SIF-Approved Document #: SIF-014-1997
(SIF Contribution #: SIF-IM-9704-037R5)

CONTRIBUTION TO SONET INTEROPERABILITY FORUM

SIF PROJECT: Network Management - Network Level Model

TITLE: Information Model for Connection Management and Fault Management at the EMS/NMS Interface

---

CONTRIBUTORS: The Network Layer Information Working Group
SONET INTEROPERABILITY FORUM

CONTACT: Wendy Teller for Ameritech
Weyand Associates
1600 Mayapple Court.
Naperville, IL 60565

Telephone: +1-630-527-6206
Fax: +1-630-527-6207
Internet: teller@mcs.com

DATE: June 27, 1997

DISTRIBUTION: SIF IM

ABSTRACT: This contribution is a draft of the requirements, scenarios and CMIP information model in support of connection management and fault management functions at the NMS/EMS interface.

This document has received the approval of the SONET Interoperability Forum (SIF).
August 28, 1997

Table of Contents

List of Figures

1. Introduction

It is the function of the SIF Network Management Information Model Working Group to derive a CMISE GDMO information model for use between the TMN NM and EM Layers. This document provides the information model and the material which was defined in support of the information model.

The document is divided into four sections. This section covers introductory information. Section 2 provides Statements of Applications which define the high level requirements for the Information Model. Section 3 provides scenarios which serve both to refine the requirements defined in Section 2 and also show how the information models support the requirements defined in Section 2. Section 4 contains the information model.

1.1 History of the Development Process and Methodology

The history of the development process is singularly marked with an ongoing effort to align the SIF information model with that of the work of others standards bodies. This has encompassed, and included, methods and models as presented by ITU, ETSI, NMF and ATMF. It is a method that, while it has at times created a need for this working group to reevaluate portions of its efforts, has at least kept this work current with the overall needs of the service providers in managing networks with scopes exceeding those of SONET alone. However; it is the charter of this group to focus on SONET requirements specifically.

1.2 Scope

This document specifies a CMISE GDMO information model for use between the TMN NM and EM Layers. It does not specify the OSI stack or CMISE functions which are used to transport the information. Specifications for OSI stack profiles and CMISE functions can be found in GR-253 and in the SIF draft document “Requirements for SIF OS Platforms”.

This specification addresses the network-view aspects of the NMS/EMS (Network Management System/Element Management System) interface needed to support A SONET network management, i.e. the management of aggregates of Network Elements such as subnetworks. It also relates the SONET Network-view and NE-view to build coherent management functions. Although it is architecturally permissible to offer NE-view only or Network-view only management services, it is to be kept in mind that the network-view is intended as an aggregate view to provide additional value. In this sense, the network-view should not constitute an opaque obstacle for reaching the NE; but it should be conceived as an organized way of seeking NE details, when needed.

This specification focuses on what is considered to be the initial functionality of

SONET network view management. It is understood that this initial set of functions, managed entities, and scenarios will be enhanced in subsequent versions.

This document addresses the following functional areas of SONET network management:

- Transport network connection management (including set-up/ modification for subnetwork connection, link connection, and trails.)
- Transport network configuration provisioning (including subnetwork provisioning, and link provisioning)
- Network fault management (including correlation, localization, notification, for both equipment and connections)

The following functional areas may be addressed in the future:

- Network connection reservation
- Loopback testing
- Network performance management (including congestion, and connection monitoring)
- Inventory including bandwidth management

Note that functional requirements, but no managed entities may be provided for the following functionality in the current specification:

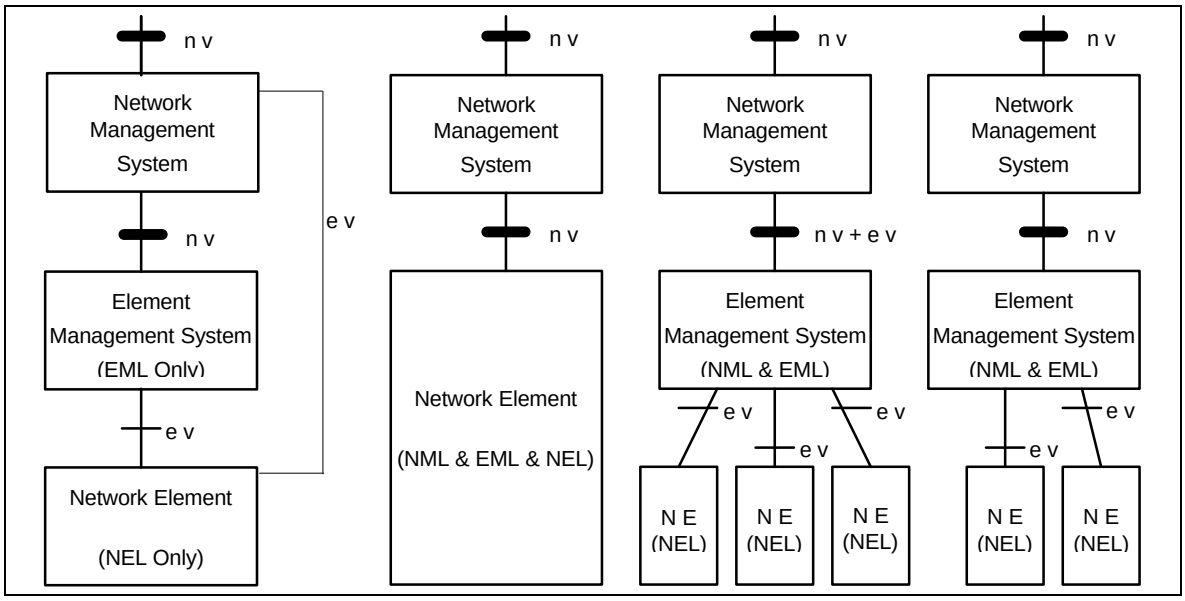
- any protection-switching, back-up functionality,
- grouping of subnetwork, or link connections, or of trails,
- routing constraints,
- scheduling or reservation,
- performance monitoring

This specification focuses on the interface functionality to manage the subnetwork, and does not provide requirements on the management systems themselves. Only Public Data Networks are addressed in this specification, the management of a Private Data Network is not included.

1.3 Network Management Architecture

In order to understand which functions will be used in this information model it is necessary to define the philosophy for the network management architecture. One can view the network with its associated network management layers as shown in Figure 1-1. We show here the Network Elements (NEs), EMSs and

NMSs, but none of the higher layer management systems. (We are not precluding the use of the higher layers of management by omitting them, but they are not our current focus.) The EMS may address Network Management Layer (NML) and Network Element Layer (NEL) functions along with Element Management Layer (EML) functions. The Network-view (nv) and the NE-view (ev) between the individual management systems are also shown. . Implementation of both the Network View and NE View together represents a specific design choice. Also, implementations that provide a "stand alone" network view (no references to NE view objects) may be defined using the objects described in this document.



ev - network element view
nv - network view

Figure 1-1 Physical Realization Examples of Multi-layer Network Management Architecture

To support the multiple architectures described in Figure 1-1, the SONET NE-view and the SONET Network-view MIBs can be combined in multiple fashions. The SONET network management interface requirements in this document address the EMS/NMS interface, regardless of the functional layers addressed by the individual EMS or NMS. The primary focus of these requirements is the NMS to EMS interaction needed to support SONET subnetwork management. With respect to Figure 1-1, these requirements are relevant to SONET NEs, EMSs, and NMSs supporting EML functions.

1.4 NE+Network-Level Management Architecture Example:

The NE+Network-Level Management Architecture (see Figure 1-2) exposes the

"SONET NE-View" between the NMS Environment and the EMS (managing the subnetwork). In this architecture, the NMS has the option to view and manage the SONET network by performing operations on the subnetwork as a whole or by performing operations on select SONET NEs. One could imagine that, for certain NMS applications, a single-entity subnetwork view would be sufficient, while for other applications a detailed view of each SONET NE comprising the subnetwork as well as their interconnections would be desirable. The models defined in this document may also be used without exposing the NE-view.

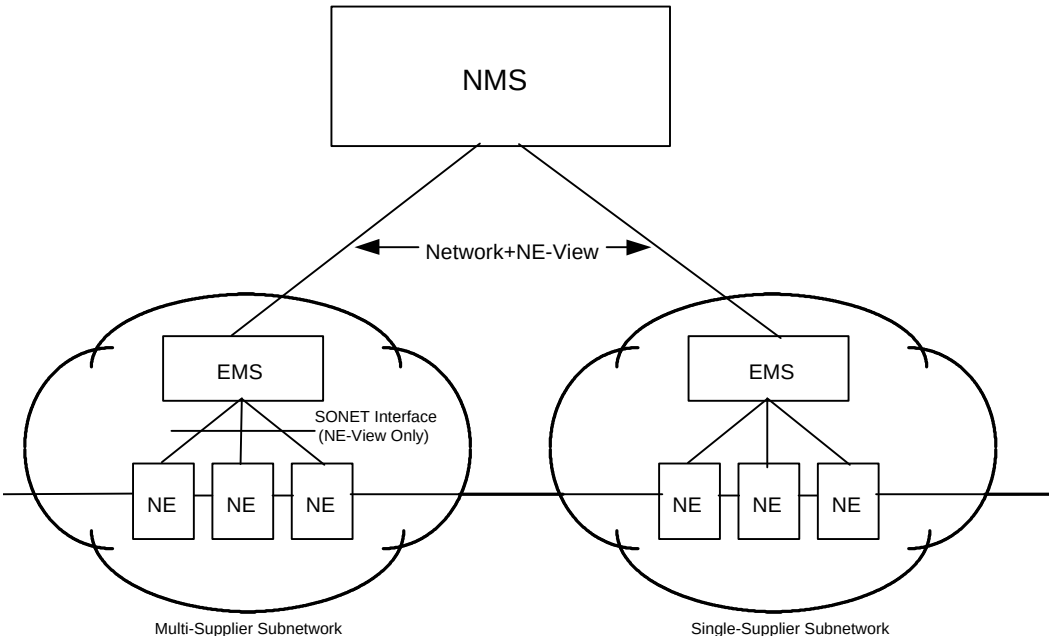


Figure 1-2 Example of NE+Network-Level Management Physical Configuration

1.5 References

The following standards contain information which was used while defining the information model in this document.

ITU-T Recommendation G.774 (09/92), *Synchronous Digital Hierarchy (SDH) Management Information Model for the Network Element View*.

ITU-T Recommendation G.774.01 (11/94), *Synchronous Digital Hierarchy (SDH) Performance Monitoring for the Network Element View*.

ITU-T Recommendation G.774.02 (11/94), *Synchronous Digital Hierarchy (SDH) Configuration of the Payload Structure for the Network Element View*.

ITU-T Recommendation G.774.03 (11/94), *Synchronous Digital Hierarchy (SDH) Management of Multiplex-Section Protection for the Network Element View*.

ITU-T Recommendation G.774.04 (07/95), *Synchronous Digital Hierarchy (SDH) Management of the Subnetwork Connection Protection for the Network Element View*.

ITU-T Recommendation G.774.05 (07/95), *Synchronous Digital Hierarchy (SDH) Management of Connection Supervision Functionality (HCS/LCS) for the Network Element View*.

ITU-T Recommendation G.805 (11/95), *Generic Functional Architecture of Transport Networks*.

ITU-T Draft Recommendation G.853-01, *Common Elements of the Information Viewpoint for the Management of a Transport Network*.

ITU-T Draft Recommendation G.853-02, *Subnetwork Connection Management Information Viewpoint*.

ITU-T Draft Recommendation G.855-01, *Management of the Transport Network - Class Library for GDMO Transport Network Model.*

ITU-T Recommendation M.3010 (05/96), *Principles for a Telecommunications management network.*

ITU-T Recommendation M.3100 (07/95), *Generic Network Information Model*.

ITU-T Recommendation Q.821 (03/93), *Stage 2 and Stage 3 Description for the Q3 Interface - Alarm Surveillance*.

ITU-T Recommendation X.710 (03/91), *Common management information service definition for CCITT applications*.

ITU-T Recommendation X.721 (02/92), *Information Technology - Open Systems Interconnection - Structure of Management Information: Definition of Management Information*.

ITU-T Recommendation X.733 (02/92), *Information technology – Open Systems Interconnection – Systems Management: Alarm reporting function*.

ITU-T Recommendation X.734 (09/92), *Information technology – Open Systems Interconnection – Systems Management: Event report management function*.

ITU-T Recommendation X.735 (09/92), *Information technology – Open Systems Interconnection – Systems Management: Log control function*.

AF-NM-0058.000, *NMS/EMS Network View Interface Requirements, and Logical MIB*, (ATM Forum Technical Committee)

AF-NM-0073-000 Letter Ballot, *M4 Network View CMIP MIB Specification* Version 1.0, November 1996.

GR-253-CORE (December 1995), *Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria (A Module of TSGR, FR-NWT-000440)*, Issue 2, (Bellcore)

TR-NWT-000496 (May 1992), *SONET Add-Drop Multiplex Equipment (SONET ADM) Generic Criteria*, Issue 3, (Bellcore).

GR-836-IMD (September 1996), *Generic Operations Interfaces Using OSI Tools - Information Model Details:Transport Configuration and Surveillance for Network Elements*, Issue 2, (Bellcore)

GR-1042-IMD (September 1996), *Generic Requirements for Operations Interfaces Using OSI Tools - Information Model Overview: Synchronous Optical Network (SONET) Transport Information Model*, Issue 2, (Bellcore)

SR-TSV-002671 (June 1993), *EML Applications for Fault Management: Subnetwork Root Cause Alarm Analysis*, Issue 1 (Bellcore).

SR-TSV-002672 (March 1994), *EML Applications for Fault Management: Intelligent Alarm Filtering for SONET*, Issue 1 (Bellcore).

GR-2869-CORE (October 1996), *Generic Requirements for Operations Based on the TMN Architecture*, Issue 2 (Bellcore).

SIF-NM-9606-050 (Draft), *Requirements for SIF OS Platforms*.

1.6 Definitions

This section lists the terms and definitions used in this document. Many of these terms have different definitions in other standards documents. Attempts were made to adopt the most common definition of each term. The definition includes the context in which the term is used. Where possible, the relationship between a term and the real world is provided.

KEY:

- "double quotes" - term is defined elsewhere in this glossary
- *italics* - element of the SIF Information Model (e.g., object, attribute, etc.)

1.6.1 access group

A group of co-located "network trail termination points" within a "layer network domain." The *sonetAccessGroup* managed object class is used to represent an access group in the model.

1.6.2 administrative domain

A set of network and administrative resources grouped for management purposes. For the NMS/EMS interface, the grouping of managed objects corresponds to the resources managed by an EMS agent. An administrative domain will typically encompass a number of "layer network domains" associated with distinct transport layers. The *networkR1* managed object class is used to represent an administrative domain in the model.

1.6.3 Client/Server (C/S) Pointers

The client pointer attribute (clientLayerNCTP) identifies the object instances of the related sonetNetworkCTP (and subclasses) in the clientserver layer. The sonetNetworkCTPs may belong to multiple sonetLayerNetworkDomains. The server pointer attribute (serverLayerNTTP) identifies the object instance of the related networkTTP (and subclasses) in the server layer.

1.6.4 characteristic information

A signal with a specific rate and format, which is transferred on "network connections" [G.853-01]. Characteristic information is also associated with termination points independent of whether or not they are supporting connections. The potential signal formats include signals of the SONET hierarchy (VT/STS/OC) and digital tributary signal formats (DS1, DS3).

1.6.5 connection management

A management application designed to manage the set-up, release, and modification of subnetwork connections and trails in a layer network. Connection management will generally require application functions in multiple TMN management layers (i.e., SML, NML, EML, and NEL). Although the focus is on configuration management, functions in other functional areas (e.g., fault, performance management) may be included in the application. SIF functional requirements for a SONET connection management application are described in G.805.

1.6.6 Connection Termination Point (CTP)

A Connection Termination Point is a managed object that terminates a link connection. See M.3100.

1.6.7 Downstream Connectivity Pointer (DCP)

Defined in M.3100, the downstream connectivity pointer attribute points to the termination point managed object, within the same managed element, that receives information (traffic) to this termination point at the same layer, or is null.

1.6.8 drop

The port on a SONET network element where the service to an end customer may be connected, e.g., a tributary card on a SONET ADM. For example, a drop for a DS1 customer service may be provided by a VT1.5 card terminating a VT1.5 trail.

1.6.9 drop and continue

The ability of a SONET add-drop multiplex to pass the same signal (STS/VT) that is being dropped onto the outgoing OC-N signal [SR-2672].

1.6.10 dynamic subnetworks

A management capability that allows modification of a subnetwork's properties in terms of termination points or contained subnetworks and links under partitioning. A typical modification would be the reallocation of sNTPs.

1.6.11 Element Management Layer (EML)

An abstraction of the functions provided by systems which manage each network element on an individual basis.

1.6.12 Element Management System (EMS)

A management system, which provides functions at the Element Management Layer, and could also include functions at the Network Management Layer. The "administrative domain" associated with the EMS agent could be delimited by geographical area, topology, or supplier product (as examples) within the provider's network.

1.6.13 hardwired multiplex

An add-drop multiplex configuration in which specific VT/STS-1 time-slots are dedicated to specific low-speed ports [SR-2672].

1.6.14 layer network

A "topological component" that includes other topological components, transport entities, and transport processing functions that describes the generation,

transport and termination of a particular characteristic information [G.853-01]. As an example, a layer network may be associated with SONET STS-1 transport.

1.6.15 layer network domain

The part of a layer network which is managed by a management system [G.853-01]. For the NMS/EMS interface, the relevant management system is the EMS. The *sonetLayerNetworkDomain* managed object class is used to represent a layer network domain in the model.

1.6.16 link

A “topological component” that provides transport capacity between two endpoints in different subnetworks via a fixed (i.e., inflexible routing) relationship[1]. The endpoints are “subnetwork termination point pools.” Multiple links may exist between a pair of subnetworks. A link also represents a set of “link connections.” The *sonetLink* managed object class is used to represent a link in the model.

1.6.17 link connection

A “transport entity” that represents the fixed capacity of transfer of “characteristic information” transparently across a link. A link connection is delineated by “network connection termination points” or “NE-view connection termination points.” The “network connection termination points” can be associated with “subnetwork termination points” by relationship. The *sonetLinkConnection* managed object class is used to represent a link connection in the model.

1.6.18 Management Applications Function (MAF)

An application process participating in system management. The management application function includes an agent (being managed) and/or manager [G.784].

1.6.19 multiple partitioning views

A management capability that supports more than one scheme of partitioning subnetworks. This allows the use of different partitioning schemes for different functional areas or different management systems.

1.6.20 network connection

A “transport entity” formed by a series of contiguous "link connections" and/or "subnetwork connections" between subnetwork termination points. A network connection may extend across “layer network domains” associated with more than one “administrative domain.” A network connection is not represented by an object class in the model.

---

[1] An exception to the “fixed” relationship is when protection mechanisms are applied.

1.6.21 network connection termination point

An extremity of a “link connection” [G.855-01]. It is also a network level abstraction of a nodal (NE) view connection termination point. The *networkCTP* managed object class is used to represent a network connection termination point in the model.

1.6.22 network connection termination point bidirectional

A network connection termination point that represents the functionalities of both a network connection termination point source and network connection termination point sink in the model.

1.6.23 network connection termination point sink

A unidirectional network connection termination point that is intended to be bound to the output of a unidirectional “link connection.” The *sonetNetworkCTPSink* managed object class is used to represent a network connection termination point sink in the model.

1.6.24 network connection termination point source

A unidirectional network connection termination point that is intended to be bound to the input of a unidirectional “link connection.” The *sonetNetworkCTPSource* managed object class is used to represent a network connection termination point source in the model.

1.6.25 Network Element (NE)

A system that supports at least “NEFs” and may also support “Element Management Layer” Functions/Mediation Functions. It cannot be further decomposed into managed elements in the context of a given management function.

1.6.26 Network Element Function (NEF)

A function within a SONET entity that supports the SONET based network transport services, e.g. cross-connections.

1.6.27 Network Element Layer (NEL)

An abstraction of functions related specifically to the technology, vendor, and the network resources or network elements that provide basic communications services.

1.6.28 network element view (NE view)

The network element view is the network view representating information received directly from network elements which is defined in other information models such as GR-1042.

1.6.29 Network Management Layer (NML)

An abstraction of the functions provided by systems which manage network elements on a collective basis as subnetworks, and/or as individual entities.

1.6.30 Network Management System (NMS)

An entity which implements functions at the Network Management Layer. It may also include Element Management Layer functions.

1.6.31 NMS Environment - *under study*

A set of Network Management Systems (NMS) which cooperate to manage one or more subnetworks.

1.6.32 network trail termination point

An extremity of a "trail" [G.855-01]. It is also a network level abstraction of a nodal (NE) view trail termination point. A network trail termination point includes trail termination functions that ensure integrity of information transport on an end-to-end basis (see Figure I.2 in G.855-01 for a mapping between termination point managed objects [G.855-01] and termination functions and access points in a functional network architecture [G.853-01]). The *sonetNetworkTTP* managed object class is used to represent a network trail termination point in the model.

1.6.33 network trail termination point bidirectional

A network trail termination point that represents the functionalities of both a network trail termination point source and network trail termination point sink in the model.

1.6.34 network trail termination point sink

A network trail termination point that is intended to be bound to the output of a unidirectional trail. The *sonetNetworkTTPSink* managed object class is used to represent a network trail termination point sink in the model.

1.6.35 network trail termination point source

A network trail termination point that is intended to be bound to the input of a unidirectional trail. The *sonetNetworkTTPSource* managed object class is used to represent a network trail termination point source in the model.

1.6.36 network view

The network view is an abstracted view of the network which would include some level of NE connectivity.

1.6.37 NE-view connection termination point

A managed object class defined in an information model for network elements

which represents the termination of a link connection.

1.6.38 NE-view trail termination point

A managed object class defined in an information model for network elements which represents the termination of a trail.

1.6.39 partitioning

The decomposition of a “subnetwork” into its component “subnetworks” and “links” in a way that reflects the internal structure (topology) of that “subnetwork” or the way that it will be managed. Partitioning may be based on a variety of factors including architecture (e.g., BLSR), supplier product line, or administrative considerations.

1.6.40 point-to-point route

A point-to-point route consists of two end points and, optionally, intermediate points in a layer network. Each end point or intermediate point will consist of a set of sNTPs in the same layer network.

1.6.41 protection

In the network view, protection refers to the ability to switch service from a primary “transport entity” to a preconfigured backup “transport entity” in response to a detected failure on the primary “transport entity.” Protection mechanisms may be entirely within a “layer network” or may involve a server layer mechanism supporting client transport services, and may be activated on the basis of a variety of monitoring. Trail protection is a protection method applied in a “layer network” when a defect condition is detected in the same “layer network.” Subnetwork connection protection is applied in the client layer network when a defect condition is detected in a server layer network, sub-layer or other transport layer network.

1.6.42 restoration

In the network view, restoration refers to an application in which the NMS responds to a confirmed failure by requesting a new connection. This action may be viewed by the EMS as a release of the failed connection and the set-up of a new connection.

1.6.43 route

A sequence of geographical or topological points or components through which “transport entities” may be established. “Transport entities” within a route have common endpoints and common intermediate points (if intermediate points are specified). The role of route endpoint may be played by a “subnetwork” or “access group;” an intermediate point may be associated with a “subnetwork,” or

"subnetwork termination point pool." In an unpartitioned subnetwork view, the route includes all potential connections in the "subnetwork" between "access groups;" in a fully-partitioned subnetwork view, the route may include all available connections supported by a specific set of paths (links) and NEs (fabrics).

1.6.44 route capacity

A measure of the transport capacity available on an end-to-end basis on a specific "route" within a "layer network domain."

1.6.45 route separation

A qualitative measure of the relationship between "routes" assigned to primary and protection connections.

1.6.46 service

Service or transport service is the communications product purchased by a customer, represented by a "subnetwork connection." Services come in many forms, including switched services, data services, and private line services.

1.6.47 state

State is an attribute type representing the condition or an object instance. See X.721.

1.6.48 subnetwork

A "topological component" used to effect routing of a specific "characteristic information" [G.853-01]. A subnetwork is associated with a specific "layer network." Within a given layer, "partitioning" may be applied to decompose a subnetwork into its component subnetworks and links. The *sonetSubnetwork* managed object class is used to represent a subnetwork in the model.

1.6.49 subnetwork connection (SNC)

A "transport entity" that transfers information across a subnetwork [G.853-01]. A point-to-point subnetwork connection connects two "subnetwork termination points." An SNC may be either a stand-alone SNC, or a concatenation of SNCs and link connections. The *sonetSubnetworkConnection* managed object class is used to represent a subnetwork connection in the model.

1.6.50 Subnetwork Management System (subNMS) - *under study*

A Network Management System, which is managing one or more subnetworks, and which is managed by one or more Network Management Systems.

1.6.51 subnetwork termination point

An abstraction that represents the binding between a "subnetwork" and either a

"network connection termination point", "NE-view connection termination point", "network trail termination point" or a "NE-view trail termination point." It also represents the potential for connection across a subnetwork [G.855-01]. A subnetwork termination point is represented in the model by the *sonetSNTP* object class.

1.6.52 subnetwork termination point bidirectional

An abstraction that represents both a sink and a source subnetwork termination point. An *sonetSNTPBidirectional* object class represents this abstraction in the model.

1.6.53 subnetwork termination point pool

A set (possibly empty) of subnetwork termination points at the frontier of a given subnetwork [G.855-01]. In SONET, a subnetwork termination point pool is used to terminate a link. An *sonetSNTPPool* object class represents this abstraction in the model.

1.6.54 subnetwork termination point sink

A subnetwork termination point that represents the potential binding of the output of a unidirectional "subnetwork connection" and either a "network connection termination point source" or a "network trail termination point sink." The *sonetSNTPSink* object class represents this abstraction in the model.

1.6.55 subnetwork termination point source

A subnetwork termination point that represents the potential binding of the output of either a unidirectional "link connection" or a "network trail termination point source" and the input of a unidirectional "subnetwork connection." The *sonetSNTPSource* object class represents this abstraction in the model.

1.6.56 subnetwork capacity

A measure of the total/available transport capacity within a subnetwork. This measure may be useful in planning for network topology changes. (See "route capacity.")

1.6.57 Termination Point (TP)

A Termination Point is a managed object that terminates "transport entites" such as "trails" and connections. See M.3100.

1.6.58 Time-Slot Assignment (TSA)

The capability to flexibly assign add-dropped signals, but not through signals. Through signals maintain the same time-slots on the incoming and outgoing signals [SR-2671].

1.6.59 Time-Slot Interchange (TSI)

The capability to flexibly assign both add-dropped signals and through signals [SR-2671].

1.6.60 topological component

An architectural component, used to describe the transport network in terms of the topological relationships between sets of points within the same “layer network” [G.853-01]. Examples of topological components include “layer network,” “subnetwork,” and “link.”

1.6.61 trail

A "transport entity" which consists of an associated pair of "unidirectional trails" capable of simultaneously transferring information in opposite directions between their respective inputs and outputs [G.853-01]. A trail also represents the transfer of “characteristic information” between “network trail termination points” or “NE-view trail termination points.” A trail is supported by a “subnetwork connection” or “network connection” and in addition includes trail termination functions that ensure integrity of information transport (i.e., via monitoring) on an end-to-end basis. A trail may be established to directly support an end-to-end network service or to provide “link connections” within the client layer. A trail is represented in the model by the *sonetTrail* managed object class.

1.6.62 trail termination function

A transport processing function that allows the monitoring of information transport on trails on an end-to-end basis. Two types of trail termination functions are defined: a trail termination source adds monitoring information to the “characteristic information” input at one end; a trail termination sink removes the monitoring information and presents the “characteristic information” at the other end.

1.6.63 Trail Termination Point(TTP)

A Trail Termination Point is a managed object that terminates a trail. It represents the access point to a subnetwork. See M.3100.

1.6.64 transport entity

An architectural component which transfers information between its inputs and outputs within a “layer network” [G.853-01]. Examples of transport entities include “subnetwork connection,” “link connection,” and “trail.”

1.6.65 Upstream Connectivity Pointer (UCP)

Defined in M.3100, the upstream connectivity pointer attribute points to the termination point managed object, within the same managed element, that sends information (traffic) to this termination point at the same layer, or is null.

1.7 Acronyms

ADM	Add Drop Multiplexer
AIS	Alarm Indication Signal
APS	Automatic Protection System
ATMF	Asynchronous Transfer Mode Forum
BLSR	BiDirectional Line Switched Ring
BML	Business Management Layer
C/S	Client/Server
CMISE	Common Management Information Service Element
CTP	Connection Termination Point
DCN	Digital Communication Network
DCP	Downstream Connectivity Pointer
DCS	Digital Cross-connect System
DSn	Digital Signal hierarchy, layer n
EFD	Event Forwarding Discriminator
EML	Element Management Layer
EMS	Element Management System
E-R	Entity-Relationship
ETSI	European Telecommunications Standards Institute
GDMO	Guidelines for the Definition of Managed Objects
ITU-T	International Telecommunications Union
LOF	Loss of Frame
LOP	Loss Of Pointer
LOS	Loss of Signal
MAF	Management Application Function
MIB	Management Information Base
NE	Network Element
NEF	Network Element Function
NEL	Network Element Layer
NMF	Network Management Forum

NML	Network Management Layer
NMS	Network Management System
NSA	Non-Service-Affecting
OC-N	Optical Carrier - level N
OOF	Out Of Frame
OS	Operating System
OSI	Open Systems Interconnection
OTDR	Optical Time Delayed Reflectometry
QOS	Quality Of Service
PM	Performance Monitoring
RCAA	Root Cause Alarm Analysis
RDI	Remote Defect Indication
RFI	Remote Failure Indication
SA	Service-Affecting
SLM	Signal Label Mismatch
SML	Service Management Layer
SNC	SubNetwork Connection
SOA	Statement Of Application
STS-N	Synchronous Transport Signal level N
STS-Nc	Synchronous Transport Signal level N, concatenated
TCA	Threshold Crossing Alarm
TMN	Telecommunications Management Network
TP	Termination Point
TSA	Time-Slot Assignment
TSI	Time-Slot Interchange
TTP	Trail Termination Point
UCP	Upstream Connectivity Pointer
UPSR	UniDirectional Path Switched Ring
VT	Virtual Tributary

2. Statements of Application

A Statement of Application (SOA) defines the service provider's functional requirements for a management application. In this section, SOAs are presented for Connection Management and Fault Management applications. Each SOA represents a broad set of requirements that may have impact on management applications at various TMN layers as well as interfaces between management systems supporting the applications. Following each SOA section is a section that indicates which particular SOA requirements are supported in this version of the information model for the NMS/EMS interface.

2.1 Connection Management SOA

The objectives of connection management are:

- the ability to setup and tear down connections in a subnetwork
- the ability to report and control the configuration and status of components of a subnetwork

2.1.1 Service Provider Marketing Request

The typical components of a service request are type of service and need for an alternative path. Type of service information may include service rate (e.g.OC-3, OC-12, DS1, DS3), payload mapping (STS-1, STS-3c), Add Drop Multiplexing and QOS.

Alternate paths may be controlled at different TMN levels. Example protection mechanisms include:

1. Protection within a subnetwork where the EMS controls the details of the protection mechanism.

2. Path protection in which the NEs containing the path terminations control the path switching. This mechanism requires that the NMS configure primary and secondary paths and create an association between them.

3. NML restoration, in which the NMS responds to a confirmed failure by requesting a new connection.

Availability of bandwidth is a tool needed for a marketing request. Marketing must do a quick check to see if the service the customer is ordering is available. A service request may require the ability to add scheduling for service turn on or turn off, and the addition of customer name and location. A service request may also include or require time of day, "Start Date", "Termination Date", or some other scheduling mechanism as well as STATE information to indicate that the

facilities are reserved.

Marketing will need the ability to locate endpoints which includes the geographic as well as the logical address. Customers may require a specific drop port which implies that specific subnetwork or network element details may be required.

2.1.2 Subnetwork & NE Considerations

In support of an Inventory Application, information about facilities, ports, drop plugs, cross-connects, timeslot availability, intermediate and end test points, and state information (in service, assigned, unassigned, alarmed, outstanding conditions) must be provided at an external interface. Because the network itself is assumed to contain the most important source of network inventory information, stewardship (primary responsibility for creation/deletion/modification) of inventory data is assumed to lie more within the EMS than within the NMS.

In support of a Connection Management Application, the following topology information must be supported in the Connection Management Model:

- Partitioning

- Logical and physical geographic and termination point information are needed at the NMS to allow tariff based path diversity. (This could be a means of "route selection" by the user or presented to the user.) This also relates to finding end points A & Z, and links between subnetworks.

- The model should be rich enough to allow a network wide view at the NML (all subnetworks) and a subnetwork wide view at the EMLs.

- Network Topology Autodiscovery is needed, but can be a later modeling effort.

In support of a Connection Management Application, the following information must be supported in the Connection Management Model:

- rates - selection of a specific facility such as DS1.

- protect scheme which allows the following strategies:

    1. Protection within subnetwork - a protected subnetwork connection or trail within a subnetwork may be requested by the NMS and associated with the subnetwork connection (e.g.: ring protection); the NMS does not have visibility into the details of the protection mechanism.

2. Path protection in which the NEs containing the path terminations control the path switching requires that the NMS configure primary and secondary paths and create an association between them. The paths and their association are passed to the EMSs whose domains contain the terminating NEs. An EMS requested to establish intermediate subnetwork connections receives two (or possibly one) requests for subnetwork connections for which an association is not necessary. (Reversion to the original is an application decision, but the model must be rich enough to provide the proper notification upon restoral of service to revert to original configuration).

3. NML restoration, in which the NMS responds to a confirmed failure by requesting a new connection would be viewed by the EMS as a release of the failed connection and the set-up of a new connection. The Manage Pending Network Changes MAF in the NMS coordinates these actions. At the NMS/EMS interface this involves subnetwork connection set-up and release actions. (Reversion to the original is an application decision, but the model must be rich enough to provide the proper notification upon restoral of service to revert to original configuration).

- drops available - We must distinguish between fiber and wire drop interfaces and be able to retrieve rate/format supported at drop.

- bandwidth availability including scheduling which requires the ability to add scheduling for service turn on/off and the addition of customer name and location. A service request may also include or require time of day, "Start Date", "Termination Date", or some other scheduling mechanism as well as STATE information to indicate that the facilities are reserved.

  Reservation of bandwidth will also be required. The ability to query to determine the capacity remaining and trigger a notification to add more, noting time differences from reservations is required. If bandwidth falls below the threshold additional capacity will be ordered. This will require the ability to add THRESHOLDING, or THRESHOLD Crossing Information in the model.

- Assign priorities to circuits/components/sessions. This could be a vehicle for supplying "bandwidth on demand". While assignment of priorities may be an application issue, it may have implications on the model.

- Gather unused facilities (DS1, DS3. STS-1s. etc.) into pool(s) of unused facilities (DS1, DS3. STS-1s. etc.). Information required includes total

available bandwidth between two ports on a subnetwork, and partitioned views (contained subnetworks and links) of routes representing that total available bandwidth.

- Ability to accept any manually entered preferred routes or explicitly preferred route separation assignments.

- Query for capacity within a subnetwork or subnetworks.

2.1.3 Additional Functional and Model Requirements

The following additional functions are required:

- Reserve all paths. This information will be forwarded to the network level management function for combining with other subnetwork paths.

- The NML will build the end-to-end connections for Primary and Secondary Paths. All paths are linked to unique circuit order identifier. The circuit order identifier could be a "service number" attribute returned via the information model to the service provider application.

- Monitor all alarms while provisioning.

- Mark all termination points and all links used as in-service. Refine the STATEs within the model.

- Test end-to-end connections to ensure working order. Modeling for testing consists of two parts: state management of resources to be tested and managing testing resources. We need to add states to objects in our model to allow resources to be taken out of service for testing.

- Notify requester of SUCCESS or FAILURE of the connection establishment.

- Notify all downstream OSs (including the Inventory Application) that the circuit is assigned and can't be reused, must be monitored etc.

- Support EFDs. This should be a standard function of CMISE supporting notifications to all management applications that requested this type of notification.

2.2 Connection Management Functions Supported

Only a subset of the requested Connection Management Functions are supported in this initial version of the model. The table below shows which

Connection Management Functions are supported by this information model:

CONNECTION MANAGEMENT FUNCTION	FUNCTION SUPPORTED
Network Topology	
Partitioning	Yes
A & Z End Points	Yes
Geographic Location	Yes
Route selection	Yes
Rates x	Yes
Protection Schemes	
Subnetwork Protection	No
Path Protection	No
NMS initiated Connection Restoration	Not applicable at this interface.
Drops Available	Yes
Scheduling and Reservation	No
Bandwidth Availability	Sufficient information is available to calculate route capacity
Subnetwork Pools of unused facilities	No
Establish Primary and Secondary Connections	
Establish Connection	Yes
Establish Monitored Connection	No
Release Connection	Yes
Mark TPs and Links as IN-SERVICE	No
Notify Connection SUCCESS or FAILURE	Yes

2.3 Fault Management SOA

2.3.1 Motivations for Fault Management Operations

The primary objective of Fault Management is the timely identification, isolation,

diagnosis and resolution of network deficiencies. Fault Management provides the following network management benefits:

- decreased time to isolate and resolve network deficiencies
- decreased need for experienced network technicians
- better information from which to manage vendor activities
- increased network availability
- better information from which to manage operations activities
- early detection of potential problems will allow correction before full failure occurs.

The fundamental strategy for fault isolation and problem diagnosis is the sensible collection and classification of alarms from various systems. User and application connectivity will be maintained by submitting a reconfiguration request to the Reconfiguration Management processes.

Fault Management includes functions that enable fault detection and fault isolation and allow for the correction of abnormal behavior. They include alarm and trouble monitoring capabilities such as alarm surveillance, collection, classification, correlation, notification, display, review, suspension and clearing, and fault localization, isolation and testing. SONET Fault Management involves network facilities and network equipment. (For clarification, network facilities include DS1s, OC-3s, etc. Network equipment includes network elements (NEs) such as Digital Cross-Connect Systems (DCSs) and SONET Add-Drop Multiplexers (ADMs).) Alarms can originate from entities defined as networks, subnetworks, and the ring interconnections. (A set of NEs can be defined as a subnetwork. Subnetworks are viewed as logical entities consisting of multiple physical nodes.) Fault Management responds in a reactive mode to network alarms and to Performance Management requests (based on analysis of network events, PM may generate alarms).

Alarms can be considered either service affecting or transient and are categorized as one of four severity codes: critical, major, minor or informational (i.e., an event). Multi-level alarms will identify self-correcting errors, errors not affecting system operation, errors requiring operator intervention, alarms for critical processes and alarms which notify of any service-threatening situation, e.g., overload conditions.

At a minimum, alarms will contain the following information:

- Originating resource (network element, facilities, element management system, Performance Monitoring, etc.)
- Trouble explanation (e.g., defined alarm code)
- Severity code

- Date and time alarm condition occurred
- Duration of alarm condition
- Status of alarm (e.g., active, cleared, acknowledged, etc.)

Fault Management's span of control begins with the receipt of either an external or internal alarm and ends with the request for the generation of a trouble ticket.

2.3.2 Fault Management Operations

The following text describes the operations required by a service provider in performing the SONET Fault Management process.

1. Set Parameters
   - Establish criteria for reliability, availability and survivability. As defined in GR-2869-CORE Generic Requirements for Operations Based on the TMN Architecture:
   - Reliability refers to measures of the mean time between failure and the mean time to repair equipment.
   - Availability refers to the percentage of time services and resources are ready for use.
   - Survivability refers to the robustness of the network in the presence of faults.
   - Register facilities for monitoring
   - Define alarm types to be screened
   - Set up alarm/event thresholds and other event criteria
   - Assign priorities to circuits/components/sessions
   - Set up rule base

2. Monitor Network Status
   - Monitoring events and alarms
   - Maintain log of events and alarms
   - Status request of any component on the network (automated and manual)
   - Collect/poll for prescreened system events or alarms
   - Collect performance information
   - Filter duplicate and informational messages
   - Suppress recurring messages
   - Suppress minor or informational alarms
   - Archive events
   - Continual data analysis and results review
   - Generate event and alarm reports

3. Detect Network Deficiencies
   - Update the user interface (network topology)

- Retrieve alarms
- Reformat alarms
- Identify alarms
- Identify alarm implications
- Archive alarms

4. Receive notification of a fault condition from alarms, from the Performance Management System (e.g., exceeding thresholds for PM triggers), or from a customer (via a trouble report or a Customer Network Management system).

5. Isolate Trouble
- Heuristic (methodical and intuitive) analysis of alarms
- Correlate alarms
- Review correlation results
- Determine source alarms
- Suppress related alarms
- Test to further isolate trouble and recommend further action for problem resolution.

6. Diagnose Fault
- Analyze event log
- Perform root cause analysis
- Make an informed hypothesis concerning the source of the trouble
- Determine condition severity
- Suggest tests
- Utilize remote test access and local diagnostic tools to test resources and provide multiple test points along the circuit path
- Analyze test results
- Suggest appropriate actions based upon diagnostic results

7. Resolve Trouble
- Initiate the fault resolution process
- Maintain connectivity to production applications
- Automatically generate trouble ticket request
- Track status of the trouble
- Escalate as necessary
- Send requests to:
- Reconfiguration Management to reconfigure network or reroute traffic
- Connection Management to order new services where required

8. Restore Service
- Initiate full backup, recovery and restore procedures
- Test the end-to-end connections to assure they work.

- Clear alarms
- Modify alarm status indicators
- Update user interface
- Mark inventories with failed resources and new routes, all nodes with pertinent state information, etc.
- Notify all downstream OSs that reconfiguration has occurred.
- Notify users of reconfiguration, reroute or restoral.
- Verify restoral of service.
- Close trouble log and archive

9. Restore Original Configuration

2.4 Fault Management Operations Supported

Only a subset of the requested Fault Management Functions are supported in this initial version of the model. The table below shows which Fault Management Operations are supported by this information model:

Fault Management Operations	Function Supported
1. Set Parameters	
* Establish criteria for reliability, availability and survivability. As defined in GR-2869-CORE Generic Requirements for Operations Based on the TMN Architecture:	No
* Reliability refers to measures of the mean time between failure and the mean time to repair equipment.	No
* Availability refers to the percentage of time services and resources are ready for use.	No
* Survivability refers to the robustness of the network in the presence of faults.	No
* Register facilities for monitoring	No
* Define alarm types to be screened	No
* Set up alarm/event thresholds and other event criteria	No
* Assign priorities to circuits/components/sessions	No
* Set up rule base	No
2. Monitor Network Status	
* Monitoring events and alarms	Yes
* Maintain log of events and alarms	Yes
* Status request of any component on the network (automated and manual)	No
* Collect/poll for prescreened system events or alarms	Yes
* Collect performance information	Yes
* Filter duplicate and informational messages	Yes
* Suppress recurring messages	Yes

Fault Management Operations	Function Supported
* Suppress minor or informational alarms	Yes
* Archive events	Yes
* Continual data analysis and results review	Yes
* Generate event and alarm reports	Yes
3. Detect Network Deficiencies	
* Update the user interface (network topology)	No
* Retrieve alarms	Yes
* Reformat alarms	Yes
* Identify alarms	Yes
* Identify alarm implications	Yes
* Archive alarms	Yes
4. Receive notification of a fault condition from alarms, from the Performance Management System (e.g., exceeding thresholds for PM triggers), or from a customer (via a trouble report or a Customer Network Management system).	Yes for fault conditions from alarms, no for other conditions
5. Isolate Trouble	
* Heuristic (methodical and intuitive) analysis of alarms	Yes
* Correlate alarms	Yes
* Review correlation results	Yes
* Determine source alarms	Yes
* Suppress related alarms	Yes
* Test to further isolate trouble and recommend further action for problem resolution.	No
6. Diagnose Fault	
* Analyze event log	Yes
* Perform root cause analysis	Yes
* Make an informed hypothesis concerning the source of the trouble	Yes
* Determine condition severity	Yes
* Suggest tests	No
* Utilize remote test access and local diagnostic tools to test resources and provide multiple test points along the circuit path	No
* Analyze test results	No
* Suggest appropriate actions based upon diagnostic results	No
7. Resolve Trouble	
* Initiate the fault resolution process	No
* Maintain connectivity to production applications	No
* Automatically generate trouble ticket request	No
* Track status of the trouble	No
* Escalate as necessary	No
* Send requests to:	No

Fault Management Operations	Function Supported
* Reconfiguration Management to reconfigure network or reroute traffic	No
* Connection Management to order new services where required	No
8. Restore Service	
* Initiate full backup, recovery and restore procedures	No
* Test the end-to-end connections to assure they work.	No
* Clear alarms	No
* Modify alarm status indicators	No
* Update user interface	No
* Mark inventories with failed resources and new routes, all nodes with pertinent state information, etc.	
* Notify all downstream OSS that reconfiguration has occurred.	No
* Notify users of reconfiguration, reroute or restoral.	No
* Verify restoral of service.	No
* Close trouble log and archive	No
9. Restore Original Configuration	

3. Model Verification Scenarios

Scenarios have been developed to validate and refine the information model. Scenarios demonstrate how features of the model are used in the context of an applications process.

3.1 Connection Management Scenarios

The scenarios in this section are Network Creation and Network Connection Configuration. The scenarios include both the network view only and the network view with the NE view. Figure 3-1, shown below, illustrates the assumed network architecture for these scenarios.

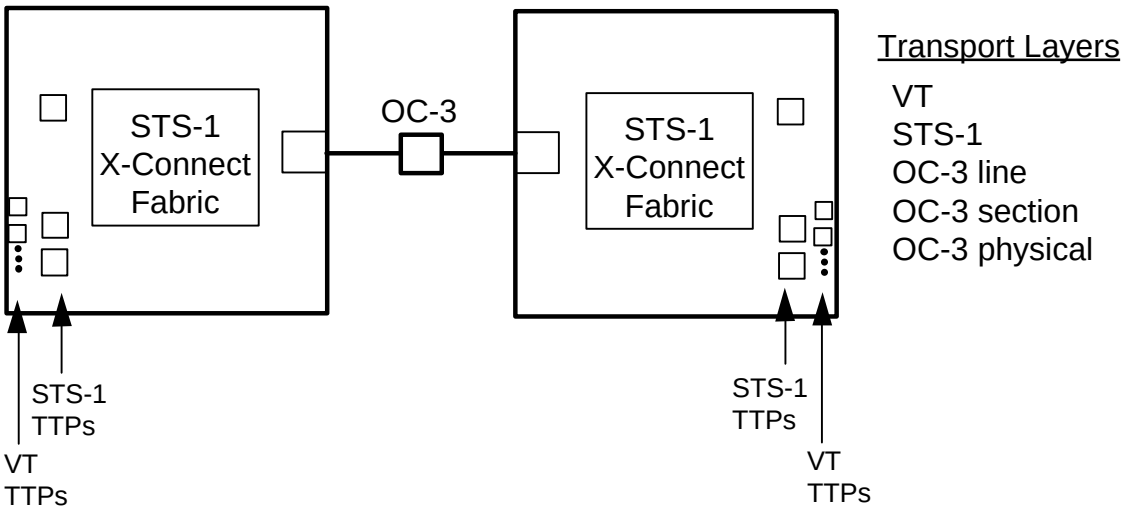


Figure 3-1 Example SONET Network Architecture

Figure 3-1 shows two SONET OC-3 ADMs connected together, through a regenerator, in a point-to-point configuration. These ADMs are configured as terminal multiplexers and they only provide STS-1 cross-connection capability. Not shown are potential DS1 or DS3 drop cards. This network architecture, while extremely simple, allows us to validate the information model without unnecessary complication.

3.1.1 Network Creation (and autodiscovery)

This function involves populating the EMS Management Information Base (MIB) with managed objects representing network resources and reporting the creation of these objects to the NMS. First, layer network resources are created. Then, subnetwork topologies are created. The creation of subnetwork topologies can be a cooperative process between the EMS, providing default views, and the NMS, requesting NMS specific views. This process is discussed below.

3.1.1.1 Layer Network Resource Creation

3.1.1.1.1 Network View Only

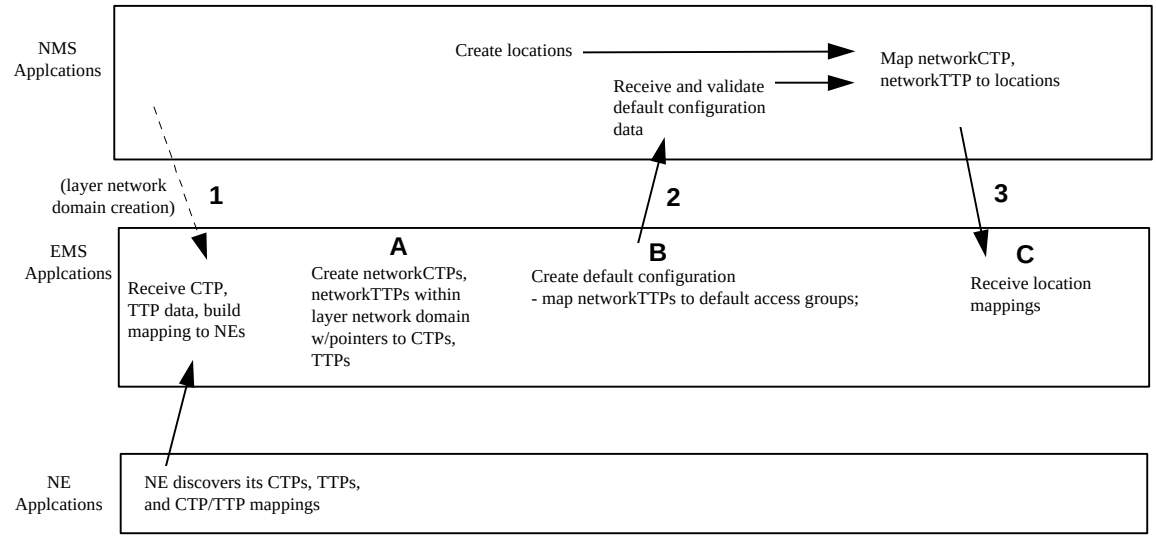


Figure 3-2 Processes and Message Flows - Build Inventory and Map to Nodal/Geographic Objects (Network View Only)

Figure 3-2 shows the applications processes and message flows used to create layer network resources for the network view. The process starts with an inventory download from the NMS and/or an autodiscovered upload from the NEs and ends with network termination points (trail and connection) which reflect NE terminations points and are mapped to access groups, NEs, and geographic locations. The following message flows illustrate exchanges of information between the NMS and the EMS:

- **Message Flow 1 (if not autodiscovered)** - This flow represents inventory download. Object instance(s) of layerNetworkDomain will be created in the MIB as a result of the NMS sending the EMS M-CREATE message(s). The networkCTP and networkTTP object instances are created by the EMS via autodiscovery.
- **Message Flow 2** - This flow represents notifications from the EMS informing the NMS of object creations and attribute value changes as a result of termination point discovery (layerNetworkDomain, accessGroup, networkCTP, and networkTTP) and termination point mappings to accessGroups and network elements.
- **Message Flow 3** - This flow represents the NMS mapping termination points (M-SET on locationName attributes) to geographic locations.

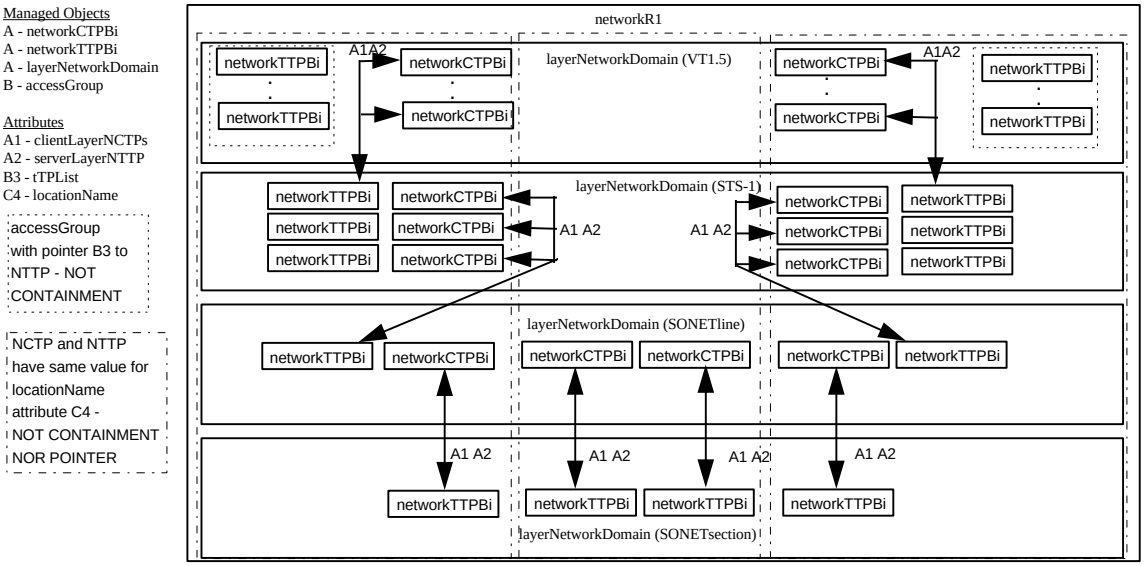


Figure 3-3 Object Instances for Layer Network Resource Creation (Network View Only)

Figure 3-3 shows the managed objects created during the layer network resource creation process. Also shown are the key relationships between managed objects, in the form of name bindings and pointer attributes. Name bindings are represented by solid-line boxes contained within solid-line boxes (e.g., layerNetworkDomain is contained within networkR1). Pointer relationships are shown by arrows and by dashed boxes. Finally, the “A”, “B”, and “C” references refer to “A”, “B”, and “C” in the previous figure. These references represent a sequence of object creation from “A” objects to “B” objects to “C” objects (and associated relationships).

3.1.1.1.2 Network View + NE View

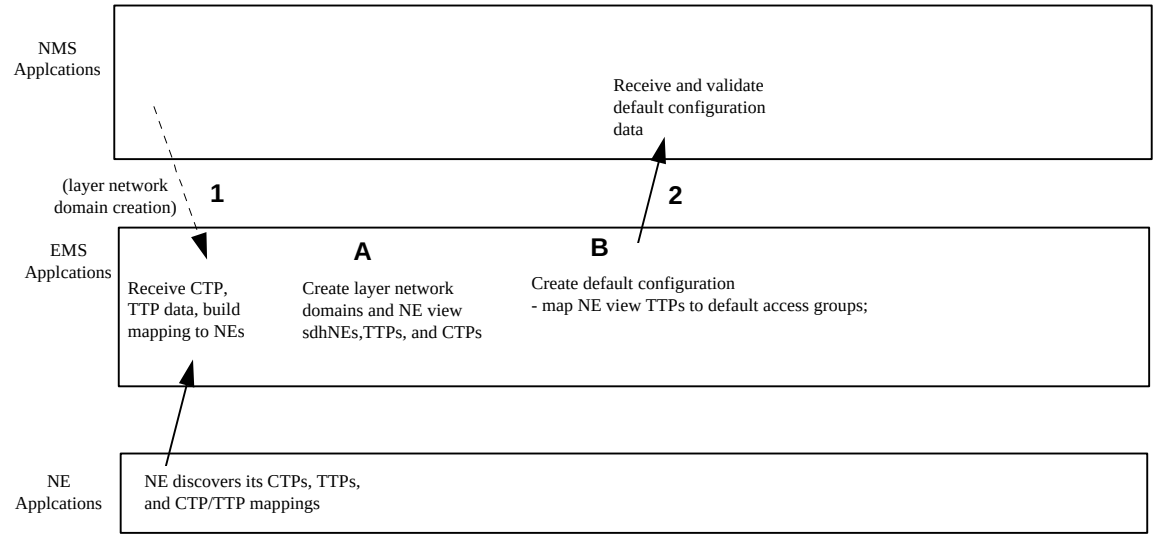


Figure 3-4 Processes and Message Flows - Build Inventory and Map to Nodal/Geographic Objects (Network + NE View)

Figure 3-4 shows the applications processes and message flows used to create layer network resources for the combined network and NE view. The process starts with an inventory download from the NMS and/or an autodiscovered upload from the NEs and ends with NE termination points (trail and connection) mapped to access groups and sdhNEs (which include geographic locations). The following message flows illustrate exchanges of information between the NMS and the EMS:

- **Message Flow 1 (if not autodiscovered)** - This flow represents inventory download. Object instance(s) of layerNetworkDomain will be created in the MIB as a result of the NMS sending the EMS M-CREATE message(s). The NE View CTP and TTP object instances are created by the EMS via autodiscovery.
- **Message Flow 2** - This flow represents notifications from the EMS informing the NMS of object creations and attribute value changes as a result of termination point discovery (layerNetworkDomain, accessGroup, sdhNE, NE View CTPs, and NE view TTPs) and termination point mappings to accessGroups.

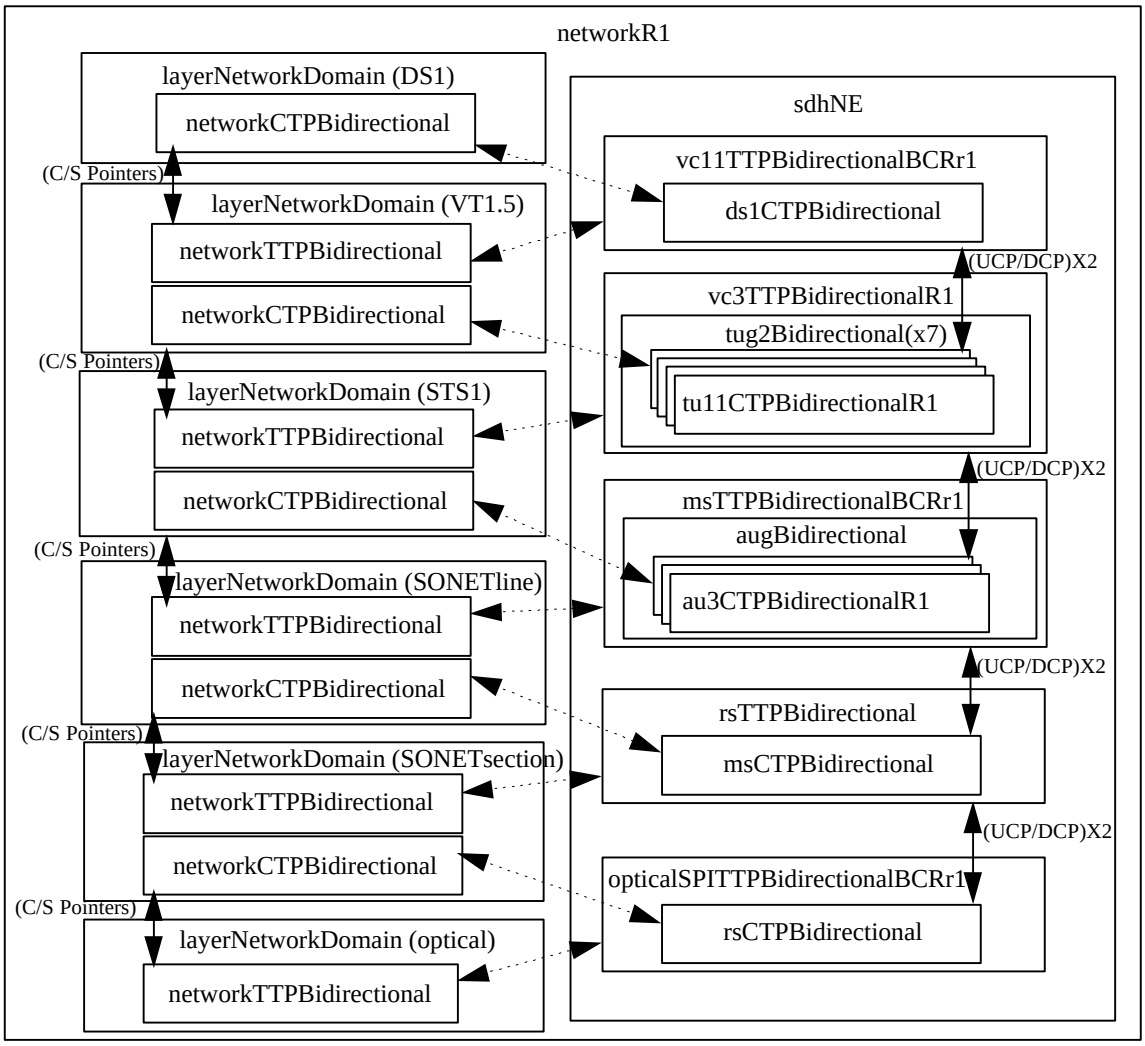


Figure 3-5 Relationships Between Network and NE View Termination Points

Figure 3-5 shows both network and NE view termination point managed objects for the SONET network described in Figure 3-1. This figure illustrates the network termination points that are abstracting the NE termination points. If only the network view is provided by the EMS, the NE view termination points will not be visible to the NMS. If the NE view is provided in addition to the network view, the network termination points will not be instantiated. Instead, the NE view termination points will be visible to the NMS. The UCP/DCP are upstream and downstream connectivity pointers which may exist (note: this assumes connections exist). The C/S Pointers are the clientLayerNCTPs and serverLayerNTTP attributes. The dotted arrows show the network view TPs that abstract the NE view TPs. All layers from optical to DS1 are shown, unlike the object instance figures which show only the SONET section to VT1.5 layers.

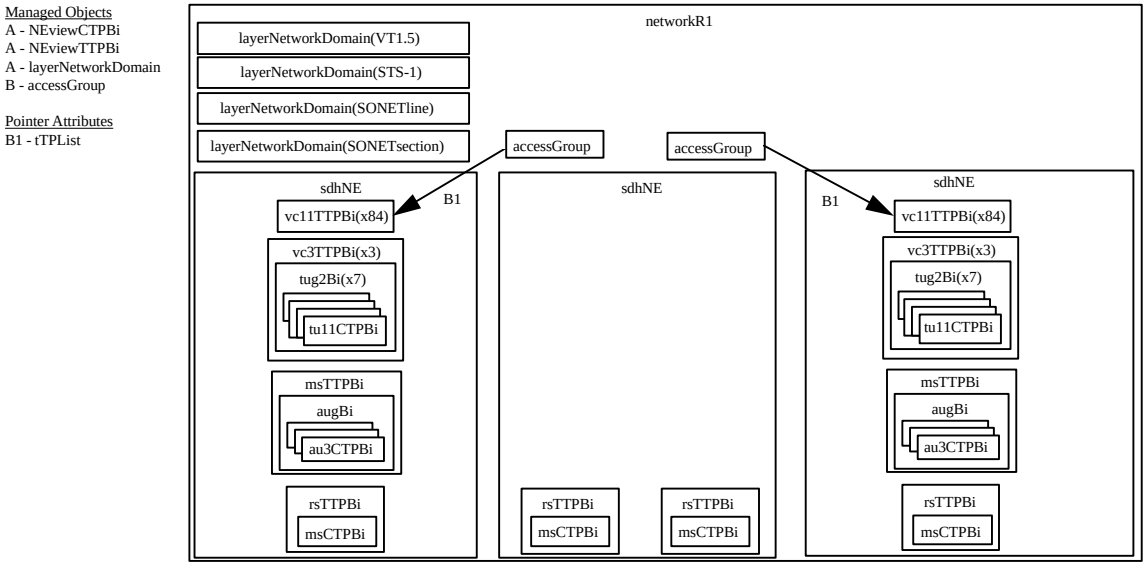


Figure 3-6 Object Instances for Layer Network Resource Creation (Network + NE view)

Figure 3-6 shows the managed objects created during the layer network resource creation process for the network and NE view. Also shown are the key relationships between managed objects, in the form of name bindings and pointer attributes. Name bindings are represented by solid-line boxes contained within solid-line boxes (e.g., layerNetworkDomain is contained within networkR1). Pointer relationships are shown by arrows and by dashed boxes. Finally, the “A”, “B”, and “C” references refer to “A”, “B”, and “C” in the previous figure. These references represent a sequence of object creation from “A” objects to “B” objects to “C” objects (and associated relationships).

3.1.1.2 Subnetwork Topology Creation

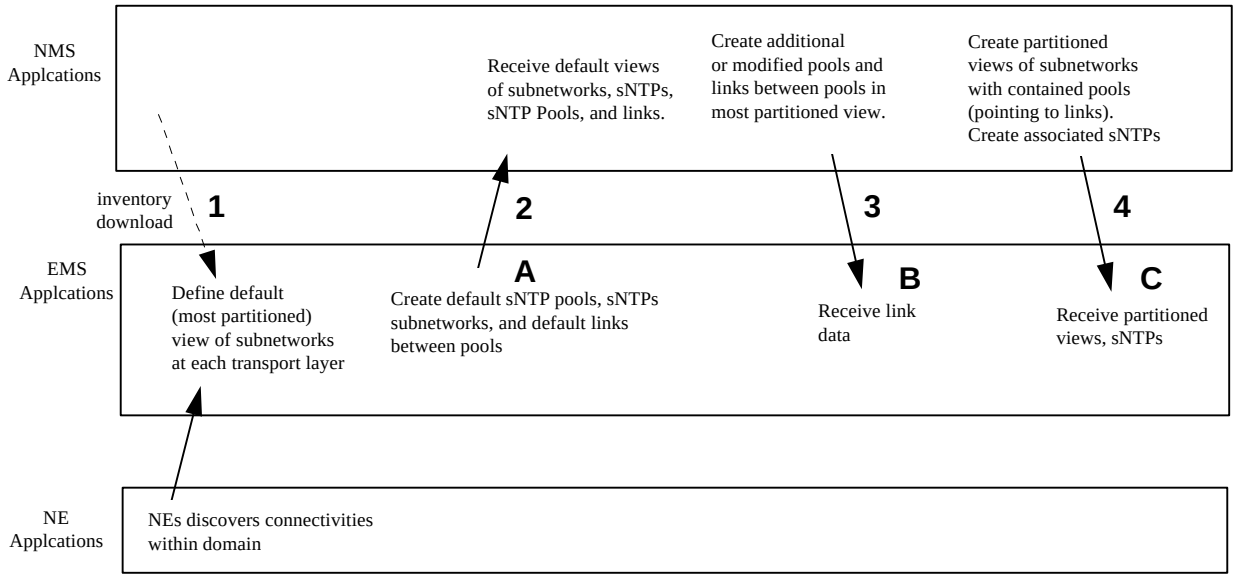


Figure 3-7 Processes and Message Flows - Build Subnetworks, Links, and Partitioned Views

Figure 3-7 shows the applications processes and message flows used to create subnetwork topologies. The process starts with an inventory download from the NMS and/or an autodiscovered upload from the NEs and ends with partitioned views of subnetworks, links, and subnetwork termination points. The following message flows illustrate exchanges of information between the NMS and the EMS:

- **Message Flow 1 (if not autodiscovered)** - This flow represents inventory download. Object instances of subnetwork, sNTP, sNTTPool, and link will be created in the MIB as a result of the NMS sending the EMS M-CREATE messages.
- **Message Flow 2** - This flow represents notifications from the EMS informing the NMS of object creations and attribute value changes as a result of subnetwork topology discovery (subnetwork, sNTP, sNTPPool, and link) and subnetwork termination point mappings to network or NE termination points.
- **Message Flow 3** - This flow represents the NMS creating link object instances (M-CREATE) and relating these to existing sNTPPools (M-SET on linkPointer).
- **Message Flow 4** - This flow represents the NMS creating partitioned views of subnetworks by creating subnetworks, sNTPs, and sNTPPools (M-CREATE) and relating these to existing managed objects (M-SET).

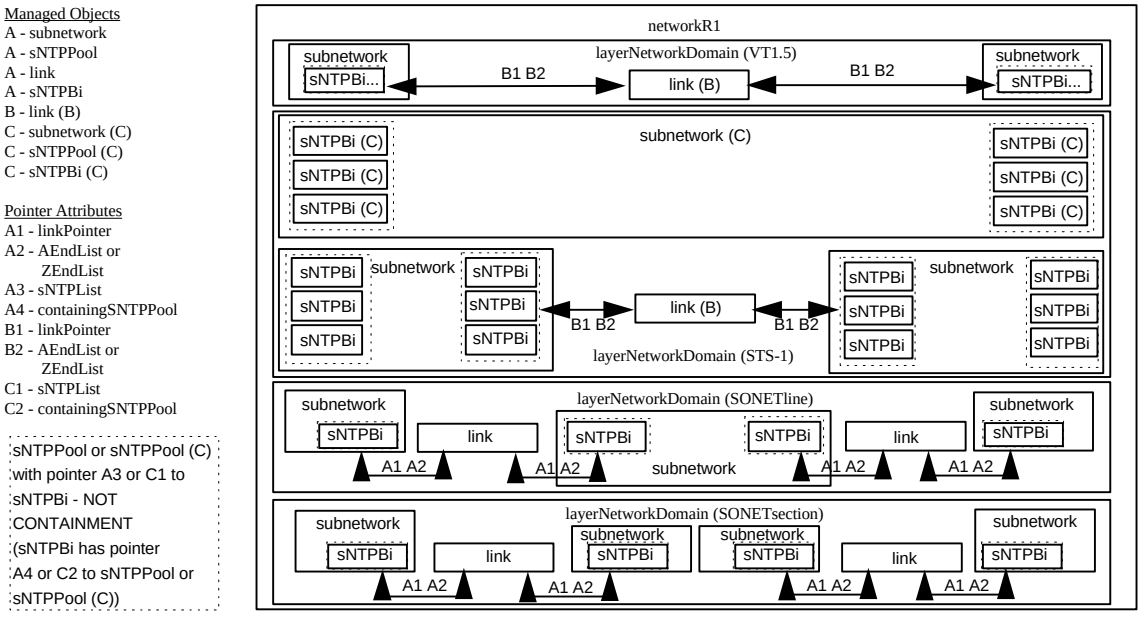


Figure 3-8 Object Instances for Subnetwork Topology Creation

Figure 3-8 shows the managed objects created during the subnetwork topology creation process. Also shown are the key relationships between managed objects, in the form of name bindings and pointer attributes. Name bindings are represented by solid-line boxes contained within solid-line boxes (e.g., layerNetworkDomain is contained within networkR1). Pointer relationships are shown by arrows and by dashed boxes. Finally, the “A”, “B”, and “C” references refer to “A”, “B”, and “C” in the previous figure. These references represent a sequence of object creations from “A” objects to “B” objects to “C” objects (and associated relationships). Note: “sNTPBi...” means many sNTPBi objects.

3.1.2 Network Connection Configuration

This function involves the set-up and tear-down of connections to support customer services. This scenario illustrates the processes required to set-up connections. First, server layer trails and client layer link connections are created. Then, subnetwork connections are created.

3.1.2.1 Layer Network Connection Configuration

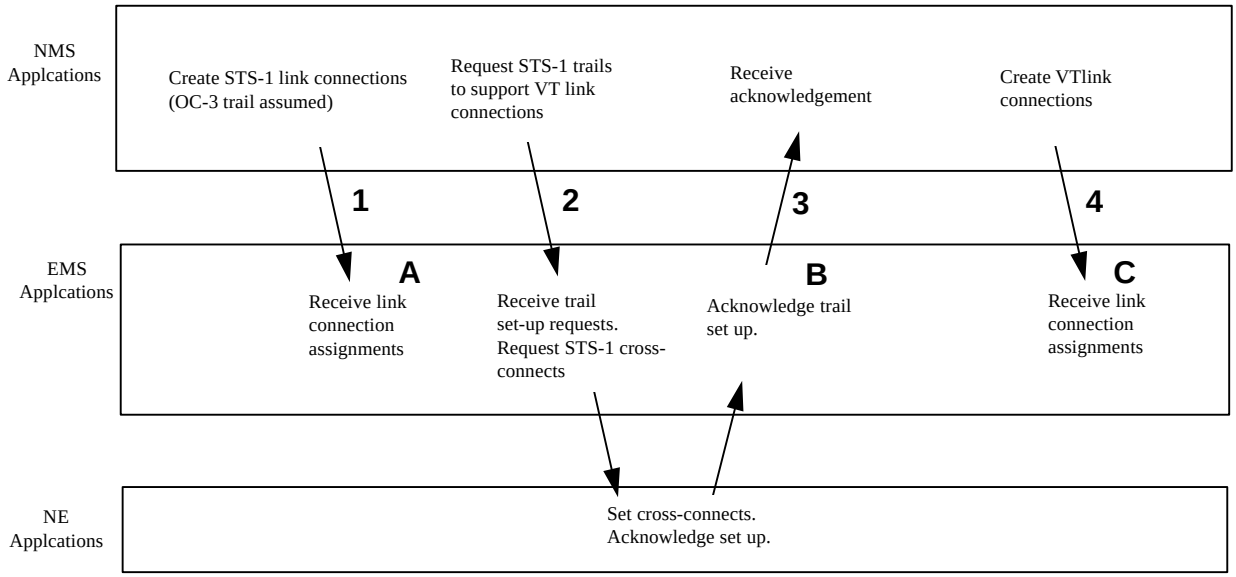


Figure 3-9 Processes and Message Flows: Create Link Connections, Supporting Trails in Each Layer

Figure 3-9 shows the applications processes and message flows used to configure layer network connections. The process starts with a request for STS-1 link connections and ends with the creation of VT link connections. The following message flows illustrate exchanges of information between the NMS and the EMS:

- **Message Flow 1** - This flow represents requests for STS-1 link connections. The NMS sends the EMS an M-ACTION on a link object instance requesting a setupLinkConnection. The EMS will respond to this request (not shown) with the identification of the created linkConnections or a failure notification.
- **Message Flow 2** - This flow represents requests for STS-1 trails to support VT link connections. The NMS sends the EMS an M-ACTION on a layer network domain object instance requesting a setupTrail.
- **Message Flow 3** - The EMS will respond to the trail request with the identification of the created trails or a failure notification.
- **Message Flow 4** - This flow represents requests for VT link connections. The NMS sends the EMS an M-ACTION on a link object instance requesting a setupLinkConnection. The EMS will respond to this request (not shown) with the identification of the created linkConnections or a failure notification.

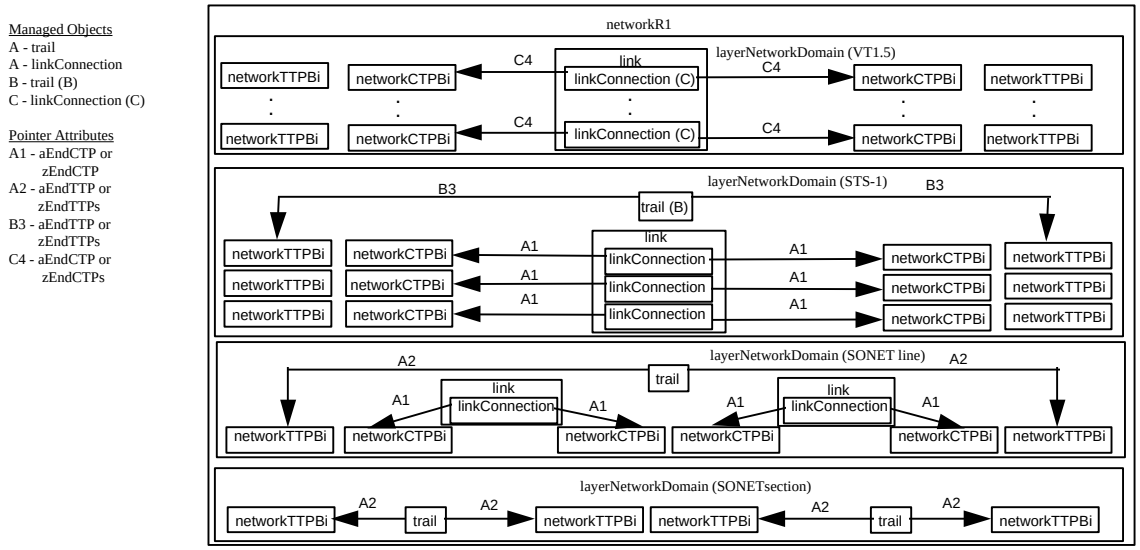


Figure 3-10 Object Instances for Layer Network Connection Configuration

Figure 3-10 shows the managed objects created during the layer network connection configuration process. Also shown are the key relationships between managed objects, in the form of name bindings and pointer attributes. Name bindings are represented by solid-line boxes contained within solid-line boxes (e.g., layerNetworkDomain is contained within networkR1). Pointer relationships are shown by arrows and by dashed boxes. Finally, the “A”, “B”, and “C” references refer to “A”, “B”, and “C” in the previous figure. These references represent a sequence of object creations from “A” objects to “B” objects to “C” objects (and associated relationships). **Note: This figure shows the network view only case. If the NE view was also supported, the network TP object instances would not be instantiated. Instead, the sdhNE and NE view TP object instances would be instantiated as shown in Figure 3-6. The link connections and trails would point to these object instances instead of the network TPs, using Figure 3-5 as a guideline.**

3.1.2.2 Subnetwork Connection Configuration

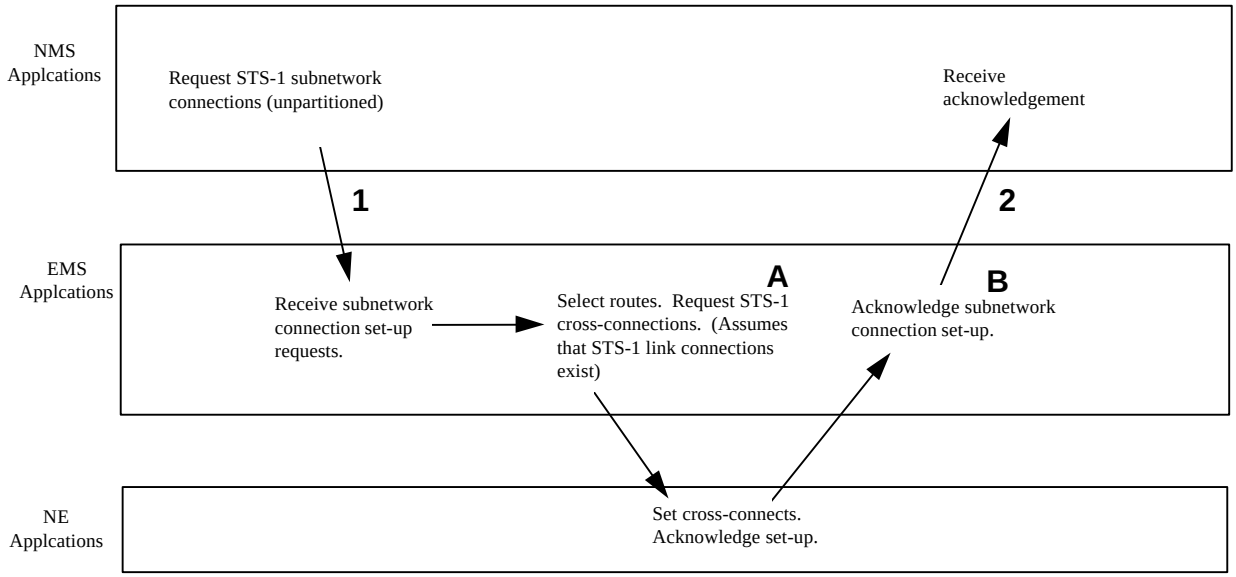


Figure 3-11 Processes and Message Flows: Set-up Subnetwork Connections

Figure 3-11 shows the applications processes and message flows used to configure subnetwork connections. The process starts with a request for STS-1 subnetwork connections (unpartitioned) and ends with acknowledgement of the creation of subnetwork connections. The following message flows illustrate exchanges of information between the NMS and the EMS:

- **Message Flow 1** - This flow represents requests for STS-1 subnetwork connections. The NMS sends the EMS an M-ACTION on a subnetwork object instance requesting a setupSNC.
- **Message Flow 2** - The EMS will respond to the set-up subnetwork connection request with the identification of the created subnetwork connections or a failure notification.

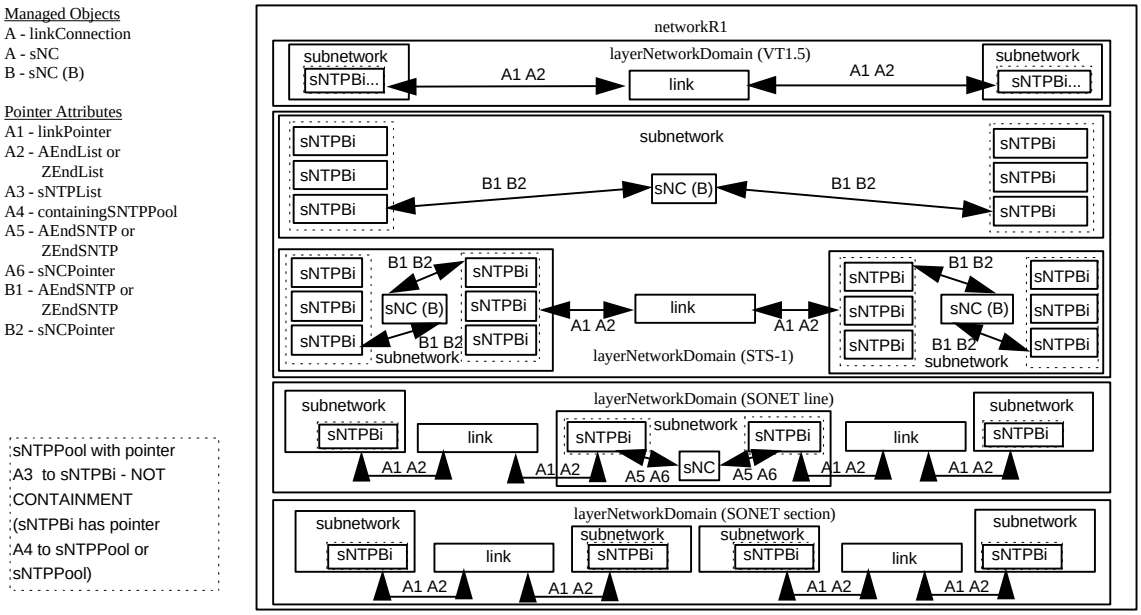


Figure 3-12 Object Instances for Subnetwork Connection Configuration

Figure 3-12 shows the managed objects created during the subnetwork connection configuration process. Also shown are the key relationships between managed objects, in the form of name bindings and pointer attributes. Name bindings are represented by solid-line boxes contained within solid-line boxes (e.g., layerNetworkDomain is contained within networkR1). Pointer relationships are shown by arrows and by dashed boxes. Finally, the “A” and “B” references refer to “A” and “B” in the previous figure. These references represent a sequence of object creations from “A” objects to “B” objects (and associated relationships). Note: Link connections, though necessary, are not shown in this figure.

3.2 Connection Configuration with Status Reporting Scenarios

This section gives an information model solution for providing Connection Configuration with Status Reporting in support of the following requirements:

R1 - The NMS requires progress information during the setup process.

R2 - On Failure:

- The EMS will report completed portions (e.g., component SNCs/link connections).
- The NMS will have the option to:

- a) completely undo. EMS will report portions it was not able to undo. As an option, the NML may specify the "undo" at setup time.

- b) keep partial setup

R3 - The NMS will have the capability to cancel while in progress. In response to a cancel request, the EMS will completely undo. It will also report portions "unable to undo".

This section gives scenarios and functional descriptions of managed objects classes for realizing the application. The approach uses a new object class modelled after the “Current Alarm Summary Control” object class [Q.821] would emit periodic reports via stimulation by a “Scanner” object class. The scanner object class was found to contain many conditional packages not needed for this application so the “Management Operations Schedule” object has been selected in lieu of the scanner object.

3.2.1 Connection Configuration with Status Reporting

Connection Configuration with Status Reporting allows an agent (EMS) to report on the status of in-progress subnetwork connections that have been requested by a manager (NMS). The functional components of the Connection Configuration service are illustrated in Figure 3-13. The agent’s Connection Configuration function receives connection requests, confirms connection completions, and provides data for status reporting. Status reporting for a given subnetwork connection object instance begins when the setup/release request is received and ceases when the agent completes all steps in the operation (e.g., sets up NE cross connections). A Connection Configuration Status Control function contains the criteria for reporting and generates the status reports.

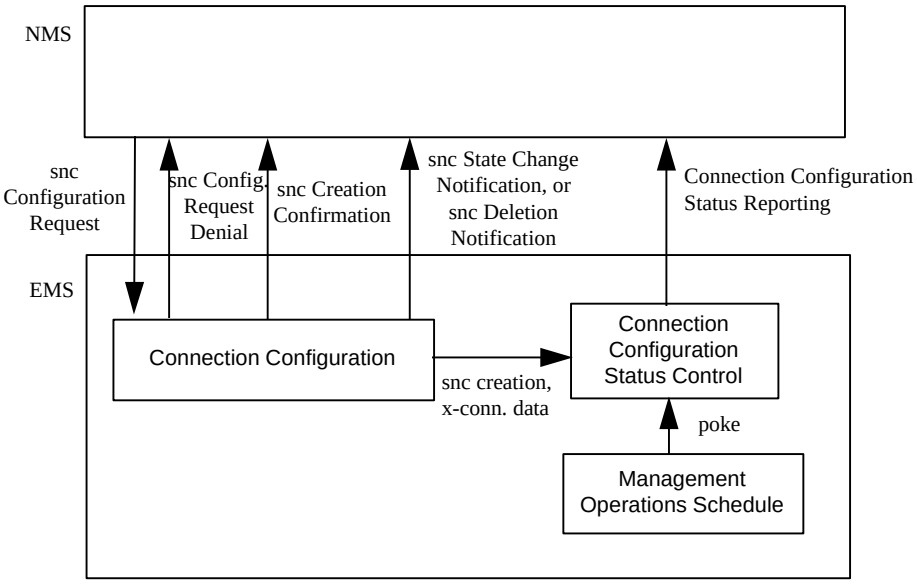


Figure 3-13 Connection Configuration Status Reporting Scenario

A report contains the identity of each subnetwork connection under setup, an indicator of whether the operation is setup or release, and a percentage measure of the progress achieved in completing the operation. The percentage indicates the number of NE cross-connects (completed or released) out of the total needed the complete the operation. Status reporting is done periodically at close intervals (on the order of seconds) as directed (via a “poke”) by a Management Operations Schedule function.

An example of the logic flows in a scenario for Connection Configuration with status reporting is shown in Figure 3-14. This illustrates the NMS/EMS interactions in Figure 3-13 in somewhat more detail and indicates the logical context for each request/response and notification.

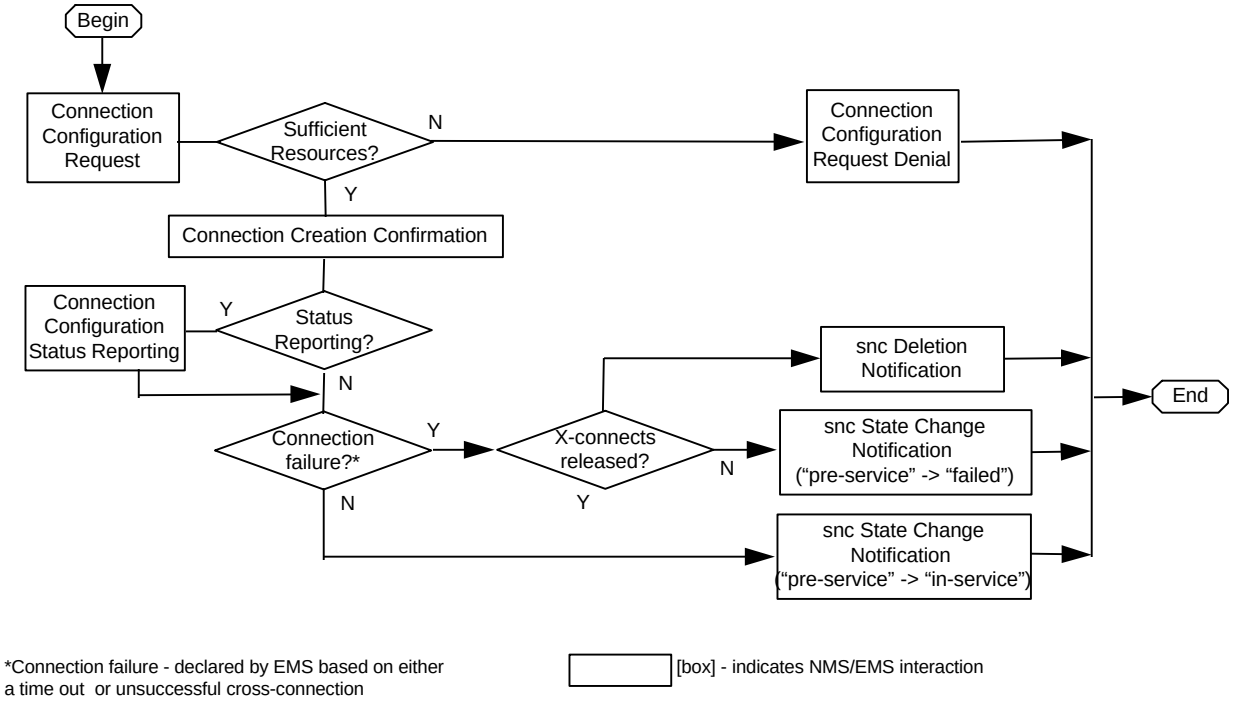


Figure 3-14 EMS Application Logic Example

The names of states for an snc are in quotes to indicate that work is needed to define these states in detail.

3.2.2 Functional Definitions of Object Classes

3.2.2.1 Connection Configuration Status Control Object Class

This object class is a somewhat modified form of the currentAlarmSummaryControl object class [Q.821]. The Connection Configuration Status Control object class is a class of support objects that enables the generation of reports on the status of subnetwork connections that have been requested by a manager and are in the process of being established or released through NE cross connections. An object is included in this summary report if:

- the object is a subnetwork connection object instance, and
- the manager requests reporting

An object will be removed from the list when it is setup, released, or failed.

The semantics of associated attributes are as follows:

- Connection Configuration Status Control Id - distinguished value can be used as a Relative Distinguished Name (RDN) for instance of object class.

- Object List - identifies a list of objects to be included in the connection configuration summary report.

The semantics of the associated notification is as follows:

- Connection Configuration Summary Report - identifies subnetwork connection object instances currently being setup or released and gives the current value of the percentage of associated cross-connects that have been successfully established or released.

The semantics of an associated conditional package is as follows:

- Empty List Suppression - suppresses notifications when the object list is empty.

3.2.2.2 Management Operations Schedule [Q.821]

The Management Operations Schedule object class is a class of support objects that provide the ability to schedule a management service to occur periodically. The period is specified by an Interval, with the first occurrence of the service (coinciding with the start of the first interval) specified as the Begin Time. The end of the time span during which the service can occur is defined by the End Time.
The objects that will supply the service are defined by the Affected Object Class and Affected Object Instances (e.g. the Current Alarm Summary Control object when providing the Current Alarm Summary Reporting Service). The Destination Address specifies the destination of the service. The Administrative State is used to allow/inhibit the operation of the schedule. The Operational State describes whether the object is capable of performing its function(s).
This object class is a subclass of the Top object class.
The semantics of associated attributes are as follows:

a) Administrative State

   The semantics of the Administrative State attribute type is described in X.731.

b) Affected Object Class

   The Affected Object Class attribute type identifies the object class affected by a scheduled management operation.

c) Affected Object Instances

   The Affected Object Instances attribute type identifies the object instances on which a scheduled management operation will be performed.

d) Begin Time

The Begin Time attribute type indicates the starting time for a management function.

e) Destination Address

The Destination Address attribute type identifies the destination to which selected event reports will be sent. The Destination Address may be an application entity title or address group. If no Destination Address is specified in the request, the address of the invoker is assumed.

f) End Time

The End Time attribute type indicates the termination time of a management function.

g) Interval

The Interval attribute type indicates the time between occurrences of a given activity described by an instance of the Management Operations Schedule object class. The interval can be specified in seconds, minutes, hours, or days.

h) Operational State

The semantics of the Operational State attribute type is described in X.731.

I) Schedule Id

The Schedule Id is an attribute type whose distinguished value can be used as an RDN when naming an instance of the Management Operations Schedule object class.

3.3 Fault Management Scenarios

This section describes general characteristics of SONET Fault Management scenarios that originate with the detection of a failure(s) by network elements. Examples of root cause for the failures in these scenarios include fiber cuts and equipment failures. Other scenarios for fault management such as response to a customer reported fault may be handled in future versions of the model.

The GR-2869 fault management scenario for “Network Detected Trouble” [GR-2869, Section 6.7.2] describes a set of basic functions and flows between functions. Figure 3-13 shows a portion of the functional flow for this scenario focussing on the subset of EMS functions applicable to a network-detected fault scenario such as a fiber cut. Functions in the Business Management Layer (BML) are omitted for simplicity.

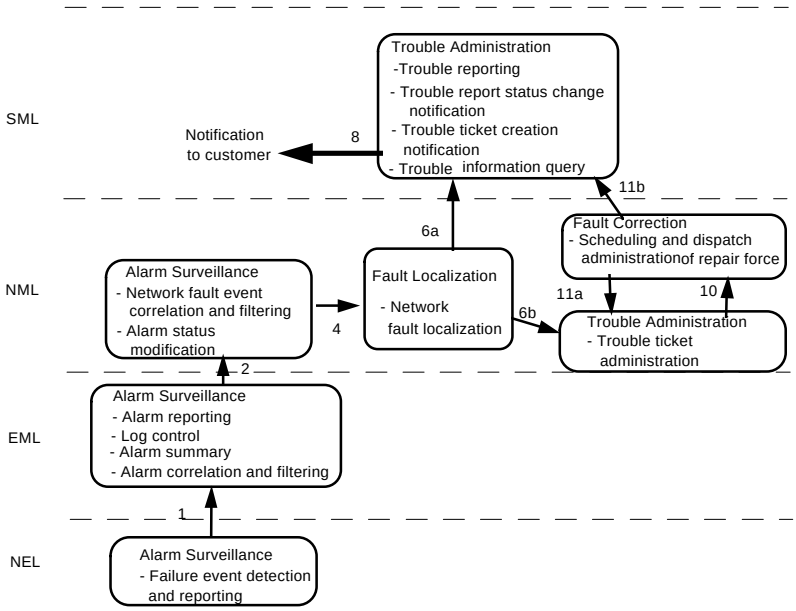


Figure 3-15 Network Detected Trouble scenario from GR-2869

Failure events detected in the Network Element Layer (NEL) are passed to Element Management Layer (EML) applications functions belonging to the Alarm Surveillance group. EML surveillance functions for SONET have been described in SR-2671 (root cause alarm analysis) and SR-2672 (alarm filtering). Many of the concepts used for our fault management model are derived from SR-2671.

3.3.1 Assumptions

1. The EMS will likely have knowledge of detailed operating characteristics and behaviors of the subnetwork/NEs that will not be visible to an NMS. This will enable the EMS to provide unique management capabilities related to:

- algorithms for root cause failure determination
- protection and reconfiguration mechanisms
- affected services
- states/status of transmission resources

Therefore, even though the NMS has visibility on the subnetwork connections/trails within an EMS's domain, the EMS may contain additional, useful knowledge.

2. The capability of the EMS to perform root cause analysis will help to reduce traffic on the DCN between the EMS and NMS.

3. The EMS will not have visibility on higher layer connections within a subnetwork connection unless the higher layer connections terminate within the EMS's domain.

4. The NMS can query the EMS for the list of current affected services and current alarms in the subnetwork.

5. There may be more than one fault event detected due to a single failure mechanism.

The EML network detected fault management application involves analysis functions, supporting data and supporting data acquisition, and results reporting functions. The following sections address EML application functions and interface data.

3.3.2 EMS Application Functions

A generic scenario for the network detected faults EMS application involves six functional steps:

1. Receive, terminate, log, and understand autonomous messages received from NEs within the subnetwork being managed.

2. Analyze the information received using supporting data (equipment hierarchy, topology, and transmission resource assignment) and knowledge of the current status for the subnetwork to filter alarms and report the filtered alarm information.

3. Analyze the information received using supporting data (equipment hierarchy, topology, and transmission resource assignment) and knowledge of the current status for the subnetwork to recognize and identify the root cause of the received message flow.

4. Analyze the information received using the supporting data and knowledge of the current status of the subnetwork to characterize the severity of the root cause problem (e.g., Service-affecting (SA) vs. Non-service-affecting (NSA), Critical/Major/Minor).

5. Analyze the information received using the supporting data and knowledge of the current status of the subnetwork to determine the impact of the root cause problem (i.e., identify affected transmission resources).

6. Report the root cause information along with the associated severity and services impact.

In summary alarms will be received from the NEs, logged at the EMS, and fed into the alarm analysis routine. The analysis routine will create records in the RCAA log and issue RCAA notifications to be sent to an Event Forwarding Discriminator (EFD). The EFD will filter the notifications and send the resulting Event Reports to the NMS. Figure 3-16, which is a variant of a picture from SR 2671, shows the process.

It is possible that the alarm analysis routine cannot find a root cause. In this case the alarm analysis routine will place the information in the uncorrelated NEL alarms in RCAA alarm records.

The EMS may be configured to pass high priority alarms to the NMS (via a specific EFD). This will allow the NMS to be immediately informed of high priority alarms even though the Alarm Analysis Routine takes a long time.

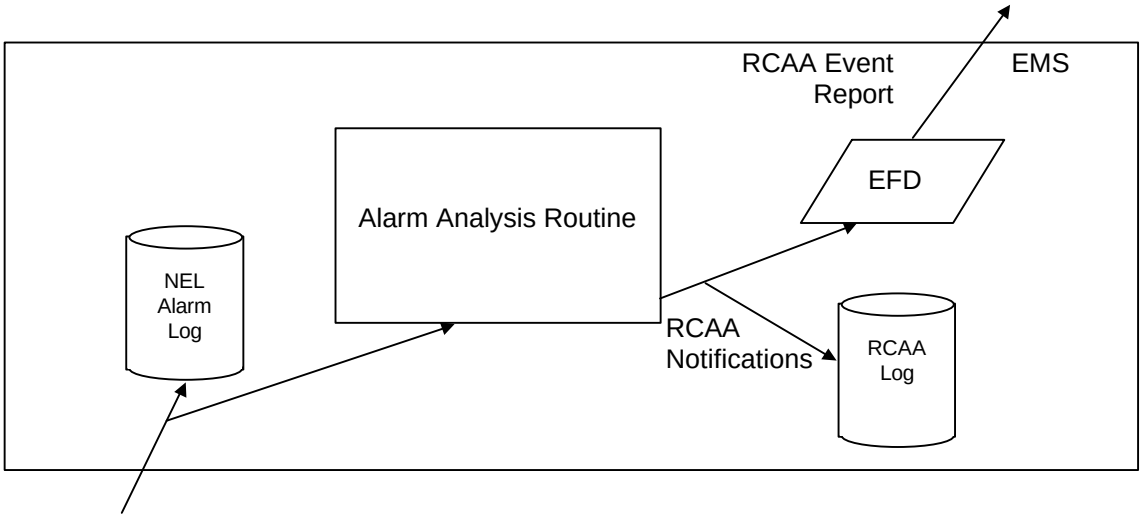


Figure 3-16 Alarm and RCAA Message Flow

The following explain the objects and notifications used to model Fault Management:

Log object from X.721 and X.735 is used to store incoming event reports and local system notifications. The NEL Alarm Log and the RCAA Log will be instances of the Log object class.

Records in the NEL Alarm Log will be instances of the alarmRecord class defined in X.721 and will contain information about NEL Alarm Event Reports from the NEs.

A RCAA Record is a subclass of the alarmRecord object class. It will contain information about the root cause of alarms.

An Alarm Analysis Routine object is used as the source of RCAA notifications.

The notification is sent to an EFD. EFDs are defined in X.721 and X.734 and are used to define the conditions that shall be satisfied by a potential event report before that event report is forwarded to a particular destination.

If the notification satisfies the conditions of the EFD an RCAA Event Report is sent from the EMS to the NMS to report the results of the Alarm Analysis

Routine.

3.3.3 Requirements for Fault Management for NEL Detected Faults

Requirements were derived from the scenario defined in the previous section. The following requirements are supported by the model presented in this document.

1. The EMS shall log alarms and events from NEs including TCAs.

2. The EMS shall filter via Intelligent Alarm Filtering and/or Root Cause Alarm Analysis alarms/events/TCAs from the NE and pass the results to the NMS within a discrete period of time.

3. Results passed to NMS (as defined in 2) shall include a reference to the set of NEL alarm events that have been resolved within that message.

4. The EMS shall maintain a log of results passed to the NMS (as defined in 2), retrievable by the NMS.

5. Subsequent analysis messages can follow those messages defined in 2. These messages shall also be passed to the NMS and tagged to the original.

6. The NMS shall be able to filter RCAA messages from the EMS related to severity, time, type and location.

7. The NMS shall be able to query the EMS for a list of the EMS's outstanding alarms.

8. The EMS shall identify the affected transmission resources.

9. "Clear" messages shall be passed to the NMS as soon as alarms are "cleared" at the lower layers.

10. The NMS needs access to the CMIP NE view of the equipment.

The following requirements are not supported with the current model, but will be supported with future versions of the model.

1. The NMS shall be able to optionally block "subsequent analysis messages" (as described in above) from the EMS. Blocked messages shall be logged.
2. The NMS shall be capable to ask the EMS to "resync" fully or partially its NE alarm info.
3. Acknowledgements made by the NMS may be passed to the EMS.
4. The discrete time period defined above shall be configurable.

3.3.4 Information Included in an RCAA Notification

This section discusses the information included in a RCAA notification that is sent from the EMS to the NMS.

The following requirements were derived from the requirements and from the Fault Management scenarios:

1. The RCAA notification should contain at least all the information contained in the associated NEL notification if the root cause was a NEL alarm.
2. The RCAA notification should contain information equivalent to the information contained in the associated NEL notification if the root cause was not a NEL alarm. For example if the root cause was a fiber cut the alarmed object will be a link which is not in the NE view.
3. The RCAA notification should contain sufficient information so that RCAA alarms can be filtered on severity, time, type and location.
4. The RCAA notification should contain information so that the high priority NEL alarms that have been resolved by this alarm message can be retrieved.
5. The RCAA notification should contain a reference to any earlier RCAA message(s) that are related to this message.
6. The RCAA notification should contain an indication whether this message represents a setting, update or clearing of an alarm.
7. The RCAA notification should indicate whether the alarm is Service Affecting (SA) or Not Service Affecting (NSA).
8. The RCAA notification should indicate whether this alarm represents one or more than one NEL alarms.
9. The RCAA notification should indicate the transmission resources affected by this RCAA alarm. “Transmission resources” means link, link connection, subnetwork connection or trail.

Given these requirements Table 3-1 lists the parameters that are included in the RCAA notification. The first column identifies the parameter. The second column indicates the standard that defines the parameter. If the parameter has not been defined in a standard this is indicated with a “n/a”. The third column indicates whether the parameter is optional or required in the standard from which it is drawn. The fourth column indicates whether this parameter should be optional for RCAA notifications. The last column indicates which of the requirements above motivated the inclusion of this parameter.

Parameter	Standard	Optional in standard	Optional in RCAA Notification	Requirements
Managed Object Class (Alarm Analysis Routine Object Class)	X.710	Mandatory	Mandatory	1,2
Managed Object Instance (Alarm	X.710	Mandatory	Mandatory	1,2

Analysis Routine Object instance)				
Event Type	X.710	Mandatory	Mandatory	1,2,3
Event Time	X.710	Optional	Mandatory	1,2,3
probableCause	X.733	Mandatory	Mandatory	1,2
SpecificProblems	X.733	Optional	Optional	1,2
perceivedSeverity	X.733	Mandatory	Mandatory	1,2,3,6
backedUpStatus	X.733	Optional	Optional	1,2
backUpObject	X.733	Optional	Optional	1,2
trendIndication	X.733	Optional	Optional	1,2
thresholdInfo	X.733	Optional	Optional	1,2
notificationIdentifier	X.733	Optional	Mandatory	1,2,4
correlatedNotification	X.733	Optional	Optional	1,2,5
stateChangeDefinition	X.733	Optional	Optional	1,2
monitoredAttributes	X.733	Optional	Optional	1,2
proposedRepairAction	X.733	Optional	Optional	1,2
additionalText	X.733	Optional	Optional	1,2
additionalInformation	X.733	Optional	Optional	1,2
alarmedObjectClass	n/a	n/a	Mandatory	1,2
alarmedObjectInstance	n/a	n/a	Mandatory	1,2
location	n/a	n/a	Mandatory	3
serviceAffecting	n/a	n/a	Mandatory	7
numberOfNELAlarms	n/a	n/a	Mandatory	8
affectedTransmissionResources	n/a	n/a	Mandatory	9
highestPriorityNELalarmRecords	n/a	n/a	Mandatory	4

Table 3-1 Parameters Included in RCAA Notification

Any parameter which was mandatory in X.710 or X.733 is mandatory in an RCAA notification. Any parameter that was optional in X.710 or X.733 is optional here unless the parameter allowed us to meet one of the requirements noted above.

The sonetAlarmAnalysisRoutine object emits the RCAA notifications. We must also send the object class and the object instance of the object which is actually in alarm. The alarmedObjectClass and alarmedObjectInstance provide this information.

The attribute: affectedTransmissionResources should provide sufficient object pointers so that the NMS can determine the set of the transmission resources that are affected by the single failure. Here transmission resources refers to links, link connections, trails and subnetwork connections. A single failure can affect many transmission resources. Consider for example that a OC-48 fiber which has been cut will affect over 1344 DS1s. To limit the information which must be sent we will send only the “bottom of the transmission hierarchy”. This will be enough information so that the NMS can determine the other transmission

resources either because it is aware of the topology at the EMS or through querying the EMS. We must be sure that we have sufficient information in the EMS so that the NMS can determine all affected resources.

To show that we can derive all dependent transmission resources given the transmission resource at the bottom of the transmission hierarchy we must consider both a hierarchy based on partitioning and layering. Considering partitioning first, given a failure in a subnetwork connection we can scope and filter to find all subnetwork connections which have the failed subnetwork connection as a component. A similar strategy will work if the key transmission source is a link connection. If the key transmission resource is a link one can follow a similar strategy for all link connections contained in the link. For trails the process is more complex. One can first find the TTPs which terminate the trail. The TTPs will point to CTPs which are supported. One can then scope the link connections which are terminated by these CTPs for the affected link connections. One might also want to find the associated subnetwork connections which can be done by scoping the SNTPs which point to the CTPs. The SNTPs will point to the subnetwork connections.

Deriving dependent transmission resources across layering must depend on the connection between TTPs and CTPs again. Suppose that trail has failed. The trail points to TTPs. The TTPs will point to CTPs which support link connections. The link connections will have failed because the supporting TTPs have failed. The CTPs are pointed to by snTPs which can be found by scoping and filtering. The snTPs may point to subnetwork connections which have failed. By following the trail of CTPs, snTPs, subnetwork connections, and TTPs through the layering and using those entities to find the associated transmission entities one can derive the transmission entities associated by key transmission entity.

The use of notificationIdentifier and correlatedNotification allow the correlation between RCAA notifications.

3.3.5 Information Included in the RCAA Record

This section discusses the information that is included in a RCAA Alarm Record. The following requirements were derived from the Fault Management SOA and from the Fault Management scenarios:

1. The RCAA Alarm Record should contain at least all the information contained in the associated RCAA notification.
2. The RCAA Alarm Record should indicate whether the root cause represented by this alarm is still active.
3. The RCAA Alarm Record should contain information so that all NEL alarms that have been resolved by this alarm message can be retrieved.

One can accomplish the requirements 1 and 2 above by including all the information defined for the RCAA notification. In order to determine which alarms are currently active the NMS will have to correlate the rcaaRecord which clear past alarms with the rcaaRecord which annunciated the clears. This can be done by looking at the perceived severity and the entity instance. One can also place a reference to the rcaaRecord which represents the setting of the alarm in the correlatedNotifications attribute of the rcaaRecord which represents the clearing of the alarm.

To meet the last requirement we must add an additional attribute which lists all correlated NEL alarms which have not already been included in the highestPriorityNELalarmRecords. This new attribute is called lowerPriorityNELalarm records.

The table below shows the information which will be kept in the rcaaRecord.

Parameter	Standard	Optional in standard	Optional in RCAA Notification
Managed Object Class	X.710	Mandatory	Mandatory
Managed Object Instance	X.710	Mandatory	Mandatory
Event Type	X.710	Mandatory	Mandatory
Event Time	X.710	Optional	Mandatory
probableCause	X.733	Mandatory	Mandatory
SpecificProblems	X.733	Optional	Optional
perceivedSeverity	X.733	Mandatory	Mandatory
backedUpStatus	X.733	Optional	Optional
backUpObject	X.733	Optional	Optional
trendIndication	X.733	Optional	Optional
thresholdInfo	X.733	Optional	Optional
notificationIdentifier	X.733	Optional	Mandatory
correlatedNotification	X.733	Optional	Optional
stateChangeDefinition	X.733	Optional	Optional
monitoredAttributes	X.733	Optional	Optional
proposedRepairAction	X.733	Optional	Optional
additionalText	X.733	Optional	Optional
additionalInformation	X.733	Optional	Optional
alarmedObjectClass	n/a	n/a	Mandatory
alarmedObjectInstance	n/a	n/a	Mandatory
location	n/a	n/a	Mandatory
serviceAffecting	n/a	n/a	Mandatory
numberOfNELAlarms	n/a	n/a	Mandatory
affectedTransmissionResources	n/a	n/a	Mandatory
highestPriorityNELalarmRecords	n/a	n/a	Mandatory
lowerPriorityNELalarm Records	n/a	n/a	Mandatory

Table 3-2 Information Kept in rcaaRecord

4. Information Model

This section provides a model for the Network Connection Management and Fault Management applications. This model addresses a subset of the functions covered in the statements of applications for Connection Management and Fault Management at the NMS/EMS interface defined in Sections 2.2 and 2.4.

4.1 Inheritance Hierarchy

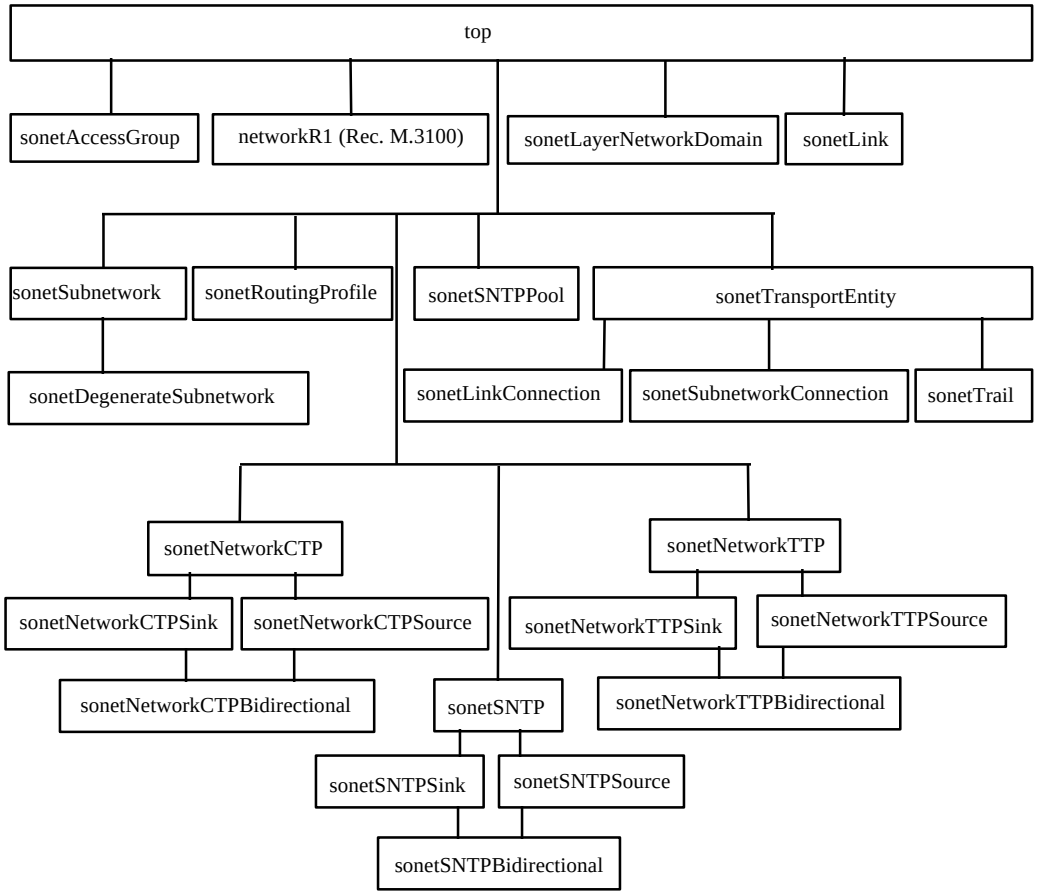


Figure 4-1 Inheritance Hierarchy

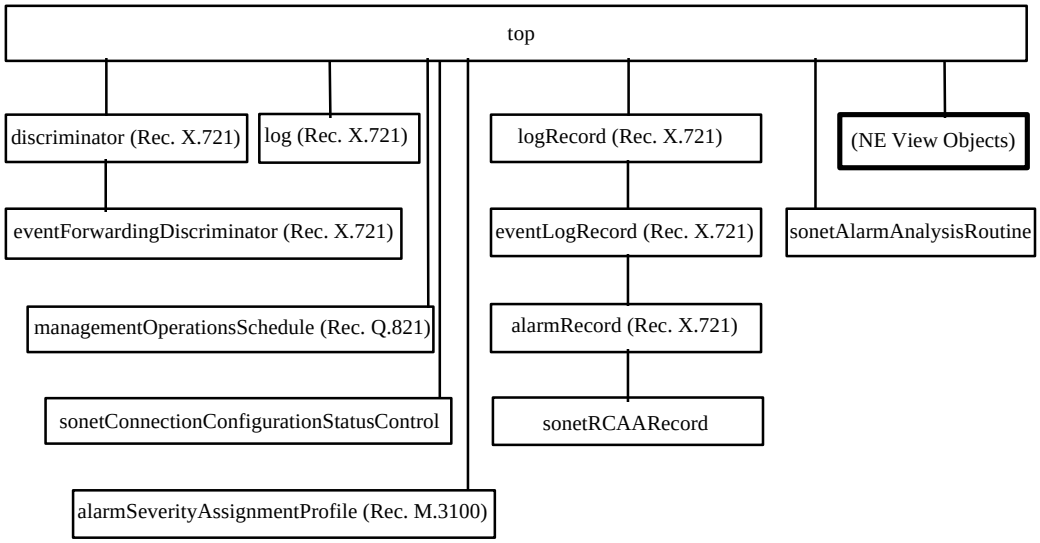


Figure 4-2 Inheritance Hierarchy (Cont'd)

4.2 Naming Hierarchy

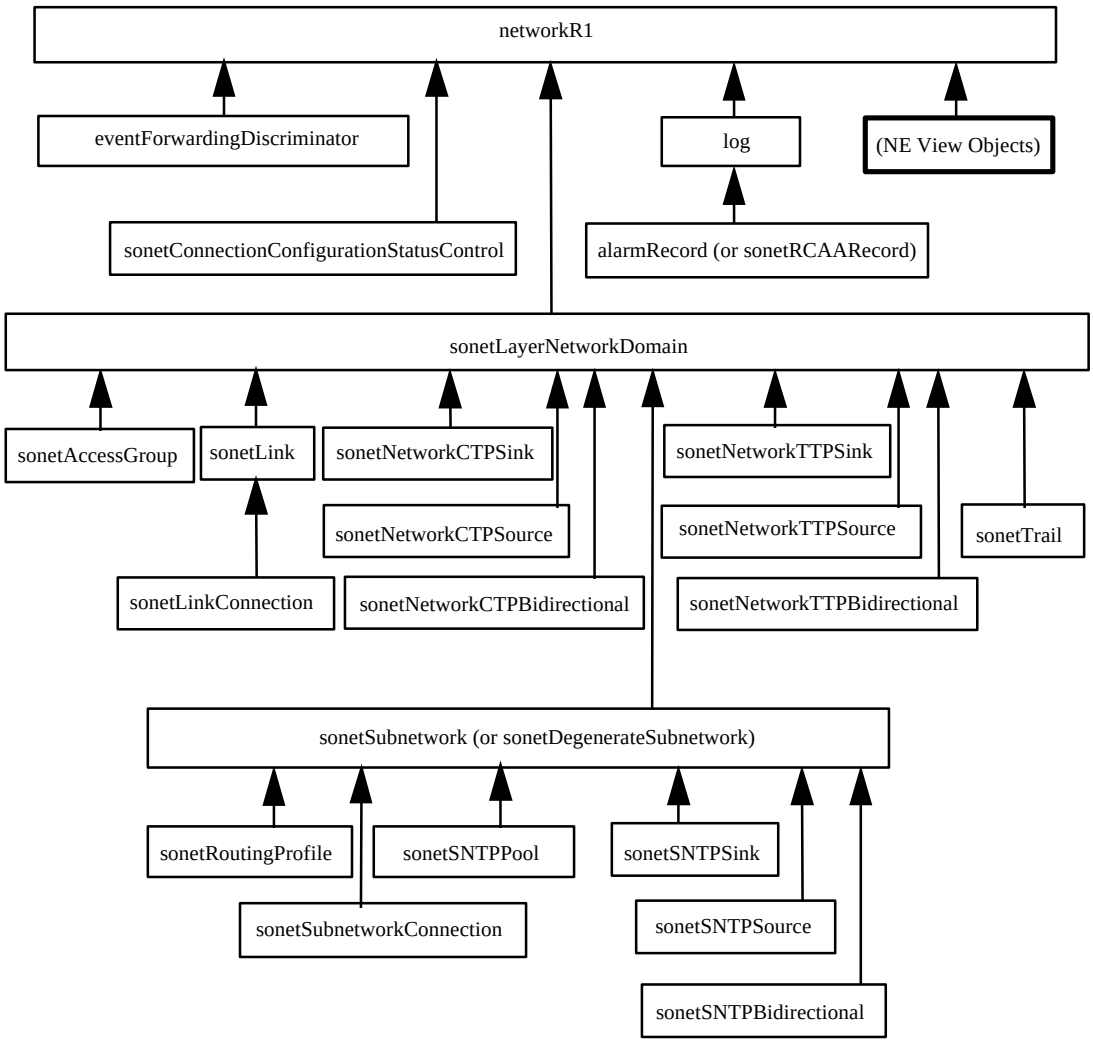


Figure 4-3 Naming Hierarchy

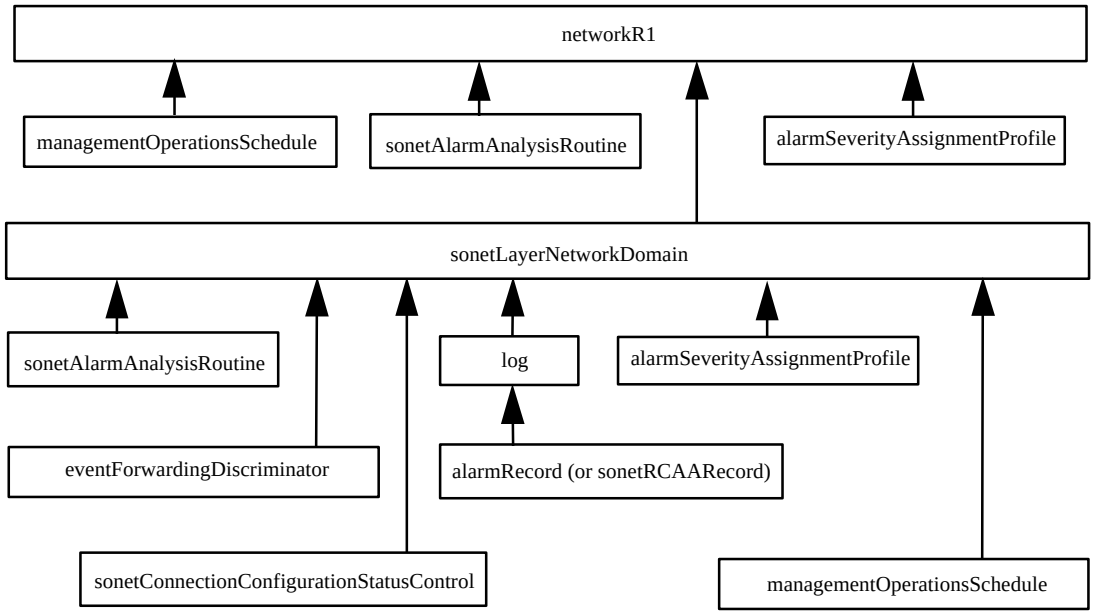


Figure 4-3 Naming Hierarchy (Cont'd)

The bold box in the Naming Hierarchy above indicates objects that are defined in the NE view (see Table 4-1). Not shown is the allowable, for existing implementations only, use of either managedElement or managedElementComplex as the top of the NE View naming hierarchy (rather than networkR1). The Network View would still use networkR1 as top.

The NE View objects may be included in an implementation where they are referenced from the defined Network View objects described in this document, and where both the NE View and the Network View are supported. Implementation of both the NE View and Network View together represents a specific design choice. Also, implementations that provide a "stand alone" network view (no references to NE view objects) may be defined using the objects described in this document. In this case the objects represented by the bold box would not be referenced by the Network View objects.

4.3 Entity-Relationship Diagram

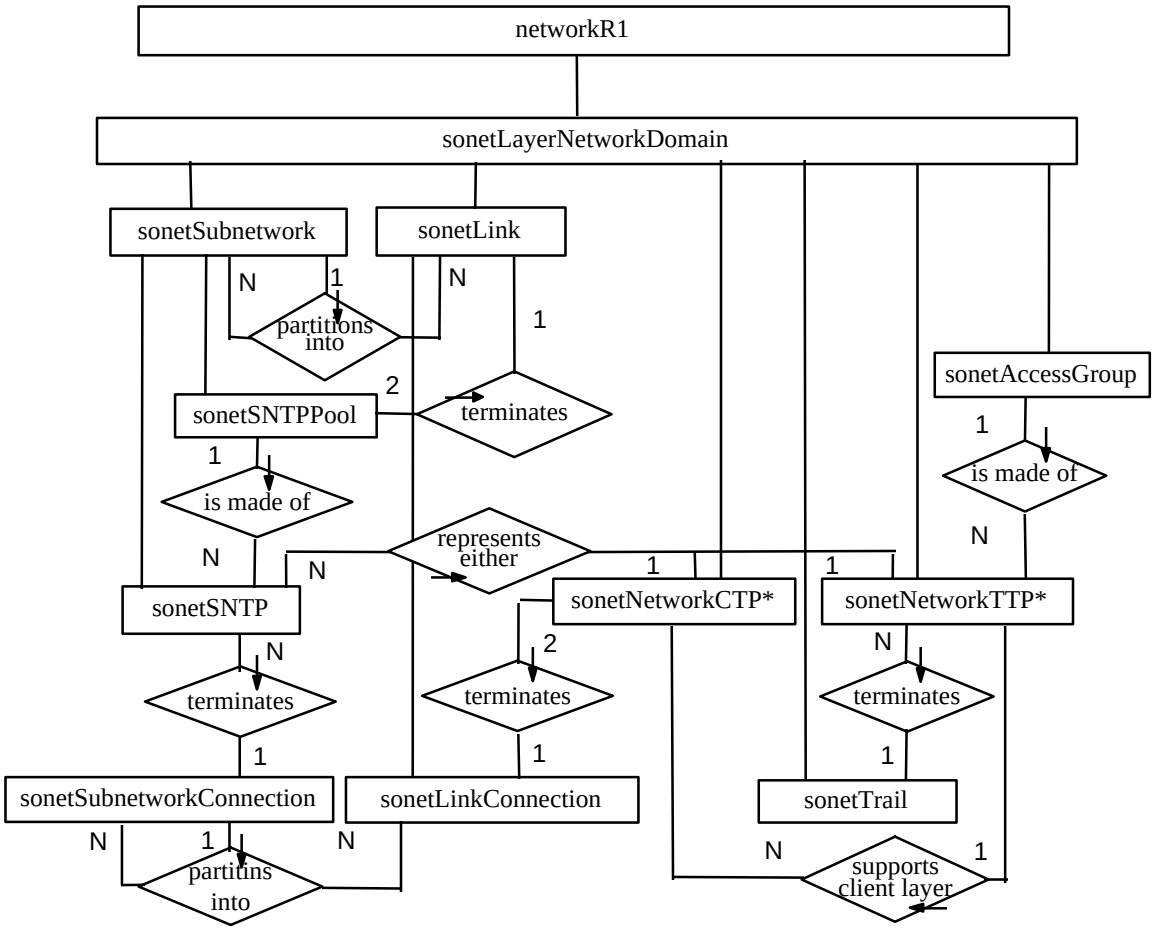


Figure 4-4 E-R Diagram

1. Generic object classes shown in diagram represent subclasses as well. For example, sONETSNTP represents sONETSNTPSink, sONETSNTPSource, and sONETSNTPBidirectional in that these subclasses may each participate in the relationships shown.
2. Vertical lines connecting object classes indicate name bindings.
3. Star (*) indicates that NE View object classes may optionally be used to participate in indicated relationships in place of network view object classes. For example, TTP from the NE View may be used in place of networkTTP.
4. Cardinalities indicate the range of object class instances that may participate in instances of a relationship.

4.4 NE View Object Classes

The following is a representative set of NE View objects which might be appropriate. Consult information models from ITU-T, Bellcore, and Committee T1 for further information. The exact list of objects to be supported is for further study.

Fragment	Object Class	Source
Network Element	managedElement	ITU-T Rec. M.3100
	sdhNE	ITU-T Rec. G.774
Hardware	equipment	ITU-T Rec. M.3100
	equipmentHolder	GR-836-IMD
	circuitPack	GR-836-IMD
	memory	GR-836-IMD
	equipmentR1	ITU-T Rec. M.3100
	equipmentHolder	ITU-T Rec. M.3100
	circuitPack	ITU-T Rec. M.3100
Generic Support	managementOperationsSchedule	ITU-T Rec. Q.821
	discriminator	ITU-T Rec. X.721
	eventForwardingDiscriminator	ITU-T Rec. X.721
	log	ITU-T Rec. X.721
	logRecord	ITU-T Rec. X.721
	eventLogRecord	ITU-T Rec. X.721
Configuration Mgt Support	objectCreationRecord	ITU-T Rec. X.721
	objectDeletionRecord	ITU-T Rec. X.721
	attributeValueChangeRecord	ITU-T Rec. X.721
	stateChangeRecord	ITU-T Rec. X.721
Basic Cross-Connection	fabricBCRr1	GR-836-IMD
	gtp	ITU-T Rec. M.3100
	crossConnection	ITU-T Rec. M.3100
	crossConnectionBCRr1	GR-836-IMD
	namedCrossConnectionBCRr1	GR-836-IMD
	mpCrossConnection	ITU-T Rec. M.3100
	mpCrossConnectionBCRr1	GR-836-IMD
	namedMpCrossConnectionBCRr1	GR-836-IMD
Termination Point	terminationPoint	ITU-T Rec. M.3100
	connectionTP Si/So/Bi [1]	ITU-T Rec. M.3100
	au3CTP SiR1/So/BiR1	ITU-T Rec. G.774
	ds1CTP Si/So/Bi	GR-836-IMD
	ds3CTP Si/So/Bi	GR-836-IMD
	msCTP Si/So/Bi	ITU-T Rec. G.774
	rsCTP Si/So/Bi	ITU-T Rec. G.774
	tu11CTP SiR1/So/BiR1	ITU-T Rec. G.774
	trailTP Si/So/Bi	ITU-T Rec. M.3100
	ds1LineTTP Si/So/Bi	GR-836-IMD
	ds1LineTTP SiBCRr1/SoBCRr1/BiBCRr1	GR-836-IMD
	ds1PathTTP Si/So/Bi	GR-836-IMD
	ds1PathTTP SiBCRr1/SoBCRr1/BiBCRr1	GR-836-IMD
	ds3LineTTP Si/So/Bi	GR-836-IMD
	ds3PathTTP Si/So/Bi	GR-836-IMD
	electricalSPITTP Si/So/Bi	ITU-T Rec. G.774

Fragment	Object Class	Source
	electricalSPITTP BiBCRr1[2]	GR-1042-IMD
	msTTP Si/So/Bi	ITU-T Rec. G.774
	msTTP BiBCRr1[3]	GR-1042-IMD
	msTTP SiBCRr1[4]	GR-1042-IMD
Termination Point (con't)	opticalSPITTP Si/So/Bi	ITU-T Rec. G.774
	opticalSPITTP BiBCRr1[5]	GR-1042-IMD
	rsTTP Si/So/Bi	ITU-T Rec. G.774
	vc11TTP SiR1/So/BiR1	ITU-T Rec. G.774
	vc11TTP SiBCRr1/SoBCRr1/BiBCRr1	GR-1042-IMD
	vc3TTP SiR1/So/BiR1	ITU-T Rec. G.774
	vc4TTP SiR1/So/BiR1	ITU-T Rec. G.774
	indirectAdapter Si/So/Bi	ITU-T Rec. G.774
	aug Si/So/Bi	ITU-T Rec. G.774
	tug2 Si/So/Bi	ITU-T Rec. G.774

Table 4-1 NE View Object Classes

1. The notation Si/So/Bi is used for termination point object classes. For exampe, connectionTP Si/So/Bi represents three separate types of object classes, namely, connectionTerminationPointSink, connectionTerminationPointSource, and connectionTerminationPointBidirectional. In addition, Bidirectional Tps are usually subclasses of both a Sink TP and a Source TP and often another Bidirectional TP. For example, msCTPBidirectional is a subclass of msCTPSink, msCTPSource and connectionTerminationPointBidirectional.

2. This object class is a subclass of the electricalSPITTPBidirectional [G.774] object class.

3. This object class is a subclass of the msTTPBidirectional [G.774] object class.

4. This object class is a subclass of the msTTPSink [G.774] object class.

5. This object class is a subclass of the opticalSPITTPBidirectional [G.774] object class.

4.5 Characteristic Information

SONET Layer	Characteristic Information	Source
Optical OC-192	opticalSTM64SPICI	SIF NLM*
Optical OC-48	opticalSTM16SPICI	ITU-T M.3100
Optical OC-12	opticalSTM4SPICI	ITU-T M.3100
Optical OC-3	opticalSTM1SPICI	ITU-T M.3100
Electrical STS-3	electricalSTM1SPICI	ITU-T M.3100
Electrical STS-1	electricalSTS1SPICI	SIF NLM*
Section STS-192	rsSTM64SPICI	SIF NLM*
Section STS-48	rsSTM16SPICI	ITU-T M.3100
Section STS-12	rsSTM4SPICI	ITU-T M.3100
Section STS-3	rsSTM1SPICI	ITU-T M.3100
Line STS-192	msSTM64SPICI	SIF NLM*
Line STS-48	msSTM16SPICI	ITU-T M.3100
Line STS-12	msSTM4SPICI	ITU-T M.3100
Line STS-3	msSTM1SPICI	ITU-T M.3100
Path STS-12c	au44cCI	SIF NLM*
Path STS-3c	au4VC4CI	ITU-T M.3100
Path STS-1	au3TU3VC3CI	ITU-T M.3100
Path VT1.5	tu11VC11CI	ITU-T M.3100
Line DS3	ds3LineCI	SIF NLM*
Line DS1	ds1LineCI	SIF NLM*
Path DS3	ds3PathCI	SIF NLM*
Path DS1	ds1PathCI	SIF NLM*

Table 4-2 Characteristic Information Values

* These values of characteristic information are not currently defined in any other documents. These values are scheduled for addition to GR-836-IMD and/or GR-1042-IMD by 4Q97. When these values are added to the GR(s), they will be removed from this document and referenced as appropriate.

4.6 States

States are supported for some object classes. The model only supports states where the need was recognized. The table below shows the states supported by each object class. See the object class definitions for the acceptable values of each state. (Note: In the object class definitions, “sonet” is added to each object class shown below.)

Classes	administrative State	operational State	usage State	tPConnection State	configuration State
AccessGroup	N	N	Y	N	N
AlarmAnalysis Routine	N	N	N	N	N
Connection Pending StatusControl	N	N	N	N	N
LayerNetwork Domain	N	N	N	N	N
Link	Y	N	Y	N	N
LinkConnection	N	Y	N	N	N
NetworkCTP	N	C	N	Y	N
NetworkTTP	Y	Y	N	Y	N
RCAARecord	N	N	N	N	N
RoutingProfile	N	N	N	N	N
SNTP	N	N	N	N	N
SNTPPool	N	N	Y	N	N
Subnetwork	N	N	N	N	N
Subnetwork Connection	Y	Y	N	N	Y
Trail	C	Y	N	N	Y

Table 4-3 States Supported(Y), Conditionally Supported (C), or Not Supported(N) by Object Class

4.7 Managed Objects

4.7.1 sonetAccessGroup

```
sonetAccessGroup  MANAGED OBJECT CLASS
    DERIVED FROM “Rec. X. 721 | ISO/IEC 10165-2”:top;
    CHARACTERIZED BY
        “Rec. M. 3100:1995”:attributeValueChangeNotificationPackage,
        “Rec. M. 3100:1995”:createDeleteNotificationPackage,
        “Rec. M. 3100:1995”:stateChangeNotificationPackage,
        “Rec. X.721 | ISO/IEC 10165-2:1992”:usageStatePackage,
        “Rec. M. 3100:1995”:userLabelPackage,
        sonetAccessGroupPackage PACKAGE
                BEHAVIOUR sonetAccessGroupBehaviour;
                ATTRIBUTES
                        sonetAccessGroupId
                                GET,
                        tTPList
                                GET-REPLACE       ADD-REMOVE;;;
REGISTERED AS {sIFNLMObjectClass 1};

sonetAccessGroupBehaviour BEHAVIOUR
    DEFINED AS
        “The sonetAccessGroup object class represents a group of co-located
networkTTPs (and subclasses) within a layer network domain.  The tTPList
attribute points to the set of networkTTPs or NE View TTPs (and subclasses of)
the sonetAccessGroup is grouping together.

        The usageState is used to describe the usage of the sonetAccessGroup.
When idle, all related TTPs are not cross-connected.  When active, some of the
related TTPs are not cross-connected.  When busy, none of the related TTPs
are not cross-connected.

        A change in the value of the usageState attribute shall cause a
stateChange notification.

        A change in the value of any of the following attributes shall cause an
attributeValueChange notification:

                tTPList
”;
```

4.7.2 sonetAlarmAnalysisRoutine

```
sonetAlarmAnalysisRoutine        MANAGED OBJECT CLASS
    DERIVED FROM “Rec. X. 721 | ISO/IEC 10165-2”:top;
    CHARACTERIZED BY
        sonetAlarmAnalysisRoutinePackage PACKAGE
                BEHAVIOUR sonetAlarmAnalysisBehaviour;
                ATTRIBUTES
                        “Rec. M.3100:1995”:alarmSeverityAssignmentProfilePointer
                                GET-REPLACE,
                        sonetAlarmAnalysisRoutineId
                                GET;
                NOTIFICATIONS
                        rcaaNotification;;;
REGISTERED AS {sIFNLMObjectClass 2};

sonetAlarmAnalysisRoutineBehaviour BEHAVIOUR
    DEFINED AS
        “The sonetAlarmAnalysisRoutine is an object which represents an
application which analyzes alarms. Once analysis is complete it issues the
rcaaNotification. The attribute alarmAnalysisRoutineId is used as an RDN.

    The alarmSeverityAssignmentProfilePointer attribute is used to flexibly
assign the severity of the alarms issued by the sonetAlarmAnalysisRoutine.”;
```

4.7.3 sonetConnectionPendingStatusControl

```
sonetConnectionPendingStatusControl  MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":operationalStatePackage,
        “Rec. M. 3100:1995”:stateChangeNotificationPackage,
        sonetConnectionPendingStatusControlPackage PACKAGE
                BEHAVIOUR

    sonetConnectionPendingStatusControlBehaviour;
                ATTRIBUTES
                        sonetConnectionPendingStatusControlId
                                GET,
                        "Rec. Q. 821":objectList
                                GET-REPLACE      ADD-REMOVE;
                NOTIFICATIONS
                        connectionConfigurationSummaryReport;;;
    CONDITIONAL PACKAGES
```

emptyListSuppressionPackage PRESENT IF "an instance supports it";
REGISTERED AS { sIFNLMObjectClass 3};

sonetConnectionPendingStatusControlBehaviour BEHAVIOUR
DEFINED AS
"This managed object class is a class of support objects that enables the generation of reports on the status of subnetwork connections and trails that have been requested by the managing system and are in the process of being established or released through NE cross connections.

The objectList attribute identifies the sonetSubnetworkConnection and sonetTrail object instances that are to be included in the connectionConfigurationSummaryReport. Objects will be added to the list by the setupSNC, releaseSNC, setupTrail, and releaseTrail actions. An object will be removed from the list when the object instance has been deleted or when the value of the configurationState in the object instance has changed.

The connectionConfigurationSummaryReport notification provides a summary report of the sonetSubnetworkConnection and sonetTrail object instances that are being established or released. The report includes the object instance of the sonetSubnetworkConnection or sonetTrail object, whether it is being established or released, and the percentage of completed or released cross-connections.

The emptyListSuppressionPackage suppresses the connectionConfigurationSummaryReport if the objectList is empty.

A change in the value of the operationalState attribute shall cause a stateChange notification.

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

objectList

";

4.7.4 sonetDegenerateSubnetwork

```
sonetDegenerateSubnetwork              MANAGED OBJECT CLASS
    DERIVED FROM sonetSubnetwork;
    CHARACTERIZED BY
        sonetDegenerateSubnetworkPackage    PACKAGE
                BEHAVIOUR sonetDegenerateSubnetworkBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 4};

sonetDegenerateSubnetworkBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class represents the degenerate case of a
subnetwork.  That is, this subnetwork allows no flexibility in the assignment of
subnetwork connections between sNTPs. There only exist fixed relationships
between sNTPs which cannot be altered by the managing system.

    The componentLinkList and componentSubnetworkList attributes are set to
null.

    The setupSNC and releaseSNC actions are used to setup and release
subnetwork connections between sNTPs with no flexibility.";
```

4.7.5 sonetLayerNetworkDomain

```
sonetLayerNetworkDomain        MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":characteristicInformationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M.3100:1995":userLabelPackage,
        sonetLayerNetworkDomainPackage PACKAGE
                BEHAVIOUR sonetLayerNetworkDomainBehaviour;
                ATTRIBUTES
                        sonetLayerNetworkDomainId
                                GET;
                ACTIONS
                        addLink,
                        releaseTrail,
                        removeLink,
                        setupTrail;;;
REGISTERED AS { sIFNLMObjectClass 5};

sonetLayerNetworkDomainBehaviour BEHAVIOUR
```

DEFINED AS

"The sonetLayerNetworkDomain object class represents the part of a layer network managed by a single administrative domain. Trails which span more than one administrative domain may be supported by sonetSubnetworkConnections and link connections within any given sonetLayerNetworkDomain.

The setupTrail action is used to explicitly create a sonetTrail object instance and associate it with the terminating sonetNetworkTTPs or NE View TTPs.

The releaseTrail action is used to explicitly delete a sonetTrail object instance and remove it's associations with sonetNetworkTTPs or NE View TTPs.

The addLink action is used to explicitly create an sonetLink object instance and associate it with the terminating sonetSNTPPools.

The removeLink action is used to explicitly delete an sonetLink object instance and remove it's associations with sonetSNTPPools." ;

4.7.6 sonetLink

```
sonetLink      MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. X.721 | ISO/IEC 10165-2:1992":administrativeStatePackage,
        "Rec. M. 3100:1995":attributeValueChangeNotificationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M. 3100:1995":stateChangeNotificationPackage,
        "Rec. X.721 | ISO/IEC 10165-2:1992":usageStatePackage,
        "Rec. M.3100:1995":userLabelPackage,
        sonetLinkPackage PACKAGE
                BEHAVIOUR sonetLinkBehaviour;
                ATTRIBUTES
                        aEndList
                                GET,
                        sonetLinkId
                                GET,
                        zEndList
                                GET;
                ACTIONS
                        addLinkCapacity,
                        releaseLinkConnection,
                        removeLinkCapacity,
```

setupLinkConnection;;;
REGISTERED AS { sIFNLMObjectClass 6};

sonetLinkBehaviour BEHAVIOUR
DEFINED AS
"A sonetLink is a topological component that provides transport capacity between two endpoints via a fixed (i.e., inflexible routing) relationship. The two endpoints are groupings of sonetNetworkCTPs or NE View CTPs. These groupings are represented at all levels of partitioning by sonetSNTPPools. Therefore, a single sonetLink object instance has pointer relationships, through the aEndList and zEndList attributes, with many sonetSNTPPool object instances. A sonetLink also represents a set of link connections.

The addLinkCapacity action is used to add additional capacity to the link. This action may require the server layer to provide additional bandwidth. If this action is successful, additional sonetNetworkCTPs or NE View CTPs and related sonetSNTPs will be created and associated with the terminating sonetSNTPPools.

The removeLinkCapacity action is used to remove excess capacity from the link. This action causes the deletion of related sonetSNTPs and sonetNetworkCTPs or NE View CTPs. This action requires that terminated link connections be explicitly released before the related SNTPs and sonetNetworkCTPs or NE View CTPs are deleted.

The setupLinkConnection action is used to explicitly create a sonetLinkConnection object instance and associate it with the terminating sonetNetworkCTPs or NE View CTPs.

The releaseLinkConnection action is used to explicitly delete a sonetLinkConnection object instance and remove it's associations with sonetNetworkCTPs or NE View CTPs.

The userLabel associated with an object instance of this object class must be unique within the context of the sonetLayerNetworkDomain.

The administrativeState is used for administratively locking and unlocking the sonetLink. When unlocked, the sonetLink functions normally. When in the locked state, the sonetLink is prohibited from adding or removing linkConnections. Also, the sonetLink is prohibited from adding or removing link capacity. Locking a sonetLink does not automatically lock the contained linkConnections. Shutting down is not supported.

The usageState is used to describe the usage of the sonetLink. When idle,

all related CTPs are not connected by link connections. When active, some of the related CTPs are not connected by link connections. When busy, none of the related CTPs are not connected by link connections. When the administrative state of the link is locked, the value of the usage state of the link cannot change.

A change in the value of the administrativeState attribute or usageState attribute shall cause a stateChange notification.

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

aEndList

zEndList

";

4.7.7 sonetLinkConnection

```
sonetLinkConnection        MANAGED OBJECT CLASS
    DERIVED FROM sonetTransportEntity;
    CHARACTERIZED BY
        sonetLinkConnectionPackage PACKAGE
                BEHAVIOUR sonetLinkConnectionBehaviour;
                ATTRIBUTES
                        aEndCTP
                                GET,
                        zEndCTP
                                GET;;;
REGISTERED AS { sIFNLMObjectClass 7};

sonetLinkConnectionBehaviour BEHAVIOUR
    DEFINED AS
```

"A sonetLinkConnection is a transport entity that represents the fixed capacity of transfer of characteristic information between sonetNetworkCTPs or NE View CTPs. Only point-to-point link connections are supported by this object class. The aEndCTP and zEndCTP attributes point to the terminating sonetNetworkCTPs or NE View CTPs.

The operationalState is used to describe the operability of the linkConnection. If the state is enabled, the linkConnection is fully or partially operational. If the state is disabled, the linkConnection is totally inoperable.

The userLabel associated with an object instance of this object class must be unique within the context of the sonetLink.";

4.7.8 sonetNetworkCTP

This managed object class is used for inheritance purposes only.

```
sonetNetworkCTP    MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":attributeValueChangeNotificationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M. 3100:1995":locationNamePackage,
        sonetNetworkCTPPackage          PACKAGE
                BEHAVIOUR sonetNetworkCTPBehaviour;
                ATTRIBUTES
                        channelIdentifier
                                GET,
                        sonetNetworkCTPId
                                GET,
                        serverLayerNTTP
                                GET,
                        tPConnectionState
                                GET;;;
    CONDITIONAL PACKAGES
        "Rec. M. 3100:1995":operationalStatePackage
                PRESENT IF "the NE View CTP the sonetNetworkCTP is
                                abstracting supports the operational state attribute.",
        "Rec. M. 3100:1995":stateChangeNotificationPackage
                PRESENT IF "the operationalStatePackage is supported.";
REGISTERED AS { sIFNLMObjectClass 8};

sonetNetworkCTPBehaviour BEHAVIOUR
    DEFINED AS
        "The sonetNetworkCTP (or subclasses) object class represents the
extremity of a potential or actual link connection.  It is also a network level
abstraction of an NE view connection termination point.

    The channelIdentifier attribute provides a group number and a channel
number to identify the timeslot associated with this NetworkCTP.

    The serverLayerNTTP attribute identifies the object instance of the related
networkTTP (and subclasses) in the server layer.
```

The locationName attribute in the locationNamePackage represents a specific geographic location where the NetworkCTP is located.

The tPConnectionState attribute provides the connection status for the NetworkCTP.

If supported, the operational state of the sonetNetworkCTP will reflect the operational state of the abstracted NE View CTP.

If supported, a change in the value of the operationalState attribute shall cause a stateChange notification. A change in the value of the tPConnectionState attribute shall cause a stateChange notification.

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

- channelIdentifier
- locationName
- serverLayerNTTP

";

4.7.9 sonetNetworkCTPBidirectional

```
sonetNetworkCTPBidirectional            MANAGED OBJECT CLASS
    DERIVED FROM sonetNetworkCTPSource,sonetNetworkCTPSink;
    CHARACTERIZED BY
        sonetNetworkCTPBidirectionalPackage  PACKAGE
                BEHAVIOUR sonetNetworkCTPBidirectionalBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 9};

sonetNetworkCTPBidirectionalBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class represents both a sink and a source
        network connection termination point.";
```

4.7.10 sonetNetworkCTPSink

```
sonetNetworkCTPSink             MANAGED OBJECT CLASS
    DERIVED FROM sonetNetworkCTP;
    CHARACTERIZED BY
```

```
        sonetNetworkCTPSinkPackage    PACKAGE
                BEHAVIOUR sonetNetworkCTPSinkBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 10};

sonetNetworkCTPSinkBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class is a unidirectional networkCTP which
terminates a unidirectional linkConnection.";
```

4.7.11 sonetNetworkCTPSource

```
sonetNetworkCTPSource   MANAGED OBJECT CLASS
    DERIVED FROM sonetNetworkCTP;
    CHARACTERIZED BY
        sonetNetworkCTPSourcePackage          PACKAGE
                BEHAVIOUR sonetNetworkCTPSourceBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 11};

sonetNetworkCTPSourceBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class is a unidirectional networkCTP which
originates a unidirectional linkConnection.";
```

4.7.12 sonetNetworkTTP

This managed object class is used for inheritance purposes only.

```
sonetNetworkTTP    MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. X.721 | ISO/IEC 10165-2:1992":administrativeStatePackage,
        "Rec. M. 3100:1995":attributeValueChangeNotificationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M. 3100:1995":locationNamePackage,
        "Rec. M. 3100:1995":operationalStatePackage,
        "Rec. M. 3100:1995":stateChangeNotificationPackage,
        sonetNetworkTTPPackage PACKAGE
                BEHAVIOUR sonetNetworkTTPBehaviour;
                ATTRIBUTES
                        clientLayerNCTPs
                                GET,
                        sonetNetworkTTPId
                                GET,
                        tPConnectionState
```

GET;;;
REGISTERED AS { sIFNLMObjectClass 12};

sonetNetworkTTPBehaviour BEHAVIOUR
DEFINED AS
"The sonetNetworkTTP object class represents the potential extremity of a sonetTrail. It is also a network level abstraction of an NE View trail termination point.

The clientLayerNCTPs attribute identifies the object instances of the related sonetNetworkCTPs (and subclasses) in the client layer. The sonetNetworkCTPs may belong to multiple sonetLayerNetworkDomains.

The administrativeState is used for administratively locking and unlocking the sonetNetworkTTP. When unlocked, the sonetNetworkTTP functions normally. When in the locked state, the sonetNetworkTTP is removed from service.

The operationalState describes the operability of the NetworkTTP. When enabled, the NetworkTTP is either partially or fully operable. When disabled, the NetworkTTP is totally inoperable.

The locationName attribute in the locationNamePackage represents a specific geographic location where the NetworkTTP is located.

The tPConnectionState attribute provides the connection status for the NetworkTTP.

A change in the value of the administrativeState attribute, operationalState attribute, or tPConnectionState attribute shall cause a stateChange notification.

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

clientLayerNCTPs

locationName
";

4.7.13 sonetNetworkTTPBidirectional

sonetNetworkTTPBidirectional MANAGED OBJECT CLASS
DERIVED FROM sonetNetworkTTPSource,sonetNetworkTTPSink;
CHARACTERIZED BY

sonetNetworkTTPBidirectionalPackage PACKAGE
BEHAVIOUR sonetNetworkTTPBidirectionalBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 13};

sonetNetworkTTPBidirectionalBehaviour BEHAVIOUR
DEFINED AS
"This managed object class represents both a sink and a source network trail termination point.";

4.7.14 sonetNetworkTTPSink

sonetNetworkTTPSink MANAGED OBJECT CLASS
DERIVED FROM sonetNetworkTTP;
CHARACTERIZED BY
sonetNetworkTTPSinkPackage PACKAGE
BEHAVIOUR sonetNetworkTTPSinkBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 14};

sonetNetworkTTPSinkBehaviour BEHAVIOUR
DEFINED AS
"This managed object class is a unidirectional sonetNetworkTTP which terminates a unidirectional sonetTrail.";

4.7.15 sonetNetworkTTPSource

```
sonetNetworkTTPSource   MANAGED OBJECT CLASS
    DERIVED FROM sonetNetworkTTP;
    CHARACTERIZED BY
        sonetNetworkTTPSourcePackage        PACKAGE
                BEHAVIOUR sonetNetworkTTPSourceBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 15};

sonetNetworkTTPSourceBehaviour BEHAVIOUR
    DEFINED AS
        “This managed object class is a unidirectional sonetNetworkTTP which
originates a unidirectional sonetTrail.”;
```

4.7.16 sonetRCAARecord

```
sonetRCAARecord  MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":alarmRecord;
    CHARACTERIZED BY
        sonetRCAARecordPackage PACKAGE
                BEHAVIOUR sonetRCAARecordBehaviour;
                ATTRIBUTES
                        alarmedObjectClass
                                GET,
                        alarmedObjectInstance
                                GET,
                        faultLocation
                                GET,
                        serviceAffecting
                                GET,
                        numberOfNELAlarms
                                GET,
                        affectedTransmissionResources
                                GET,
                        highestPriorityNELalarmRecords
                                GET,
                        lowerPriorityNELalarmRecords
                                GET;;;
REGISTERED AS { sIFNLMObjectClass 16};

sonetRCAARecordBehaviour BEHAVIOUR
```

DEFINED AS

"The sonetRCAARecord is generated by the root cause alarm analysis routine based on one or more NEL alarm event reports. It contains all the information contained in the RCAA Notification as well as the lowerPriorityNELalarmRecords attribute which lists all correlated NEL alarm records which were not included in the highestPriorityNELalarmRecords. The correlatedNotifications will contain the notificationIdentifier of previous RCAA notifications to which this notification is either an update or a clear.
Instances of the sonetRCAARecord will be created by the alarm analysis routine. Instances will be deleted per the configuration of the log.";

4.7.17 sonetRoutingProfile

```
sonetRoutingProfile MANAGED OBJECT CLASS
    DERIVED FROM “Rec. X. 721 | ISO/IEC 10165-2”:top;
    CHARACTERIZED BY
        “Rec. M. 3100:1995”:createDeleteNotificationPackage,
        sonetRoutingProfilePackage PACKAGE
                BEHAVIOUR sonetRoutingProfileBehaviour;
                ATTRIBUTES
                        maxHops
                                GET,
                        routeDescriptionList
                                GET,
                        sonetRoutingProfileId
                                GET;;;
REGISTERED AS { sIFNLMObjectClass 17};

sonetRoutingProfileBehaviour BEHAVIOUR
    DEFINED AS
```

“The sonetRoutingProfile object class represents a set of topological routing constraints that can be applied to a new sonetSubnetworkConnection or sonetTrail during setup. Subnetwork connections or trails shall be rejected if the routing criteria cannot be met.

The maxHops attribute is the maximum number of network elements that the new connection or trail may traverse. This attribute may be set to null to indicate that the maxHops criteria does not apply.

The routeDescriptionList attribute is a list of objects (such as sonetLinks, sonetSubnetworks, sonetSNTPPools, etc.) and their use in routing (prohibited, mandatory, preferred).

Objects may be referenced by the routeDescriptionList as being excluded, mandatory, or preferred. A mandatory object must be used. An excluded object must not be used. A preferred object should be used if possible.

A sonetSubnetworkConnection or sonetTrail may be referenced by the routeDescriptionList as same route or diverse route. A new sonetSubnetworkConnection or sonetTrail being created must follow the same route as a sameRoute referenced object. A new sonetSubnetworkConnection or sonetTrail must follow a different route than a referenced object referred to as diverseRoute.";

4.7.18 sonetSNTP

This managed object class is used for inheritance purposes only.

```
sonetSNTP   MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":attributeValueChangeNotificationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        sonetSNTPPackage        PACKAGE
                BEHAVIOUR sonetSNTPBehaviour;
                ATTRIBUTES
                        containingSNTPPool      GET-REPLACE,
                        reflectedTP             GET,
                        sNCPointer              GET,
                        sonetSNTPId             GET;;;
REGISTERED AS { sIFNLMObjectClass 18};

sonetSNTPBehaviour BEHAVIOUR
    DEFINED AS
        "A sonetSNTP is an abstraction that represents the binding between a
sonetSubnetwork and a CTP (either network or NE view) or a TTP (either
network or NE view). It also represents the potential for connection across a
sonetSubnetwork.

The containingSNTPPool attribute identifies the sonetSNTPPool object
instance which this sonetSNTP object instance (subclass) is related to, if any.

The reflectedTP attribute identifies the CTP (network or NE View) or TTP
(network or NE view) object instance the sonetSNTP (subclass) is related to.

The sNCPointer attribute points the sonetSubnetworkConnection, if any, the
sonetSNTP (subclass) is terminating.
```

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

containingSNTPPool

reflectedTP

sNCPointer
";

4.7.19 sonetSNTPBidirectional

```
sonetSNTPBidirectional          MANAGED OBJECT CLASS
    DERIVED FROM sonetSNTPSource,sonetSNTPSink;
    CHARACTERIZED BY
        sonetSNTPBidirectionalPackage  PACKAGE
                BEHAVIOUR sonetSNTPBidirectionalBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 19};

sonetSNTPBidirectionalBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class represents both a sink and a source
        subnetwork termination point.";
```

4.7.20 sonetSNTPPool

```
sonetSNTPPool      MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":attributeValueChangeNotificationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M. 3100:1995":stateChangeNotificationPackage,
        "Rec. X.721 | ISO/IEC 10165-2:1992":usageStatePackage,
        "Rec. M. 3100:1995":userLabelPackage,
        sonetSNTPPoolPackage PACKAGE
                BEHAVIOUR sonetSNTPPoolBehaviour;
                ATTRIBUTES
                        linkPointer
                                GET-REPLACE,
                        sNTPList
                                GET-REPLACE      ADD-REMOVE,
                        sonetSNTPPoolId
                                GET;;;
```

CONDITIONAL PACKAGES
poolCapacityManagementPackage PRESENT IF "the sonetSNTPPool represents the end of a link which crosses the EMS domain boundary.";
REGISTERED AS { sIFNLMObjectClass 20};

sonetSNTPPoolBehaviour BEHAVIOUR
DEFINED AS
"This managed object class represents a set of (possibly empty) sonetSNTPs at the frontier of a given sonetSubnetwork. An sonetSNTPPool may also terminate a sonetLink. The linkPointer attribute points to the terminating sonetLink, if any. The sNTPList attribute points to the set of sonetSNTPs the sonetSNTPPool is grouping together.

If supported, the addPoolCapacity ACTION increases the pool capacity of the SNTPPool by adding SNTPs to the pool.

If supported, the removePoolCapacity ACTION decreases the pool capacity of the SNTPPool by removing SNTPs from the pool.

The usageState is used to describe the usage of the sonetSNTPPool. When idle, all related SNTPs are not cross-connected. When active, some of the related SNTPs are not cross-connected. When busy, none of the related SNTPs are not cross-connected.

A change in the value of the usageState attribute shall cause a stateChange notification.

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

linkPointer

sNTPList

";

4.7.21 sonetSNTPSink

sonetSNTPSink MANAGED OBJECT CLASS
DERIVED FROM sonetSNTP;
CHARACTERIZED BY
sonetSNTPSinkPackage PACKAGE

```
            BEHAVIOUR sonetSNTPSinkBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 21};

sonetSNTPSinkBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class is a unidirectional sonetSNTP which
terminates a unidirectional sonetSubnetworkConnection.";
```

4.7.22 sonetSNTPSource

```
sonetSNTPSource  MANAGED OBJECT CLASS
    DERIVED FROM sonetSNTP;
    CHARACTERIZED BY
        sonetSNTPSourcePackage        PACKAGE
            BEHAVIOUR sonetSNTPSourceBehaviour;;;
REGISTERED AS { sIFNLMObjectClass 22};

sonetSNTPSourceBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object class is a unidirectional sonetSNTP which
originates a unidirectional sonetSubnetworkConnection.";
```

4.7.23 sonetSubnetwork

```
sonetSubnetwork            MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":attributeValueChangeNotificationPackage,
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M. 3100:1995":userLabelPackage,
        sonetSubnetworkPackage PACKAGE
            BEHAVIOUR sonetSubnetworkBehaviour;
            ATTRIBUTES
                componentLinkList
                    GET-REPLACE      ADD-REMOVE,
                componentSubnetworkList
                    GET-REPLACE ADD-REMOVE,
                sonetSubnetworkId
                    GET,
                subnetworkType     GET;
            ACTIONS
                setupSNC,
```

releaseSNC;;;
REGISTERED AS { sIFNLMObjectClass 23};

sonetSubnetworkBehaviour BEHAVIOUR
DEFINED AS
"This managed object class represents a topological component used to effect routing of a specific characteristic information. A sonetSubnetwork is associated with a specific layer network. Within a given layer, partitioning may be applied to decompose a sonetSubnetwork into its component sonetSubnetworks and links.

The componentLinkList and componentSubnetworkList attributes identify the components of the sonetSubnetwork which exist at the next lower level of partitioning.

The subnetworkType attribute provides information on the element level resource(s) this object instance is abstracting.

The setupSNC and releaseSNC actions are used to setup and release sonetSubnetworkConnections between sonetSNTPs contained by the sonetSubnetwork.

A change in the value of any of the following attributes shall cause an attributeValueChange notification:

componentLinkList

componentSubnetworkList

subnetworkType
";

4.7.24 sonetSubnetworkConnection

sonetSubnetworkConnection MANAGED OBJECT CLASS
DERIVED FROM sonetTransportEntity;
CHARACTERIZED BY
“Rec. X.721 | ISO/IEC 10165-2:1992”:administrativeStatePackage,
sonetSubnetworkConnectionPackage PACKAGE
BEHAVIOUR sonetSubnetworkConnectionBehaviour;
ATTRIBUTES
aEndSNTP

GET,
componentLinkConnectionList
GET,
componentSubnetworkConnectionList
GET,
configurationState
GET,
relatedRoutingProfile
GET,
zEndSNTPs
GET;;;
REGISTERED AS { sIFNLMObjectClass 24};

sonetSubnetworkConnectionBehaviour BEHAVIOUR
DEFINED AS
"A sonetSubnetworkConnection object instance represents a transport entity that transfers information across a sonetSubnetwork. A sonetSubnetwork connection is terminated by subnetwork termination points (aEndSNTPs and zENdSNTPs).

A sonetSubnetworkConnection can be composed of sonetSubnetworkConnections and sonetLinkConnections which exist at the next lower level of partitioning. The componentLinkConnectionList and componentSubnetworkConnectionList attributes identify these components of the sonetSubnetworkConnection.

The configurationState attribute describes the procedural status relative to the process of creation or deletion of a sonetSubnetworkConnection object instance. The configurationState indicates that the sonetSubnetworkConnection is in either a preservice, inservice, postservice, or configuration failure state.

The relatedRoutingProfile attribute points to the sonetRoutingProfile object instance which contains the routing profile for the sonetSubnetworkConnection.

A change in the value of the administrativeState attribute or configurationState attribute shall cause a stateChange notification.

The administrativeState is used for administratively locking and unlocking the sonetSubnetworkConnection. When unlocked, the sonetSubnetworkConnection functions normally. When in the locked state, the sonetSubnetworkConnection is prohibited from the transport of characteristic information.

The operationalState describes the operability of the sonetSubnetworkConnection. When enabled, the sonetSubnetworkConnection

is either partially or fully operable. When disabled, the sonetSubnetworkConnection is totally inoperable.

The userLabel associated with an object instance of this object class must be unique within the context of the sonetSubnetwork.";

4.7.25 sonetTrail

```
sonetTrail     MANAGED OBJECT CLASS
    DERIVED FROM sonetTransportEntity;
    CHARACTERIZED BY
        sonetTrailPackage  PACKAGE
                BEHAVIOUR sonetTrailBehaviour;
                ATTRIBUTES
                        aEndTTP
                                GET,
                        configurationState
                                GET,
                        relatedRoutingProfile
                                GET,
                        zEndTTPs
                                GET;;;
    CONDITIONAL PACKAGES
        "Rec. X.721 | ISO/IEC 10165-2:1992":administrativeStatePackage
        PRESENT IF "the trail can be administratively locked and unlocked.";
REGISTERED AS {sIFNLMObjectClass 25};

sonetTrailBehaviour BEHAVIOUR
    DEFINED AS
        "A sonetTrail object class represents the transfer of information between
sonetNetworkTTPs or NE View TTPs.  The aEndTTP and zEndTTPs attributes
point to the terminating sonetNetworkTTPs or NE View TTPs.

        If supported, the administrativeState is used for administratively locking
and unlocking the sonetTrail.  When unlocked, the sonetTrail functions normally.
When in the locked state, the sonetTrail is prohibited from the transport of
characteristic information.

    The relatedRoutingProfile attribute points to the sonetRoutingProfile object
instance which contains the routing profile for the sonetTrail.

    If supported, a change in the value of the administrativeState attribute shall
cause a stateChange notification.  A change in the value of the
```

configurationState attribute shall cause a stateChange notification.

The configurationState attribute describes the procedural status relative to the process of creation or deletion of a sonetTrail object instance. The configurationState indicates that the sonetTrail is in either a preservice, inservice, postservice, or configuration failure state.

The operationalState describes the operability of the sonetTrail. When enabled, the sonetTrail is either partially or fully operable. When disabled, the sonetTrail is totally inoperable.

The userLabel associated with an object instance of this object class must be unique within the context of the sonetLayerNetworkDomain.";

4.7.26 sonetTransportEntity

This managed object class is used for inheritance purposes only.

```
sonetTransportEntity                MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X. 721 | ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "Rec. M. 3100:1995":createDeleteNotificationPackage,
        "Rec. M. 3100:1995":operationalStatePackage,
        "Rec. M. 3100:1995":stateChangeNotificationPackage,
        "Rec. M. 3100:1995":userLabelPackage,
        sonetTransportEntityPackage PACKAGE
                BEHAVIOUR sonetTransportEntityBehaviour;
                ATTRIBUTES
                        "Rec. M.3100:1995":directionality GET,
                        sonetTransportEntityId      GET;;;
REGISTERED AS { sIFNLMObjectClass 26};

sonetTransportEntityBehaviour BEHAVIOUR
    DEFINED AS
        "This managed object represents a transport entity which transfers
information transparently from input to output, either uni-directionally or
bidirectionally.

        A change in the value of the operationalState attribute shall cause a
stateChange notification.";
```

4.8 Packages

4.8.1 emptyListSuppressionPackage

```
emptyListSuppressionPackage PACKAGE
    BEHAVIOUR emptyListSuppressionBehaviour;
REGISTERED AS {sIFNLMPackage 1};

emptyListSuppressionBehaviour BEHAVIOUR
    DEFINED AS
        "The emptyListSuppressionPackage suppresses the
connectionConfigurationSummaryReport if the objectList in the
connectionConfigurationStatusControl object is empty .";
```

4.8.2 poolCapacityManagementPackage

```
poolCapacityManagementPackage PACKAGE
    BEHAVIOUR poolCapacityManagementPackageBehaviour;
    ACTIONS
        addPoolCapacity,
        removePoolCapacity;
REGISTERED AS {sIFNLMPackage 2};

poolCapacityManagementPackageBehaviour BEHAVIOUR
    DEFINED AS
        "Indicates that the sonetSNTPPool represents the end of a link which
crosses the EMS domain boundary.";
```

4.9 Attributes

4.9.1 sonetAccessGroupId

```
sonetAccessGroupId ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
    MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
    BEHAVIOUR sonetAccessGroupIdBehaviour;
REGISTERED AS {sIFNLMAttribute 1};

sonetAccessGroupIdBehaviour BEHAVIOUR
    DEFINED AS
        "The sonetAccessGroupId is an attribute type whose distinguished value
```

can be used as an RDN when naming an instance of the sonetAccessGroup object class.";

4.9.2 aEndList

```
aEndList      ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.EndList;
    MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
    BEHAVIOUR           aEndListBehaviour;
REGISTERED AS {sIFNLMAttribute 2};

aEndListBehaviour BEHAVIOUR
    DEFINED AS
        "This attribute identifies the aEnd SNTPPools the link is terminated by.";
```

4.9.3 aEndCTP

```
aEndCTP     ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.CTP;
    MATCHES FOR EQUALITY;
    BEHAVIOUR           aEndCTPBehaviour;
REGISTERED AS {sIFNLMAttribute 3};

aEndCTPBehaviour BEHAVIOUR
    DEFINED AS
        "The aEndCTP attribute is a pointer to the subclass of sonetNetworkCTP
or NE View CTP object instance terminating the aEnd of the link connection.";
```

4.9.4 aEndSNTP

```
aEndSNTP   ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.SNTP;
    MATCHES FOR EQUALITY;
    BEHAVIOUR           aEndSNTPBehaviour;
REGISTERED AS {sIFNLMAttribute 4};

aEndSNTPBehaviour BEHAVIOUR
    DEFINED AS
        "This attribute identifies the aEnd sonetSNTP that terminates the
sonetSubnetwork connection.  In the case of a unidirectional sonetSubnetwork
connection, the object instance identified will be of the sonetSNTPSource object
class.  In the case of a bidirectional sonetSubnetwork connection, the object
instance identified will be of the sonetSNTPBidirectional object class.";
```

4.9.5 aEndTTP

```
aEndTTP      ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.TTP;
    MATCHES FOR EQUALITY;
    BEHAVIOUR              aEndTTPBehaviour;
REGISTERED AS {sIFNLMAttribute 5};

aEndTTPBehaviour BEHAVIOUR
    DEFINED AS
        “This attribute identifies the aEnd sonetNetworkTTP or NE View TTP that terminates the sonetTrail.  In the case of a unidirectional sonetTrail, the object instance identified will be of the sonetNetworkTTPSource or NE View TTPSource object class.  In the case of a bidirectional sonetTrail, the object instance identified will be of the sonetNetworkTTPBidirectional or NE View TTPBidirectional object class.”;
```

4.9.6 affectedTransmissionResources

```
affectedTransmissionResources  ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.AffectedTransmissionResources;
    MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
    BEHAVIOUR              affectedTransmissionResourcesBehaviour;
REGISTERED AS {sIFNLMAttribute 6};

affectedTransmissionResourcesBehaviour BEHAVIOUR
    DEFINED AS
        "The affectedTransmissionResources attribute indicates the key transmission resources which are affected by this alarm.  Any other transmission resource which depends on this resource either through layering or partitioning will also be affected.";
```

4.9.7 sonetAlarmAnalysisRoutineId

```
sonetAlarmAnalysisRoutineId ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
    MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
    BEHAVIOUR              sonetAlarmAnalysisRoutineIdBehaviour;
REGISTERED AS {sIFNLMAttribute 7};

sonetAlarmAnalysisRoutineIdBehaviour BEHAVIOUR
    DEFINED AS
```

"The sonetAlarmAnalysisRoutineId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetAlarmAnalysisRoutine object class.";

4.9.8 alarmedObjectClass

```
alarmedObjectClass          ATTRIBUTE
    WITH ATTRIBUTE SYNTAX  SIFNLMMod.ObjectClass;
    MATCHES FOR EQUALITY;
    BEHAVIOR alarmedObjectClassBehavior;
REGISTERED AS {sIFNLMAttribute 3};

alarmedObjectClassBehavior BEHAVIOR
    DEFINED AS
        "The object class of the object instance which the alarm analysis routine has determined to be the root cause of the event.";
```

4.9.9 alarmedObjectInstance

```
alarmedObjectInstance      ATTRIBUTE
    WITH ATTRIBUTE SYNTAX  SIFNLMMod.ObjectInstance;
    MATCHES FOR EQUALITY;
    BEHAVIOR alarmedObjectInstanceBehavior;
REGISTERED AS {sIFNLMAttribute 4};

alarmedObjectInstanceBehavior BEHAVIOR
    DEFINED AS
        "The object instance of the object which the alarm analysis routine has determined to be the root cause of the event.";
```

4.9.10 channelIdentifier

```
channelIdentifier ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.ChannelIdentifier;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR channelIdentifierBehaviour;
REGISTERED AS {sIFNLMAttribute 6};

channelIdentifierBehaviour BEHAVIOUR
    DEFINED AS
        "This attributes provides a group number (if applicable to the network domain) and a channel number.  The group number (if applicable) specifies the timeslot of the AUG or TUG (VT Group) within its server multiplex. The value
```

shall be the integer which represents the position of the group timeslot in temporal order. The first timeslot shall be numbered one. The channel number specifies the timeslot of the networkCTP within its group or server multiplex. The value shall be the integer which represents the position of the timeslot in temporal order. The first timeslot shall be numbered one.";

4.9.11 clientLayerNCTPs

clientLayerNCTPs ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ClientNCTPs;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR clientLayerNCTPsBehaviour;
REGISTERED AS {sIFNLMAttribute 7};

clientLayerNCTPsBehaviour BEHAVIOUR
DEFINED AS
"This attribute identifies the object instances of the related sonetNetworkCTP (and subclasses) in the client layer. The sonetNetworkCTPs may belong to multiple sonetLayerNetworkDomains.";

4.9.12 componentLinkList

componentLinkList ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ComponentList;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR componentLinkListBehaviour;
REGISTERED AS {sIFNLMAttribute 8};

componentLinkListBehaviour BEHAVIOUR
DEFINED AS
"The componentLinkList attribute represents an association between a sonetSubnetwork and its component links at the next lower level of partitioning. A sonetSubnetwork may be associated with zero or more component links.";

4.9.13 componentLinkConnectionList

componentLinkConnectionList ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ComponentList;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR componentLinkConnectionListBehaviour;
REGISTERED AS {sIFNLMAttribute 9};

componentLinkConnectionListBehaviour BEHAVIOUR
DEFINED AS
"The componentLinkConnectionList attribute represents an association between a sonetSubnetworkConnection and its component link connections. A sonetSubnetwork connection may be associated with zero or more component link connections.";

4.9.14 componentSubnetworkList

componentSubnetworkList ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ComponentList;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR componentSubnetworkListBehaviour;
REGISTERED AS {sIFNLMAttribute 10};

componentSubnetworkListBehaviour BEHAVIOUR
DEFINED AS
"The componentSubnetworkList atttribute represents an association between a sonetSubnetwork and its component sonetSubnetworks at the next lower level of partitioning. A sonetSubnetwork may be associated with zero or more component sonetSubnetworks.";

4.9.15 componentSubnetworkConnectionList

componentSubnetworkConnectionList ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ComponentList;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR componentSubnetworkConnectionListBehaviour;
REGISTERED AS {sIFNLMAttribute 11};

componentSubnetworkConnectionListBehaviour BEHAVIOUR
DEFINED AS
"The componentSubnetworkConnectionList atttribute represents an association between a sonetSubnetwork connection and its component sonetSubnetwork connections. A sonetSubnetwork connection may be associated with zero or more component sonetSubnetwork connections.";

4.9.16 configurationState

configurationState ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ConfigurationState;
MATCHES FOR EQUALITY;
BEHAVIOUR configurationStateBehaviour;
REGISTERED AS {sIFNLMAttribute 12};

configurationStateBehaviour BEHAVIOUR
DEFINED AS
"The configurationState attribute describes the procedural status relative to the process of creation or deletion of sonetSubnetworkConnection and sonetTrail object instances. The possible states of the configurationState attribute are: preservice, inservice, postservice, and configuration failure state.

Preservice: The state that exists after the creation of the transport entity object instance (resulting from a set-up request) and prior to either the successful completion of all supporting NE-level cross-connections or the failure of a set-up process. The transport entity does not support service during the preservice state.

Inservice: The state that results after the successful completion of all supporting NE-level cross-connections. The transport entity may provide service while in the inservice state.

Postservice: The state that exists after a request for deletion of the transport entity object instance has been received and prior to either the successful tear-down of all supporting NE-level cross-connections or the failure of the tear-down process. The transport entity does not support service while in the postservice state.

Configuration failure: The state of a transport entity that exists after either:

a) the failure to undo the cross-connections in an aborted set-up process, or

b) an aborted tear-down process.

The transport entity does not support service during the configuration failure state.";

4.9.17 sonetConnectionPendingStatusControlId

sonetConnectionPendingStatusControlId ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR sonetConnectionPendingStatusControlIdBehaviour;
REGISTERED AS {sIFNLMAttribute 13};

sonetConnectionPendingStatusControlIdBehaviour BEHAVIOUR
DEFINED AS
"The sonetConnectionPendingStatusControlId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetConnectionSetupStatusControl object class.";

4.9.18 containingSNTPPool

containingSNTPPool ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ContainingSNTPPool;
MATCHES FOR EQUALITY;
BEHAVIOUR containingSNTPPoolBehaviour;
REGISTERED AS {sIFNLMAttribute 14};

containingSNTPPoolBehaviour BEHAVIOUR
DEFINED AS
"This attribute identifies the sonetSubnetworkTPPool the sonetSNTP is related to.";

4.9.19 highestPriorityNELalarmRecords

highestPriorityNELalarmRecords ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.AlarmRecords;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR highestPriorityNELalarmRecordsBehavior;
REGISTERED AS {sIFNLMAttribute 5};

highestPriorityNELalarmRecordsBehaviour BEHAVIOUR
DEFINED AS
"The highestPriorityNELalarmRecords attribute is a set of NEL alarm record instances which are correlated to this RCAA record. These NEL alarms are considered to be of primary importance in determining the RCAA result.";

4.9.20 sonetLayerNetworkDomainId

sonetLayerNetworkDomainId ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR sonetLayerNetworkDomainIdBehaviour;
REGISTERED AS {sIFNLMAttribute 15};

sonetLayerNetworkDomainIdBehaviour BEHAVIOUR
DEFINED AS
“The sonetLayerNetworkDomainId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetLayerNetworkDomain object class.”;

4.9.21 sonetLinkId

```
sonetLinkId   ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
    MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
    BEHAVIOUR           sonetLinkIdBehaviour;
REGISTERED AS {sIFNLMAttribute 16};

sonetLinkIdBehaviour BEHAVIOUR
    DEFINED AS
        "The sonetLinkId is an attribute type whose distinguished value can be
used as an RDN when naming an instance of the sonetLink object class.";
```

4.9.22 linkPointer

```
linkPointer    ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.LinkPointer;
    MATCHES FOR EQUALITY;
    BEHAVIOUR           linkPointerBehaviour;
REGISTERED AS {sIFNLMAttribute 17};

linkPointerBehaviour BEHAVIOUR
    DEFINED AS
        “The linkPointer attribute identifies the sonetLink object instance which is
associated to the sonetSNTPPool or sonetAccessGroup object instance.”;
```

4.9.23 faultLocation

```
faultLocation ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.FaultLocation;
    MATCHES FOR EQUALITY, SUBSTRINGS, SET-COMPARISON, SET-
            INTERSECTIONS;
    BEHAVIOUR           faultLocationBehaviour;
REGISTERED AS {sIFNLMAttribute 6};

faultLocationBehaviour BEHAVIOUR
    DEFINED AS
        "The faultLocation will contain a possibly empty list of geographic
locations at which the root cause alarm is present.  For example if the root cause
is determined to be a fiber cut the faultLocation would list the geographic
locations of the end points of the fiber.";
```

4.9.24 lowerPriorityNELalarmRecords

```
lowerPriorityNELalarmRecords     ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.AlarmRecords;
    MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
    BEHAVIOUR               lowerPriorityNELalarmRecordsBehavior;
REGISTERED AS {sIFNLMAttribute 7};

lowerPriorityNELalarmRecordsBehaviour BEHAVIOUR
    DEFINED AS
        "The lowerPriorityNELalarmRecords attribute is a set of NEL alarm record
instances which are correlated to this RCAA record and which have not been
referenced in highestPriorityNELalarmRecords.";
```

4.9.25 maxHops

```
maxHops      ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.MaxHops;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR           maxHopsBehaviour;
REGISTERED AS {sIFNLMAttribute 18};

maxHopsBehaviour BEHAVIOUR
    DEFINED AS
        "The maxHops attribute is the maximum number of network elements that
a connection or trail may traverse.";
```

4.9.26 sonetNetworkCTPId

```
sonetNetworkCTPIdATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
    MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
    BEHAVIOUR           sonetNetworkCTPIdBehaviour;
REGISTERED AS {sIFNLMAttribute 19};

sonetNetworkCTPIdBehaviour BEHAVIOUR
    DEFINED AS
        "The sonetNetworkCTPId is an attribute type whose distinguished value
can be used as an RDN when naming an instance of the sonetNetworkCTP (or
subclasses) object class.";
```

4.9.27 numberOfNELalarms

```
numberOfNELAlarms       ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.Integer;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR           numberOfNELAlarmsBehaviour;
REGISTERED AS {sIFNLMAttribute 8};

numberOfNELAlarmsBehaviour BEHAVIOUR
    DEFINED AS
        "The numberOfNELAlarms attribute will indicate the number of NEL
alarms which are correlated to this record.  If only one alarm is correlated the
alarm analysis routine was unable to correlate the NEL alarm to other alarms
and this is a raw alarm.";
```

4.9.28 serviceAffecting

```
serviceAffecting      ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.Boolean;
    MATCHES FOR EQUALITY;
    BEHAVIOUR             serviceAffectingBehaviour;
REGISTERED AS {sIFNLMAttribute 9};

serviceAffectingBehaviour BEHAVIOUR
    DEFINED AS
        "The serviceAffecting attribute will indicate whether the alarm analysis routine has determined this alarm to be service affecting.";
```

4.9.29 sonetNetworkTTPId

```
sonetNetworkTTPId ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
    MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
    BEHAVIOUR             sonetNetworkTTPIdBehaviour;
REGISTERED AS {sIFNLMAttribute 20};

sonetNetworkTTPIdBehaviour BEHAVIOUR
    DEFINED AS
        “The sonetNetworkTTPId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetNetworkTTP (or subclasses) object class.”;
```

4.9.30 tTPList

```
tTPList               ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.TTPList;
    MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
    BEHAVIOUR             tTPListBehaviour;
REGISTERED AS {sIFNLMAttribute 21};

tTPListBehaviour BEHAVIOUR
    DEFINED AS
        “This attribute identifies the (subclasses of) networkTTPs or NE View TTPs grouped in the sonetAccessGroup.”;
```

4.9.31 reflectedTP

```
reflectedTP   ATTRIBUTE
```

```
    WITH ATTRIBUTE SYNTAX SIFNLMMod.ReflectedTP;
    MATCHES FOR EQUALITY;
    BEHAVIOUR           reflectedTPBehaviour;
REGISTERED AS {sIFNLMAttribute 22};

reflectedTPBehaviour BEHAVIOUR
    DEFINED AS
        "This attribute identifies the object instance the SNTP is abstracting.";
```

4.9.32 relatedRoutingProfile

```
relatedRoutingProfile       ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.RoutingProfile;
    MATCHES FOR EQUALITY;
    BEHAVIOUR           relatedRoutingProfileBehaviour;
REGISTERED AS {sIFNLMAttribute 23};

relatedRoutingProfileBehaviour BEHAVIOUR
    DEFINED AS
        "The relatedRoutingProfile attribute is a pointer to the sonetRoutingProfile
object instance which is associated with the sonetSubnetworkConnection or
sonetTrail object instance.";
```

4.9.33 routeDescriptionList

```
routeDescriptionListATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.RouteDescriptionList;
    MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
    BEHAVIOUR           routeDescriptionListBehaviour;
REGISTERED AS {sIFNLMAttribute 24};

routeDescriptionListBehaviour BEHAVIOUR
    DEFINED AS
        "The routeDescriptionList attribute is a list of objects (such as links,
sonetSubnetworks, sonetSNTPPools, etc.) and their use in routing (excluded,
mandatory, preferred).";
```

4.9.34 sonetRoutingProfileId

```
sonetRoutingProfileId       ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
    MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
```

BEHAVIOUR sonetRoutingProfileIdBehaviour;
REGISTERED AS {sIFNLMAttribute 25};

sonetRoutingProfileIdBehaviour BEHAVIOUR
DEFINED AS
"The sonetRoutingProfileId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetRoutingProfile object class.";

4.9.35 serverLayerNTTP

serverLayerNTTP ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.ServerNTTP;
MATCHES FOR EQUALITY;
BEHAVIOUR serverLayerNTTPBehaviour;
REGISTERED AS {sIFNLMAttribute 26};

serverLayerNTTPBehaviour BEHAVIOUR
DEFINED AS
"This attribute identifies the object instance of the related networkTTP (and subclasses) in the server layer.";

4.9.36 sNCPointer

sNCPointer ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.SNCPointer;
MATCHES FOR EQUALITY;
BEHAVIOUR sNCPointerBehaviour;
REGISTERED AS {sIFNLMAttribute 27};

sNCPointerBehaviour BEHAVIOUR
DEFINED AS
"This attribute identifies the object instance of the sonetSubnetworkConnection that the SNTP terminates.";

4.9.37 sonetSNTPId

sonetSNTPId ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR sonetSNTPIdBehaviour;
REGISTERED AS {sIFNLMAttribute 28};

sonetSNTPIdBehaviour BEHAVIOUR
DEFINED AS
"The sonetSNTPId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetSNTP object classes or subclasses.";

4.9.38 sNTPList

sNTPList ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.SNTPList;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR sNTPListBehaviour;
REGISTERED AS {sIFNLMAttribute 29};

sNTPListBehaviour BEHAVIOUR
DEFINED AS
"This attribute identifies the sonetSNTPs grouped in the sonetSNTPPool.";

4.9.39 sonetSNTPPoolId

sonetSNTPPoolId ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR sonetSNTPPoolIdBehaviour;
REGISTERED AS {sIFNLMAttribute 30};

sonetSNTPPoolIdBehaviour BEHAVIOUR
DEFINED AS
"The sonetSNTPPoolId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetSNTPPool object classes or subclasses.";

4.9.40 sonetSubnetworkId

sonetSubnetworkId ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR sonetSubnetworkIdBehaviour;
REGISTERED AS {sIFNLMAttribute 31};

sonetSubnetworkIdBehaviour BEHAVIOUR
DEFINED AS
"The sonetSubnetworkId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetSubnetwork object class.";

4.9.41 subnetworkType

subnetworkType ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.SubnetworkType;
MATCHES FOR EQUALITY;
BEHAVIOUR subnetworkTypeBehaviour;
REGISTERED AS {sIFNLMAttribute 32};

subnetworkTypeBehaviour BEHAVIOUR
DEFINED AS
"The subnetworkType attribute provides information about the element level resource(s) the subnetwork is abstracting.";

4.9.42 tPConnectionState

tPConnectionState ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.TPConnectionState;
MATCHES FOR EQUALITY;
BEHAVIOUR tPConnectionStateBehaviour;
REGISTERED AS {sIFNLMAttribute 33};

tPConnectionStateBehaviour BEHAVIOUR
DEFINED AS
"This attribute provides the connection status for NetworkCTPs and NetworkTTPs. The value of this attribute will be not connected only if ALL abstracting SNTPs are not involved in a subnetwork connection. Otherwise, the value will reflect whether the SNTP(s) is sinking, sourcing, or sinking and sourcing a subnetwork connection(s).";

4.9.43 sonetTransportEntityId

sonetTransportEntityId ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR sonetTransportEntityIdBehaviour;
REGISTERED AS {sIFNLMAttribute 34};

sonetTransportEntityIdBehaviour BEHAVIOUR
DEFINED AS
"The sonetTransportEntityId is an attribute type whose distinguished value can be used as an RDN when naming an instance of the sonetTransportEntity object class or subclasses.";

4.9.44 zEndCTP

zEndCTP ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.CTP;
MATCHES FOR EQUALITY;
BEHAVIOUR zEndCTPBehaviour;
REGISTERED AS {sIFNLMAttribute 35};

zEndCTPBehaviour BEHAVIOUR
DEFINED AS
"The zEndCTP attribute is a pointer to the sonetNetworkCTP or NE View CTP (and subclasses) object instance terminating the zEnd of the link connection.";

4.9.45 zEndList

zEndList ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.EndList;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR zEndListBehaviour;
REGISTERED AS {sIFNLMAttribute 36};

zEndListBehaviour BEHAVIOUR
DEFINED AS
"This attribute identifies the zEnd SNTPPools the link is terminated by.";

4.9.46 zEndSNTPs

zEndSNTPs ATTRIBUTE
WITH ATTRIBUTE SYNTAX SIFNLMMod.SNTPs;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
BEHAVIOUR zEndSNTPsBehaviour;
REGISTERED AS {sIFNLMAttribute 37};

zEndSNTPsBehaviour BEHAVIOUR
DEFINED AS

"This attribute identifies the zEnd sonetSNTPs that terminate the sonetSubnetwork connection. In the case of a unidirectional sonetSubnetwork connection, the object instances identified will be of the sonetSNTPSink object class. In the case of a bidirectional sonetSubnetwork connection, the object instances identified will be of the sonetSNTPBidirectional object class.";

4.9.47 zEndTTPs

```
zEndTTPs    ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SIFNLMMod.TTPs;
    MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
    BEHAVIOUR           zEndTTPsBehaviour;
REGISTERED AS {sIFNLMAttribute 38};
```

zEndTTPsBehaviour BEHAVIOUR
DEFINED AS

"This attribute identifies the zEnd networkTTPs or NE View TTPs that terminate the sonetTrail. In the case of a unidirectional sonetTrail, the object instances identified will be of the networkTTPSink or NE View TTPSink object class. In the case of a bidirectional sonetTrail, the object instances identified will be of the networkTTPBidirectional or NE View TTPBidirectional object class.";

4.10 Actions

4.10.1 addLink

```
addLink             ACTION
    BEHAVIOUR       addLinkBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        "M.3100:1995":generalErrorParameter;
    WITH INFORMATION SYNTAX SIFNLMMod.AddLinkInformation;
    WITH REPLY SYNTAX SIFNLMMod.AddLinkReply;
REGISTERED AS {sIFNLMAction 1};
```

addLinkBehaviour BEHAVIOUR
DEFINED AS

"The addLink action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetLayerNetworkDomain (and subclasses) to setup a link. Upon receipt of this request, the managed system creates a sonetLink object instance and sets the related linkPointer attribute in the sonetSNTPPool object instances. In the information syntax, the managing system needs to specify at least one, but not

necessarily at every level of partitioning, sntpPool in both aEndSNTPPools and zEndSNTPPools. The managed system is responsible for maintaining consistency among sNTPPools and links at all levels of partitioning. Also, the agreedUserLabel is returned to the managing system. The agreedUserLabel will be the same as the suppliedUserLabel (if any), unless another label is required to satisfy the uniqueness criteria.";

4.10.2 addLinkCapacity

```
addLinkCapacity     ACTION
    BEHAVIOUR     addLinkCapacityBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        "M.3100:1995":generalErrorParameter;
    WITH INFORMATION SYNTAX SIFNLMMod.AddLinkCapacityInformation;
    WITH REPLY SYNTAX SIFNLMMod.AddLinkCapacityReply;
REGISTERED AS {sIFNLMAction 2};

addLinkCapacityBehaviour BEHAVIOUR
    DEFINED AS
        "The addLinkCapacity action request is sent by a managing system (e.g.,
the NMS) to the managed system (e.g., the EMS) to direct the sonetLink (and
subclasses) to add link capacity. Upon receipt of this request, if server layer
bandwidth is available, the managed system creates two (subclasses of)
sonetNetworkCTP or NE View CTP object instances for each integral increment
in link capacity specified in the request. In addition, the managed system will
also create two associated sonetSNTP object instances (subclasses of) for each
level of partioning at which the sonetLink appears. If server layer bandwidth is
not available, the managed system can either reconfigure the server layer to
provide additional bandwidth or deny the request.";
```

4.10.3 addPoolCapacity

```
addPoolCapacity     ACTION
    BEHAVIOUR addPoolCapacityBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        “M.3100:1995”:generalErrorParameter;
       WITH INFORMATION SYNTAX SIFNLMMod.AddPoolCapacityInformation;
       WITH REPLY SYNTAX SIFNLMMod.AddPoolCapacityReply;
REGISTERED AS {sIFNLMAction 3};
```

addPoolCapacityBehaviour BEHAVIOUR
DEFINED AS
"The addPoolCapacity action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetSNTPPool (and subclasses) to add pool capacity. Upon receipt of this request, if server layer bandwidth is available, the managed system creates one sonetNetworkCTP or NE View CTP or GTP object instance (subclasses of) for each integral increment in pool capacity specified in the request. In addition, the managed system will also create one associated sonetSNTP object instance (subclasses of), for each integral increment, and associate it with both the CTP or GTP and the sonetSNTPPool. Other sonetSNTPs may be created, based on the new CTP's or GTP's relationships to other levels of partitioning. If server layer bandwidth is not available, the managed system can either reconfigure the server layer to provide additional bandwidth or deny the request.";

4.10.4 releaseLinkConnection

releaseLinkConnection ACTION
BEHAVIOUR releaseLinkConnectionBehaviour;
MODE CONFIRMED;
PARAMETERS
"M.3100:1995":generalErrorParameter;
WITH INFORMATION SYNTAX
SIFNLMMod.ReleaseLinkConnectionInformation;
REGISTERED AS {sIFNLMAction 4};

releaseLinkConnectionBehaviour BEHAVIOUR
DEFINED AS
"The releaseLinkConnection action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetLink (and subclasses) to release a link connection. Upon receipt of this request, the managed system deletes the link connection object instance.";

4.10.5 releaseSNC

releaseSNC ACTION
BEHAVIOUR releaseSNCBehaviour;
MODE CONFIRMED;
PARAMETERS
"M.3100:1995":generalErrorParameter;
WITH INFORMATION SYNTAX SIFNLMMod.ReleaseSNCInformation;

REGISTERED AS {sIFNLMAction 5};

releaseSNCBehaviour BEHAVIOUR
DEFINED AS
"The releaseSNC action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetSubnetwork (and subclasses) to release a sonetSubnetwork connection. Upon receipt of this request, the managed system deletes the sonetSubnetworkConnection object instance and sets the related sNCPointers to null.";

4.10.6 releaseTrail

releaseTrail ACTION
BEHAVIOUR releaseTrailBehaviour;
MODE CONFIRMED;
PARAMETERS
"M.3100:1995":generalErrorParameter;
WITH INFORMATION SYNTAX SIFNLMMod.ReleaseTrailInformation;
REGISTERED AS {sIFNLMAction 6};

releaseTrailBehaviour BEHAVIOUR
DEFINED AS
"The releaseTrail action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetLayerNetworkDomain (and subclasses) to release a sonetTrail. Upon receipt of this request, the managed system deletes the sonetTrail object instance.";

4.10.7 removeLink

removeLink ACTION
BEHAVIOUR removeLinkBehaviour;
MODE CONFIRMED;
PARAMETERS
"M.3100:1995":generalErrorParameter;
WITH INFORMATION SYNTAX
SIFNLMMod.RemoveLinkInformation;
REGISTERED AS {sIFNLMAction 7};

removeLinkBehaviour BEHAVIOUR
DEFINED AS

"The removeLink action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetLayerNetworkDomain (and subclasses) to remove the specified link. Upon receipt of this request, the managed system deletes the link object instance and sets the linkPointer attributes in the related sonetSNTPPools object instances to null. This action will fail if there are any sonetLinkConnection object instances named by the specified link.";

4.10.8 removeLinkCapacity

```
removeLinkCapacity        ACTION
    BEHAVIOUR     removeLinkCapacityBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        "M.3100:1995":generalErrorParameter;
    WITH INFORMATION SYNTAX
        SIFNLMMod.RemoveLinkCapacityInformation;
REGISTERED AS {sIFNLMAction 8};

removeLinkCapacityBehaviour BEHAVIOUR
    DEFINED AS
        "The removeLinkCapacity action request is sent by a managing system
(e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetLink
(and subclasses) to remove excess link capacity. Upon receipt of this request,
the managed system deletes two (subclasses of) sonetNetworkCTP or NE View
CTP object instances for each integral decrement specified in the request. In
addition, the managed system will also delete two associated sonetSNTP object
instances (subclasses of) for each level of partioning at which the sonetLink
appears. This action will fail if there are an insufficient number of available
sonetNetworkCTPs or NE View CTPs.";
```

4.10.9 removePoolCapacity

```
removePoolCapacity        ACTION
    BEHAVIOUR     removePoolCapacityBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        "M.3100:1995":generalErrorParameter;
    WITH INFORMATION SYNTAX
        SIFNLMMod.RemovePoolCapacityInformation;
    REGISTERED AS {sIFNLMAction 9};
```

removePoolCapacityBehaviour BEHAVIOUR
DEFINED AS
"The removePoolCapacity action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetSNTPPool (and subclasses) to remove excess pool capacity. Upon receipt of this request, the managed system deletes one sonetNetworkCTP or NE View CTP or GTP object instance (subclasses of) for each integral decrement specified in the request. In addition, the managed system will also delete all associated sonetSNTP object instances (subclasses of), at every level of partitioning at which the CTP's or GTP's are abstracted. This action will fail if there are an insufficient number of available sonetNetworkCTPs or NE View CTPs or GTPs.";

4.10.10 setupLinkConnection

setupLinkConnection ACTION
BEHAVIOUR setupLinkConnectionBehaviour;
MODE CONFIRMED;
PARAMETERS
"M.3100:1995":generalErrorParameter;
WITH INFORMATION SYNTAX SIFNLMMod.SetupLinkConnectionInformation;
WITH REPLY SYNTAX SIFNLMMod.SetupLinkConnectionReply;
REGISTERED AS {sIFNLMAction 10};

setupLinkConnectionBehaviour BEHAVIOUR
DEFINED AS
"The setupLinkConnection action request is sent by a managing system (e.g., the NMS) to the managed system (e.g., the EMS) to direct the sonetLink (and subclasses) to setup a link connection. Upon receipt of this request, the managed system creates a sonetLinkConnection object instance. Also, the agreedUserLabel is returned to the managing system. The agreedUserLabel will be the same as the suppliedUserLabel (if any), unless another label is required to satisfy the uniqueness criteria.";

4.10.11 setupSNC

setupSNC ACTION

```
    BEHAVIOUR      setupSNCBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        "M.3100:1995":generalErrorParameter;
    WITH INFORMATION SYNTAX SIFNLMMod.SetupSNCInformation;
    WITH REPLY SYNTAX SIFNLMMod.SetupSNCReply;
REGISTERED AS {sIFNLMAction 11};

setupSNCBehaviour BEHAVIOUR
    DEFINED AS
        "The setupSNC action request is sent by a managing system (e.g., the
NMS) to the managed system (e.g., the EMS) to direct the sonetSubnetwork
(and subclasses) to setup a sonetSubnetworkConnection.  Upon receipt of this
request, the managed system creates an sonetSubnetworkConnection object
instance and sets the related sNCPointers.  Also, the agreedUserLabel is
returned to the managing system.  The agreedUserLabel will be the same as the
suppliedUserLabel (if any), unless another label is required to satisfy the
uniqueness criteria.";
```

4.10.12 setupTrail

```
setupTrail              ACTION
    BEHAVIOUR      setupTrailBehaviour;
    MODE CONFIRMED;
    PARAMETERS
        "M.3100:1995":generalErrorParameter;
    WITH INFORMATION SYNTAX SIFNLMMod.SetupTrailInformation;
    WITH REPLY SYNTAX SIFNLMMod.SetupTrailReply;
REGISTERED AS {sIFNLMAction 12};

setupTrailBehaviour BEHAVIOUR
    DEFINED AS
        "The setupTrail action request is sent by a managing system (e.g., the
NMS) to the managed system (e.g., the EMS) to direct the
sonetLayerNetworkDomain (and subclasses) to setup a sonetTrail.  Upon receipt
of this request, the managed system creates a sonetTrail object instance.  Also,
the agreedUserLabel is returned to the managing system.  The agreedUserLabel
will be the same as the suppliedUserLabel (if any), unless another label is
required to satisfy the uniqueness criteria.";
```

4.11 Notifications

4.11.1 connectionConfigurationSummaryReport

```
connectionConfigurationSummaryReport NOTIFICATION
    BEHAVIOUR connectionConfigurationSummaryReportBehaviour;
    WITH INFORMATION SYNTAX
        SIFNLMMod.ConnectionConfigurationSummaryReportInformation;
REGISTERED AS {sIFNLMNotification 1};

connectionConfigurationSummaryReportBehaviour BEHAVIOUR
    DEFINED AS
        "This notification is sent by the managed system to the managing system
to report the status of setupSNC, releaseSNC, setupTrail, and releaseTrail
requests.  This notification is sent by the managed system at intervals
determined by the managementOperationsSchedule object instance.

    The information provided in this report includes: object instances of
sonetSubnetworkConnections and sonetTrails in the objectList attribute, an
indication of whether the sonetSubnetworkConnection or sonetTrail is being
established or released, and each sonetSubnetworkConnection's or sonetTrail's
associated percentage of completion or release.  The percentage of completed
or released cross-connections is used to denote percentage of completion or
release.";
```

4.11.2 rcaaNotification

```
rcaaNotification  NOTIFICATION
      BEHAVIOUR      rcaaNotificiationBehaviour;
      WITH INFORMATION SYNTAX SIFNLMMod.RcaaInfo
            AND ATTRIBUTE IDS
            probableCause                    probableCause,
            specificProblems                 specificProblems,
            perceivedSeverity                perceivedSeverity,
            backedUpStatus                   backedUpStatus,
            backUpObject                     backUpObject,
            trendIndication                  trendIndication,
            thresholdInfo                    thresholdInfo,
            notificationIdentifier           notificationIdentifier,
            correlatedNotifications          correlatedNotifications,
            stateChangeDefinition                stateChangeDefinition,
            monitoredAttributes              monitoredAttributes,
            proposedRepairActions            proposedRepairActions,
```

```
            additionalText                  additionalText,
            additionalInformation                   additionalInformation,
            alarmedObjectClass              alarmedObjectClass,
            alarmedObjectInstance           alarmedObjectInstance,
            faultLocation                   faultLocation,
            serviceAffecting                serviceAffecting,
            numberOfNELAlarms               numberOfNELAlarms,
            affectedTransmissionResources   affectedTransmissionResources,
            highestPriorityNELalarmRecords  highestPriorityNELalarmRecords;
REGISTERED AS   { sIFNLMNotification 2};

rcaaNotificiationBehaviour BEHAVIOUR
    DEFINED AS
        "The RCAA Notification is generated by the root cause alarm analysis
routine based on one or more NEL alarm event reports.  It will contain all the
fields which are contained in a alarm record as defined in X.733. The
notificationIdentifier parameter shall be required.  The correlatedNotification
parameter shall be included if this RCAA Notification is a update or a clear of an
earlier RCAA Notification.  The Event Time shall be included and will be set to
the time of the arrival of the first NEL alarm which is correlated to this RCAA
Notification.

        In addition the alarmedObjectClass, alarmedObjectInstance,
faultLocation, serviceAffecting, numberOfNELAlarms,
affectedTransmissionResources and the higherPriorityNELalarmRecords
parameters shall be included. These parameters shall have the same values and
behaviors as the associated attributes in the RCAArecord.";
```

4.12 Name Bindings

4.12.1 sonetAccessGroup-sonetLayerNetworkDomain

```
sonetAccessGroup-sonetLayerNetworkDomain NAME BINDING
    SUBORDINATE OBJECT CLASS sonetAccessGroup AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND
    SUBCLASSES;
    WITH ATTRIBUTE sonetAccessGroupId;
    BEHAVIOUR sonetAccessGroup-sonetLayerNetworkDomain;
    CREATE
        WITH-REFERENCE-OBJECT,
        WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE
        ONLY-IF-NO-CONTAINED-OBJECTS;
```

REGISTERED AS {sIFNLMNameBinding 1};

sonetAccessGroup-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetAccessGroup are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.";

4.12.2 sonetConnectionPendingStatusControl-sonetLayerNetworkDomain

sonetConnectionPendingStatusControl-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetConnectionPendingStatusControl AND SUBCLASSES;
NAMED BY
SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND
SUBCLASSES;
WITH ATTRIBUTE sonetConnectionPendingStatusControlId;
BEHAVIOUR
sonetConnectionPendingStatusControl-sonetLayerNetworkDomain;
CREATE
WITH-REFERENCE-OBJECT ,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 2};

sonetConnectionPendingStatusControl-sonetLayerNetworkDomain
BEHAVIOUR
DEFINED AS
"Instances of sonetConnectionPendingStatusControl are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.";

4.12.3 sonetConnectionPendingStatusControl-networkR1

sonetConnectionPendingStatusControl-networkR1 NAME BINDING
SUBORDINATE OBJECT CLASS
sonetConnectionPendingStatusControl AND SUBCLASSES;
NAMED BY
SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1 AND
SUBCLASSES;

WITH ATTRIBUTE sonetConnectionPendingStatusControlId;
BEHAVIOUR sonetConnectionPendingStatusControl-networkR1;
CREATE
WITH-REFERENCE-OBJECT ,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 3};

sonetConnectionPendingStatusControl-networkR1 BEHAVIOUR
DEFINED AS
"Instances of sonetConnectionPendingStatusControl are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.";

4.12.4 sonetLayerNetworkDomain-networkR1

sonetLayerNetworkDomain-networkR1 NAME BINDING
SUBORDINATE OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1 AND SUBCLASSES;
WITH ATTRIBUTE sonetLayerNetworkDomainId;
BEHAVIOUR sonetLayerNetworkDomain-networkR1;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 4};

sonetLayerNetworkDomain-networkR1 BEHAVIOUR
DEFINED AS
"Instances of sonetLayerNetworkDomain are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.";

4.12.5 sonetLink-sonetLayerNetworkDomain

sonetLink-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetLink AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
WITH ATTRIBUTE sonetLinkId;

BEHAVIOUR sonetLink-sonetLayerNetworkDomain;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 5};

sonetLink-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetLink are created and deleted by the managing system using the addLink and removeLink actions on the sonetLayerNetworkDomain superior object. Some instances of this class may be automatically instantiated at initialization of the EMS. Some may also be automatically created or deleted by the EMS as a side effect of the creation or deletion of trails in server layers.";

4.12.6 sonetLinkConnection-sonetLink

sonetLinkConnection-sonetLink NAME BINDING
SUBORDINATE OBJECT CLASS sonetLinkConnection AND
SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLink AND SUBCLASSES;
WITH ATTRIBUTE sonetTransportEntityId;
BEHAVIOUR sonetLinkConnection-sonetLink;
REGISTERED AS {sIFNLMNameBinding 6};

sonetLinkConnection-sonetLink BEHAVIOUR
DEFINED AS
"Instances of sonetLinkConnection are created and deleted by the managing system using the setUpLinkConnection and releaseLinkConnection actions on the superior link object. Some instances of this class may be automatically instantiated at initialization of the EMS. Some may also be automatically created or deleted by the EMS as a side effect of the creation or deletion of trails in server layers.";

4.12.7 eventForwardingDiscriminator-networkR1

eventForwardingDiscriminator-networkR1 NAME BINDING
SUBORDINATE OBJECT CLASS "Rec. X.721 | ISO/IEC 10165-2:1992":
eventForwardingDiscriminator AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1
AND SUBCLASSES;
WITH ATTRIBUTE discriminatorId;
CREATE
WITH-REFERENCE-OBJECT ,

WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 7};

4.12.8 log-networkR1

log-networkR1 NAME BINDING
SUBORDINATE OBJECT CLASS "Rec. X.721 | ISO/IEC 10165-2:1992": log
AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1
AND SUBCLASSES;
WITH ATTRIBUTE logId;
CREATE
WITH-REFERENCE-OBJECT ,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 8};

4.12.9 sonetAlarmAnalysisRoutine-networkR1

sonetAlarmAnalysisRoutine-networkR1 NAME BINDING
SUBORDINATE OBJECT CLASS sonetAlarmAnalysisRoutine AND
SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1
AND SUBCLASSES;
WITH ATTRIBUTE sonetAlarmAnalysisRoutineId;
BEHAVIOUR sonetAlarmAnalysisRoutine-networkR1;
CREATE
WITH-REFERENCE-OBJECT ,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 9};

sonetAlarmAnalysisRoutine-networkR1 BEHAVIOUR
DEFINED AS
“Instances of sonetAlarmAnalysisRoutine are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.“;

4.12.10 alarmSeverityAssignmentProfile-networkR1

```
alarmSeverityAssignmentProfile-networkR1      NAME  BINDING
    SUBORDINATE  OBJECT  CLASS "Rec.
    M.3100:1995":alarmSeverityAssignmentProfile
    AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1
    AND SUBCLASSES;
    WITH ATTRIBUTE alarmSeverityAssignmentProfileId;
    CREATE
        WITH-REFERENCE-OBJECT,
        WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE
        ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS   {sIFNLMNameBinding 10};
```

4.12.11 eventForwardingDiscriminator-sonetLayerNetworkDomain

```
eventForwardingDiscriminator-sonetLayerNetworkDomain NAME  BINDING
    SUBORDINATE  OBJECT  CLASS "Rec. X.721 | ISO/IEC 10165-2:1992":
        eventForwardingDiscriminator AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND
    SUBCLASSES;
    WITH ATTRIBUTE discriminatorId;
    CREATE
        WITH-REFERENCE-OBJECT ,
        WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE
        ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS   {sIFNLMNameBinding 11};
```

4.12.12 log-sonetLayerNetworkDomain

```
log-sonetLayerNetworkDomain   NAME  BINDING
    SUBORDINATE OBJECT CLASS "Rec. X.721 | ISO/IEC 10165-2:1992": log
AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND
    SUBCLASSES;
    WITH ATTRIBUTE  logId;
    CREATE
        WITH-REFERENCE-OBJECT ,
        WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE
```

ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 12};

4.12.13 sonetAlarmAnalysisRoutine-sonetLayerNetworkDomain

sonetAlarmAnalysisRoutine-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetAlarmAnalysisRoutine AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
WITH ATTRIBUTE sonetAlarmAnalysisRoutineId;
BEHAVIOUR sonetAlarmAnalysisRoutine-sonetLayerNetworkDomain;
CREATE
WITH-REFERENCE-OBJECT ,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 13};

sonetAlarmAnalysisRoutine-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetAlarmAnalysisRoutine are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.";

4.12.14 alarmSeverityAssignmentProfile-sonetLayerNetworkDomain

alarmSeverityAssignmentProfile-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS "Rec. M.3100:1995":alarmSeverityAssignmentProfile
AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain
AND SUBCLASSES;
WITH ATTRIBUTE alarmSeverityAssignmentProfileId;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 14};

4.12.15 managementOperationsSchedule-sonetLayerNetworkDomain

managementOperationsSchedule-sonetLayerNetworkDomain NAME BINDING

SUBORDINATE OBJECT CLASS
"Rec. Q.821":managementOperationsSchedule AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
WITH ATTRIBUTE managementOperationsScheduleId;
CREATE;
DELETE
DELETES-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 15};

4.12.16 managementOperationsSchedule-networkR1

managementOperationsSchedule-networkR1 NAME BINDING
SUBORDINATE OBJECT CLASS
"Rec. Q.821":managementOperationsSchedule AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS "Rec. M.3100:1995":networkR1 AND SUBCLASSES;
WITH ATTRIBUTE managementOperationsScheduleId;
CREATE
DELETE
DELETES-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 16};

4.12.17 sonetNetworkCTP-sonetLayerNetworkDomain

sonetNetworkCTP-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetNetworkCTP AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
WITH ATTRIBUTE sonetNetworkCTPId;
BEHAVIOUR sonetNetworkCTP-sonetLayerNetworkDomain;
REGISTERED AS {sIFNLMNameBinding 17};

sonetNetworkCTP-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetNetworkCTP (subclasses of) are created and deleted by the managing system using the addLinkCapacity and removeLinkCapacity actions on associated link or sonetSNTPPool objects. Instances of this class (subclasses of) can also be created and deleted by the EMS (for example, as a consequence of operations at the element level or in server layers).";

4.12.18 sonetNetworkTTP-sonetLayerNetworkDomain

sonetNetworkTTP-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetNetworkTTP AND
SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND
SUBCLASSES;
WITH ATTRIBUTE sonetNetworkTTPId;
BEHAVIOUR sonetNetworkTTP-sonetLayerNetworkDomain;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
DELETES-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 18};

sonetNetworkTTP-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetNetworkTTP (subclasses of) can be created and deleted by the managing system. Instances of this class (subclasses of) can also be created and deleted by the EMS (for example, as a consequence of operations at the element level or in server layers).";

4.12.19 sonetRoutingProfile-sonetLayerNetworkDomain

sonetRoutingProfile-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetRoutingProfile AND
SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND
SUBCLASSES;
WITH ATTRIBUTE sonetRoutingProfileId;
BEHAVIOUR sonetRoutingProfile-sonetLayerNetworkDomain;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
DELETES-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 19};

sonetRoutingProfile-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetRoutingProfile are created and deleted automatically by the EMS as a result of the setupSNC, setupTrail and releaseSNC, releaseTrail actions. Some instances of this class may be automatically

instantiated at initialization of the EMS. Some may also be created and deleted directly by the managing system.“;

4.12.20 sonetRoutingProfile-sonetSubnetwork

```
sonetRoutingProfile-sonetSubnetwork NAME BINDING
    SUBORDINATE OBJECT CLASS sonetRoutingProfile AND
    SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS sonetSubnetwork AND
    SUBCLASSES;
    WITH ATTRIBUTE sonetRoutingProfileId;
    BEHAVIOUR sonetRoutingProfile-sonetSubnetwork;
    CREATE
        WITH-REFERENCE-OBJECT,
        WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE
        DELETES-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 20};
```

sonetRoutingProfile-sonetSubnetwork BEHAVIOUR
DEFINED AS
“Instances of sonetRoutingProfile are created and deleted automatically by the EMS as a result of the setupSNC, setupTrail and releaseSNC, releaseTrail actions. Some instances of this class may be automatically instantiated at initialization of the EMS. Some may also be created and deleted directly by the managing system.“;

4.12.21 sonetSubnetworkConnection-sonetSubnetwork

```
sonetSubnetworkConnection-sonetSubnetwork NAME BINDING
    SUBORDINATE OBJECT CLASS sonetSubnetworkConnection AND
    SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS sonetSubnetwork AND
    SUBCLASSES;
    WITH ATTRIBUTE sonetTransportEntityId;
    BEHAVIOUR sonetSubnetworkConnection-sonetSubnetwork;
REGISTERED AS {sIFNLMNameBinding 21};
```

sonetSubnetworkConnection-sonetSubnetwork BEHAVIOUR
DEFINED AS
“Instances of sonetSubnetworkConnection are created and deleted by the managing system using the setupSNC and releaseSNC actions on superior

subnetwork object. Some instances of this class may be automatically instantiated at initialization of the EMS.“;

4.12.22 sonetSNTP-sonetSubnetwork

sonetSNTP-sonetSubnetwork NAME BINDING
SUBORDINATE OBJECT CLASS sonetSNTP AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetSubnetwork AND SUBCLASSES;
WITH ATTRIBUTE sonetSNTPId;
BEHAVIOUR sonetSNTP-sonetSubnetwork;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
DELETES-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 22};

sonetSNTP-sonetSubnetwork BEHAVIOUR
DEFINED AS
“Instances of sonetSNTP are created and deleted by the managing system using the addPoolCapacity and removePoolCapacity actions on a sonetSNTPPool object. Some instances of this class may be automatically instantiated at initialization of the EMS. Some may also be created and deleted by the EMS (for example, as a consequence of operations at the element level or in server layers).“;

4.12.23 sonetSNTPPool-sonetSubnetwork

sonetSNTPPool-sonetSubnetwork NAME BINDING
SUBORDINATE OBJECT CLASS sonetSNTPPool AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetSubnetwork AND SUBCLASSES;
WITH ATTRIBUTE sonetSNTPPoolId;
BEHAVIOUR sonetSNTPPool-sonetSubnetwork;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 23};

sonetSNTPPool-sonetSubnetwork BEHAVIOUR
DEFINED AS

"Instances of sonetSNTPPool are created and deleted by the managing system. Instances of this class may also be automatically instantiated at initialization of the EMS.";

4.12.24 sonetSubnetwork-sonetLayerNetworkDomain

sonetSubnetwork-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetSubnetwork AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
WITH ATTRIBUTE sonetSubnetworkId;
BEHAVIOUR sonetSubnetwork-sonetLayerNetworkDomain;
CREATE
WITH-REFERENCE-OBJECT,
WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE
ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {sIFNLMNameBinding 24};

sonetSubnetwork-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetSubnetwork are created and deleted by the managing system. Some instances of this class may be automatically instantiated at initialization of the EMS.";

4.12.25 sonetTrail-sonetLayerNetworkDomain

sonetTrail-sonetLayerNetworkDomain NAME BINDING
SUBORDINATE OBJECT CLASS sonetTrail AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS sonetLayerNetworkDomain AND SUBCLASSES;
WITH ATTRIBUTE sonetTransportEntityId;
BEHAVIOUR sonetTrail-sonetLayerNetworkDomain;
REGISTERED AS {sIFNLMNameBinding 25};

sonetTrail-sonetLayerNetworkDomain BEHAVIOUR
DEFINED AS
"Instances of sonetTrail are created and deleted by the managing system using the addTrail and releaseTrail actions on sonetLayerNetworkDomain superior object. Some instances of this class may be automatically instantiated at initialization of the EMS. Some may also be automatically created or deleted by the EMS as a side effect of the creation or deletion of links in the same

layer.“;

4.13 Supporting Productions

```
SIFNLMMod -- {}
DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS everything

IMPORTS

ProbableCause,SpecificProblems, PerceivedSeverity, BackedUpStatus,
TrendIndication, ThresholdInfo, NotificationIdentifier,
CorrelatedNotifications,  MonitoredAttributes,
ProposedRepairActions,  AdditionalText,  AdditionalInformation,
AttributeValueChangeDefinition
FROM
Attribute-ASN1Module {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1}

Boolean,
ChannelNumber,
CharacteristicInformation,   -- This will be removed in next release of document
Directionality,
LocationName,
NameType,
RelatedObjectInstance,
UserLabel
FROM
ASN1DefinedTypesModule {ccitt recommendation m(13) gnm(3100)
informationModel(0) asn1Modules(2) asn1DefinedTypesModule(0)}

DistinguishedName
FROM
InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)}

Attribute,
AttributeId,
ObjectClass,
ObjectInstance
FROM
CMIP-1 {joint-iso-ccitt ms(9) cmip(1) version1(1) protocol(3)};
```

```
-- This section will be removed when these values are included in GR-836-IMD
-- and/or GR-1042-IMD

-- Characteristic Information OBJECT IDENTIFIERs

sIFNLMSpecificExtension OBJECT IDENTIFIER ::= {}

sIFNLMCharacteristicInformation OBJECT IDENTIFIER ::=
{sIFNLMSpecificExtension}

electricalSTS1SPICI CharacteristicInformation ::=
{sIFNLMCharacteristicInformation 1}

opticalSTM64SPICI CharacteristicInformation ::=
{sIFNLMCharacteristicInformation 2}

rsSTM64SPICI CharacteristicInformation ::= {sIFNLMCharacteristicInformation
3}

msSTM64SPICI CharacteristicInformation ::= {sIFNLMCharacteristicInformation
4}

au44cCI CharacteristicInformation ::= {sIFNLMCharacteristicInformation 5}

ds3LineCI CharacteristicInformation ::= {sIFNLMCharacteristicInformation 6}

ds3PathCI CharacteristicInformation ::= {sIFNLMCharacteristicInformation 7}

ds1LineCI CharacteristicInformation ::= {sIFNLMCharacteristicInformation 8}

ds1PathCI CharacteristicInformation ::= {sIFNLMCharacteristicInformation 9}

-- End CharacteristicInformation Section

-- module definitions

AdditionalCapacity ::= Integer

AddLinkInformation ::= SEQUENCE {
    aEndSNTPPools       EndList,
    zEndSNTPPools       EndList,
    suppliedUserLabel   UserLabel
    }
```

```
AddLinkReply ::= SEQUENCE {
    link                ObjectInstance,
    agreedUserLabel     UserLabel
    }

AddLinkCapacityInformation ::= AdditionalCapacity

AddLinkCapacityReply ::= SEQUENCE {
    aEndCTPs        SET OF ObjectInstance,
    zEndCTPs        SET OF ObjectInstance,
    aEndSNTPs       SET OF ObjectInstance,
    zEndSNTPs       SET OF ObjectInstance
    }

AddPoolCapacityInformation ::= AdditionalCapacity

AddPoolCapacityReply ::= SEQUENCE {
    cTPs        SET OF ObjectInstance,
    sNTPs       SET OF ObjectInstance
    }

AffectedTransmissionResources ::= SET OF ObjectInstance

AlarmRecords ::= SET OF ObjectInstance

Boolean::= BOOLEAN

ChannelIdentifier ::= SEQUENCE {
    groupNumber [0] GroupNumber OPTIONAL,
    channelNumber [1] ChannelNumber
    }

ClientNCTPs ::= SET OF ObjectInstance

ComponentList ::= SET OF ObjectInstance

ConfigRequestType ::= ENUMERATED {
    setup (0),
    release (1)
    }

ConfigurationState ::= ENUMERATED {
    preService (0),
```

```
    inService (1),
    postService (2),
    configurationFailure (3)
    }

ConfigurationStatus ::= SEQUENCE {
    sNCorTrail                              [0]     ObjectInstance,
    configRequestType                       [1]     ConfigRequestType,
    percentCompletedOrReleased      [2]     Integer
    }

ConnectionConfigurationSummaryReportInformation     ::=     SET     OF
ConfigurationStatus

ContainingSNTPPool ::= RelatedObjectInstance

CTP ::= RelatedObjectInstance

EndList ::= SET OF ObjectInstance

EndPoint ::= CHOICE {
    terminationPoint                [0]             ObjectInstance,
    sNTPPoolorAccessGroup   [1]             ObjectInstance
    }


Format      ::= OBJECT IDENTIFIER

GroupNumber ::= INTEGER

Integer ::= INTEGER

LinkPointer ::= RelatedObjectInstance

FaultLocation ::= SET OF LocationName

MaxHops ::= CHOICE {
    notApplicable   NULL,
    maxHops INTEGER
    }

RcaaInfo ::=  SEQUENCE {
     probableCause              ProbableCause,
      specificProblems                  [1]SpecificProblems OPTIONAL,
```

```
        perceivedSeverity               PerceivedSeverity,
        backedUpStatus                  BackedUpStatus  OPTIONAL,
        backUpObject                    [2]ObjectInstance  OPTIONAL,
        trendIndication                 [3]TrendIndication  OPTIONAL,
        thresholdInfo                   [4]ThresholdInfo   OPTIONAL,
        notificationIdentifier          [5]NotificationIdentifier  OPTIONAL,
        correlatedNotifications         [6]CorrelatedNotifications,
        stateChangeDefinition           [7]AttributeValueChangeDefinition
OPTIONAL,
        monitoredAttributes             [8]MonitoredAttributes  OPTIONAL,
        proposedRepairActions           [9]ProposedRepairActions OPTIONAL,
        additionalText                  AdditionalText  OPTIONAL,
        additionalInformation           [10]AdditionalInformation OPTIONAL,
      alarmedObjectClass          ObjectClass,
      alarmedObjectInstance             ObjectInstance,
      faultLocation               FaultLocation,
      serviceAffecting            Boolean,
      numberOfNELAlarms           Integer,
      affectedTransmissionResources         AffectedTransmissionResources,
      highestPriorityNELalarmRecords        AlarmRecords }

ReflectedTP ::= ObjectInstance

ReleaseLinkConnectionInformation ::=   UserLabel

ReleaseSNCInformation ::= UserLabel

ReleaseTrailInformation ::= UserLabel

RemoveLinkInformation ::= UserLabel

RemoveLinkCapacityInformation ::= Integer

RemovePoolCapacityInformation ::= Integer

RouteDescription ::= SEQUENCE {
    referenceObject         ObjectInstance,
    option                  RoutingOption
    }

RouteDescriptionList ::= SEQUENCE OF RouteDescription

RoutingInfo ::= CHOICE {
```

```
    routeDescriptionList [0] RouteDescriptionList,
    routingProfilePointer [1]       ObjectInstance
    }

RoutingOption ::= ENUMERATED {
    mandatory1 (0), -- must use the object in establishing the connection NOT ACCEPTING mandatory (may be keyword)
    preferred (1), -- attempt to use the object in establishing the connection
    exclude (2), -- do not use the object in establishing the connection
    sameRoute (3), -- use the same route as the reference object
    diverseRoute (4) -- use different route as the reference object
    }

RoutingProfile ::= RelatedObjectInstance

ServerNTTP ::= RelatedObjectInstance

SetupLinkConnectionInformation ::= SEQUENCE {
    aEndCTP         ObjectInstance      OPTIONAL,
    zEndCTP         ObjectInstance      OPTIONAL,
    directionality      Directionality,
    suppliedUserLabel       UserLabel
    }

SetupLinkConnectionReply ::= SEQUENCE {
    linkConnection   ObjectInstance,
    agreedUserLabel         UserLabel,
    aEndCTP ObjectInstance,
    zEndCTP ObjectInstance
    }

SetupSNCInformation ::= SEQUENCE {
    sNTPA           EndPoint,
    sNTPZs   SET OF EndPoint,
    directionality      Directionality,
    routingInfo         RoutingInfo   OPTIONAL,
    suppliedUserLabel       UserLabel
    }

SetupSNCReply ::= SEQUENCE {
    sNC         ObjectInstance,
    sNTPA     ObjectInstance,
    sNTPZs SET OF ObjectInstance,
    agreedUserLabel UserLabel
```

```
    }

SetupTrailInformation ::= SEQUENCE {
    tTPA        EndPoint,
    tTPZs       SET OF EndPoint,
    directionality      Directionality,
    routingInfo         RoutingInfo   OPTIONAL,
    suppliedUserLabel       UserLabel
    }

SetupTrailReply ::= SEQUENCE {
    trail       ObjectInstance,
    tTPA        ObjectInstance,
    tTPZs       SET OF ObjectInstance,
    agreedUserLabel         UserLabel
    }

SNCPointer ::= RelatedObjectInstance

SNTP ::= ObjectInstance

SNTPList ::= SET OF ObjectInstance

SNTPs ::= SET OF ObjectInstance

SubnetworkType ::= ENUMERATED {
    tSIADM (0),
    tSAADM (1),
    hardwiredADM (2),
    terminalMux (3),
    dCS (4),
    otherNE (5),
    uPSR (6),
    bLSR (7),
    linearChain (8),
    pointtopoint (9),
    otherSubnetwork (10)
    }

TPConnectionState ::= ENUMERATED {
    notConnected (0),
    sinkConnected (1),
    sourceConnected (2),
```

```
    bidirectionallyConnected (3)
    }

TTP ::= ObjectInstance

TTPList ::= SET OF ObjectInstance

TTPs ::= SET OF ObjectInstance
END
```