.

12.10.4.1 Management Information Main Group

Notifications are unsolicited messages sent by the JMX Agent to the management system. JMX Agent supports a feature for sending notifications to the higher level manager. The JMX Agent registers itself as an Alert Observer and Event Observer which enables the JMX Agent to receive notification.

Autonomous notifications are sent in both SNMP and TL1. JMX Agent sends autonomous notifications whenever an alert is received/updated by the system.

When the JMX Agent detects that a predetermined event has occurred, traps are generated by the agent and sent to one or more trap destinations, the Object Managers.

An Object Manager can determine what kind of event caused the agent to generate the trap by examining the data stored in the variable bindings ("varbinds") of

each trap message.[49] The varbinds contain the identity and values of MIB variables that provide information on the specific event.

To identify the trap destinations, AdventNet's SNMP API has implemented a proprietary table called the Trap Forwarding Table that contains all necessary information regarding the trap destination.

A TL1 autonomous message is sent from an agent to the appropriate Manager without an explicit input message associated with it. These are unsolicited messages from the agent to report alarmed and non-alarmed events to the manager. The TL1 autonomous messages are sent to the Managers which have established a connection with the JMX Agent using the TL1 protocol.

12.10.4.1.1 Retrieving Lost Notifications

The notification can be either persistent or non-persistent. The SNMP traps and the TL1 autonomous messages received by the Managers are stored in the database and can be retrieved later. The notification log table and the varbind log table track the notifications sent and the varbind contents, respectively. This persistence feature allows lost notifications to be retrieved.

All notifications sent to the Managers are stored in the database. If the Manager fails to receive a notification, then it can be retrieved from the database. To this end, the Manager has been equipped to access readily the notification log and varbind log tables.

Through the notification log index, it is possible to recover notifications selectively. The notification log table summarises all notifications. The details of the notifications are available in the varbind log table. The number of notifications that can be stored in the database can be specified via the maxRows attribute in the PersistenceTrapMIB. For example, if maxRows is set to 10, then only the last 10 notifications will be stored in the database.

12.10.4.1.2 Forwarding Notifications

Notifications are sent to the Managers which have registered with the JMX Agent. Details of the Managers able to receive the notifications are specified in the Forwarding table. These details are the same for the SNMPv1/v2c, but there is a considerable difference from the details that must be provided for the SNMPv3. In particular, SNMPv3 requires that security-related details of the recipient Managers be provided also.

It is possible to suspend notifications to some Managers temporarily. A Manager can be permanently unregistered by deleting the row corresponding to that Manager in the Forwarding table.

12.10.4.2 System Group

The system group is concerned with providing the Manager with details about the system. Such details include basic information required by the NMS to manage the

[49] Variable binding simply refers to the values of the variables.

EMS. Details such as version, host name, start time, up-time and ports available are exposed as scalar values by this group. Some values in the system group are exposed as tables, namely the TrapPort and the scheduler tables.

12.10.4.2.1 Log Configuration

The JMX Agent provides a feature for configuring the log files. This module provides for setTable values. The data exposed by this module are:

- name of the log files
- directory name of the log files
- number of files of the type
- number of lines in the files
- whether logging must be done or not
- whether time-stamping must be done or not
- the level at which logging must be done.

The configuration of Log Files is also carried out through some values which are exposed as table attributes. The tables involved are the Log and the Log User tables.

12.10.4.2.2 Proxy Service

The JMX Agent can act as a *proxy* for any other agent. Unlike dynamic registration, the proxy service does not register the target device's agent as a sub-agent. The value of only one OID can be obtained at one time from the agent of a target device. The Manager passes the details of the query to the JMX Agent and, in turn, the JMX Agent performs an SNMP set/get/getnext on the target device's agent. The process is illustrated in Figure 12.14.

The NMS-Framework MIB has a proxy table in the proxy service module, where all proxy details are stored.

12.10.4.2.3 Trivial File Transfer Protocol (TFTP)

TFTP is the forerunner protocol of File Transfer Protocol (FTP). TFTP is less complex than FTP and does not have all the features of FTP. TFTP is well suited to enabling simple devices without memory, such as routers and switches, to obtain their *bootstrap* information.

In the case of JMX Agent as a client of FMS, the transaction is between fault management system and the device, where a TFTP server is running. The system can be

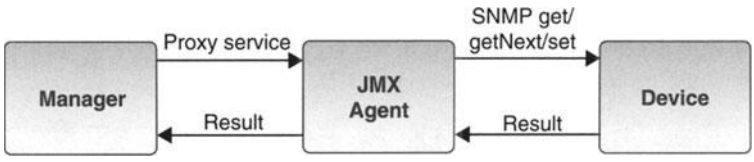


Figure 12.14 JMX agent acting as proxy.

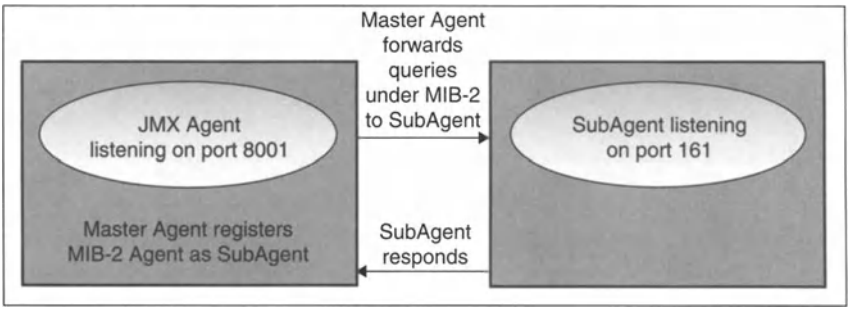


Figure 12.15 Relationship between master and sub-agent.

instructed to *get* a file, for example the device configuration files, by TFTP from a device to the fault management system. The system can also be instructed to *put* a file, for example log files of fault Management, into the device with TFTP service. The Manager ensures transfer of files between FMS and the device with a TFTP service.

To invoke TFTP service through the MIB browser of the Manager, the user must modify the values of the attributes of the NMS-Framework MIB: server name, port number, source file, destination file, TFTP mode and request, to get/put a file.

12.10.4.2.4 Registering the Sub-agent

The JMX Agent acts as the Master Agent and any number of sub-agents can be registered with the Master Agent. Any agent can be registered as a sub-agent to the JMX Agent. By using this facility, the Manager can access the sub-agent's OID in the same way that it accesses the Master Agents' OIDs. That is, the Manager can access the sub-agent's OIDs through the Master Agent.

Requests or queries are sent to the Master Agent by the Manager. Queries related to the sub-agents registered with the Master Agent are forwarded to the respective sub-agents. The sub-agents send their responses to the Master Agent, which in turn sends them on to the Manager. The varbind values processed by the Master Agent are also sent to the Manager.

The Master-Agent/Sub-Agent functional relationship is illustrated in Figure 12.15.

The management information corresponding to the Sub-Agent is exposed through a table called the SubAgent table.

Any agent can be registered as a sub-agent to a Master Agent. Sub-agents can be registered for SNMP, RMI and CORBA. Similarly, any SNMP agent can be registered as a sub-agent to the JMX Agent. Details of the SNMP agent, such as sub-agent name, port, time out and community, must be entered. In RMI and CORBA protocols, the sub-agents can be registered to the JMX Agent using the "Cascader Service".

12.10.4.3 Fault Group

While managing the EMS, the NMS may encounter events and alerts. These details are exposed to the Manager through the Fault Group of the Fault-MIB. The Fault Group can be classified as general, database, notification and configuration details.

General details include number of events generated, number of alerts generated, size of event queue, size of alert queue, user-defined properties and so on with respect to the objects being managed by the EMS. General details are exposed to the NMS.

Database details corresponding to alerts, events and severity of the alerts are exposed by the EMS as database records. There is a table devoted to each of alerts, events and web NMS severities. The Alert table exposes data about the Alert objects in the EMS, the event table exposes data about events in the EMS and the Alert Count table shows the number of alarms of a particular severity. Fault notification details can be configured to be forwarded to any specific Manager. Trap filters can be configured using the Fault Management JMX Agent. The data that are necessary for configuring the trap filters file are exposed through the trap table.

12.10.5 Accessing JMX Agent

The JMX Agent is protocol independent. By default, five protocols are supported by the JMX Agent. However, the user can plug in a protocol other than those specified using a protocol adaptor plug-in. The protocols supported at time of writing are SNMP v1/2/3, HTTP/HTML, RMI, CORBA and TL1.

The details which can be queried by the FMS are the same in all protocols. The AdventNet Agent Toolkit provides the MIB Browser, the Remote Object Browser and the TL1 Browser. The JMX Agent can be accessed using the MIB Browser for SNMP protocol, Remote Object Browser for RMI, CORBA and HTTP protocols and the TL1 Browser for TL1 protocol.

In the case of SNMP, the details which can be queried are exposed as MIBs. The SNMP Manager is the MIB Browser and can query the agent and obtain the result using the specified MIB. Thus the JMX Agent can be accessed using the SNMP v1/2/3 protocols.

"Java edition" licences are sometimes sold separately by OSS vendors for additional revenue. In the case of AdventNet, the Agent Toolkit Java Edition makes the SNMP Agent able to access the JMX Agent. The JMX Agent can be accessed using SNMP v3 protocol by specifying the user name, password and context name in the Manager. By default, authentication for the JMX Agent is given for three users.

12.11 Using Fault Management Functionality

Network events, alarm and alert grouping aspects of day-to-day fault management activities will be described in detail.

12.11.1 Network Events

Network events (or just "events") represent specific changes in status of network devices. Network events can also convey general information or the current status of the devices in a network. An operator can learn the history of a device by browsing the Event Viewer in the Client.

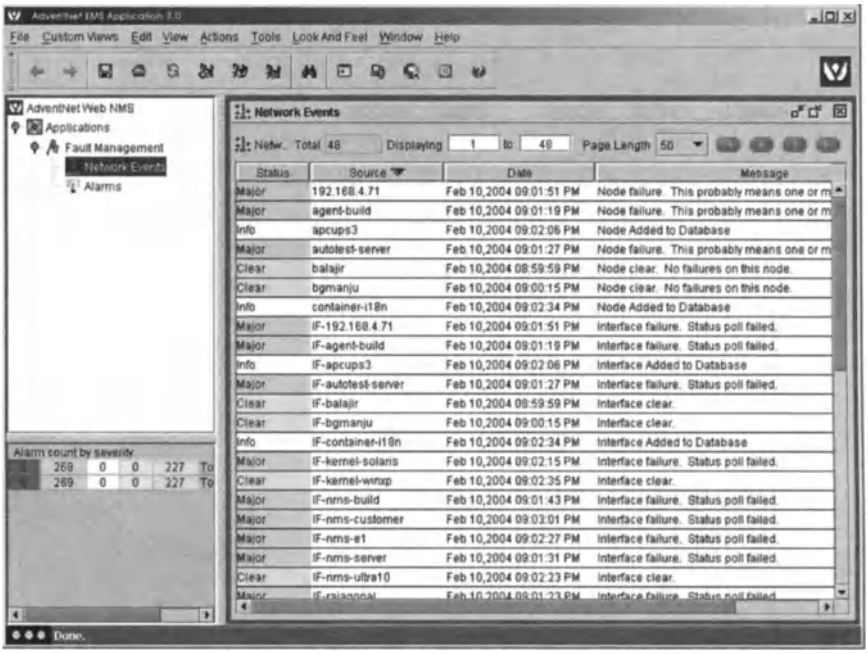


Figure 12.16 Event Viewer client.

The Event Viewer allows the operator to extract more detailed information from network events. The events database can be navigated with the aid of different properties that are provided to allow an easy and understandable view. By navigating the database, the operator can sort data according to their requirements. Navigating through the events database can be done using tools such as view range, the navigator buttons and sort.

The operator can search for one or more related events in the event database. The search operation will be executed over the entire database; it is not restricted to the displayed event viewer alone. Detailed event properties are viewed in a dialog box such as in Figure 12.17. Each event property is described in Table 12.2.

The operator can view the alarms related to a particular event.

12.11.2 Working with Alarms

Alarms are generated when a failure or fault is detected in network device(s). The alarms generated are displayed in the Alarm Viewer.

The Alarm Viewer provides a range of functions for extracting information from alarms. The alarm database can be navigated with the help of the view range, navigator buttons, column reordering and so on. The operator can also view the severity of alarms (these are colour-coded on-screen) in Figure 12.18. Alarms can also be searched using queries.

Alarm details include a brief description of the device which has failed, the owner with whom the alarm is associated and the date, time and severity of the alarm.

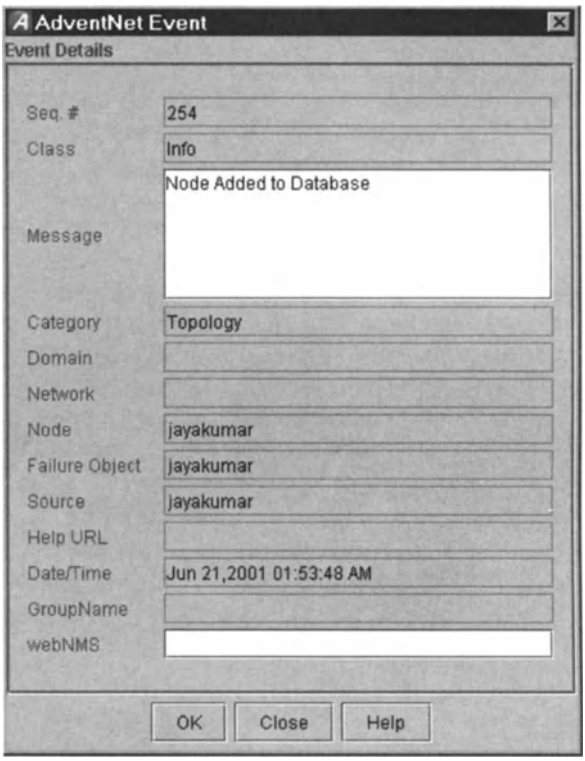


Figure 12.17 Event details.

Table 12.2 Event properties.

Property	Description
Sequence number	Unique ID for the event generated
Severity	Criticality of the event
Message	Any additional information on the event
Category	Alarm category
Domain	This is a property in the event object which holds any domain-specific information based on physical location or functional categorization, or logical categorization of the source of the event
Network	Information about the network to which the source of the event belongs is specified here
Node	For user convenience, node is used to hold any additional information about the source or the source itself of the event
Failure Object	Entity within the source that is primarily responsible for the event
Source	This property holds the information about the source of the event
Date/Time	The time at which the selected event was created

Alarms of a particular severity might be viewed in a screen such as in Figure 12.19.

Alarm details can be viewed for any particular alarm. Alarm details or properties are explained in Table 12.3.

Annotations can be added to an alarm for future reference. For example, the solution for a problem resolved by the user can be entered in the "Alarm Annotate". This

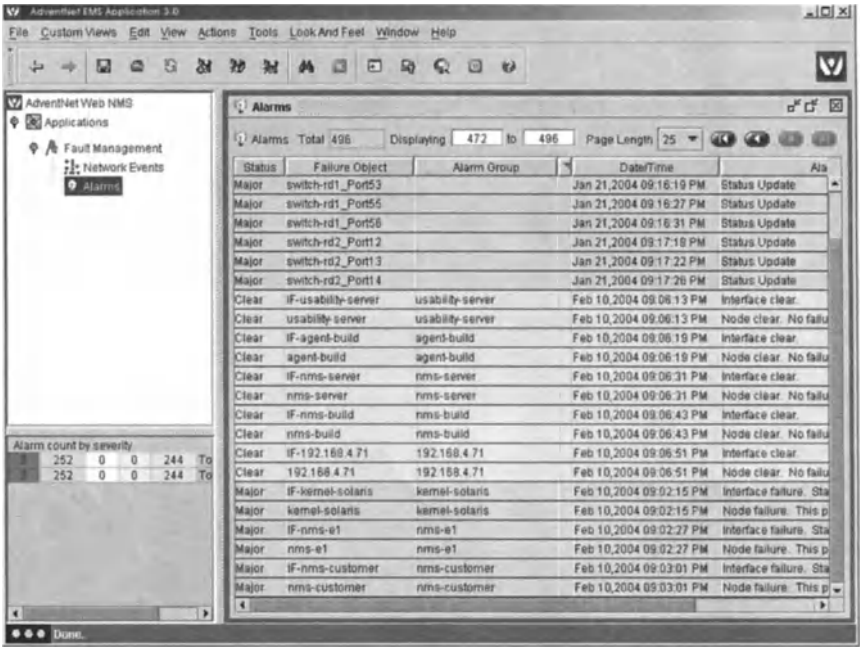


Figure 12.18 Alarms list client.

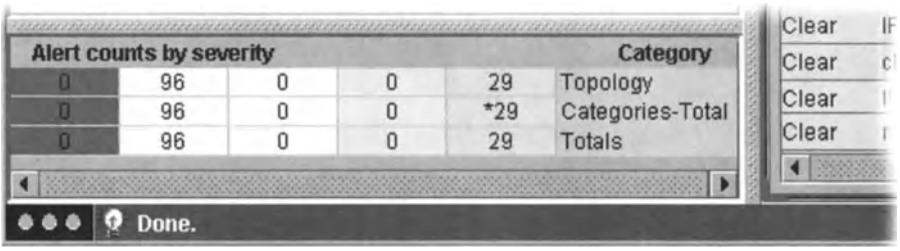


Figure 12.19 Alerts counter according to severity level.

Table 12.3 Alarm details.

Property	Description
Message	Stores additional information about the alarm
Failure object	The entity which has caused the alarm within the source
Source	Source of the alarm
Owner	Owner with whom the alarm is associated
Category	Alarm category
Created	Date and time when the alarm was first created
Modified	Date and time when the alarm was last modified
Group	Group of which the alarm is a member
Severity	Criticality of the alarm
Other alarms in this group	Other alarms present in this group

Figure 12.20 Alarm details screen for an object that has experienced failure.

enables other users to solve the same problem with less effort, by just reading the "Alarm Annotate". System or user actions on the alarm are also recorded automatically. User actions such as annotating and selecting appear under "Alarm Annotate".

The status of alarms added, updated or deleted can be viewed in the alarm history. The FMS fault management system automatically handles updating, clearing and deleting "Alarm History" records.

12.11.3 Alert Grouping

FMS provides a client-side mechanism of dynamic grouping of alerts and viewing of alerts with specific criteria. If the "Group View Mode" is set to "max", then alerts are grouped and, based on the criticality of the grouped alerts, and alerts of maximum severity in any group are shown in the client. If the value is set to "latest", then alerts are grouped and, from the grouped alerts, the latest updated alert in a group is shown in the client.

Suppose a custom view named "alert grouping" is created, with the "Group View Mode" criteria set to "max" and the three rows highlighted in grey in Figure 12.21 are grouped as a unit. From the group formed, an alert of maximum severity in that group is shown in the client. Similarly, when the "Group View Mode" criteria is set to "latest",

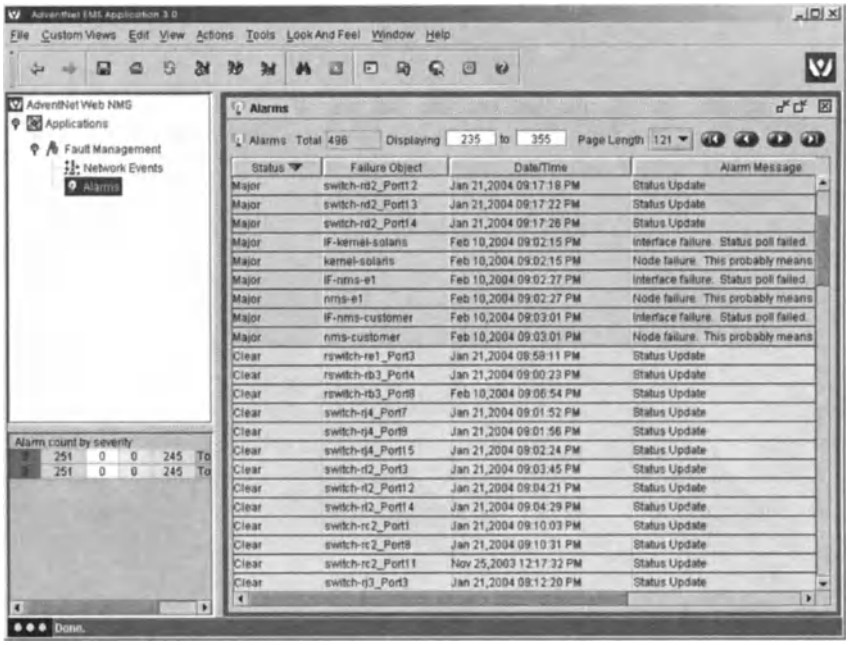


Figure 12.21 Alarms grouped by severity. Note the alarm count by severity at the lower left.

alerts are grouped based on their similarities and the latest updated alert in that group is shown in the client.

12.11.4 Tree Operation

A node can be added anywhere in the tree by specifying the parent of the node to be added. The menu found in this panel for the parent and the child need not be the same. That is, the child can have either the same tree popup menu as that of the parent node or any other tree popup menu. This feature fulfils the "Any Node Anywhere" concept whereby different types of node can be added under any given parent node.

User-defined properties for a new node can be added, modified or deleted at any time. Through this functionality, it is possible to add the key and the value into the table for the particular node. For example, an alarm node can be added as a network event. The "parent node" would be network events. The tree name might be "Test Node" and the panel name "Test Node". The output from this example is shown in Figure 12.22.

When adding a node, user-defined properties can be specified which are added to the database. For example, a user-defined property might be added under the attribute FRAME TITLE and with value *Frame Title*. Any number of user-defined properties can be added.

12.11.5 HTML User Interface

The HTML client is a browser client, that is, it runs in Internet browsers. It loads all HTML pages and is faster than the Java Client. We discuss some of the features of FMS as they function through the HTML interface.

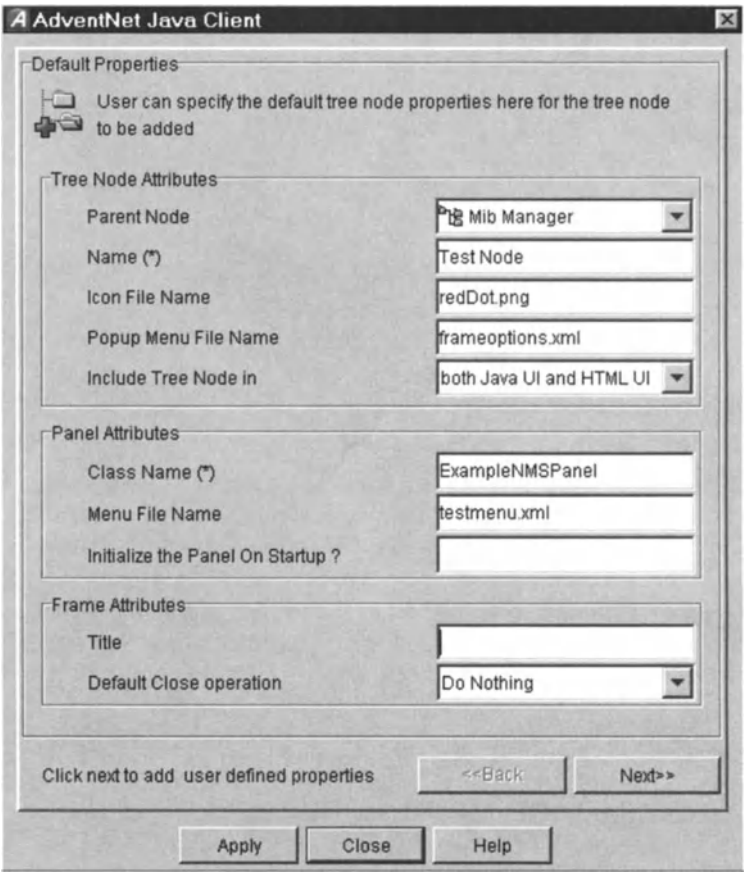


Figure 12.22 Tree node properties.

12.11.5.1 Network Events in HTML

Events, as the complete set of notifications from the network, are of two types: external and internal events. External events are actions or notifications that are external to the fault management system, for example traps which are basic notifications from the SNMP agents or management applications. Internal events, on the other hand, are notifications which are generated when FMS is integrated with any other network managements system. Examples of internal events are threshold events and status events.

Creating custom views explains the properties used for filtering required events. It also shows the column headings that can be selected for the custom view. The age of an event is a criterion to filter events based on their last-modified time.

12.11.5.2 Event Operations in HTML

Parsing a trap to generate an event is an important event operation. The agent running in any device can be configured to send some traps to the manager. We shall discuss traps and why traps are converted into events.

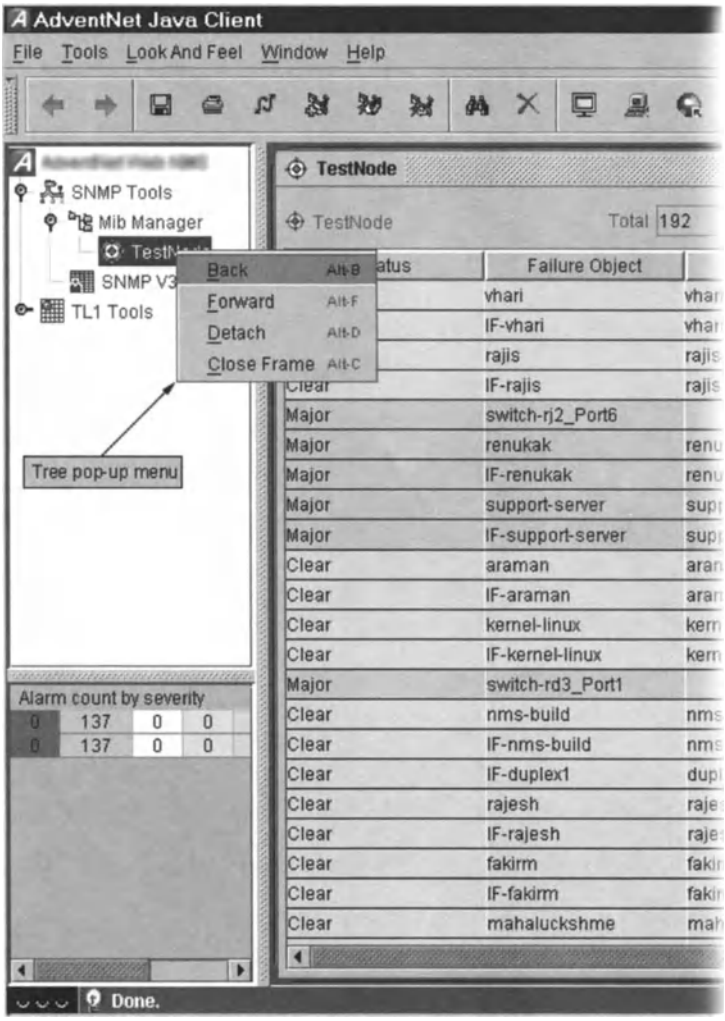


Figure 12.23 Test node.

Traps contain cryptic information, which cannot easily be understood by users. Therefore, a trap parser is required to translate or parse these traps into understandable information for easy identification.

A trap parser converts information from any device into meaningful events, that is, identification of the affected element. For example, a "link down" message may have the router and the interface as the failure object. Using the trap parsers, this can be converted to a more understandable event where the failure object is uniquely identified. This is important because it allows the tracking of the fault or failure. Trap parsers option under the "Edit" menu helps in detailed parsing of the input data to identify fully the failure object of the event. The trap parser can be edited through an HTML interface.

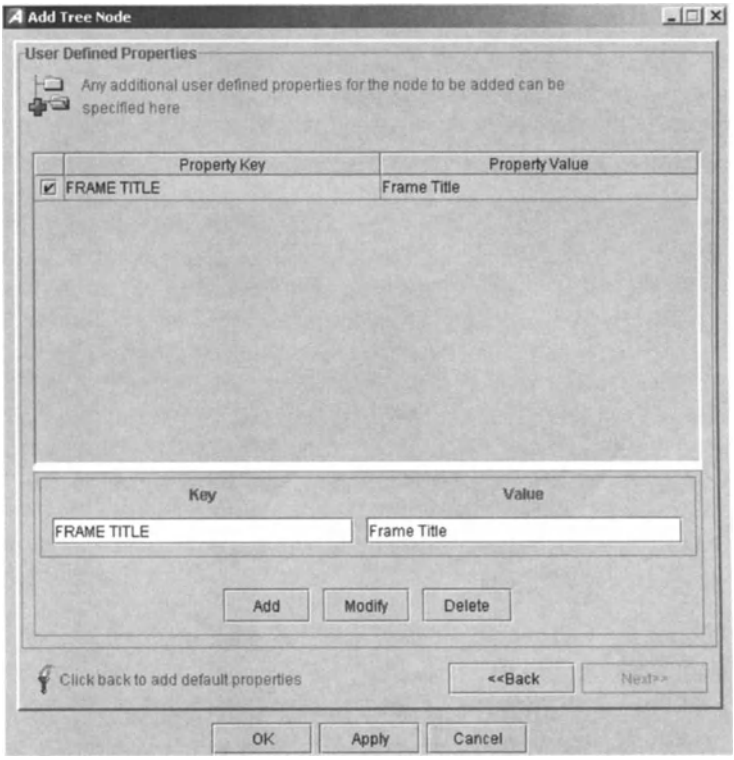


Figure 12.24 Adding a tree node.

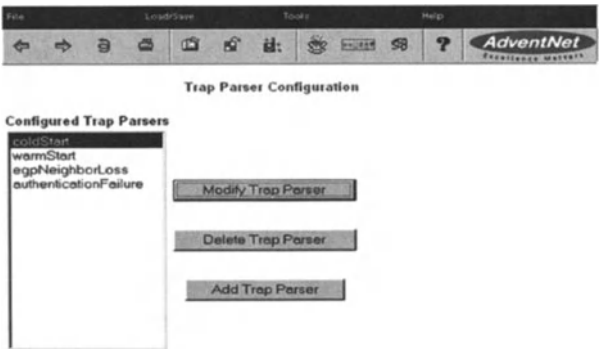


Figure 12.25 Trap parser configuration through HTML user interface.

The HTML screen in Figure 12.25 shows all trap parsers that have been configured, with radio buttons for every trap parser. By default, four trap parsers are configured. A new trap parser can be added in the HTML form shown in Figure 12.26.

Trap parser parameters are given in Table 12.4.

Trap parsers can also be loaded from MIBs or from files.

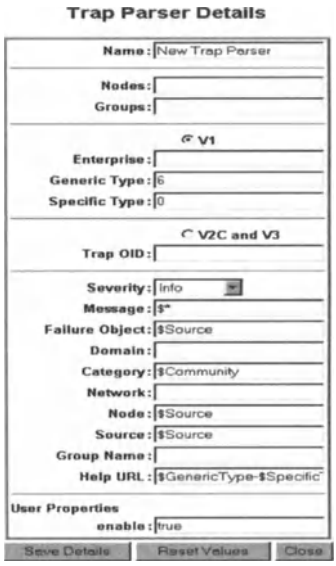


Figure 12.26 Adding a new trap parser through HTML interface.

Table 12.4 Trap parser parameters.

Property	Description
Name	The name to be assigned to the trap parser
Nodes	The nodes at which traps should be listened for
Groups	The groups from which the trap should be listened for
Enterprise	SNMP enterprise identifier in the trap
Generic Type	Trap types include cold start, warm start, link down, link up, authentication failure, enterprise specific
Specific Type	The trap parser will only be applied to a trap if the type matches
Trap OID	The trap's Object ID
Severity	The Severity to be assigned to the Event of the trap. One of the following must be assigned: All, Critical, Major, Minor, Clear, Info. The default is Info
Message	The Message to be displayed by the Event of the trap
Failure Object	The name of the failed object
Domain	The Domain name of the trap
Category	Category to which the trap should belong
Network	Network to which the object generating the trap belongs
Node	The node that generates the trap
Source	The source from where the trap is generated
Group Name	The name of the group to which the trap will be assigned

12.11.5.3 Alarms in HTML

Alarms are simply messages generated by the NMS that need the users' or administrators' attention. The devices connected with the NMS create various events which are converted/parsed into alarms, based on their significance.

By default, the alarms are classified into five categories: critical, major, minor, warning and clear. This classification is based on the severity/criticality of the alarm

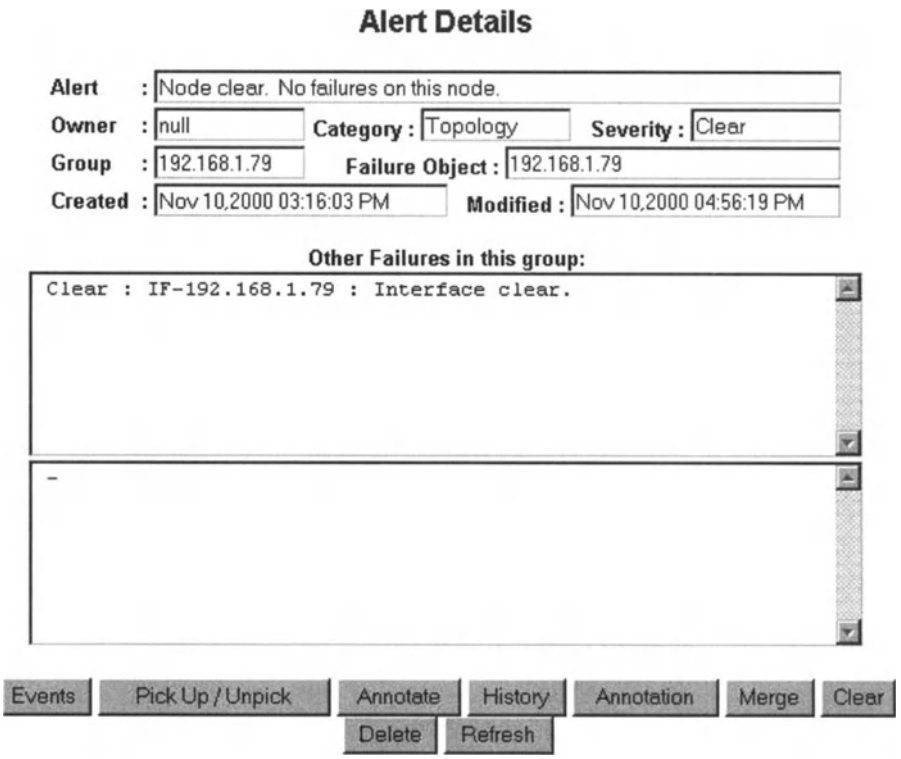


Figure 12.27 Alert details in HTML interface.

that is generated. "Alarm Viewer" is the main user interface tool used by network administrators. Some of its functions, such as sending alarms, problem management and work assignment, are central to the NMS. The operator can navigate the alarms database.

Alarm details include a brief description about the device which has failed, the owner with whom the alarm is associated, the date and time of the alarm and the alarm severity.

The alarm categories are assigned with distinct colours for easy identification. There are default colours – critical, red; major, brown; minor, yellow; warning, aqua; and clear, lime – but the colours are configurable.

Details regarding a particular alarm such as message, owner and time are maintained by the NMS. These details are seen in a form when the user clicks on a particular alarm. Figure 12.27 illustrates how these are shown in HTML format.

When an alarm occurs and is displayed, it must be handled and the problem associated with it should be solved. FMS allows alarms to be handled within the software package; functions for handling alarms include:

- viewing corresponding events
- attending to the alarm
- annotating the alarm

- viewing the alarm history
- viewing annotation for the alarm
- viewing annotation and history together
- refreshing the alarm
- clearing the alarm
- deleting the alarm.

12.12 Administration

FMS gives network administrators sophisticated event management, including alerting, automated actions, event correlation, trap/event/alert filtering and trap/event parsing.

For network management staff, fault detection is an online process that provides an indication of malfunctioning. Fault detection and notification are two functional areas which should identify problems and inform the system administrator. FMS handles error conditions, that cause users to lose the full functionality of a network resource. It also provides network administrators with event management functionality such as to generate alerts, execute automated actions, correlate events, filter traps/events/alerts and parse traps/events parsing, to detect, isolate and repair malfunctions in the network and its control subsystem.

12.12.1 Filtering and Processing Notifications

The trap port is the port at which the FMS listens for SNMP notifications. By default, it is configured to listen at port 162. Trap ports can be dynamically configured through the Fault Management Client.

12.12.1.1 Configuring Trap Filters

Traps need to be processed before they are converted into events. Trap filters are used to modify the properties of the incoming trap. Configuring filters at runtime eliminates the problem of restarting the server as it is done for configuration through configuration files. Trap filters can be configured at runtime.

The type of trap, whether it is SNMP v1, v2 or v3, is specified in the runtime configuration wizard for configuring trap filters. Similarly to SNMP v2 traps, the trap OID is used for the SNMP v3 traps. The trap filter is configured based on the type of trap. The user gives input values either as generic type (GT) or specific type (ST). When specifying ST trap values, the OID value is given.

12.12.1.2 Configuring Trap Parsers

When traps are received by the fault management system, they are filtered and the properties modified using trap filters. If the trap filter returns a trap PDU, then the traps will enter the trap parsers. Parsing traps generates meaningful events. The trap parser user interface configuration feature generates events from the SNMP traps and parsers configured for SNMP v1, v2c and v3 traps.

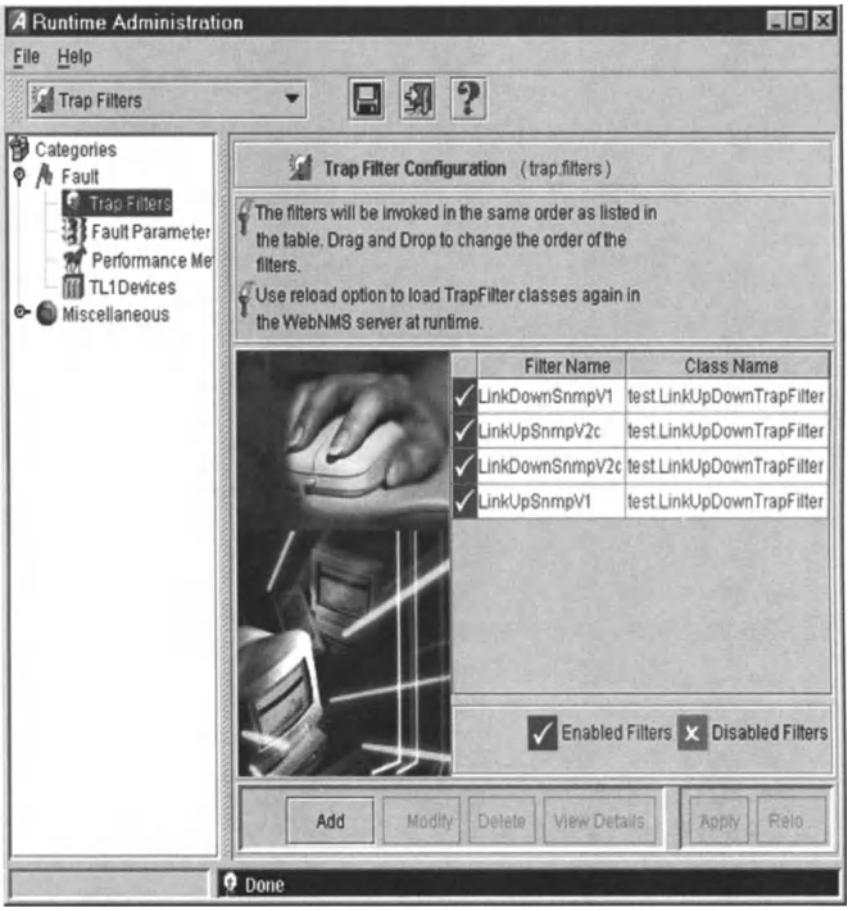


Figure 12.28 Trap filter configuration.

The fault management system allows dynamic configuration of trap parsers from the client. This can be done from the event browser of the fault client.

Rather than specifying the match criteria manually, an MIB with defined traps can be translated to trap parsers.

With FMS, the user can configure SNMP v1, v2c and v3 traps. These matching criteria determine the type of traps that will be parsed by any given trap parser.

The configuration screen allows the specification of match criteria in a set of fields. Using the values in these fields as criteria, the incoming SNMP v1 traps can be matched. An incoming trap should satisfy all the criteria specified. Upon satisfaction of all criteria, an event will be generated as configured in the trap parser.

The FMS can be used to configure trap parsers for SNMP v2c and v3 traps. The criteria that can be given are:

- **Nodes:** This field is used in order to apply a trap parser when FMS receives a trap from a particular set of nodes. This was already explained in setting match criteria with nodes for v1 traps.

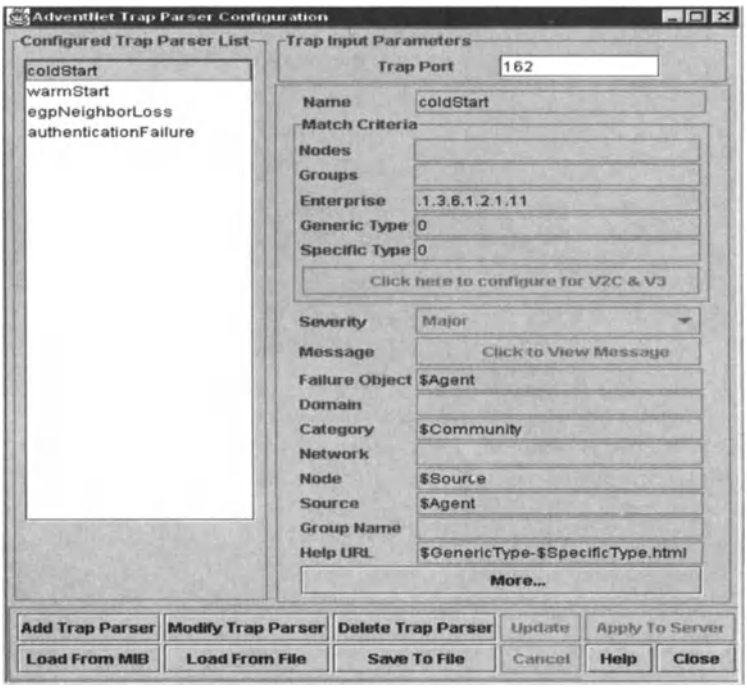


Figure 12.29 Trap parser configuration.

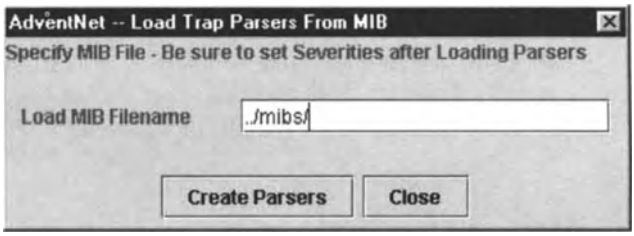


Figure 12.30 Loading trap parsers from an MIB is an easy way of configuring a trap parser.

- **Groups:** The managed objects can be grouped together and given as match criteria in this field.
- **Trap OID:** The trap type of each SNMP v2c and v3 trap could be uniquely identified by using the trap OID that is provided with the trap PDU. This trap OID can be used to define the matching criteria. The trap parser would be applied only to traps whose trap OID starts with the given trap OID value.

Event object attributes are defined by the values given in the trap parsers. If the user knows something of the event they are seeking, then selecting some values is useful in identifying the failure quickly. Fields include:

- **Severity:** Sets the state of the event, which determines the severity shown in the event browser. This severity determines how a fault/alert may be affected by this

event. For a given failure object, the severity specified for the event will be responsible for the severity of the corresponding fault/alert.

- **Message:** Corresponds to the "text" field of the event object. The value specified here will be copied on to the "text" field of the event object created by this trap parser.
- **Failure Object:** The key of the event object. This field affects how the event is processed by the fault management system. Appropriate processing by the trap parser will ensure that the failure object reflects the exact problem as notified. This can be used for tracking down the problems quickly and identifying the objects involved, rather than simply reporting raw events.
- **Domain:** Used to specify the domain name for the event. This field may or may not be used depending upon the application.
- **Category:** Used to categorise events and alerts.
- **Network:** Used to specify the name of the network from which the event was raised. This may or may not be used, depending on the application.
- **Node:** Enables using a node value for the event. This may or may not be used, depending on the application.
- **Source:** Typically the source name for a given event.
- **Group Name:** Helps to group meaningful events – and hence alerts – in a meaningful way.
- **Help URL:** The URL that refers to the file which provides detailed help for a given event.

While specifying values that are output to the event object generated, the user can use the information in the incoming trap PDU. The information in the trap PDU is accessed using tokens which represent the values of the trap fields. Incoming traps are parsed by configuring a list of trap parsers. Only one trap parser can be applied to a given trap. The match criteria in the parser determine whether or not a specific trap matches the trap parser.

12.12.2 Events

To identify the failure object corresponding to an event, traps and input events should be parsed and necessary details given. This parsing is useful in defining other fields of the event or alert. FMS allows detailed parsing of the input events by event parsers.

FMS allows dynamic configuration of event parsers from the fault management client. This can be done from the event browser of the fault client through the event parser configuration screen (see Figure 12.29).

The event parser configuration window has several parameters that help in configuring an event parser. The first parameter is the "Name" field to help identify the parser distinctly, when managing a list of parsers. The remaining fields of the event parser can be viewed as a table, where the rows correspond to the event object attributes. The output of any event parser is an event object, which will be the modified instance of the incoming event based on the definitions configured in the event parser. If any field is empty in the definition column, then that particular property of the outgoing event will be an empty string. Additional criteria can be configured

Figure 12.31 Event parser configuration details.

based on other properties of an event, which could include user properties aside from the given default event properties.

The first column in Figure 142 is for the "name of the property", which could be any valid event property. The second column specifies the matching criteria. The third column specifies the pattern in which the incoming event property must be tokenised. The fourth column defines the output value of the corresponding property in the resultant event.

The matching criteria determine whether the event will be parsed by the given event parser. The output of the event parser is an event object, which will be the modified instance of the incoming event. The attributes of the event object are defined by what are specified in the event parser.

If an event is generated by a trap, then the associated trap PDU reference will be maintained in the incoming event object. If the event object contains the trap PDU reference, then the properties of the trap can be used within the event parsers.

When an event arrives into the fault management system, the event parsers list is checked to see whether the incoming event satisfies the match criteria of the event parser. If the event parser matches, the event is passed through the corresponding event parser. The outgoing event from the parser is then matched with the remaining set of parsers. If there are further matches, the event is passed through those parsers. The process continues until no more parsers remain to be scanned.

When an event is raised, the user may need to execute certain actions. FMS provides the event filters to perform automatic actions – such as sending an e-mail, suppressing the event or generating traps – upon occurrence of an event. The user can also execute some classes, such as a custom filter action, when an event is generated. These event filters can be configured either dynamically from the fault management client or by setting match criteria and specifying the actions to be executed when an event matches the filter. Types of filter actions are:

- **Suppress filter actions:** Suppress events that match the filter criteria, either all events or multiple events of some type within some given interval.
- **Send trap actions:** Send SNMP v1 traps for events matching particular filter criteria. The traps can be configured to have specified event data and can be configured to be sent to any specified host.

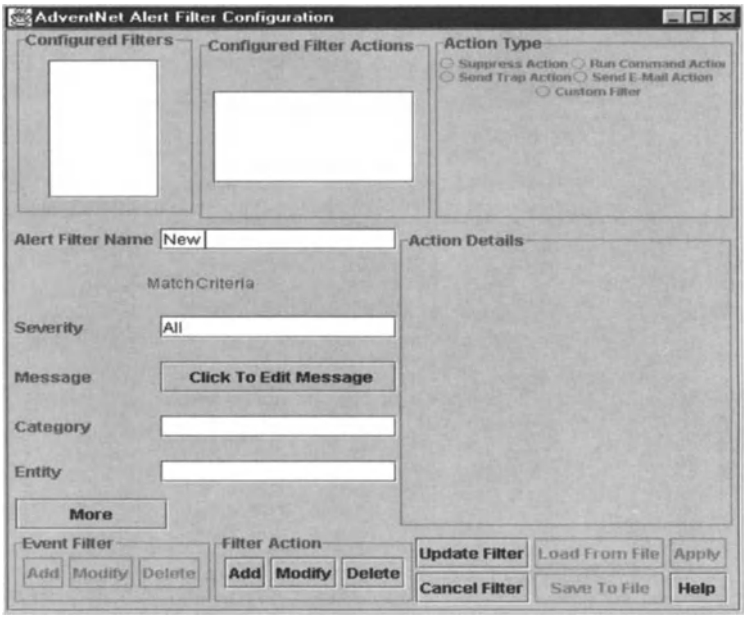


Figure 12.32 Alert filter configuration.

- **Send e-mail actions:** Send an e-mail for events matching filter criteria.
- **Run command actions:** Run a command on the server for events matching particular filter criteria. These can be used to page someone, send an email or to execute any desired command.
- **Run custom Java class filter:** Write Java code to perform actions and configure them to be applied to events matching particular filter criteria.

For a given event, all filters in the event filters list are tested to see whether they satisfy the match criteria. If the event matches an event filter, then the actions associated with that filter will be executed. After execution of the actions specified in the matching filter, the event will be checked for a match with the remaining filters in the list. If a match is found, the corresponding event filter will be executed, and so on.

12.12.3 Alerts

Alert filters are used to filter and modify the properties of the incoming alert. When an alert is received by FMS, the alert filters will be applied if the match criteria matches the alert properties. It is possible to configure certain actions on occurrence of an alert, such as sending an e-mail, or to apply custom code. FMS allows dynamic configuration of alert filters from the client allowing actions such as suppressing multiple alerts and sending traps.

An alert filter can be configured by setting the match criteria and specifying the actions to be executed when the alert matches the filter. The five actions mentioned

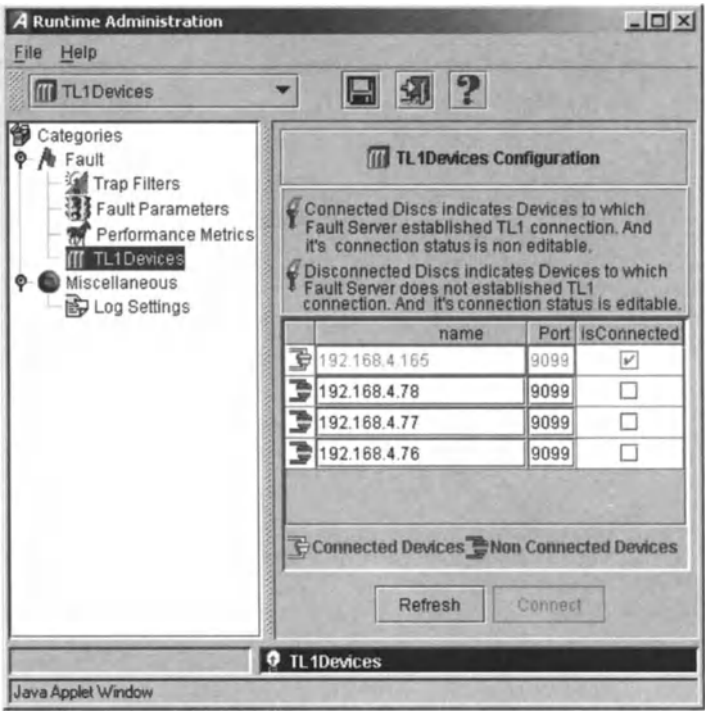


Figure 12.33 Administration of TL1 devices.

above are supported as alert filter actions. Configuration of an alert filter involves setting alert filter parameters and defining filter actions.

12.12.3.1 Configuring TL1 Devices

The startup parameters for TL1 devices can be configured using a particular configuration file. The following details are given in the user interface of TL1 configuration:

- The names of the TL1 devices.
- The port number
  —the port through which connection is made, if a session is established
  —the port(s) as specified, if a session is not established.
- The status of the TL1 device: whether or not a session has been established.

The devices which have already established a session are indicated by green. Those devices which have not established a session by then are highlighted.

12.12.3.2 Configuring Command Line Arguments

As already mentioned, dynamic configuration allows users to configure the event and alert parameters at runtime without having to restart the server. The *Parameter*

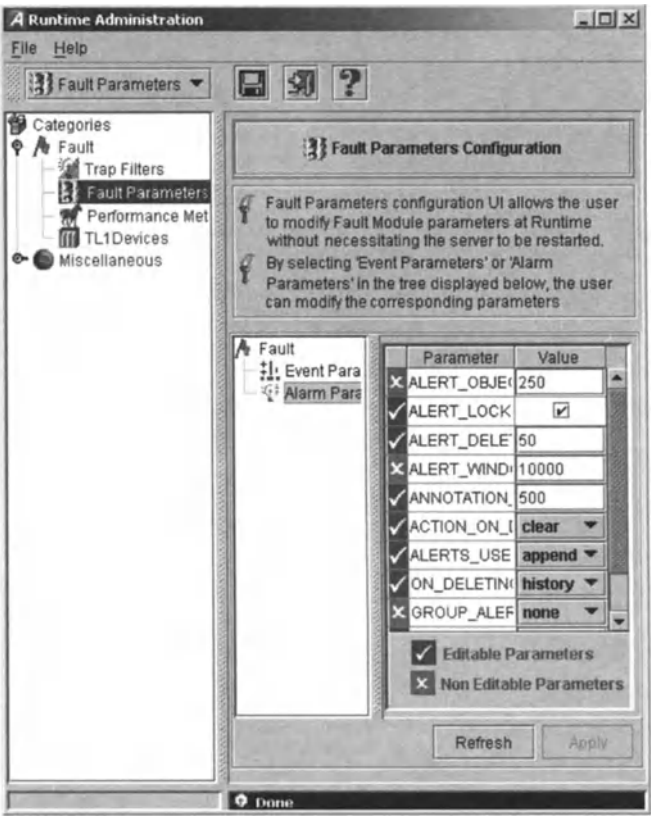


Figure 12.34 Configuring fault parameters.

column in Figure 12.34 shows the configurable parameters and the *Value* column gives the values for the corresponding parameters. Those parameters which cannot be configured dynamically are highlighted. The event and alert related parameters can be configured separately by choosing the corresponding item in the tree.

12.12.3.3 Determining Performance Metrics

Determining performance metrics at runtime eliminates the problem of restarting the server whenever a change is made. The user can measure event performance separately from alert performance by choosing the appropriate item in the tabbed pane.

There are two options for measuring the performance for both events and alerts:

- **Time-based:** Specifying the number of events/alerts and measuring the time taken to process that many events/alerts.
- **Count-based:** Specifying the time interval in minutes and measuring the number of events/alerts processed in that interval.

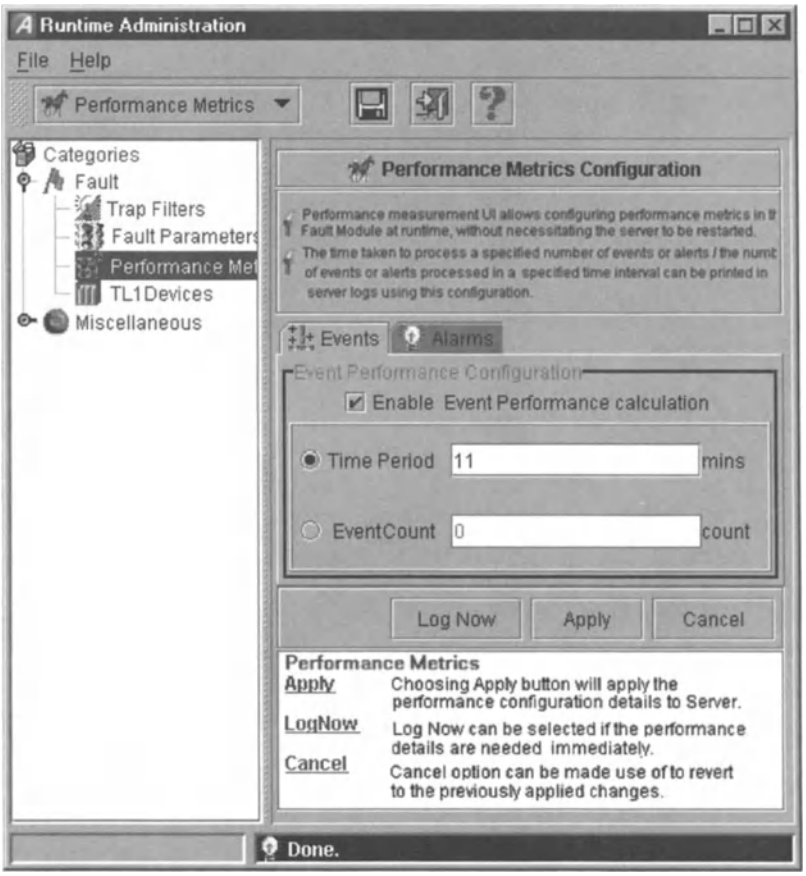


Figure 12.35 Configuring performance metrics.

12.13 Conclusion

Fault management is largely a matter of alarm and alert management. Indeed, some vendors use the term "alarm management" rather than "fault management" in their OSS modules. The fact that network devices generate alarms when they are in a fault state allows OSS for fault management to focus on the abstraction, rather than being involved in the job of examining devices and checking whether they are in a fault state. This functionality will become more advanced as devices are increasingly able to repair themselves or be aware of what type of corrective action to take if they are in a fault state. It will be increasingly rare for a human to be required to process a fault alert.

Fault management is also largely a matter of information management. An OSS vendor is not too concerned about how to go about fixing an NE. Rather, they merely create a hub to where information on faults is sent and from where instructions on how to fix the fault are sent out. The OSS then allows the service provider to

implement their own fault correction procedures according to their particular business and the needs of the network infrastructure that they have in place. Increasing intelligence in NEs will lead to some faults bypassing the OSS altogether. For example, a device that is in a fault state might know to contact a field engineer directly for repairs and only inform the central OSS if there is some ambiguity in the nature of the fault. However, it is likely that, even in that situation, faults would continue to be logged with the OSS to maintain a central record of how much maintenance time particular NEs demand and how frequently particular NEs find themselves in a fault state.

Many OSS applications allow complete control from an HTML interface. This means that a user needs only to be equipped with a web-enabled PC and a web browser in order to do their job. This reduces installation and support costs for the OSS and also allows distributed processing. From a human resources management viewpoint, web enablement makes it easy to spread workload over human resources in a way that is convenient given the peculiarities of the workforce of any service provider. See also Chapter 14 for more on recent advancements in web-based network management.

13 Traffic Management

13.1 Introduction

Traffic management is essential for the efficient flow of traffic through a network. This means that it is basic to the moment-to-moment running of a network. Traffic management is a key factor in QoS. Traffic management functionality provides advance warnings of performance-critical conditions so that network managers can re-route traffic before it affects QoS. Part of traffic management is real-time detection of network failures and correction of unavoidable network failures. This may allow users to be forewarned or otherwise proactively address the situation to minimise negative effects on QoS.

A central purpose of any traffic management system is to provide visibility into the distribution of network traffic. This visibility must span both time and location. Information on distribution of network traffic is a prerequisite for more efficient use of existing equipment.

If a traffic management system can operate across heterogeneous environments, then labour and operational costs are reduced because the information is available in a single location. Further, the single user interface to such a unified traffic management system reduces training costs.

A traffic management system also provides the more usual benefits that are derived from any effective network management software functionality. These benefits are that

- increased customer confidence and usage leads to higher revenues
- reduced customer churn decreases associated revenue loss
- higher call completion rates result in increased revenues.

These are often mentioned in the marketing literature of OSS vendors. Often, traffic management functionality is not entirely built into the OSS. Rather, a performance or fault management OSS module relies upon a hardware-based traffic management device. However, we include this chapter owing to the integral importance of traffic management to achieve the aims of OSS software, however traffic management functionality is provided.

K. Misra et al. (eds.), *OSS for Telecom Networks*

13.2 What is Traffic Management?

All traffic management systems must monitor network traffic. However, they differ in the way in which they act, when they act and how they act. The most common time to act is when a pre-configured threshold is reached, triggering an alarm. The system will then apply controls to optimise traffic flow, and possibly re-route it. The idea is to detect a network problem *before* it impacts a service – and this will be the mandate of a network manager as far as traffic management is concerned.

Real-time QoS alarms provide an exception-based mechanism to alert service providers of potential problems. When a (pre-defined) threshold is reached or crossed, the resulting alarm provides an early warning for a potential network problem. Ideally, a threshold should be set to warn of a *potential* problem so that the problem is avoidable with advance notice to network engineering staff.

An exception will induce the network manager to use traffic controls to modify the network state. Commands can be sent to the network elements individually or can be sent in groups. In addition, they can be sent to single or several exchanges. Newer traffic management systems will allow greater automation so that responses to threshold-triggered alarms can be pre-programmed, thus allowing potential problems to be averted without human intervention.

13.3 Traffic Management in ATM Networks

The greatest benefit of ATM is the well-defined way in which it manages traffic. The ATM Forum specification on traffic management is complex, and an explanation of some of the basics will now be given.

ATM offers five classes of service, and any ATM connection must satisfy a sustainable cell rate peak cell rate and maximum burst size associated with a particular class of service. These three parameters are important in describing the profile of an ATM connection.

ATM traffic management mechanisms include Call Admission Control (CAC), Resource Management, Usage Parameter Control, Network Parameter Control, Priority Control, Traffic Shaping and Explicit Forward Congestion Indication. These mechanisms collectively help define the quality of ATM service.

Under ATM Forum traffic management specifications, CAC asks for a specific QoS. The network then refers to a specific list of performance requirements for the connection, including QoS objectives for Cell Loss Ratio (CLR), Cell Transfer Delay (CTD) and Cell Delay Variation (CDV).

CLR is the ratio of lost cells to total transmitted cells. Cells can be lost when an ATM switch malfunctions, but usually cells are explicitly discarded for non-compliance or in response to network congestion. Buffer management is the strategy for deciding how and when to discard packets.[50]

ATM network buffers address congestion at the cell level. The peculiarities of ATM networks demand that there be short queues within the network. There should

[50] HJ Chao and X-L Guo, *Quality of Service Control in High-speed Networks*. Wiley, New York, 2002.

also be a minimum number of buffers to ensure that the required cell loss rate is achieved for each virtual channel. Cells can be discarded selectively, based on their priority.

The three strategies for discarding cells are:

1. Complete buffer sharing with pure pushout – Cells are permitted to pass into the network for transmission until the buffer is full, at which time a new high-priority cell can push out a buffered low-priority cell and new cells push out buffered cells of equal priority.
2. Partial buffer sharing with nested sharing for different loss priorities – A threshold is assigned to each priority and cells of a particular class are allowed to enter the buffer only if the current queue length is less than the corresponding threshold.
3. Expelling policy – Cells are squeezed out by cells of higher priority and when the number of high-priority cells exceeds a threshold, then low-priority cells at the head of the queue are expelled until high-priority cell reaches the head of the queue and is then transmitted.

CTD is the elapsed time between a cell's exit at the source and its entry at the destination. Subscribers using CBR or Variable Bit Rate – real time (VBR-rt) service categories need to specify this parameter. An IP router attached to an ATM network might delay some cells in order to reduce the peak rate and rate variance without affecting throughput. A Motion Pictures Experts Group (MPEG) code operating in a situation where delay is not a problem might operate in CBR mode.

CDV, sometimes called "cell jitter", compares the inter-cell traffic departure of a given connection with its inter-cell arrival. Subscribers to CBR or VBR-rt services must specify this value as a parameter of their desired QoS.

Each of these QoS parameters is established in the traffic contract between a subscriber and the ATM network. Conformance for CBR, VBR-rt, VBR-nrt and UBR connections are given in the "conformance definition". Provided the subscriber sends ATM User-to-Network Interface (UNI) traffic which conforms to ATM traffic specifications, the network will deliver the QoS that was negotiated.

Particular difficulties arise in ATM traffic management because ATM networks have small cell size, high speed and little control overhead in cells. The two main ATM traffic management specifications are:

- Specification I.371 by the ITU-T
- Traffic Management Specification 4.0 by the ATM Forum.

Tools for controlling congestion in packet-switched and frame relay networks are not appropriate for ATM networks because:

- ATM traffic is generally not amenable to flow control. For example, voice and video continue to generate and pump out cells, oblivious to a congested network state.
- The time in which cells are created and the time in which it is necessary for cells to be transmitted, particularly for real-time applications such as voice and video, is less than delays involved in propagation across a network. Therefore, feedback control is not viable because the time that it would take for messages on network

state to get back to a traffic management system, for the system to transmit instructions and for traffic to be re-routed according to those instructions would cause delays which degrade QoS. This would defeat the main purpose of traffic management.

- ATM networks enjoy a range of transmission speeds (kbps to Gbps). It is rare to find an implementation of a congestion control schema which accommodates such a wide range of speeds. Congestion control schema are generally suited to one or other end of the spectrum.
- While ATM networks accommodate a variety of transmission speeds, they are also used for a wide variety of applications. In particular, the diverse applications generate diverse traffic patterns. The most obvious different traffic patterns are the steady, high-rate flow of real-time applications versus data applications whose flow is not so constant. Traffic whose rate is not constant is said to be "bursty". Note that video compression can make bursty the real-time application of video.
- Owing to high switching speeds and high transmission speeds, an ATM network is generally fairly volatile even under normal operation. Designing a control scheme appropriate to ATM would require some very clever engineering, as knee-jerk responses to wide fluctuations might cause unnecessary, and wasteful, re-routes.

The main techniques for ATM traffic management are:

1. resource management using virtual paths
2. connection admission control
3. usage parameter control
4. selective cell discard
5. traffic shaping.

Discussion of points 3 and 4 will be omitted because they are technical and beyond the scope of this book. Since ATM is a well-established technology, the interested reader will easily find an abundance of reading materials on these topics.

13.3.1 Resource Management Using Virtual Paths

Network resource management is about separating traffic according to service characteristics. Virtual paths are used by the ATM Forum for traffic control based on network resource management.

A Virtual Path Connection (VPC) groups similar Virtual Channel Connections (VCCs). Characteristics of the VPC are inherited and enjoyed by the VCCs. In a user-to-user application, the VPC extends between a pair of UNIs. The user must ensure that the total demand from the VCCs can be accommodated by the VPC. In a user-to-network application, the VPC extends between a UNI and a network node. The network is aware of and accommodates the QoS of the VCCs within the VPC. In a network-to-network application, the VPC extends between two network nodes, and the network is aware of and accommodates the QoS of the VCCs within the VPC.

The main QoS parameters are cell loss ratio, cell transfer delay and cell delay variation. These are all affected by the resources allocated to the VPC. Where a VCC passes through multiple VPCs, the performance of the VCC depends on that of the consecutive VPCs and how the VCC is handled by nodes such as switches or concentrators between and on VPCs. Handling by the nodes affects QoS through nodes' processing speed and cell prioritisation.

An OSS should be aware of the capacity assigned to the VPCs and of the action and performance of the nodes. Indeed, in designing VPCs and VCCs, the user should be given information and assistance based on the software's knowledge of the performance and capacity characteristics of the network components.

13.3.2 Connection Admission Control

Connection admission control protects the network from excessive loads. When a user requests a VPC or VCC, the user specifies the traffic characteristics. The user selects a QoS from the QoS classes that are available for the network. The network accepts the connection only if the resources are available to provide that QoS. Acceptance of the connection creates a "traffic contract" with the user: the user is bound to ensure that the traffic does not exceed the level specified and the network is bound to provide the QoS promised.

ATM connections are policed by network operators at the UNI and Network–Node Interface (NNI) according to traffic contracts. If a connection does not adhere to a traffic contract, then action is taken against the cells which violate the traffic contract. Such action includes discarding or reducing the priority with which they are tagged. A multiplexer with a built-in traffic-shaping function is used to delay violating cells in a buffer and transmit them to the next node once the cells adhere to the prevailing traffic contract.

The Peak Cell Rate (PCR) is the maximum rate at which cells are generated on a connection. CDV expresses the variation in the rate at which cells are generated. The Sustainable Cell Rate (SCR) is the maximum rate of an ATM connection averaged over the duration of the connection. Burst tolerance is the maximum variation in the rate at which cells are generated with respect to the SCR. In short, the SCR and burst tolerance are analogous to PCR and CDV with respect to a throughput average taken over the duration of the connection.

Traffic parameters are summarised in Table 13.1.

Table 13.1 Traffic parameters.

Traffic parameter	Source	
	Constant bit rate (CBR)	Variable bit rate (VBR)
Peak cell rate (PCR)	Relevant	Relevant
Cell delay variation (CDV)	Relevant	Relevant
Sustainable cell rate (SCR)	Not relevant	Relevant
Burst tolerance	Not relevant	Relevant

13.4 Congestion Control in ATM Networks

ATM networks promise large bandwidth and guaranteed QoS wherever there are effective traffic management schemes. Nortel, for example, has developed a method of testing the robustness of flow control algorithms and studied network topologies in which two different congestion control schemes co-exist. Both pure ATM and IP-over-ATM networks have been studied.

An Available Bit Rate (ABR) simulator is used to evaluate performance, which can overlay any set of Virtual Circuit (VC) routes on any topology. Nortel implemented five flow control algorithms, identified 10 performance metrics, studied key behaviours, source-rate oscillations and interference in rate control by switch nodes which are not bottlenecked and established criteria for the ability of an algorithm to perform well regardless of changes in topology, feedback delays and traffic.

After evaluating the interoperability of pairs of ABR flow control algorithms in diverse ATM networks, Nortel concluded that heterogeneous configurations with two different ABR flow control algorithms operating simultaneously cause users to receive disparate bandwidths if:

1. The algorithm at a bottlenecked switch is much more oscillatory than the algorithm at the non-bottlenecked switches.
2. At least one VC traverses only the more oscillatory algorithm and at least one VC traverses both algorithms.

13.5 Traffic Shaping

Traffic shaping or traffic conditioning is the enforcement of the rules on network packets based on QoS policies. In networks which allow specification of QoS, it is necessary to specify the traffic profile for a "connection" to decide how to allocate various network resources.

Traffic shaping ensures that traffic entering at an edge or a core node adheres to a specified profile, since the function is carried out at a node. In other words, traffic shaping is the forcing of traffic to conform to some specified behaviour. This is usually a worst case or a worst-plus-average case. For example, the worst case allowable might be 100 Mbps of data for a maximum of 2 seconds and the average allowed over any seconds might be 50 Mbps. Traffic shaping is about regulating data transmission – both the average rate and the intensity of bursts ("burstiness"). The most obvious use of traffic shaping is to reduce congestion, and it can be used ultimately to allow a carrier to achieve QoS goals and SLA requirements. This can be critical where services that are promised include real-time applications such as audio and video.

13.5.1 Traffic-shaping Methods

Traffic shaping is where the rate of traffic flow is adjusted or optimised to ensure quality parameters are satisfied within the limitations of the telecom network.

Examples of traffic-shaping methods include:

- reducing the peak cell rate
- limiting burst lengths
- reducing peak-to-peak CDV by appropriately spacing cells in time
- queue service schema.

Given some switches that are connected together, along with a signalling architecture that enables people to make calls, the problem is how to deliver the promised QoS. This requires a resource management strategy, since congestion is the greatest factor in data loss. Of course, costs need to be kept sufficiently low so that a profit can be made from the charging scheme in place. The most important traffic parameters proposed for resource management are mean bitrate, peak bitrate, variance of bitrate, burst length, burst frequency, cell-loss rate and cell-loss priority. These parameters are considered in three states: their actual values, their measured values and their effective values as experienced by the customer.

The functions involved in traffic shaping are:

1. **Acceptance Function:** Each switch can accept a virtual circuit request based on the traffic parameters declared by the customer. Acceptance is given if the resulting traffic mix will not cause the switch to fail in order to achieve its QoS goals.
2. **Policing Function:** Given that a switch at the edge of the network has accepted a virtual circuit request, it must ensure that the customer equipment fulfils its side of the traffic contract. The policing function estimates the parameters of incoming traffic and takes action if they measure traffic exceeding agreed parameters. This action could be to drop the cells or to mark them as low cell-loss priority.
3. **Charging Function:** This function computes a charge from the estimated and agreed traffic parameters.
4. **Traffic-shaping Function:** Traffic shaping occurs in the customer equipment. A common analogy is, if the policing function is the policeman and the charging function is the judge, then the traffic shaper is a lawyer. Like a lawyer, it allows the service to go as close as possible to the minimum QoS line in the sand without crossing it, and at the least cost. In particular, the traffic shaper uses information about the policing and charging functions to change traffic characteristics of the customer's stream to ensure lowest charge, smallest cell loss, etc.

In Chapter 6, we briefly introduced Service Level Agreements (SLAs). An SLA also specified the Traffic Conditioning Agreement (TCA) by the rules needed to realise the service which is specified in the SLA, and this specifies both what the service provider *and* client agree to do so that the promised quality of service is delivered. A TCA specifies classifier rules, traffic profiles and shaping rules that are to be applied to each traffic classification. A TCA encompasses all traffic conditioning rules that are explicitly given in the SLA, and also makes explicit the rules that must be satisfied to achieve the service requirements. In the TCA, the traffic profile comprises the properties of a traffic stream in time, that is, bits per second, burstiness specification. As a traffic stream enters the domain to which the TCA applies, packets are classified into a traffic aggregate by a filter on the relevant router at the edge of

the domain. The filter is associated with a traffic profile that specifies the information rate and how that rate is to be measured, the burst size.

An OSS might include functionality to manage the traffic-shaping functionality of network hardware. An example is the TEMPo product from Tenor.

Normally, all the users on a network with a single Internet connection get a share of the overall connection speed. Users demanding more data can be problematic because they get a bigger share of the Internet link and so can slow other users. In a managed office, this might breach an agreement with tenants who are paying for a specific connection speed.

In an Internet context, traffic shaping is simply controlling the speed of Internet traffic. For a managed office, some limit may be imposed on the usage of each tenant. Type of traffic, speed of traffic or the amount of data might be used to specify that limit.

The traffic smoothing function allows equal allocation of bandwidth to each Internet traffic flow. The result is that the available bandwidth is predictably allocated to each concurrent data transfer.

13.5.2 Traffic-shaping Devices

Several vendors offer traffic-shaping devices but their effectiveness requires managers to know what it is that they need to shape. Some traffic-shaping devices combine information gathering with the traffic-shaping function, allowing more effective traffic shaping. This illustrates what was once spoken by a wise network manager: "You can't manage what you can't see." The PacketShaper 2500 by Packeteer does this. In particular, it automatically classifies a variety of traffic types, for example the Real-time Protocol (RTP) and Resource Reservation Protocol (RSVP), each of which may run over one of many protocols. What is useful in the age of next-generation value-added services is that the device recognises traffic types using application signatures, as opposed to TCP and UDP port numbers. Non-IP flows can also be classified, such as flows based on IPX or on IBM's SNA or on AppleTalk.

Most traffic-shaping devices use queuing algorithms. The queuing technique prioritises traffic according to its class. Different classes of traffic are then shunted to different queues. Each queue is serviced with a frequency depending on the priority accorded to the traffic in the queue.

Active intervention in TCP connections between end stations allows more granular control over bandwidth and latency of TCP flows than does the use of queuing algorithms. In fact, most bandwidth management devices rely on queuing mechanisms for traffic management. Such active intervention involves actions such as altering TCP window size, altering maximum segment size and altering retransmission times.

The first step in the running of a traffic-shaping device is traffic discovery. The unit would be attached to an Internet access router and run, and a web browser can usually be used to view a network traffic profile. Such a profile will show current and peak rates for the applications running. It will also show the number of times each application has been instantiated. It should also give information on its success in achieving the user-specified rules.

A unit might, for example, be designed to control traffic headed to and from WAN pipes running at E1 (2.048 Mbps) or slower, so theoretically any higher rate would suffice. Even when handling flows consisting entirely of short (64-byte) packets, PacketShaper does not drop traffic until the load exceeds 55 Mbps. Latency is low and constant, with measurements around 500 μs.

After enabling traffic shaping and configuring PacketShaper to constrain bandwidth to T1 (1.544 Mbps), it is possible to send UDP traffic through PacketShaper at rates of up to 2.3 Mbps. This is approximately 50% faster than should have been allowed by the shaping contract.

A baseline test for traffic shaping devices determines the maximum number of concurrent TCP connections that can be handled. Tarantula from ArrowPoint Communications (acquired by Cisco Systems) opens TCP connections and downloads 246-byte HTML objects to verify each connection. Close to 8200 concurrent connections are possible before transfers start to fail, which is enough for a unit designed to police a T1/E1 link.

Another test scenario emulated multiple flows, comprising four file transfers, a RealMedia flow and two SAP R/3 purchase orders. Without traffic shaping, the file transfers consumed over 90% of the bandwidth. In this case, a traffic-shaping rule might restrict file-transfer connections to 300 kbps with the possibility of bursting up to 512 kbps if other applications allow. Response time for the SAP sessions should fall as a result and aggregate file-transfer bandwidths average between 300 and 512 kbps.

13.6 IP Traffic Management

Multiprotocol label switching (MPLS) is now a key technology in IP networks. MPLS implements a regime whereby packets are tagged so that they can each take the fastest route through a network and be reassembled in the correct order at the destination.

Various products are available for MPLS traffic engineering. One of the major applications for IP networks, and especially IP VPNs, is voice-over IP (VoIP). VoIP is valuable because of the lower cost of VoIP voice services. However, it is also challenging because it demands higher QoS than is usually needed in IP networks. Therefore, it is not surprising that MPLS traffic engineering products often give particular attention to the problem of QoS with a view to enabling effective VoIP services. We give an overview of some of the approaches that are being taken.

13.6.1 Cisco Systems, Inc.

Cisco recognises that the quality and priority of VoIP traffic in a converged network are a key concern. Fortunately, QoS functions and management tools have evolved so that network-wide services can be scaled, including protecting voice sessions in a converged network and monitoring SLAs. Traffic classification, prioritization, queuing and traffic and bandwidth management capabilities are available in Cisco routers, which focus on VoIP.

The CiscoWorks2000 QoS Policy Manager has a library of voice policies with preset QoS properties for routers and switches based on their role in the network, for example access versus core. VoIP SLAs are measured to provide information on latency, jitter and dropped packets, and to perform trend analysis.

13.6.2 Cisco and MPLS

Increasing deployments of MPLS networks for traffic engineering benefits are helping service providers to manage their bandwidth better in order to ensure high performance levels for VoIP and other traffic. Traffic engineering enables traffic forwarding decisions to be made based on a greater number of variables than those considered by traditional routing protocols.

Using MPLS features, service providers can guarantee bandwidth between endpoints across an IP-VPN that uses MPLS. Note that these benefits are often tied to the use of other products of the solution provider. For example, such MPLS features provided by Cisco are geared towards networks which use the Cisco 12 000 Series Internet routers and Cisco 7500 Series routers, although they do not necessarily *only* work in such networks. Such MPLS technologies allow providers to guarantee services in a way that is analogous to Layer 2 Frame Relay and ATM.

Cisco MPLS guaranteed bandwidth services' core features are:

- Traffic engineering which works at the control plane to allow multiple MPLS tunnels with different priorities between pairs of sites.
- The use of classification, marking, policing, shaping, queuing and congestion avoidance to compensate for congestion in the engineered tunnel.

MPLS fast rerouting establishes a backup network path with bandwidth and service quality requirements. If there is an interface or link failure, then traffic can be rerouted in milliseconds. The automatic bandwidth allocator dynamically varies the bandwidth of an MPLS traffic-engineered tunnel based on measured traffic load for extra-efficient bandwidth utilization.

13.6.3 Cisco and RSVP

RSVP is an IP service which allows applications to request QoS guarantees from a network. In data networks, if resources are not available somewhere along the way, then data are queued or delayed, usually with little noticeable effect on application performance. With voice calls, delay must be minimised to the extent necessary to permit voice conversations. Unless sufficient bandwidth is available, it is better to keep voice traffic off the network entirely, rather than allowing the traffic on and delaying or dropping the packets to an extent which reduces voice quality.

In cases where end-to-end resources for a VoIP session are not immediately available, network operators have several configuration options. Voice traffic can be redirected to the PSTN, for example, or users can receive an "equipment-busy" tone.

VoIP WANs must account for the fact that the customer access link – which is generally the slowest network segment – is most likely to become congested.

An enhancement to the RSVP protocol in Cisco equipment ensures that network resources are available end-to-end for the duration of voice sessions. This is accomplished using RSVP as a call admission control mechanism in Cisco H.323 VoIP networks. The technology synchronises RSVP procedures with H.323 Version 2 setup procedures to ensure that the required QoS for a call is maintained across the IP network.

Cisco's RSVP implementation has been enhanced to address scalability concerns. Cisco IOS contains RSVP scalability enhancements, which integrates Integrated Services (IntServ) and Differentiated Services (DiffServ) QoS mechanisms. IntServ allows a strong bandwidth guarantee for RSVP traffic, whereas DiffServ provides specific treatments to different classes of traffic. RSVP typically performs admission control, classification, policing and scheduling of data packets on a per-flow basis. It also keeps a database of information for each flow. If a carrier has 200 000 RSVP flows in the network core, then scalability could be a concern. On the other hand, DiffServ does not offer strong bandwidth guarantees but scales well. The RSVP scalability enhancements enable network managers to rely on RSVP's admission control and control-plane strengths while allowing DiffServ to handle the forwarding-plane functions of classification, policing and queuing.

13.6.4 Compagnie Financière Alcatel

Alcatel's approach to traffic engineering for MPLS networks seeks to provide:

- centralised and multi-vendor traffic engineering
- 30% more efficient usage of MPLS networks
- integration with an MPLS performance management tool.

In a non traffic-engineered network, traffic takes the shortest path to its destination. In this configuration, traffic is by definition uncontrolled and QoS is hard to control. The result is network "hot-spots" with congested links, while most links are little used. A workaround solution that is used by many carriers is to over-provision, increasing the capacity of links in the hot areas. By some measures, average link utilization in a backbone is 15% breaching even the 80:20 rule. Hence over-provisioning is economically inefficient. Further, the method itself is hard to apply because rapid traffic growth makes it difficult to predict where hot-spots will arise. A bad prediction will lead to an investment where the additional capacity will not be used, while the real hot-spots remain hot.

Alcatel's product "ALMA Vision – Traffic Engineering" is a centrally functioning MPLS network optimization tool. It implements algorithms that increase the overall bandwidth used for services by up to 30%. In multi-vendor networks, ALMA can activate Label Switched Paths (LSPs). An LSP is a virtual end-to-end connection used in MPLS networks to simulate circuit-switched networks. Once LSPs are activated, ALMA can update parts of the network using its "GRATE" algorithms.

Most networks contain a mixture of traffic under MPLS and other protocols, like pure IP. Therefore, it is usually impractical to predict the bandwidth that can be reserved on a link for LSPs owing to the sharing of that link with other traffic. For example, a link may have enough bandwidth at a certain moment but, minutes later,

best effort traffic may consume the bandwidth available on the link so that the path used by the LSP is congested. A traffic engineering solution should measure and predict bandwidth utilization of all existing LSPs and derive the traffic matrix from this. It should also measure the bandwidth utilization of all links to detect congestion.

If the traffic matrix changes significantly so that an LSP configuration ceases to be optimal, then the LSPs should be re-routed by a central traffic management to reach a new optimal LSP configuration. On the other hand, if the bandwidth for traffic engineered LSPs is decreasing, a traffic engineering tool may decide to reroute certain LSPs to steer them clear of congested links, possibly making those links less congested.

Service-aware traffic engineering is a solution that comprises intelligent performance management, link utilization monitoring, LSP bandwidth monitoring and the ability to optimise dynamically the configuration of LSPs.

It is clear that performance management goes hand-in-hand with traffic management. However, performance is the end-goal of traffic management and there are many metrics for performance. Indeed, it may be that performance management software sets the parameters by which traffic management software operates. The traffic management software *serves* the performance management software. In turn, the network manager sets the performance metrics where they need to be in order to meet prevailing business goals, the most obvious of which are SLAs and guaranteed services.

13.6.5 Dyband Corporation

Dyband intelligent IP traffic management enables service providers to control bandwidth consumption by IP address for up to 50 000 individual subscribers and/or groups. By evaluating the bandwidth capacity/demand ratio 100 times per second, Dyband can shape individual and aggregate bandwidth usage based on real-time traffic conditions.

Dyband also allows service providers to define and manage any number of SLAs at distinct price points and provides real-time visibility into bandwidth consumption across the network. Historical statistics are provided for capacity analysis, "just-in-time" provisioning, usage-based billing and for customer/technical support.

13.6.6 Fujitsu

The STM-4c Network Interface product is an example of a product that relies on traffic management functionality to achieve a higher level purpose. In this case, the purpose is improved DSL performance. The STM-4c operates in conjunction with Fujitsu's FDX DSL Access Concentrator DSLAM platform. The aim is to provide higher DSL bandwidth.

The STM-4c Network Interface allows the FDX solution to provide fast Internet connections and VoDSL services. Video applications, and also point-to-point 2 Mbps leased line services, benefit from the increased network capacity offered by the STM-4c interface, permitting fast and economic deployment.

The STM-4c interface introduces back-haul efficiencies into the core network, using traffic management and shaping capabilities of Fujitsu's FDX system ensuring optimum use of core transmission bandwidth.

The FDX DSL Access Concentrator forms part of Fujitsu's broadband access equipment, developed to provide network operators with a choice of access solutions.

13.7 General Traffic Management Products

We now review some of the products that are available for traffic management.

13.7.1 Intrarom SA

Intrarom is a Romanian manufacturer of telecommunication and information systems. Intrarom offers the hardware-based Real-time Traffic Management System (RTTMS).

There are various events which can have a serious effect on the service provision. Undesirable spikes in traffic load require some response from network management to ensure that services continue to be provided. The following undesirable effects are associated with such spikes:

- Abnormal increases in traffic demand due to foreseen events such as national or religious holidays, or an international sporting event. Traffic demand can also increase unusually because of unforeseen events such as natural disasters or political crises.
- Focused overloads and mass calling, such as for telemarketing campaigns.
- Congestion in connected networks.
- Delays in the provision of additional circuits or equipment.
- Failures or planned outages of transmission systems and exchanges.
- Long duration of calls to ISPs.
- Increased traffic load due to intelligent network services.

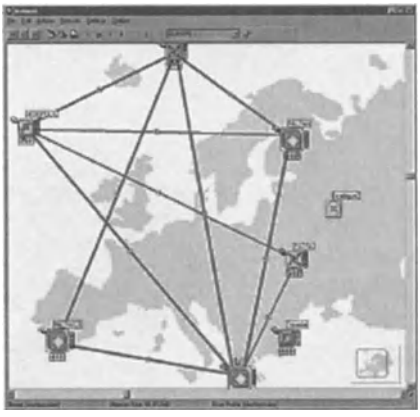


Figure 13.1 RTTMS – network overview window.

These events can lead to congestion, which, if uncontrolled, may seriously degrade the quality of service and generate loss of revenue for the network operator. With the aid of the RTTMS, overload situations can be dealt with promptly and network capacity utilization can be optimised. This can result in increased return on the capital invested in the network and in an improved ratio of effective to ineffective calls.

13.7.2 Vitesse

The hardware-based traffic management products offered by Vitesse provide policing, congestion management, quality provisioning through shaping and scheduling, interworking and switching of layer 2, 3 and 4 data packets and ATM cells. This applies up to bandwidths of several Gbps. Interworking is also referred to as Segmentation and Reassembly (SAR).

The Vitesse Traffic Management Line includes three product families:

- PaceMaker: OC-48 Traffic Manager with ATM SAR
- Monitor: OC-48 ATM OAM and Protection Switching Processor
- OSCAR: OC-48 AALS SAR with Traffic Manager.

The benefits that the Vitesse traffic management suite seeks to deliver are:

- integration enabling equipment manufacturers a time-to-market advantage
- single-chip, thus low power requirements, solution with packet processing
- OC-48 traffic and congestion management solution
- QoS and cost of service capabilities and services.

Traffic management functionality is needed for or, at least, can be applied to various types of NE. Intrarom provides solutions for a variety of networking equipment including:

- multi-service switches
- terabit routers
- ATM core switches
- DSLAM control cards
- multiservice add/drop multiplexers
- digital cross connects.

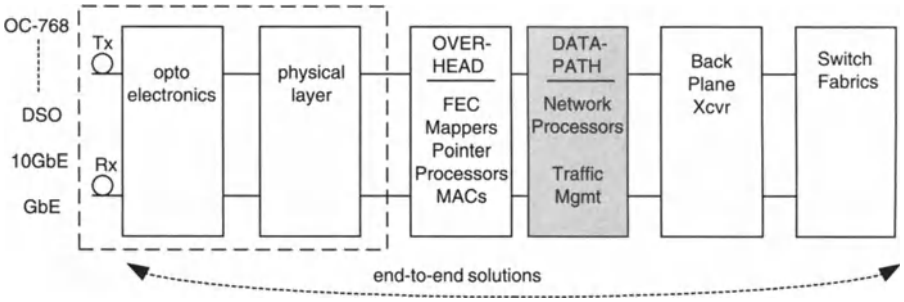


Figure 13.2 Vitesse traffic management application domain.

Vitesse follows the strategy of combining components in the physical layer such as cables and connectors physical network elements, (PHYs), framers, network processors, traffic managers and switch fabrics. A switch fabric is a generic interface which allows network nodes to communicate without having detailed information on how the connection between the nodes is established. With this approach, Vitesse semiconductor solutions seek to offer a seamless integrated architecture spanning PHYs to fabric and supporting a wide range of data rates.

13.8 Conclusion

Traffic management is often carried out at the hardware level, and so the details of how it works are not always the concern of OSS systems. However, OSS systems lean heavily on traffic management functionality in order to ensure that performance parameters are met, which impacts on such business factors as SLAs and guaranteed service agreements. We have explained the basics of traffic management and the main business motivations for traffic management.

14 Web-based Telecommunications Systems Management

14.1 Introduction

Almost all new OSS allow access through a web browser. At its foundation, this is just rendering a screen into HTML, and so we would hope that web-based network management means more than the ability to view information in HTML. Web-based network management means that the sharing and using of network management OSS systems can be as distributed as the web itself.

We shall present two approaches to web-based management. The first is based on a web-based telecom network management system that was set up in Moldova. The second presents a proposal for advanced web-based management tools.

14.2 Case Study of Web-based Telecom Network Management

This section is based on the work of Veaceslav Sidorenco of the Technical University of Moldova, Faculty of Radioelectronics.[51]

Web-based information processing technologies progressed rapidly from the static HTML-based information presentation toward more dynamic interaction between client-side, end-user and sophisticated servers.

Important in orienting dynamic web technologies towards industrial needs was the convergence of the following phenomena:

1. Common Interface Models (CIMs) were defined by Desktop Management Task Forces (DTMF). These provided a standard way to describe and share the enterprise-wide management information in a web environment based on object-oriented technologies. The CIM permits implementation of CORBA, Java Management API (JMAPI) and Microsoft's Common Object Model (COM).
2. In addition to existing industry standards, two new management technologies have been proposed and are under discussion by the IETF:
   - HyperMedia Management Schema (HMMS), which is an extensible data model representing the managed environment.
   - HyperMedia Management Protocol (HMMP), which is a communication protocol embodying HMMS to run over traditional HTTP.

[51] We thank Veaceslav Sidorenco for allowing us to include his work.

K. Misra et al. (eds.), *OSS for Telecom Networks*

The Web-based Enterprise Management (WBEM) standardization initiative seeks to integrate previously incompatible enterprise management standards.

14.2.1 Main WBEM Technology Goals

The main goals of the WBEM technique are oriented toward solving industry problems in the domain of distributed enterprises such as telecommunications and data communications systems. This goal has to be achieved without strict requirements of changing the architecture of such well-known management technologies as SNMP, CORBA and TMN, and integrating them at upper presentation level of systems architecture.

WBEM tools must be able to perform scaleable, platform-independent, application-independent functions of dynamic distance monitoring of the current state of distributed sub-systems and objects. They should also provide web-based tools for control and management of configurations and parameters of objects and processes.

14.2.2 National-Level Telecom Network Management

The telecommunications system of Moldova is under reconstruction, now having a mixed infrastructure of old communication hardware and new equipment. The overall management of the national telecommunications system is generally manual and needs to be unified and automated under the requirements of the ITU-T standards concerning Telecommunications Management Networks and Systems (TMS).

The main reason for a national-level TMS is the necessity for increasing the quality of telecom services through automation of basic control and administration functions, such as monitoring, provisioning, troubleshooting, planning and optimization. Another objective is to reduce telecommunications network operation and maintenance costs.

A project of Mold-TMS powered by WBEM was proposed based on TMN where the subsystems under management were Fault, Configuration, Accounting, Performance and Security (FCAPS). In this model, every NE of the network under management is associated with an agent which is capable of measuring the current state of the parameters from some specified management area, and to generate:

- statistical information about accumulated monitored values
- alerts concerning exceptional events of hardware and software faults, performance degradation, security problems and accounting malfunctions.

Control agents must be able to change structure and parameters of the objects under remote control.

The national telecommunications system of Moldova comprises three international stations and over 50 regional intermediate or terminal analog/digital stations. The regional stations are connected by copper, optical and radio-relay wireless links.

Each network node is equipped with TMN's local-area networks, with Windows NT-based servers, Microsoft SQL servers and control and monitoring agents for

implementing FCAPS management functions. Most agents are implemented as ActiveX components of types:

- automation servers
- automation controllers
- controls
- COM objects
- documents
- containers.

Performance management agents are software modules working as services which can generate time-scheduled and on-request information messages, formatted as SQL statements and/or Dynamic HTML pages that can be read by any ActiveX-enabled web browser. Performance management agents are able to monitor both in tabular and graphical form information on network traffic using different time bases, statistics of successful and unsuccessful call requests and traffic overloads.

Fault management agents can detect abnormal situations in the network. They can seek and inform of network-generated alarms, localise faults, and begin the search for the source of the fault by narrowing down the area from which the fault could possibly have originated using techniques such as alarm correlation. They can also initiate trouble-handling sequences, such as the issuing of work orders, generate reports on fault statistics and start recovery and initialization procedures.

Configuration management agents put objects into service and remove them from service. They install new objects, remove outdated and defective objects and provide connectivity re-routing when exceptions are encountered, such as a cable break or power failure in a router. The network under management is presented in the form of a database. The user interface for this database might be implemented using a package such as MapX ActiveX Geoinformation System Component from MapInfo.

Accounting management agents are responsible for accurate collection and processing of accounting records, maintaining billing policy and updating tariff information.

Security management agents are responsible for corporate immunity against corruption and intrusion. Security management agents monitor different security-related events and generate alerts which may be regarded as of very high priority, certainly comparable with fault management alerts.

The WBEM-enhanced Mold-TMS is under testing, having implemented many types of agent and an integrated software system at its National Telecommunication Control Center, a national NOC.

14.2.3 Conclusions from Case Study

The practical implementation of a national-level TMS using WBEM technology demonstrated the efficiency of the architecture and system design. Using a browser such as Internet Explorer with a reduced number of ActiveX components and HTML-speaking agents at the server sides, it is possible to generate sufficiently sophisticated scaleable control and monitoring schemes for distributed object management over a pan-national telecom system.

The Moldovan project was groundbreaking at the dawn of the twenty-first century and is still a rare example of how a national telecom network might be run in an entirely centralised way. We now turn to a more advanced approach to the problem of developing a distributed environment for web-based network management.

14.3 Developing a Distributed Environment for Web-based Network Management

This section is based on the work of Flavio Spolidoro of Empresa Brasileira de Telecomunicacoes and Noemi Rodriguez of the Departamento de Informatica, PUC-Rio, Brazil.[52]

The use of scaleable management tools is becoming more important for integrated management of different network platforms, and a flexible environment for web-based distributed network management would be a valuable step forward. The environment proposed includes services for monitoring and asynchronous event handling. The dynamic characteristics of this environment allow the administrator to extend or modify functionality without stopping the system, and also allow the administrator to redefine dynamically the servers supporting the network management application.

Traditionally, network management applications consume a great deal of processing resources. As institutional networks grow in size, the conventional network management architecture, in which a centralised management application is responsible for monitoring and handling events of the entire network, ceases to be feasible. On the other hand, most computer networks present high percentages of idle processor time. Therefore, it is natural to distribute the workload associated with network management among available machines. Even if dedicated servers carry out network management, distribution means that new servers can be added as needed, instead of replacing a server with a more powerful machine every time it reaches its limit. This distributed approach has been analysed in several publications.

Once we decide to decentralise the management application, creating a distributed application with processes running on several machines, we must decide what communication infrastructure will be used. Traditional management protocols are too low-level and designed for communication with network devices. A higher level communication model allows us to abstract from specific protocols and to integrate management of devices with other requirements, such as management of network services.

Distributed components models have gained maturity in recent years. Among these models, CORBA is a de facto standard and deals well with heterogeneity and interoperability. It therefore seems natural to use CORBA's infrastructure as a basis for a distributed management application. However, the traditional use of CORBA, using only static language bindings, usually C++ or Java, would result in a highly static management application. Network management presents dynamic requirements: the network administrator should be able to include new functionality without the need to stop the running application.

[52] We thank Flavio Spolidoro and Noemi Rodriguez for allowing us to include their work.

When we distribute the network management application, dynamic requirements grow: the application must adapt to changing conditions in the environment, such as processor load, in order to take advantage of available processing resources effectively.

Another important trend in network management is to offer web-based interfaces. This allows network state to be viewed and acted on remotely, helping to organise technical support services. Further, with the current popularity of web browsers, their use as network management interfaces means less software installation cost and a gentler learning curve for different technical levels of network administrators.

Spolidoro and Rodriguez presented a dynamic distributed network management environment that uses CORBA as a basis for distribution and provides a web interface. More specifically, we investigate the use of Lua, an interpreted language, and LuaCorba, a dynamic CORBA binding, to create this dynamic distributed management environment. Lua glues together the CGI, CORBA and SNMP components which are used to access end devices.

14.3.1 WebComm Architecture

WebComm comprises the Event Handler Service and the Monitoring Service. Its architecture is illustrated in Figure 14.1.

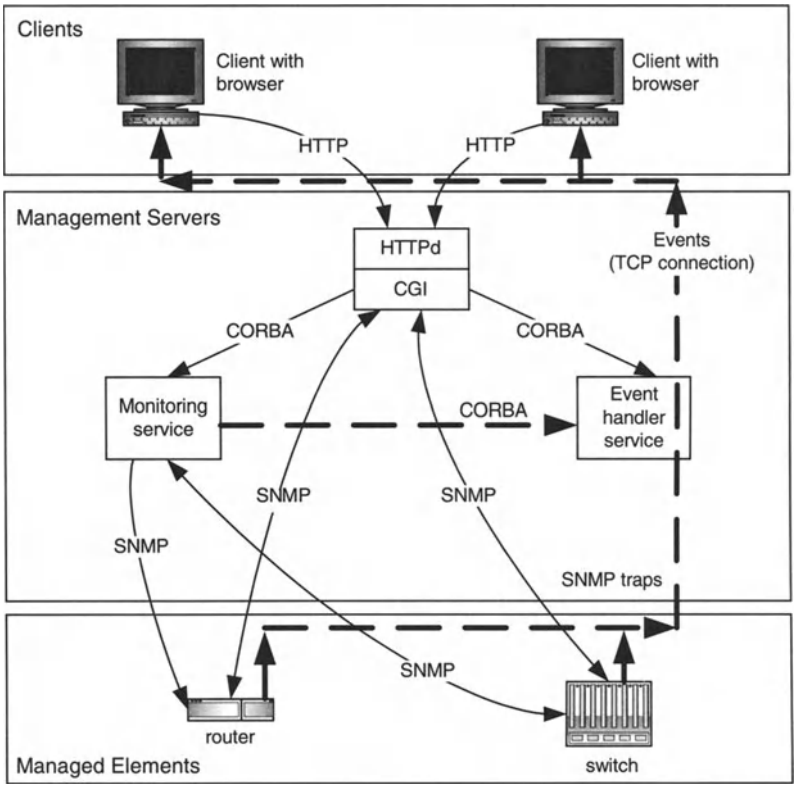


Figure 14.1 WebComm architecture.

The user interface is web-based. A CGI application built with CGILua scripts manages dynamic page construction and manages access to WebComm's services.

The Event Handler Service is responsible for receiving, handling (filtering and correlating), and forwarding asynchronous events to the client application, which is the web browser.

The reception of events and alarms is fundamental for a network management application which includes fault management. Event processing is thus a central part of WebComm: the architecture of the Event Handler Service, described below, allows the event processing load to be distributed over several machines.

The Monitoring Service is responsible for monitoring network elements. This service can poll managed elements for state information in order to generate statistical graphs or to identify fault conditions. When an abnormal condition is identified, the Monitoring Service can send an event to the Event Handler Service. The information it produces can also be retrieved synchronously by the CGI application. Both the Event Handler Service and Monitoring Service have CORBA interfaces. These CORBA interfaces may be used to reconfigure these services. By using CORBA interfaces, the user interface application becomes a module that is independent of the management application and may be built in any programming language and platform that has an Object Request Broker (ORB) implemented. An alternative user interface application may be built, for instance, with the HP OpenView platform. WebComm uses SNMP as the management protocol for communication with network devices.

14.3.2 Event Handler Service

Events that are handled by the Event Handler Service are asynchronous notifications, such as SNMP traps or higher level messages, having as their main purpose reporting faults to the administrator or management application itself, thus allowing a manual or automatic corrective action to be carried out immediately. However, the direct reception of events is not advisable. In large networks, the number of events received can be so high as to make it difficult for the operator to identify the real cause of the problem because the user interface is so "polluted" with events. In some telecommunications networks, such as EMBRATEL's Frame Relay and ATM networks, a management application receives around 130 000 events daily.

To avoid the pollution of the user interface with events, some applications apply *correlation* and *filtering* mechanisms on events. The Event Handler Service allows the user to define functions which will be executed upon the reception of an event, eventually masking some events and generating other new ones. This allows the generation of more significant events, helping the operator to identify the root cause of the problem.

To allow event processing to be distributed among different machines, the Event Handler Service can be decomposed into several processes, forming a hierarchical structure. This structure is made up of two kinds of process: the *Event Dispatcher* and the *Event Agents*. Figure 14.2 presents the architecture of this service with some event agents and one event dispatcher. Communication between the event service processes is also based on CORBA.

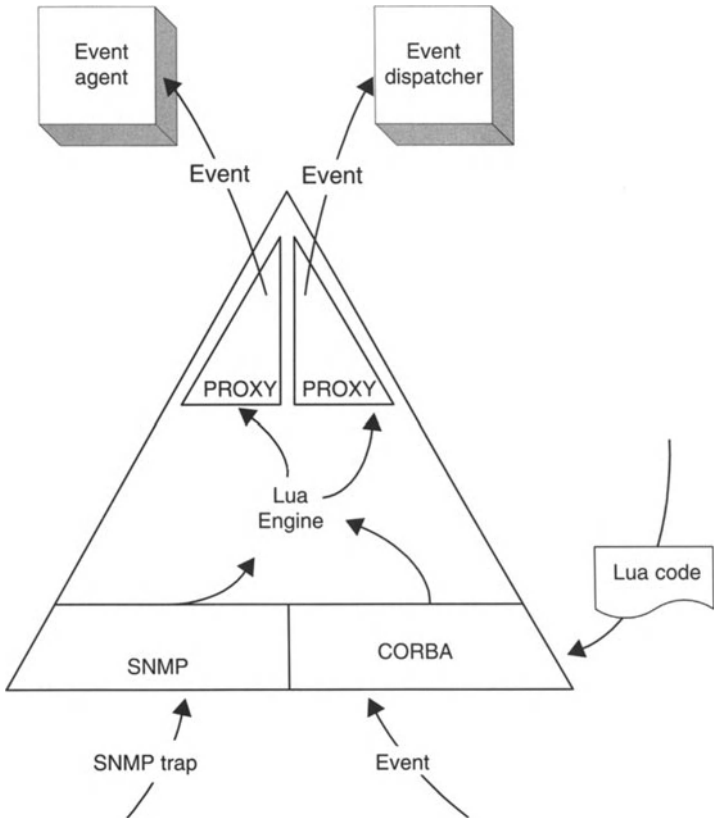


Figure 14.2 Event agent.

All the configuration of this service is done using Lua and CORBA. It is possible to define dynamically the way the events will be handled and to configure user domains which will receive events. With domain configuration it is possible to set up different user profiles, such as user groups who will receive provisioning events and others who will receive alarms.

An *Event Agent* is a process responsible for receiving and processing events, eventually generating new events. Event agents provide a CORBA interface containing methods for agent configuration and for registering new events. For agent configuration, one of the methods in this interface allows the programmer to pass a chunk of Lua code containing function definitions. The CORBA call for registering an event is one of the two possible ways for an agent to receive an event. This method receives events in a standard event format, which conforms to the ITU X.733 recommendation. Event agents may also receive SNMP traps.

An event agent can deliver filtered or generated events either to an event dispatcher or to another event agent. Several event agents can be linked forming a hierarchy similar to a tree, in which each level receives and correlates events generated by inferior levels (see Figure 14.2). As a result, the load of processing events

may be distributed over event agents in different servers. The structure of this tree can be dynamically redefined; the CORBA configuration method at each agent can be used to redefine the agent to which it should send its events.

WebComm event agents are started manually or at machine boot. They then remain idle until receiving a configuration through their CORBA interface.

The *Event Dispatcher* is the process responsible for delivering events to client applications. It is also responsible for storing the events in a database, which can be anything from a relational database to a simple file. In WebComm, the asynchronous reception of events is made possible through the use of Java applets. Applets are used by users who are distributed throughout the network and allow the server to communicate asynchronously with a client, thus escaping the client–server paradigm imposed by the web. The event dispatcher can be regarded as a specialised event agent for distributing events to clients. It also receives events through its CORBA interface. This interface is identical with the event agent interface with an additional method which allows the caller to retrieve the TCP port at which the event dispatcher is awaiting clients' connections. This method is necessary for applets to know where the event dispatcher is awaiting connections, since applets communicate with the event dispatcher using TCP sockets.

When the event dispatcher is started, it waits for client TCP connections. Because of security restrictions, an applet is only able to open connections to programs being executed in the same machine as the web server that has sent the applet to the web browser. The applet can only open connections to applications running in other machines if it is a signed applet.

The event dispatcher is again configured by passing to it Lua code, which performs the distribution and storage of the events. There are three ways to distribute an event: to every user, to a domain of users or to one user. Tables are used to structure data in Lua, making it is easy to define simple structures, such as arrays and records, and more complex structures, such as linked lists and sets.

14.3.3 Monitoring Service

The Monitoring Service is responsible for periodically monitoring and collecting information from the network elements. Such information is useful in performance management and for identifying abnormal conditions.

The Monitoring Service can be broken down into *Monitor* processes, which are responsible for collecting and monitoring the information in the network devices. These processes are composed of a CORBA interface, for receiving its configuration, and a Lua engine, which executes the monitoring according to the Lua code. Monitors can be executed in different machines, allowing load distribution.

The user interface application is basically a CGI application, a set of CGILua scripts. These scripts implement the user interface and the network management application, setting up the services described in the previous sections. As the remote services use CORBA interface, the user interface application could be implemented in any kind of language that has an ORB implemented. We chose CGILua for implementing WebComm because of its flexibility and simplicity. The simplicity of Lua syntax makes the development of the application easy and quick. An application

built with CGILua may also be extended, allowing the user to add new tools to the management application.

Lua libraries may be loaded into the CGILua application as in any other Lua program. The CGILua application loads the LuaCORBA library aiming at accessing the remote services described in the previous sections, and loads the LuaMan++ library to access managed devices directly.

WebComm is a multi-user application with each user having their own management environment. A user management environment has a diagram of the managed network and a set of management tools. Management tools are small CGILua scripts which can be included through an upload or edited in the browser. A simple and useful tool, for example, is a script to ping the network devices. The WebComm application supports the creation, discovery and editing of network configurations. It also allows the user to query and update MIB-II information of managed devices.

As examples of how the remote services are used to create management tools, we next discuss asynchronous event reception and traffic monitoring in WebComm. Events are reported to an applet running in the client browser. The connection between applet and dispatcher is established as displayed in Figure 14.3. A CGILua script calls the method in the CORBA interface of the event dispatcher that returns a TCP port. In particular, it is the TCP port in which the event dispatcher awaits client

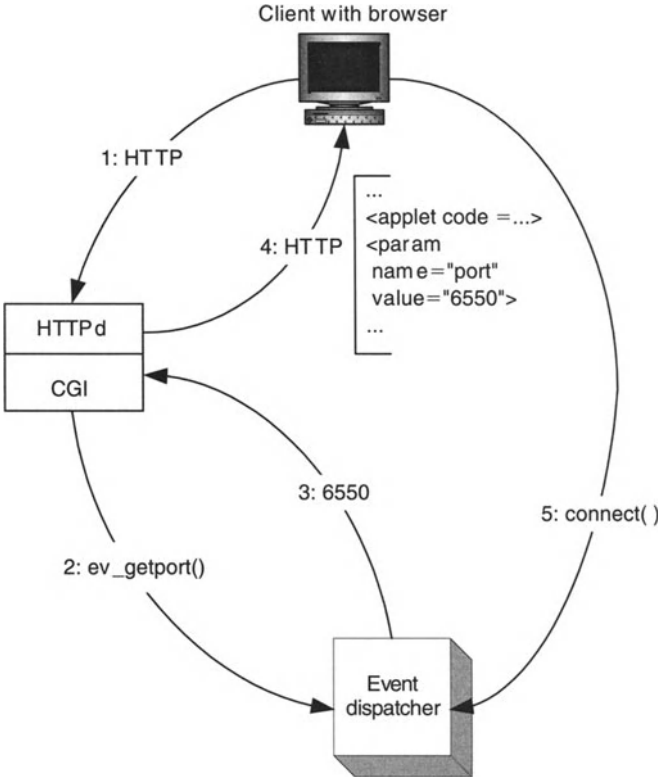


Figure 14.3 WebComm client connects to Event Dispatcher for receiving events.

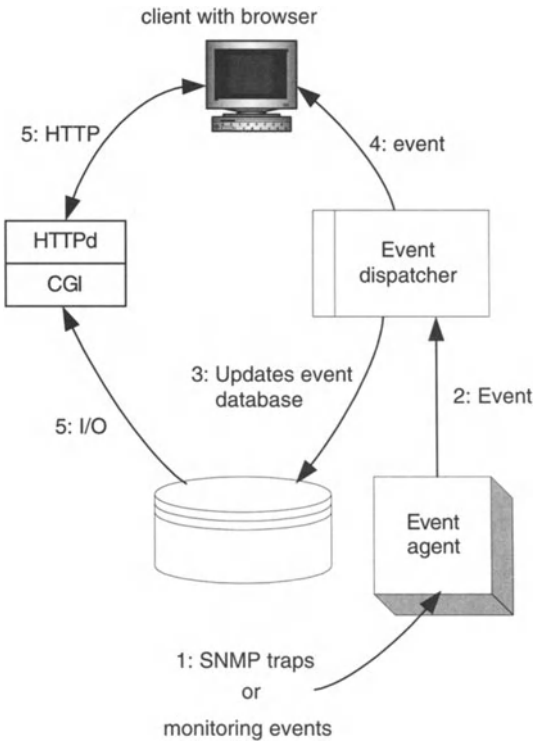


Figure 14.4 WebComm implements asynchronous event reception.

connections. Next, the CGI script returns an HTML page to the web browser, encapsulating an applet which receives the TCP port as an argument. The applet then establishes a connection with the event dispatcher and becomes ready to receive events.

Once the applet is connected to the event dispatcher, the applet is ready to receive events. When any event reaches the event dispatcher, the event dispatcher stores the event in an event database and sends the event to the connected clients (see Figure 14.4). When the applet receives the event, using HTTP, it activates a CGILua script that accesses the event database and rebuilds the page containing the event list.

This access to the event database is required because alarms which were previously active may have been eliminated, by the correlation mechanism, with the reception of the new event. For example, *link up* alarms can eliminate *link down* alarms of the same network interface. In contrast to many web-based management applications, WebComm does not use the applet as the graphical interface to display the event list. Its graphical interface is HTML-based.

In the event manager screen, events are sorted as they are received. Some fields of the X.733 standard format are exhibited, for instance the *severity*, which determines the row colour. Similar events are filtered and the number of similar events is exhibited in the column *count*. The field *event time* displays the date of the latest event.

The traffic monitor interface allows the user to see statistical graphs describing input and output activity at each interface in managed network devices.

The traffic monitor is based on the Monitoring Service. The user determines whether host traffic is monitored at that host, when adding a host to the network diagram. If so, then a monitor is chosen to be reconfigured in order to also monitor all the interfaces in the new host. Mechanisms for choosing monitors which keep the load balanced between them are issues for future study. Traffic information is collected from managed devices through SNMP at 5-minute intervals and stored in a database. This information can be displayed on various temporal scales.

14.3.4 Summary of WebComm Services

The main services offered by WebComm are asynchronous event reception and monitoring.

The Event Handler Service maps SNMP traps to X.733 events and filters and correlates events. This service can be provided in a distributed fashion, where agents can be configured to work as a hierarchical tree, creating multiple levels for handling events.

The Monitoring Service is responsible for collecting and monitoring information from managed network devices, so that alarms can be sent to the Event Handler Service to report abnormal conditions to the user.

The mere use of a web-based interface does not introduce any important shift towards a distributed management paradigm. In WebComm, the web-based interface makes the entire management application available locally and remotely. The two main services in WebComm – events and monitoring – use CORBA as the main distribution mechanism. Hence they are independent of the user interface application, which could be rewritten in a different platform, even using a compiled language with a CORBA binding, while still using Lua for service configuration.

14.4 Products

We now give an overview of some of the web-based network management products available.

Optivity Software (Bay Networks) provides solutions for the campus and enterprise environment by using a web-based network management approach. The OmniView application provides tabular and graphical views of network statistics for quick diagnostics and monitoring of network health collected from Bay Networks devices and industry-standard MIBs.

Network Health (Concord Corporation) is a family of web-based software applications that automate the collection, analysis and reporting of critical network data. Network Health discovers and collects vital data from existing devices in the network, condensing it into graphical presentations.

The Nebula product line includes Nebula-ICC, Nebula-ATM, Nebula-NMS, Nebula-SNMS, Nebula Performance Monitor and Nebula MIB Explorer. The Nebula set of tools offers autodiscovery, real-time graphical reporting, multi-vendor MIB support, management policies for any ATM product or feature that can be defined without the need for programming. Nebula also offers an ATM-specific, web-based reporting and plotting capability, authenticated and encrypted network traffic,

confining SNMP traffic to secure domains, protection against IP spoofing, replaying and capturing and user ID and application access restriction on nodes.

14.4.1 Single Interface to Data and Voice Networks

AT&T offers a suite of extranet-based network management capabilities that will make it easier for business customers to monitor and control their corporate data and voice networks. AT&T Interactive Advantage provides a secure integrated platform where business customers can access network management tools through a single interface.

Security features enable customers to restrict employee access so that the information can be protected within their own organization. For instance, the voice network manager might have access limited to voice services only, or the data manager might have access limited according to the employee's security authorization.

Businesses can obtain information about their corporate networks from a single source, including details on performance, maintenance, repairs, provisioning and service orders. Interactive Advantage also offers collaborative tools where customers can communicate with their AT&T support teams through electronic bulletin boards where reports and messages can be posted.

Voice Network Services management tools are available through Interactive Advantage. Customers can monitor their corporate networks' real-time call attempts, and call-completion data online to manage their toll-free facilities and resources better. Network and traffic data reports can be electronically retrieved for proactive network planning to help managers achieve optimal circuit performance. Potential network problems or even opportunities can be quickly identified using online performance analysis tools for on-network and off-network calling patterns and trunk group monitoring.

Data Network Services management tools simplify management of frame relay network ports and Permanent Virtual Circuits (PVCs). Customers can view graphical and tabular reports with their browser or download directly into a spreadsheet application where data can be stored and manipulated. Interactive Advantage can also be used to improve the timeliness of tracking and managing trouble tickets in addition to placing service order requests.

14.4.2 Roaming Ulysses

The Ulysses platform is an example of a platform that accommodates a web-based management approach. Because the information is available through a Java-based platform, the information can be distributed through the web as for network management staff located at geographically disparate locations.

In short, Ulysses operates on the basis that switch-based systems cannot deliver features needed to support new requirements in planning and designing wireless networks, and migration to new technologies, such as 3G, is particularly difficult. Further, Intelligent Network (IN) platforms are limited by the need for proprietary modifications. Logica therefore developed its Ulysses platform to provide a general architecture and common set of facilities for developing and deploying various

advanced wireless mobility services. The architecture accommodates mobility applications including home location registers, authentication centres and roaming platforms.

Ulysses can host multiple co-located applications, allowing new services to be rapidly deployed on existing hardware platforms, and all of its advanced features can be managed from a central point. This ensures easy integration with the rest of an operator's network management infrastructure. Ulysses has a distributed architecture, allowing components to be geographically remote.

The database and operations and management processor (OMP) (which is implemented using SQL Server) layers are logically separate systems. Indeed, they can be deployed on separate platforms, allowing Ulysses deployments to be tailored. Common components are reused in a variety of service implementations, enabling new applications to be built on well-tested and field-proven code and data structures.

The database servers are deployed in a live/shadow combination so that subscriber data are maintained in two real-time locations. The live and shadow systems are synchronised with each other and with the disk database stored in the OMP.

Ulysses supports centralised operations, administration and management (OAM) via the OMP component, which provides mediation with the subscriber administration and network management systems. The OMP is responsible for:

- startup and shutdown of database platforms
- modifications to secure databases
- maintenance and event reporting.

The main system administration terminal is Java-based and can be operated locally and remotely via a Java-enabled system. Therefore, the OMP can be integrated into a web-based network management schema.

14.5 Conclusion

The development of web-based network management is consistent with the growth of the web and its convenience to the enterprise. This chapter has presented a case study and several technologies that are being used to enabled web-based network management. Although the web is suited for making information available more easily to people throughout an organisation, this is not automatically so. For example, there is a finite set of standards that allowed the Moldova project to succeed.

Now that one set of interfaces has been developed and implemented, we can see that progress will be made difficult in the future to the same extent that it has been in the past with OSS. That is, new software must be able to talk to the CORBA interfaces or be compatible with whatever standards have been used. Further, because these standards have been implemented in multiple systems across geographically wide areas, the integration projects of the future might dwarf the OSS system integration projects of today. However, the standards being used for web-based management are generally more open than the proprietary systems of the past. This means that the standards of today should have a long lifespan and when they are modified there will be a high degree of backward compatibility.

Bibliography

1. AdventNet Ltd. "AdventNet Configuration Management Server 3.0 User Manual", 2002.
2. AdventNet Ltd. "AdventNet Fault Management Server 3.0 User Manual", 2002.
3. Aidarous, S and Plevyak, T (Eds). "Telecommunications Network Management: Technologies and Implementations", 1998, Series on Network Management (Aidarous, S and Plevyak, T, series editors), IEEE Press.
4. Alcatel product documentation.
5. Bing, B. "High-speed Wireless ATM and LANs", 1998, Wireless Communications Series, Artech House Publishers.
6. *Capacity* magazine.
7. Chen, G and Kong, Q. "Integrated Telecommunications Management Solutions", 2000, Series on Network Management (Aidarous, S and Plevyak, T, series editors), IEEE Press.
8. Clarity International Ltd. "Clarity Configuration Manager User Manual", 2002.
9. Clarity International Ltd. "Clarity Service Manager User Manual", 2002.
10. Cramer Systems product documentation – available on website.
11. Deri, L. "JLocator – Web-based asset location system", 2000, University of Pisa.
12. Dimension Data product documentation – available on website.
13. GE Smallworld product documentation – available on website.
14. *Global OSS* magazine.
15. ITU-T public documents.
16. Luise, M and Pupolin, S. "Broadband Wireless Communications", 1998, Springer-Verlag.
17. Raman, LG. "Fundamentals of Telecommunications Network Management", 1999, Series on Network Management (Aidarous, S and Plevyak, T, series editors), IEEE Press.
18. Servin, C. "Telecommunications: Transmission and Network Architecture", 1996, Springer-Verlag.
19. Sidorenco, V. "Web-based Telecommunications System Management", 1999, Faculty of Radioelectronics, Technical University of Moldova.
20. Neuman de Souza, J and Boutaba, R. "Managing QoS in Multimedia Networks and Services", 2000, Kluwer Academic Publishers.
21. Spolidoro, F and Rodriguez, N. "Distributed Environment for Web-based Network Management", 2001, Empresa Brasileira de Telecomunicacoes, Brazil.

22. Telemanagement Forum. "eTOM Business Process Framework", Document No. GB921, May 2001.
23. Telemanagement Forum. "Telecom Operations Map", Document No. GB910, March 2000.
24. Tenor Networks product documentation.
25. Tsang, DHK and Kuhn, PJ (Eds). "Broadband Communications: Convergence of Network Technologies", 2000, Kluwer Academic Publishers.
26. Varro product documentation.
27. Venieris, I and Hussmann, H (Eds). "Intelligent Broadband Networks", 1998, Wiley.
28. Vittesse product documentation.

Abbreviations

3GPP	Third-generation Mobile Partnership Project
3GPP2	Third-generation Mobile Partnership Project II
ABR	Available Bit Rate
ADM	Add/Drop Multiplexer
(A)DSL	(Asynchronous) Digital Subscriber Line
AMDF	Automated Main Distributing Frame
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
BE	Back-end
BER	Bit Error Ratio
B-ISDN	Broadband ISDN
BPR	Business Process Re-engineering
CAC	Call Admission Control
CCM	Clarity Configuration Manager
CDMA	Code Division Multiple Access
CDR	Call Detail Record
CDV	Cell Delay Variation
Centrex	CENtralised EXchange
C-Flag	Contingency Flag
CEO	Chief Executive Officer
CIM	Common Interface Model
CLI	Command Line Interface
CLR	Cell Loss Ratio
CMIP	Common Management Information Protocol
CMS	Configuration Management Server
CNM	Customer Network Management
COE	Central Office Equipment
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPE	Customer Premises Equipment
CRM	Customer Relationship Management

CSN	Compact Service Node
CTD	Cell Transfer Delay
CTP	Connection Termination Point
DB	Database
DiffServ	Differentiated Services
DIM	Device Intelligent Module
DOM	Domain Object Model
DSL	Digital Subscriber Line
DTD	Document Type Definition
DTMF	Dual-tone Multi-frequency
(D)WDM	(Dense) Wave Division Multiplexing
DXC	Digital Cross Connect
EAI	Enterprise Application Integration
ECC	Embedded Communication Channel
EJB	Enterprise Java Beans
EMS	Element Management System
ERM	Enterprise Resource Management
ERP	Enterprise Resource Planning
eTOM	e-Business Telecom Operations Map
FAB	Fulfilment, Assurance and Billing
FCAPS	Fault, Configuration, Accounting, Performance and Security
FE	Front-end
FMS	Fault Management Server
FTP	File Transfer Protocol
GIS	Geographical Information System
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GSMP	General Switch Management Protocol
GT	Generic Type
GUI	Graphical User Interface
HEC	Header Error Correction
HFC	Hybrid Fibre–Coaxial Cable
HMMP	HyperMedia Management Protocol
HMMS	HyperMedia Management Schema
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
IAD	Integrated Access Device
IDE	Integrated Development Environment
IDL	Interface Description Languages
IDR	Indirect Data Retrieval
IETF	Internet Engineering Taskforce
IIOP	Internet Inter-ORB Protocol
ILMI	Interim Local Management Interface
IN	Intelligent Network
IP	Internet Protocol
IntServ	Integrated Services

ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
ISV	Independent Software Vendor
ITU-T	International Telecommunications Union – Telecommunications Standardization Sector
J2EE	Java 2 Platform Enterprise Edition
JDBC	Java Database Connectivity
JFC	Java Foundation Classes
JMAPI	Java Management API
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KPI	Key Performance Indicator
KQI	Key Quality Indicator
LAN	Local Area Network
LNP	Local Number Portability
LSP	Label Switched Path
MAC	Media Access Control
MAN	Metropolitan Area Network
MBeans	Managed (Java) Beans
MDF	Main Distribution Frame
MIB	Management Information Base
MMS	Multimedia Messaging Service
MPEG	Motion Pictures Experts Group
MPLS	Multiprotocol Label Switching
MSC	Mobile Switching Centre
MTTR	Mean Time to Repair
MVC	Model/View/Controller
NAR	Network Management System Archive
NE	Network Element
NEMS	Network Element Management System
NGOSS	Next-generation OSS
NML	Non-machine Language
NMS	Network Management System
NNI	Network-to-network Interface
NOC	Network Operations Centre
OAM	Operations, Administration and Management
O&M	Operations and Maintenance
OC-x	Optical Channel – Speed of Connection
OID	Object Identifier
OMC	Operations Management Centre
OMP	Operations Management Processor
OPS	Operations
ORB	Object Request Broker
OSS	Operational Support System

OTDR	Optical Time Domain Reflectometer
PBM	Policy-based Management
PCR	Peak Cell Rate
PDH	Pleisiochronous Digital Hierarchy
PDU	Protocol Data Unit
PHY	Physical Network Element
POP	Point of Presence
PSTN	Public System Telephone Network
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RAN	Radio Access Network
RDBMS	Relational Database Management System
RFC	Request for Comment
RMI	Remote Method Invocation
RMO	Resource Management and Operations
ROI	Return on Investment
RSVP	Resource Reservation Protocol
RTP	Real-time Protocol
RTTMS	Real-time Traffic Management System
R/W	Read/Write
SAR	Segmentation and Reassembly
SCE	Service Creation Environment
SCP	Service Control Point
SCR	Sustainable Cell Rate
SDH	Synchronous Digital Hierarchy
SI	System Integrator
SIP	Strategy, Infrastructure and Product Management
SLA	Service Level Agreement
SLM	Service Level Management
SN	Service Node
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical Network
S/P	Supplier/Partner
SPRM	Supplier/Partner Relationship Management
ST	Specific Type
SQM	Service Quality Management
TC	Transmission Convergence
TCA	Traffic Conditioning Agreement
TCP/IP	Transmission Control Protocol/Internet Protocol
TCS	TL1 Command Set
TDM	Time Division Multiplexing
TFTP	Trivial File Transfer Protocol
TINA	Telecommunication Information Networking Architecture
TL1	Transaction Language One
TMF	Telemanagement Forum
TMN	Telecommunication Management Network

TMS	Telecommunications Management Networks and Systems
TOM	Telecom Operations Map
TTP	Trail Termination Point
UDP	User Datagram Protocol
UI	User Interface
UME	UNI Management Entity
UMTS	Universal Mobile Telephone Service
UNI	User-to-network Interface
VBR-rt	Variable Bit Rate – Real Time
VC	Virtual Circuit
VCC	Virtual Channel Connection
VLAN	Virtual LAN
VoDSL	Voice-over Digital Subscribe Line
VoIP	Voice-over IP
VP	Virtual Path
VPC	Virtual Path Connection
VPL	Virtual Path Link
VPN	Virtual Private Network
WAN	Wide Area Network
WAP	Wireless Application Protocol
W-ATM	Wireless ATM
WBM	Web-based Network Management
WBEM	Web-based Enterprise Management
WSMT	Wireless Services Measurements Team
XML	Extensible Markup Language

Index