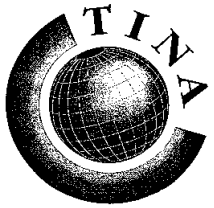


Telecommunications
Information
Networking
Architecture
Consortium

TINA-C Deliverable

Issue Status: Publicly Released

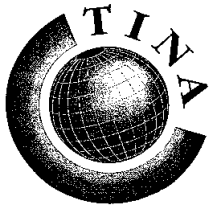
Network Resource Architecture

Version: 3.0

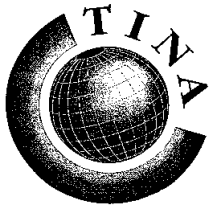
Date of Issue: February 1997

Table 1:

Page	Company	Company Contact (Address, Telephone, Fax)



Telecommunications
Information
Networking
Architecture
Consortium



Telecommunications
Information
Networking
Architecture
Consortium

Network Resource Architecture
Version 3.0

Abstract: This document specifies basic seperations used in the control and management of TINA Network Resource functions. A reusable group of technology independent components is described to perform these functions. The work is part of the TINA-C specification effort and as such an element in the TINA-C architecture. This document is intended to replace the 1994 Connection Management Architecture Baseline.

Keywords: Network Resource, Connection Management, Open Distributed Processing, Telecommunication Management Network, Managed Objects, G.803 Functional Architecture.

Author(s): **Chelo Abarca, Jan Forslow, Takeo Hamada, Stephanie Hogg, Hong Beom Jeon, Dae Seok Kim, Hahn Young Lee, Narayanan Natarajan, Frank Steegmans**

Editor: **Frank Steegmans**

Type: **TINA-C Baseline**

Document Label: **NRA_v3.0_97_02_10**

Date: **February 10, 1997**

File: **/u/tinac/96/resources/viewable/nra_v3.0.ps**
also available via http://www.tinac.com/

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Purpose

The purpose of this document is to show the functionality of setting up, maintaining and releasing telecommunication connections, as well as managing these resources in a TINA-C consistent architecture. The document describes the server functions and how its clients (DPE, service and management applications) can make use of it. It also describes a generic network information model that supports the Network Resource Architecture.

This document provides a view of the current status of the work in TINA-C core team in the subject of Network Resource Management as by December 1996. It is expected that future versions of the document will answer the open issues identified in Section 8.

1.2 Audience

This document should be read by those concerned with the specification, design and implementation of communication networks and services in a TINA environment.

1.3 Document History

This document is based on the 'Connection Management Architecture' (CMA) of 1994 [13], which was in its turn based on Version 1.0 released in December 1993. While the CMA was mainly focussing on the management of telecommunication connections, this document, the Network Resource Architecture (NRA), also covers the other management area's specified in TMN.

Just as the CMA of 1994 incorporated the work done in 1994, this document tries to capture the work done in 1995 and 1996 in the Network Resource area.

1.4 How to Read This Document

This document is the first to be released of a series of documents that covers the work done in 1995 and 1996. Rather than waiting for the other documents to be finished this document is early released so that feedback can be carried forward into the other documents. As a result, this version contains material that might be better suited for other types of documents than an architecture. This material will also be covered in the other documents that will be produced later on (Network Resource Information Model, Network Components Specification, etc.). In the next version of this document it will be dropped and just referred to[1]. This version is intended to provide the reader with as much 'new' material as possible. Material that did not significantly change, is not yet revised but will be in the next release. It is temporarily put in the appendices.

1. Which material will be covered in other documents and dropped from the NRA will depend on the new documents. Therefore no specific list is presented.

Readers are friendly requested to focus on the content of this document rather than on the form. The authors are aware of a number of inconsistencies and minor flaws for which they apologize. These will also be cleared out in the next release.

1.5 Prerequisites

This document assumes the reader is familiar with the following:

- Overall Concepts and Principles for TINA [3];
- TINA Management Architecture [7].

Having read the following documents is not essential in order to understand the NRA document but it will make reading much easier:

- Information Modelling Concepts [10];
- Computational Modelling Concepts [4];
- Engineering Modelling Concepts [6].

Related reading material (currently outdated, but a new release is planned):

- Network Resource Information Model [1];
- Network Component Specification [2].

1.5.1 Document Organization

This document is structured as follows:

- Section 2 “Business Model and Network Resource Architecture” situates the the Network Resource Architecture in the bigger ‘TINA Business Model’ picture;
- Section 3 “Overview of a TINA network” presents a brief resource model of a TINA network from the perspective of control and management of the network;
- Section 4 “Connection Management” describes the connection management architecture from all TINA viewpoints;
- Section 5 “Configuration Management” describes how the topological configuration is managed in a TINA network;
- Section 6 “Fault Management” describes how fault management is implemented in aTINA network;
- Section 7 “Accounting Management” describes how accounting management is implemented in aTINA network.

Section 2 and Section 3 introduce the concepts of the NRA and are consequently considered to be descriptive. Prescriptive parts in Section 4 are indicated in the chapter on a case by case basis. Section 5, Section 6 and Section 7 contain brand new and maybe controversial material. Therefore they are descriptive in this release. Dependent on the feedback, prescriptive parts will be introduced in the next release.

1.6 Main Inputs

Main inputs for the work in the Network Resource Architecture are the following:

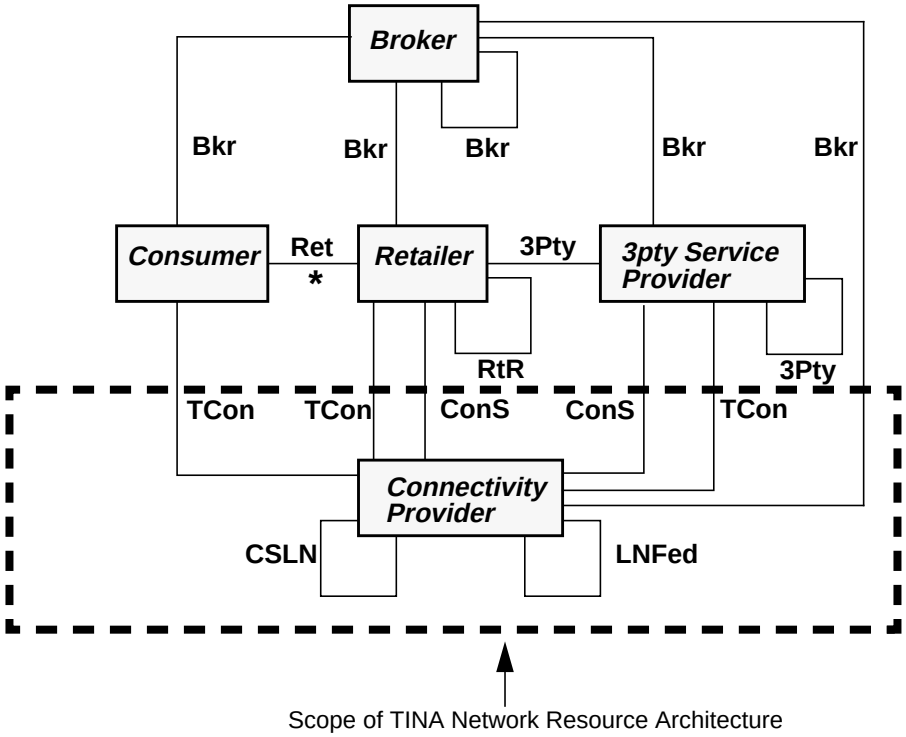
- Generic management principles from existing standards on TMN and OSI management (see [7]);
- Network Information Model descriptions from existing standards like ITU-TS G.803 “Architecture of Transport Networks Based on the SDH” [28], and M.3100 “Generic Network Information Model” [29];
- 1992 INA results of Communication Management Architecture [25];
- Core-team response to the RFR/S on the Connectivity Service Reference Point (Cons-RP) [21]. The NRA is updated towards the unified response, but is not completely compliant because of a number of technical reasons outside the scope of this discussion;
- Core-team response to the RFR/S on the Terminal Connectivity Reference Point (Tcon-RP) [22];
- Core-team response to the RFR/S on the Retailer Reference Point (Ret-RP) [23];

2. Business Model and Network Resource Architecture

The objective of this section is to describe the scope of the TINA Network Resource Architecture from the perspective of the TINA Business Model and the TINA Reference Points described in the TINA Reference Points document [16]. For the reference points under its scope, the network resource architecture provides the foundation for the detailed specification of the reference points.[1] The discussion in this section assumes familiarity with the business model concepts described in the TINA Reference Points document [16].

2.1 Network Resource Architecture and Inter-Domain Reference Points

Figure 2-1 illustrates the scope of the TINA Network Resource Architecture in terms of TINA inter-domain reference points. (Figure 2-1 is adapted from [16].)



*Note: The interactions between Consumer and Retailer for the management of terminal flow connections are also under the scope of the network resource architecture.

Figure 2-1. Scope of Network Resource Architecture in terms of Inter-Domain Reference Points

1. It should be noted that reference point specifications also draw upon concepts from the TINA Service Architecture and the TINA Computing Architecture.

As shown in the figure, the scope of the Network Resource Architecture covers the following inter-domain reference points:

- *Connectivity Service Reference Point (ConS-RP)*: This inter-domain reference point comprises of interfaces for interactions between a TINA stakeholder in the Connectivity Provider business and a TINA stakeholder that is either in the Retailer business or in the Third-Party Service Provider business. The connectivity provider provides, to a retailer or a third-party service provider, capabilities for the establishment, release, modification, and management of *network flow connections*. The network that a network flow connection spans is called a *connectivity layer network* and it is made up of one or more *layer networks*. (See Section 3.)
- *Terminal Connection Reference Point (TCon-RP)*: This inter-domain reference point comprises of interfaces for interactions between a TINA stakeholder in the Connectivity Provider business and a TINA stakeholder that is either in the Consumer business, the Retailer business, or the Third-Party Service Provider business. The latter three businesses are together referred to as *connectivity consumer*s. TCon-RP arises in configurations where a connectivity consumer is responsible for the management of *network flow end points* (end points of a network flow connection) including link connections on the access links of a layer network. In such cases, TCon-RP comprises of interfaces for interactions between a connectivity provider and a connectivity consumer concerning establishment, release, and modification of network flow endpoints and access link connections. See the TCon Reference Point RFR for details [22].
- *Layer Network Federation Reference Point (LNFed-RP)*: A transport connection across a layer network is called a *trail*. (See Section 3.) It is possible that a layer network consists of several portions, each portion belonging to a different administrative domain. The LNFed-RP occurs in such configurations. In such cases, management of a trail (setup, release, and modification) involves cooperation between the different connectivity provider stakeholders. LnFed-RP comprises of interfaces for interactions between two connectivity provider stakeholders towards the provision of trails that span their administrative domains.
- *Client-Server Layer Network Reference Point (CSLN-RP)*: This reference point is a consequence of the client-server relationship that may exist between two layer networks. This relationship occurs when the transport services of one layer network (the server layer) are used to configure topological links in the other layer network (the client layer). See Section 3. CSLN-RP occurs in configurations where the client layer network and the server layer network belong to different administrative domains. CSLN-RP comprises of interfaces for interactions between the two connectivity provider stakeholders towards the management of links in the client layer network using trails in the server layer network.

2.2 Network Resource Architecture and Intra-Domain Reference Points

The intra-domain reference points under the scope of the network resource architecture are listed below:

- *End-to-End Communication Reference Point (EECom-RP)*: This is an intra-domain reference point in the administrative domain of a retailer or a third-party service provider stakeholder. This reference point deals with interactions between objects that provide *communication session* capabilities and objects that use these capabilities to realize generic and/or service specific session functions. The communication session capabilities associated with this reference point deal with establishment, release, modification, and management of bindings between *stream interfaces*. See Section 4 for details.
- *Network Management Layer Reference Point (NML-RP)*: This is an intra-domain reference point in the administrative domain of a connectivity provider stakeholder. The portion of a layer network in the administrative domain of a connectivity provider stakeholder consists of a top level subnetwork that is further partitioned into one or more lower level subnetworks. This partitioning may occur in several levels, and the lowest level subnetwork consists of an individual network element. The NML-RP deals with capabilities for managing a subnetwork at some level (other than the lowest level). The scope of NML-RP is similar to that of M4 Network View that is currently being defined by the ATM Forum.
- *Element Management Layer Reference Point (EML-RP)*: This is an intra-domain reference point in the administrative domain of a connectivity provider stakeholder.The EML-RP deals with capabilities for managing the lowest level subnetwork that consists of an individual network element. The scope of EML-RP is similar to that of M4 Network Element View that has been defined by the ATM Forum.

3. Overview of the TINA Network Modeling Concepts

This section presents an overview of the TINA network model. The topological and connectivity concepts of the TINA network model are briefly described. The objective of this discussion is to present a network resource model from the perspective of control and management of the network. This section presents only a brief description of the TINA network resource model. A detailed definition of the information model for these resources is presented in the TINA Network Resource Information Model (NRIM) document [1].

3.1 Overall Structure of a TINA Network

A network modeled by the TINA Network Resource Architecture, hereafter referred to as a TINA Network, is a transport network that is capable of transporting multimedia information. The information traffic carried by the network will be heterogeneous in terms of data formats, bandwidth requirements, and other Quality of Service (QoS) characteristics. The network traffic in a TINA network can consist of inter-related multimedia streams, and the TINA network transports such traffic ensuring stream synchronization. The application level end points (TINA logical view) in the TINA network model are stream interfaces associated with TINA applications or multimedia devices.[1] See Figure 3-1.

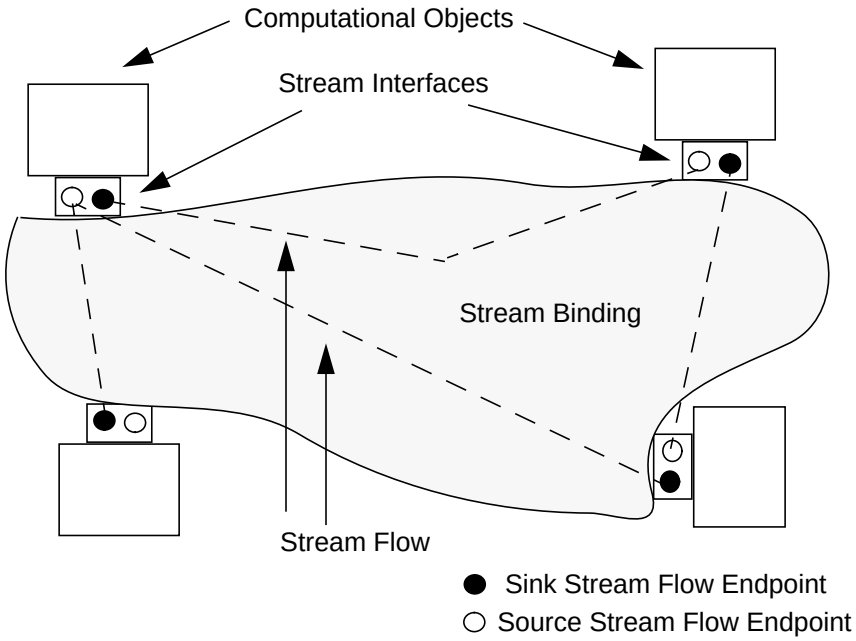


Figure 3-1. TINA Network (Logical View)

1. The Kernel Transport Network that transports request and reply messages for computational objects interaction via operational interfaces is a higher level abstraction that can be modeled using the TINA network resource model.

Figure 3-1 represents the logical view (application level, end-to-end view) of the TINA network model. Connectivity resources at this level are called *Stream Flow*s (SF). A stream flow is bounded by two or more *Stream Flow End Point*s (SFEP). An SFEP is either an information source or sink, but not both (i.e., SFEPs are unidirectional). A source SFEP can be bound to one or more sink SFEPs (providing point-to-multipoint connectivity). *Stream Interface* (SI) is a modeling construct that aggregates SFEPs belonging to an application or a multimedia device. A *Stream Binding* is a modeling concept that represents a collection of stream flows that have been grouped together for some purpose at the application level.

An SFEP can terminate only one stream flow. Associated with a stream flow end point is the characteristic information accepted/delivered at that SFEP. These properties include frame structure identification, QoS, etc. The frame structure and QoS of the source and sink stream flow end points bound by a stream flow need not be identical but must be compatible[2].

In the physical view, a TINA network is divided into two main components: one is the *Connectivity Layer Network* (CLN) and the other is formed by the communication resources contained in *Customer Premises Equipment* (CPE), see Figure 3-2. A CPE may be either a simple terminal device (telephone or multimedia device) or a computing system in which TINA compliant applications are deployed. A connectivity layer network is a transport network consisting of a heterogeneous collection of switching resources, transmission resources, and adapters. A connectivity layer network is made up of components of different technologies, such as ATM, Frame Relay, narrow band ISDN, wireless, SDH, or PDH, and is capable of transporting different types of information (Figure 3-2 shows some of the possible technologies).[3] Note that it is possible that a CPE is attached to several such networks.

A communication endpoint at which a connectivity layer network accepts or delivers information is called a *Network Flow End Point* (NFEP), see Figure 3-3. From the perspective of a connectivity layer network, an NFEP may be either a source, sink, or both (contrast this with an SFEP). Associated with an NFEP is a characteristic information accepted/delivered at the NFEP. These properties include frame structure identification, QoS, etc. A *Network Flow Connection* (NFC) is a connectivity resource that transports information between a group of NFEPs. An NFC has one of the following configurations:

- A point-to-point bidirectional connection between two NFEPs
- A point-to-point unidirectional connection between two NFEPs. One of the end points is designated as the root NFEP, and the other end point is designated as the leaf NFEP. Information is transported from the root NFEP to the leaf NFEP.

2. See [5] for details.
3. The current version of NRIM and NRA specifications do not address management of connectionless networks, such as the Internet. It is expected that this limitation will be overcome in the next issue of these specifications.

- A point-to-multipoint unidirectional connection between two or more NFEPs. One of them is designated as the root NFEP, and the others are designated as the leaf NFEPs. Information is transported from the root NFEP to the leaf NFEPs.

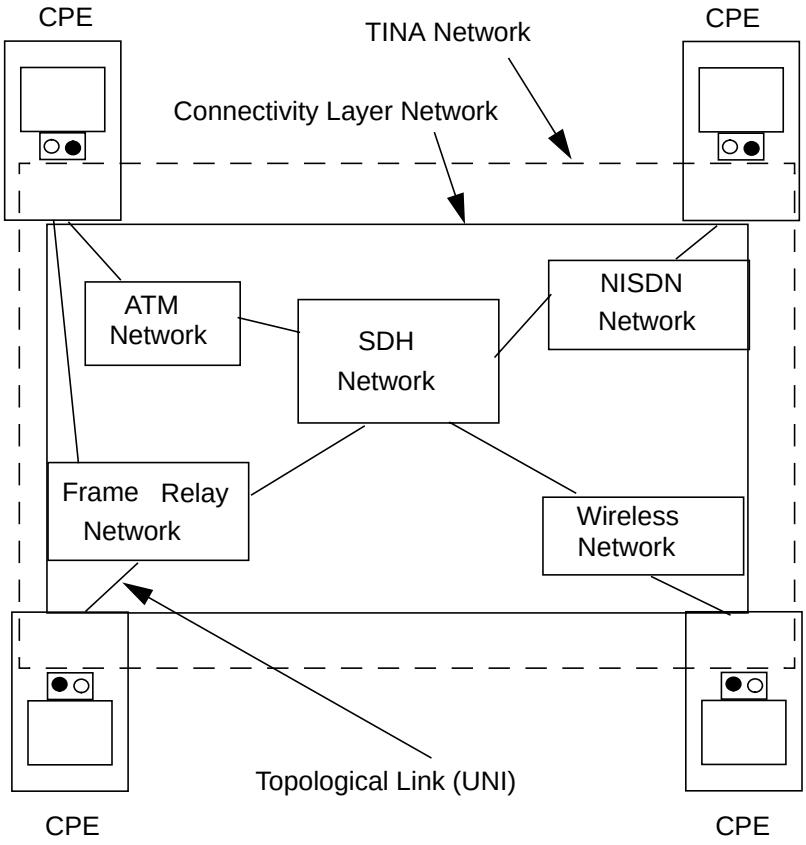


Figure 3-2. TINA Network (Physical View)

From the connectivity perspective, a stream flow is composed of one or more Network Flow Connections and two or more *Terminal Flow Connection*s (TFC), see Figure 3-3. A terminal flow connection is a connectivity resource that transports information either from an SFEP to an NFEP, or vice versa, or to another SFEP within the same CPE. It is possible that the frame structure and QoS associated with the SFEP and NFEP bound by a terminal flow connection are different, in which case the TFC performs the necessary adaptation. It is possible that multiple TFCs have the same NFEP; i.e., several stream flows can be multiplexed over a single network flow connection.

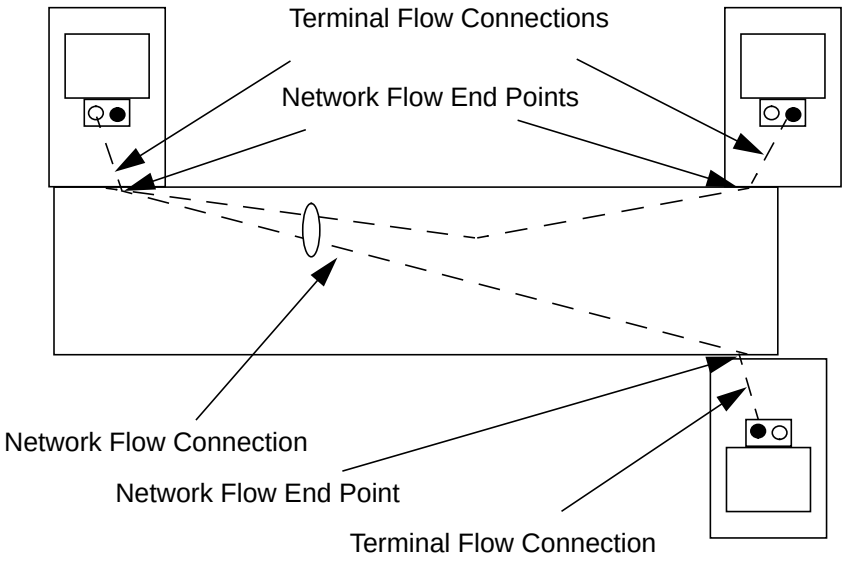


Figure 3-3. Stream Flow, Network Flow Connection, and Terminal Flow Connection

In the networking literature, the concept of *Layer Network* (LN) is used to denote a network that is based on a single technology and that transports information of a specific format, referred to as the characteristic information of the layer network. Examples of layer networks are: ATM Virtual Path (VP) network, ATM Virtual Channel (VC) network, SDH VC4 Path network, and Frame Relay (FR) network. A connectivity layer network is made up of one or more layer networks. A layer network may be related to another layer network in a connectivity layer network in one of two ways:

- **Peer-to-Peer Relationship:** This is the case when information delivered by one layer network is adapted and given as input to the other layer network, and vice versa. This relationship is symmetric, and is referred to as *layer interworking relationship*. An example of this relationship is adaptation of Frame Relay to ATM (VP/VC), and vice versa.
- **Client-Server Relationship:** This is described later in Section 3.2. This is the case when a group of link connections in one layer (the *client layer*) network is served by a trail in the other layer (the *server layer*) network. Such a group of link connections is called as a *topological link*.

3.2 Structure of a Layer Network

The network resource that transports information across a layer network between two or more end points in the layer network is called a *trail*. The end points are called *Network Trail Termination Points* (NWTTP). Thus, a trail is defined relative to a layer network. For example, a virtual path (VP) trail is a resource that transports ATM VP cells across a VP layer network. Depending on the nature of the layer network, a trail may or may not be directly related to the physical network. For example, a trail in the SDH path layer gives a logical view of the transport capacity, that is not necessarily related to the physical network. Whereas a trail in the physical media layer is related directly to an actual fibre (SDH physical optical section) or coaxial cable (SDH physical electrical section). A trail may have one of the following configurations: point-to-point bidirectional, point-to-point unidirectional, or point-to-multipoint unidirectional.

3.2.1 Subnetworks and Links

A layer network is decomposed into *subnetworks* that are interconnected by *topological links* between them, see Figure 3-4. In general (e.g., in G.805), a link represents a topological relationship between two subnetworks and the potential for connectivity between the subnetworks. In the TINA Network Resource Information Model (NRIM), the concept of a link is more specific, and is called as *topological link*. A topological link represents a group of transport resources (called *link connections*) that transport information between the subnetworks. Further, a topological link is configured using exactly one trail in the underlying server layer network, see Section 3.2.2.[4]

Each subnetwork may be further decomposed into smaller subnetworks interconnected by topological links until the desired level of detail is revealed. This will generally be when the subnetwork is equivalent to a single network element (switch or digital cross-connect). The network resource that transports information across a subnetwork between two or more end points in the subnetwork is called a *subnetwork connection*, see Figure 3-4. Reflecting the partitioning of a layer network into subnetworks and topological links, a trail is made up of one or more subnetwork connections and link connections. Similarly, reflecting the partitioning of a subnetwork into subnetworks and topological links, a subnetwork connection may also be made up of one or more subnetwork connections and link connections, see Figure 3-4.

The partitioning concept is useful for defining:

- Significant administrative boundaries between network operators jointly providing end-to-end paths within a single layer.
- Management domain boundaries (i.e., scope of management functions or systems) within the portion of a layer network that is under the control of a single network operator.

4. The prefix “topological” is used to emphasize use of this specific configuration.

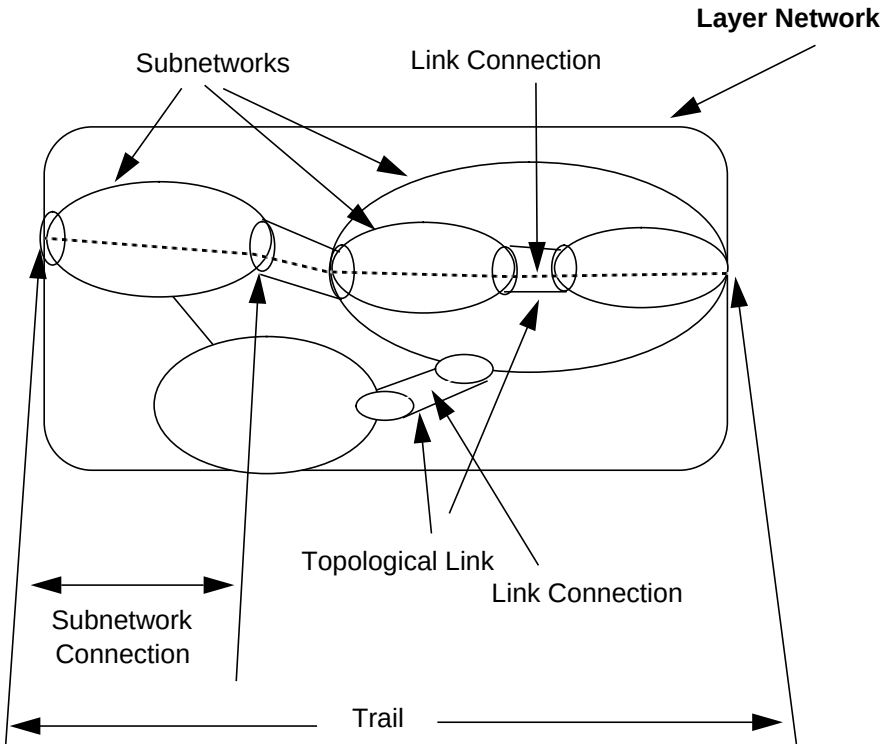


Figure 3-4. Structure of a Layer Network

3.2.2 Client-Server Relationship

As mentioned in Section 3.1, two layer networks may have a client-server relationship. This relationship exists when the transport capabilities of one layer (the server layer) are used in the other layer (the client layer). More specifically, this relationship is established when link connections in the client layer network are provided by a trail in the server layer network. See Figure 3-5. Some examples of the client-server relationship between layer networks are listed below:

- Link connections in a ATM VP layer network are provided by a path (the name of the trail in the path layer) in the SDH path layer network.
- Link connections in a ATM VC layer network are provided by a trail in a VP layer network.
- Link connections in a SDH path layer network are provided by a section (name of the trail in the transmission media layer network) in the transmission media layer network.

The end points of a link connection are called *Network Connection Termination Points* (NWCTP). It should be noted that while a trail or a subnetwork connection may be either a point-to-point or a point-to-multipoint connection, a link connection is always a point-to-point connection.

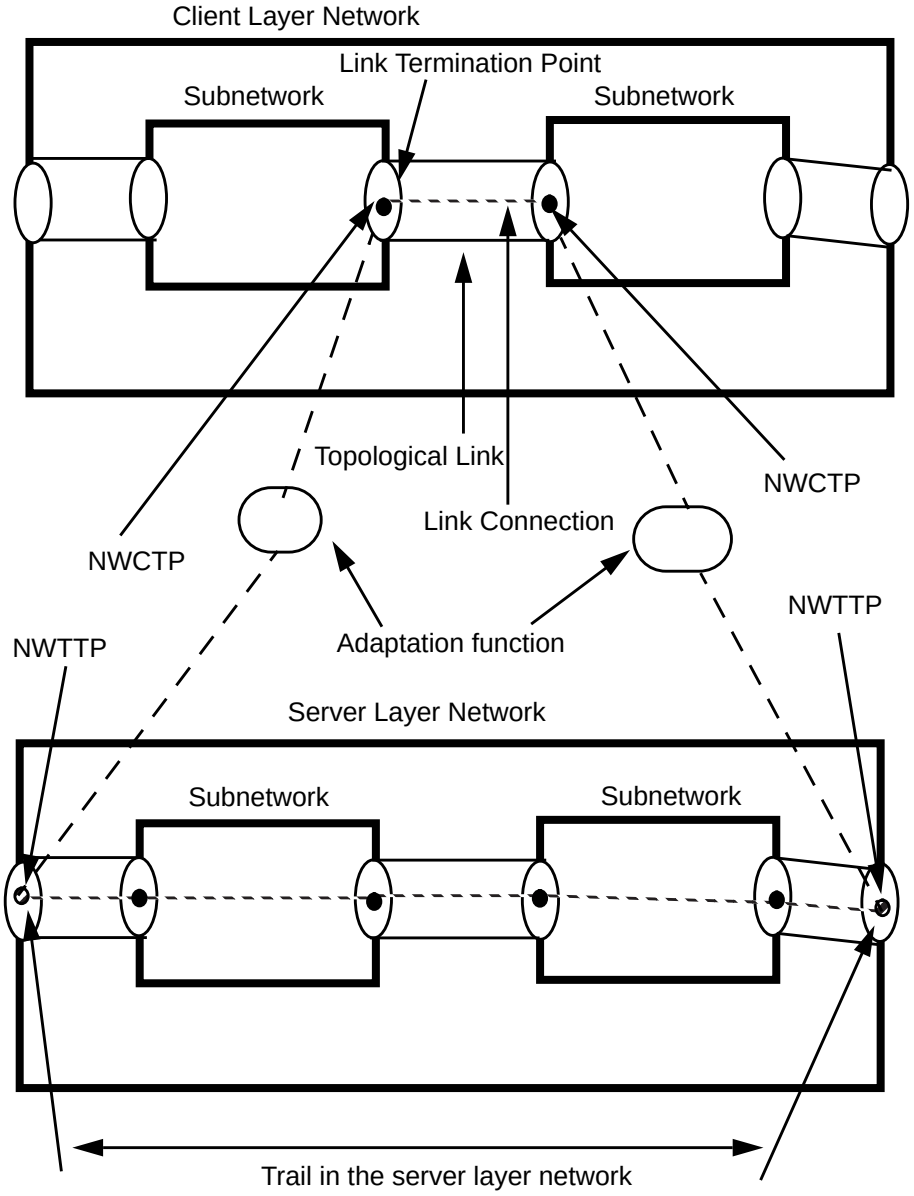


Figure 3-5. Client-Server Relationship

3.2.3 Edges and Network Connection Termination Points

As described in Section 3.2.1, it is possible that a subnetwork is partitioned into two or more subnetworks interconnected by topological links. Reflecting this partitioning, a subnetwork connection may also be made up of two or more subnetwork connections and link connections. In general, the lifetime of a subnetwork connection and its component subnetwork connections and link connections may be different. That is, when a subnetwork connection is deleted, some of the component subnetwork connections and link connections may continue to exist. Similarly, a subnetwork connection may be set up using existing subnetwork connections and link connections. These capabilities are useful for rerouting trails and subnetwork connections upon failures. To allow for this generality, the TINA NRIM distinguishes between an end point of a subnetwork connection and an end point of a link connection. This distinction is especially important for unidirectional subnetwork connections and link connections. The two kinds of end points are distinguished using the concepts of network connection termination points and edges as described below (see Figure 3-6):

- **Network Connection Termination Point (NWCTP):** A termination of a link connection is called a network connection termination point. A link connection is only a point-to-point connection, and thus a link connection has only two network connection termination points.
- **Edge:** An extremity of a subnetwork connection is called an edge. A point-to-point subnetwork connection has two edges, and a multipoint subnetwork connection has more than two edges. An edge of a subnetwork connection is bound to a network connection termination point. This binding may change during the lifetime of the subnetwork connection.

Reflecting the partitioning levels of a subnetwork, a subnetwork connection in a composite subnetwork is partitioned into subnetwork connections in the component subnetworks. In such a situation, an edge of the composite subnetwork connection and an edge of the component subnetwork connection will be bound to the same NWCTPs. See Figure 3-6.

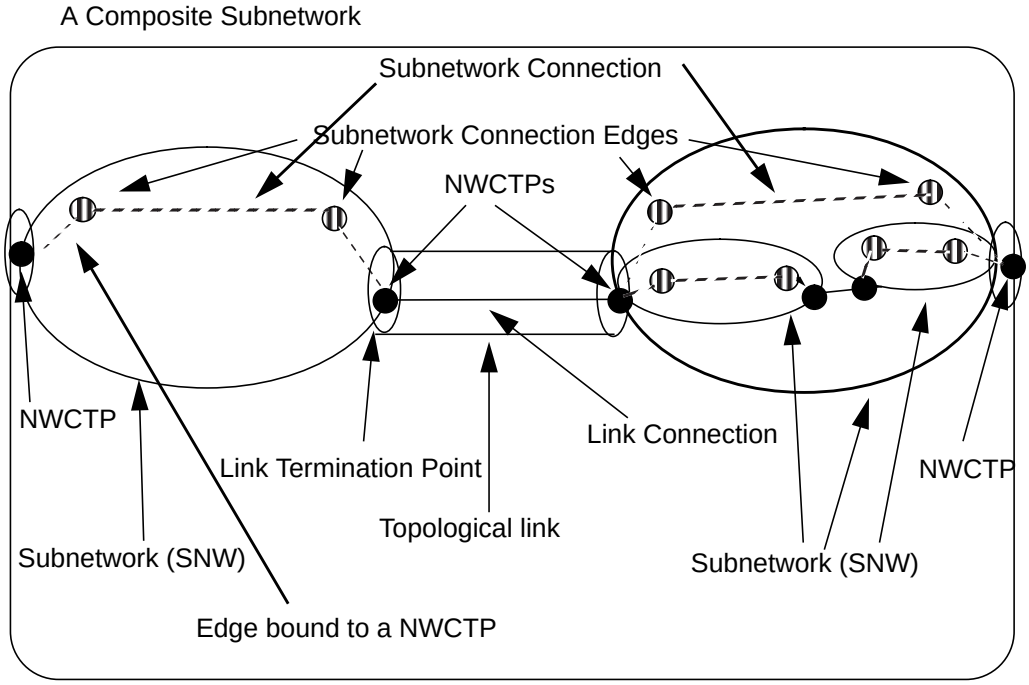


Figure 3-6. Relationship Between Edges and NWCTPs

3.2.4 Tandem Connection

A tandem connection is an arbitrary series of contiguous subnetwork connections and/or link connections and is used to represent an arbitrary segment of a trail. The extremities of a tandem connection are either network connection termination points or trail termination points. Just as in the case of a trail or a subnetwork connection, a tandem connection may be either point-to-point or point-to-multipoint.

Figure 3-7 illustrates tandem connections. As can be seen from the figure, a limiting case of a tandem connection is a subnetwork connection. The concept of tandem connection is very useful in situations where a trail spans multiple subnetworks and the subnetworks are under the control of different administrations.

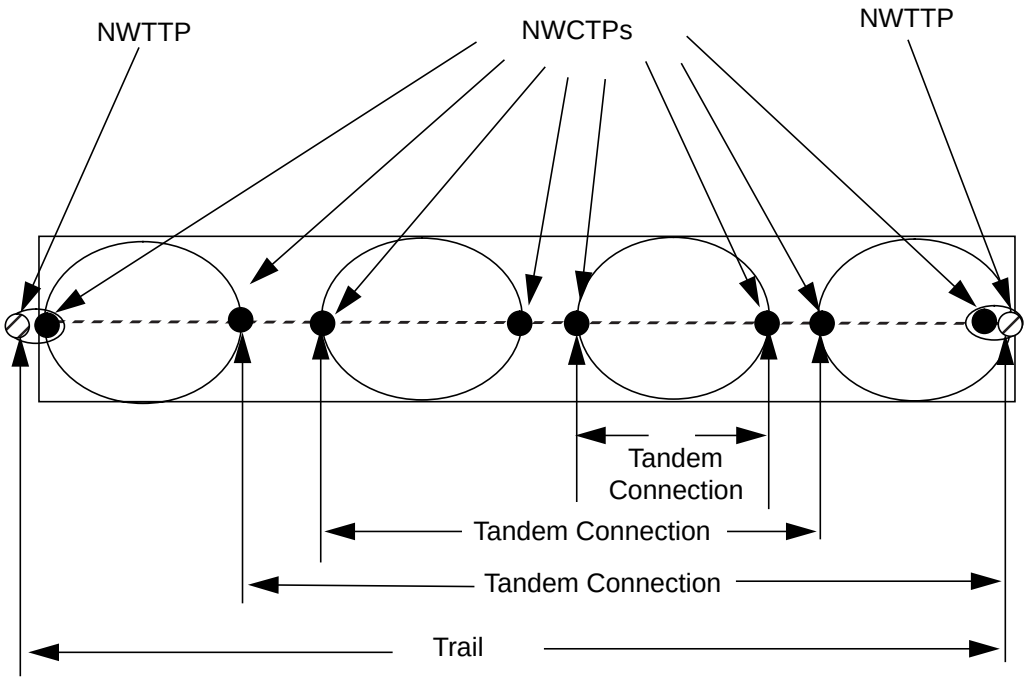


Figure 3-7. Tandem Connection

3.3 Domain Concepts

In network management, the concept of a *domain* is typically used to refer to a collection of resources that have some common properties with regard to their management. The TINA NRIM defines two types of domains:

- **Administrative Domain:** An administrative domain is a collection of resources that are under the control of a single network administration. (Thus, this notion is associated with ownership of resources.) This notion is fundamental since a global telecommunication network may have parts under the control of different administrations, and these administrations have to cooperate among them (exchange information and offer management capabilities) to provide services to customers. To cope with the (technical and business) complexities of large scale networks, a network administration should have a detailed view of resources in its domain, and only an abstracted (or summarized) view of resources in other administrative domains. The concept of administrative domain is applied in two levels: at the level of the connectivity layer network, and at the level of layer networks. The portion of the connectivity layer network

that is under the control of a single administration is called a *Connectivity Layer Domain* (CLD). The portion of a layer network that is under the control of a single administration is called a *Layer Network Domain* (LND). The reason for defining administrative domains in two levels is that it facilitates the modeling of the relationships between layer networks, and between layer networks and a connectivity layer network.

- **Management Domain:** A management domain is a collection of resources within an administrative domain that are under the control of a management function, such as connection management function or fault management function. Note that a management domain does not span administrative domains. An administrative domain may consist of one or more management domains. A resource may belong to multiple management domains but always belongs to only one administrative domain.

4. Connection Management

The mission of Connection Management (CM) functions in a TINA-C consistent/compliant network is to support telecommunication services (distributed applications) in their need for connections. These telecommunication services range from the simple telephony service currently offered by the local network operator to more complicated services like multi-media conference calling, broadband virtual private networks, multi-party services, and mobility services. Connection Management functions may also support other management services (for instance, a fault management application that needs a connection through a specific trunk in order to test that trunk), and the interconnection of DPE nodes.

The complete set of CM functions is mostly addressed as the Connection Management Architecture (CMA).

In this document*, connection* can mean different things depending on the abstraction level. On the highest level it means a binding between stream interfaces of computational objects. This association is broken down into smaller connections, often in a recursive way. For instance the association between physical endpoints in an ATM network is also a connection. On the lowest level of abstraction a connection is the association of ports internal to a switch.

The results of the ‘unification’ [15], ‘communication session’ and ‘Reference Point’ [16] work as well as the introduction of the business model, as mentioned earlier, highly influenced the contents of this chapter. The focus shifted from a pure (network) resource description towards the integration of this resource description in the complete architecture. Although not (completely) covered in this version, the nodal part of the connectivity is not anymore regarded as out of the scope of the TINA connection management architecture.

This all resulted in the redefinition of the connection graphs, the introduction of the session concept in the upper layers of the connection management architecture and the specification of a more elaborated computational model than before.

This chapter consists, just as most other chapters in this document, of 5 sectionS. Section 4.1 introduces and defines the scope of the chapter. It also provides the reader with sufficient background material to understand the sequel. This section is recommended reading material for both TINA-C CM beginners and veterans. Section 4.2 summarizes the functional requirements for the TINA-C CM. Section 4.3 describes the information model. Section 4.4 covers the computational viewpoint of the CM architecture. Section 4.5 explains the dynamic behaviour of the previously described computational objects by explaining a number of scenario’s.

4.1 Scope

This section defines the scope of the TINA-C connection management. Section 4.1.1 is a high level introduction to the CM. Section 4.1.2 (overview) illustrates the several concepts that are used in the CM by means of an example. Section 4.1.3 deals with some background concepts, while Section 4.1.4 and Section 4.1.5 introduce Connection Graph and Session respectively.

4.1.1 Introduction

The goal of the Connection Management specification effort is:

- To specify reusable functions which manage network transport resources, independently of transmission and switching technology. These functions will be made available to clients such as service and management applications and the DPE. The functions will be deployed as a set of distributed, interoperable software components.
- To specify a generic network information model to support the management activity mentioned above.

Connection Management relies on the functionality of the TINA Distributed Processing Environment (DPE) (see [6] for a description). At the same time, the DPE Kernel Transport Network (KTN)[1] that interconnects the DPE nodes can be built using Connection Management services. The KTN is used to support the operational interfaces that setup and control services and connections, replacing the current signalling approach. The definition of the KTN is currently under study as part of the DPE specification effort (see Section 8).

The TINA-C CM is a layered architecture as (partially) demonstrated in Figure 4-1. As a result, the architecture described in this chapter can be split and depending on the implementation partially supported. For example, a Connectivity Service provider could support the Cons (connectivity level) and Tcon (Terminal connectivity) reference points to accommodate TINA-C service providers and users, but implement proprietary CM functions in his domain. Another example, where a non TINA-C CM functionality might be useful or even desirable is at the layer network level. Current legacy systems mostly have a well defined and standardized way of dealing with CM functionality. This functionality could be used instead of the Connection Performer approach, while the layer network level TINA approach (LNCs and TLAs) is maintained.

1. An analogy of a KTN is a signalling network.

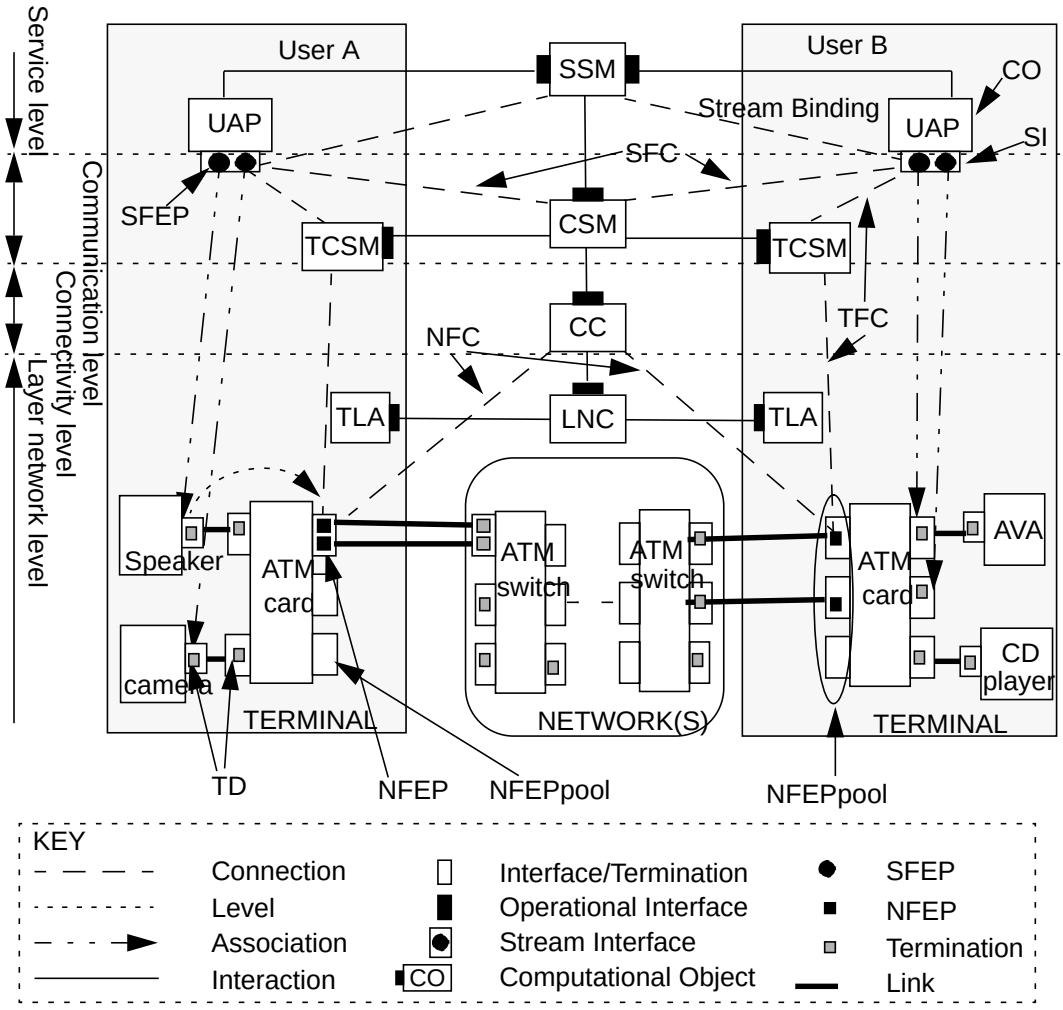


Figure 4-1. Service-oriented components of the CMA (ATM example, simplified)

4.1.2 Overview

The CMA structure is based on the TINA network modeling concepts. It is divided into a number of conceptual levels, that work together to transform service layer communication requirements into constructs that can be implemented in the resource layer. Each level has its own set of environmental and organizational concepts (such as session for service oriented levels and network or subnetwork for resource oriented levels), connection concepts (such as stream bindings, flow connections, and trails), detailed information models (such as service session graph and connection graph), and associated computational objects.

This overview seeks to introduce these conceptual levels, their scope, and their relations to each other. Figure 4-1 introduces the service oriented levels: service, communication and connectivity. We have included the service level, even though it is not part of the NRA, to illustrate the use of the connection management architecture.

The service level is a client of the connection management architecture. It uses the services of the communication level offered by the CSM to setup service-level connections. From the computational viewpoint, the service session is provided by a Service Session Manager (SSM), an (optional) User Service Session Manager (USM), and an User Application (UAP) computational objects (COs). The UAP and USM represent a user to the service. The UAP is located on the user terminal and may support Stream Interfaces (SIs).

At this level, connections are described in terms of a stream binding between SIs that support multipoint to multipoint, multimedia connections. Stream bindings are described by service level information models, such as the service session graph [20]. Stream interfaces can be considered specialized computational interfaces, and are always associated with a CO. However, connections may be established between virtual devices, such as a camera or a speaker, as well as COs.

Stream bindings are set up in a distributed manner. Each CO participating in the binding must set up its SI and associated Stream Flow End Points (SFEPs). This process could involve associating SFEPs with particular Terminal Devices (TDs). For instance, Figure 4-1 shows SFEPs that are associated with a speaker and a camera. The SSM is responsible for setting up the overall stream binding. To do this, it translates the stream binding description into a set of Stream Flow Connections (SFCs), and uses a communication session to establish them. Figure 3-1shows the relation between stream binding and SFCs.

Each communication session is associated with a single client service session. The communication session has two components: a terminal component and a network component. From the computational viewpoint, the former is maintained by a Terminal Communication Session Manager (TCSM) and the latter is maintained by a Communication Session Manager (CSM). The communication session is responsible for end-to-end service (application) level connections: that is, between SFEPs. Figure 4-1 indicates this by the dashed lined marked SFC linking the CSM with SFEPs. In TINA, these connections are SFCs, which are modeled by the Logical Connection Graph (LCG), see Section 4.3.1.

The CSM uses the SFEP descriptions to determine the associated TCSM and Network Flow End Point (NFEP) or NFEPpools. In Figure 4-1, user A's SFEPs are associated with an NFEPpool, while user B's SFEPs are associated with pre-configured NFEPs. The CSM contacts each TCSM associated with the SFC so that the terminal and network parts of the SFC can be connected. The CSM translates the SFCs to Network Flow Connections (NFCs) and uses the connectivity session to establish the NFCs. Refer to Figure 3-3 to see the relation between SFCs and NFCs, and the scope of NFCs.

Each connectivity session is associated with a client communication session. From the computational viewpoint, the connectivity session is controlled by a Connection Coordinator. The connectivity session is concerned with end-to-end network connections: that is between NFEPs. Figure 4-1 indicates this by dashed lines marked NFC linking the CC with the NFEPs. Note that NFEPs are a technology independent representation of network termination points. Figure 4-1 shows NFEPs mapped to termination points in an ATM network.

The TINA connectivity session presents a technology independent network view. One that is independent of complications such as multiple connectivity provider domains or establishing connections over different network technologies. This view is supported by the

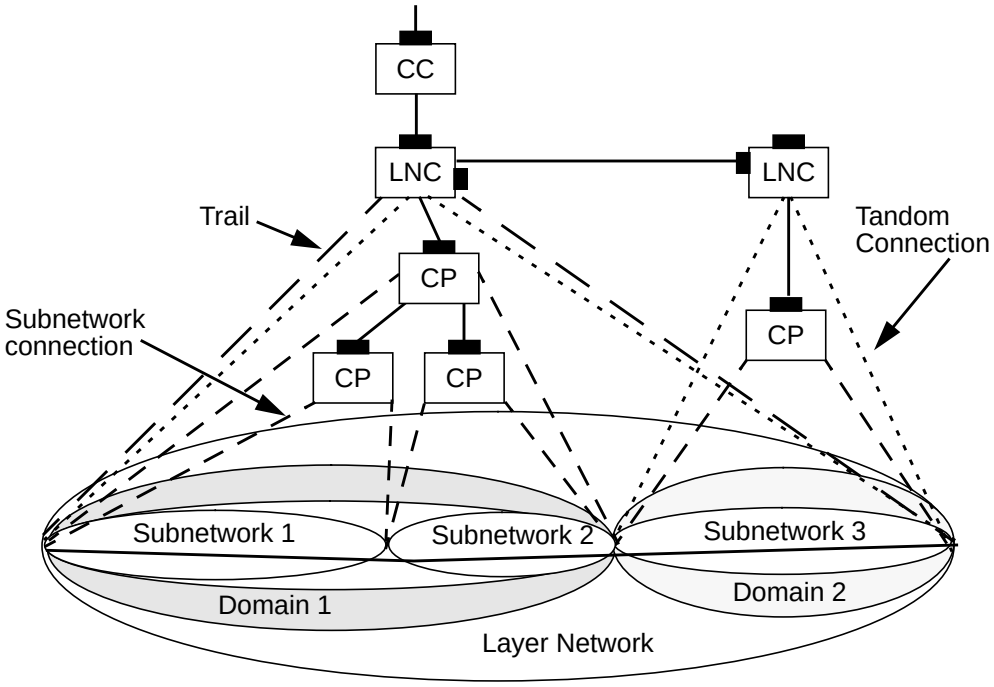


Figure 4-2. Resource-oriented levels of the CMA (simplified)

NFCs, which are modelled by the Physical Connection Graph (PCG). However, to establish connections, this abstract view must be translated to a suitable technology dependent layer network view. The connectivity session components, determine the layer network(s) to use, map NFCs to trails, and locate a layer network coordinator to establish the trail.

A layer network represents a particular networking technology, such as ATM (see Figure 4-1). Unlike the service oriented levels, there is no concept of sessions. From the computational viewpoint, a Layer Network Coordinator (LNC) can act on behalf of the layer network to setup end-to-end connections across that network technology, independent of connectivity provider domains and subnetworks. These connections are described by trails. LNCs also act on behalf of a domain within the layer network. LNCs representing the same layer network can cooperate with each other to complete trails initiated in one domain. These connections are described by tandem connections (TC).

Figure 4-2 shows the LNC's capabilities. To connectivity session clients, the LNC presents an end to end view of a network technology, and offers services to establish trails across the entire network. To a peer (components acting on behalf of other domains of the same layer network), the LNC offers services to setup tandem connections to complete trails that are terminated outside their domain.

The layer network level also requires terminal components, see Figure 4-1. The connectivity provider components interact with the terminal components to establish or select NFEPs. Once a NFEP is selected, the connection between SFEP and NFEP can be completed by the TCSM, which must be informed of the selection by the Terminal Layer Adapter (TLA) or the CSM.

The last conceptual level is the subnetwork. A subnetwork is associated with a particular network technology within a single domain. From the computational viewpoint, a Connection Perform (CP) offers management services to establish subnetwork connections, see Figure 4-2. The CP manages the subnetwork to establish and control these connections. These components are always technology dependent and represent an idealized TINA view of the network. It is possible to replace computational components below the layer network level with technology specific components if desired. Subnetworks are recursive and correspond, at the lowest level, to the element management layer. CPs at this level interact with NEs to set up part of a connection.

Table 4-1. Conceptual levels and associated information and computational concepts and objects

Level	Environment concept	Information model	Connection concept	Connection terminators	Associated COs
Service	service session	service session graph	stream binding	SI	UAP, SSM, USM, SSF
Communication	communication session	logical connection graph	stream flow connection	SFEP	CSM, TCSM, CSMF
Connection	connectivity session	physical connection graph	network flow connection	NFEP	CC, FCC, CCF
Layer network	network, domain	trail, tandem connection	trail, tandem connection	NWTP	LNC, TM, TCM
Subnetwork	subnetwork, element	subnetwork connection	subnetwork connection	Edge	CP

Table 4-1 summarizes the conceptual levels. It lists the conceptual levels, environmental concepts, information models, connection concepts and associated COs. Sessions are used to provide connection services oriented to aiding service level communication. Session related components last only for the life-time of the session and then are destroyed. While the network and subnetwork control components, though providing a connection service, are aimed at managing a resource layer and so their lifetime is much longer. It is related to the lifetime, status, and other factors (e.g load) of the resources they control.

Also note the different connection concepts at each level. Differences between these connections include the level of abstraction (e.g. technology dependent or independent), the scope of the connection (e.g in a network, domain or subnetwork), and type of termination. There are also many similarities. Apart from the multipoint-to-multipoint stream bindings, connections models are based on root and leaf terminations associated with branches of

the connection. Similar operations can be formed on each category of connection. Finally, the topological and directional characteristics of a connection remain unchanged when mapped from one level to another.

Not all computational objects have been introduced, to simplify diagrams. See Section 4.4 for a comprehensive description of all COs. As well as the environment related COs, there are a number of connection related COs or interfaces, such as the Flow Connection Controller (FCC), Trail Manager (TM), and Tandem Connection Manager (TCM). These COs (or interfaces) are controlled by operations derived from the information model, and can be viewed as related. Connection COs generally act on environment COs to establish connections in another domain or at a lower level. These CO's were omitted earlier, partly to enhance the readability of the diagrams, and partly because object separations vary between the conceptual levels and are not prescriptive.

Also, factory COs, such as the Communication Session Manager Factory (CSMF) and Connection Coordinator Factory (CCF), are introduced at the session oriented layers to handle some life cycle aspects of session related COs. Factories objects are used to create objects related to a particular service. These objects were not described in the diagrams as we were not considering life cycle issues in our overview.

Finally, to actually establish a connection at one level from a connection at another, the connection terminations of each type of connection must be associated. This could be relatively simple when one termination is an abstraction of a lower level termination (e.g. NFEPs and NWTTPs) or connection terminations are co-located. Or it could be quite complex, such as linking the service level terminations to network connections. This latter case is so complex, it is described by a connection model of its own: the Terminal Flow Connection (TFC).

4.1.3 Separation of Call and Connection Control

The TINA connection management architecture is based on the principle of the separation of call and connection control. This principle emerged long ago in ITU-T. The exact meaning of this principle has not always been the same. Initially it meant that in order to use resources efficiently it was regarded to be efficient if end-users first negotiate about the actual bearer connection they need before they really start the process of claiming resources for this connection. Therefore two separate layers were introduced. One to cope with end-to-end user control (call control) and one with bearer control (connection control).

Note that the terminology used here comes from the IN world. In TINA, the concept of call has been translated into the service session concept. Connection is supported by the overall connection management architecture, including the communication session level.

Later more benefits of separating the end-user and service related concerns from the concerns of controlling and managing the needed network resources were identified. The separation now has more the nature of a separation between functions in the Service Management Layer versus functions in the Network Management Layer. For Connection Management this is shown in Figure 4-1 in the separation between service components (i.e., service session control) and CSM.

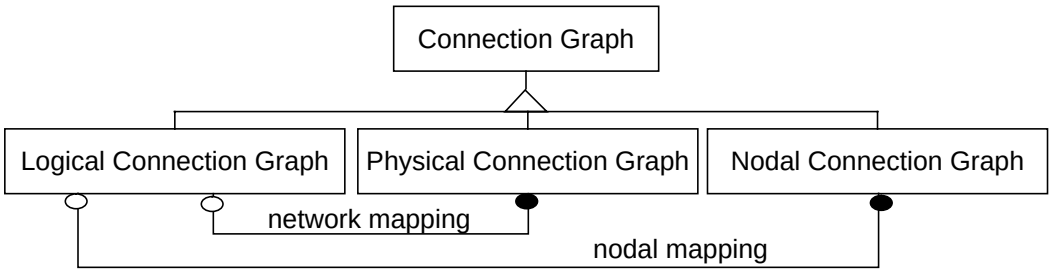


Figure 4-3. Connection Graph classes

4.1.4 Connection Graphs

The connection graph concept is used by clients of the Communication Session Manager (CSM), the Connection Coordinator (CC), and the Terminal CSM (TCSM) to specify their connectivity needs in a technology independent manner. The information object classes used in the connection graph fragment are based on a directed graph. The *connection graph* object is defined as a container object for the components of the graph. Examples of the use of the connection graph can be found in Section 4.3.

The detailed specifications of the connection graph concept was originally defined in cooperation with the Eurescom project P103 (see [49]). However, the TINA connection graph has recently been simplified to reduce the complexity of setting up connections.

The connection graph concept is closely related to the communication session, defined in the “Definition of Service Architecture” [20]. The communication session relates the connections in a transport service to the parties involved. A *connection graph* is a specification of connections between parties involved. A communication session is directly associated with one connection graph.

A *connection graph* contains flow connections, which are described in terms of flow end points. *A flow end point (FEP)* object represents a source, sink, or sink/source of an information stream. FEPs can be grouped in *flow end point pools*.

A FEP can be designated as a root or leaf of a flow connection. Each leaf FEP is associated with a particular branch of the flow connection. A *flow connection* object represents the connection between flow end points. It may be point to point (a connection with one single *branch*) or point to multi-point (a connection with several *branch*es). All point to multipoint connections are unidirectional.

The concept of connection graph is used at a different level of abstraction by CSM, CC and TCSM. Thus there are different specializations of the *connection graph* class, into the *logical connection graph*, *physical connection graph* and *nodal connection graph*, respectively. The term *logical* refers to application level connectivity, *physical* refers to the network, and *nodal* to the nodes between which the flow connections are defined. The flow connections and flow end points are also specialized.

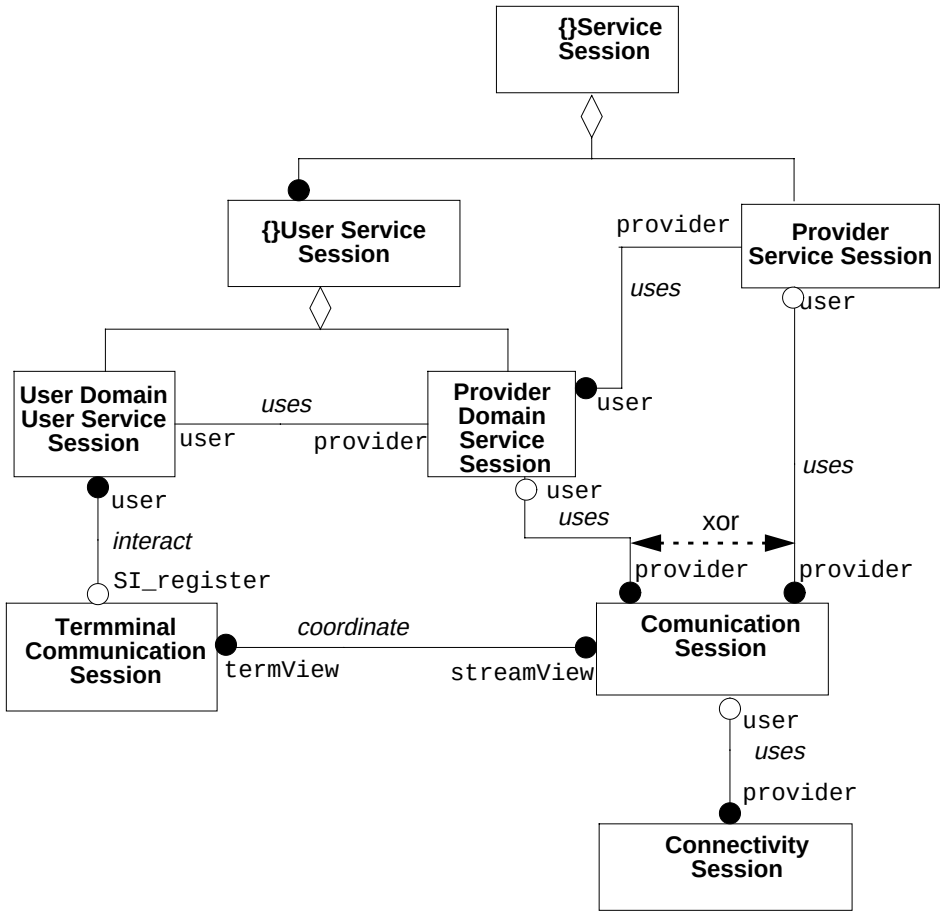


Figure 4-4. Session control relations

4.1.5 Sessions

A session is a temporary relation among a group of resources that are assigned to collectively fulfil a task or objective for a period of time. Sessions are an environment concept: they represent the resources and logic required to provide an instance of a service. This environment includes generic concepts such as state and ownership. This environment can be extended to support a particular view of resources, for instance the logical view of the TINA network model (Section 3.1). It is supported by attributes related to resources in the session, relations between resources in the session, and relations to other sessions. Unlike traditional resource oriented concepts, such as layer networks, sessions are transient and exist only for limited periods of time.

It should be noted that sessions are a high-level, abstract concept relating to the provision and control of instances of service use. They need to be related to the detailed information (and computational) models used. The connection graphs provide the detailed information model of connections that is used by both the communication and connectivity session.

Session concepts are introduced in the service architecture, see [20]. Three kinds of sessions were identified in that document: access sessions, service sessions, and communication sessions. A fourth session, the connectivity session, is identified in this document. This new session reflects the relations between stream flow connections and network flow connections, and supports the business roles involved in establishing a connection. Both the communication session and connectivity session relate to the connection management service which the CMA provides.

This section will concentrate on introducing communication and connectivity sessions. Section 4.1.2 has already introduced service session concepts and its relation to the communication session, which we will describe further in this section. Access sessions setup a security and management environment to allow to instantiate services between domains. In the case of connection management, the domains relate to the retailers or third party service providers and the connectivity providers. The relation of access session to communication connectivity sessions requires further study. Appendix D provides an introduction to the use of access sessions in relation to the NRA.

4.1.5.1 Communication session

A communication session serves as an environment for establishing and managing service level connections. It is used to establish and maintain a number of stream flow connections (see Section 3.1). These connections can be described by a connection graph (see Section 4.1.4 and Section 4.3.1). The connection graph is the basis for describing the relation between the communication session and its clients: i.e. the service session. Note the different scope of these concepts: connection graphs are a detailed, information model of connectivity while the communication session is a high-level abstraction of service provision specialized to provided connections.

The communication session supports the service provider (i.e. the retailer or third party service provider) part of the connection management service. It needs to establish a user relation with a connectivity session, associated with a connectivity provider, to actually establish communications. In a way, it can be viewed as supporting the user domain (i.e. service provider domain) part of a connection management service.

To establish these stream flow connections, the communication session must map them to a network related view, described by network flow connections (see Section 3.1). To establish and control the network part of the connections, the communication session needs to establish a user relation with a connectivity session, as shown by Figure 4-4. To do this, the communication session must determine the connectivity provider that can support its connections, and request a connectivity session. It then uses this connectivity session to establish and maintain the network part of its connections.

As shown in Figure 4-4, the communication session can be divided into two parts: the terminal part (terminal communication session) and the network part. A terminal communication session exists for each terminal. It offers services to the user domain part of the user service session to aid the set up of the connection between service-level and network connection end points within the terminal. It is used by the overall communication session

maintain terminal flow connections and their relations to stream and network flow connections. In this way the communication session establishes and maintains the end-to-end service-level connections.

4.1.5.2 Connectivity session

A connectivity session serves as an environment for establishing and managing network connections. It is used to establish and maintain a number of network flow connections (see Section 3.1). These connections can be described by a connection graph (see Section 4.1.4 and Section 4.3.1). The connection graph is the basis for describing the relation between the connection session and its clients: i.e. the communication session. The connectivity session is a high-level abstraction of service provision specialized to provided connections, this time at the network level.

The connectivity session supports the connectivity provider part of the connection management service. In a way, it can be viewed as supporting the provider domain (i.e. connectivity provider domain) part of a connection management service. This connectivity provider service provides a technology independent view of network connection. The connectivity session needs to map this abstract view to a technology dependent view offered by the supporting layer networks.

A connectivity session is responsible for determining the layer network (or networks) responsible for the connection. The layer networks are responsible for establishing the actual connection and interacting with the consumer domains as necessary. A connectivity session may need to determine which interactions are necessary with other connectivity providers to establish a connection.

4.1.5.3 Relations between sessions

It is important to realize that the communication and connectivity sessions relate to the same connection management service. The communication session offers service level view of connection to its clients. But to actually establish connections, it needs to relate to terminal communication sessions and connectivity sessions.

Figure 4-4 gives a formal information model of the relation between the sessions. Figure 4-5 relates this model to particular instances of service, communication and connectivity sessions. This figure gives a computational view, similar to Figure 4-1. Note that a service session has user domain parts, and provider domain parts, that are supported by the UAP and the USM and SSM respectively in the computational model. The communication session relates to the CSM and TCSM components, where each TCSM supports a terminal communication session within the overall communication session. Finally, the connectivity session relates to the CC and FCC components, which support overall session control and individual connection control respectively.

A communication session is controlled by a service session, normally through a PSS (provider service session) or a PD_USS (provider domain user service session), see Figure 4-4. Only one PSS or PD_USS may control a communication session at a time. This relation translates into interactions between the SSM and CSM in the computational diagram (Fig-

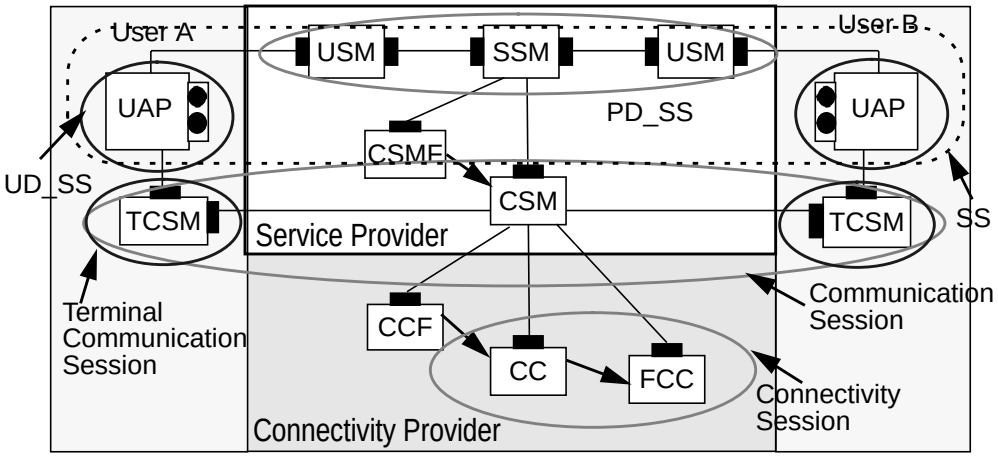


Figure 4-5. Instance of communication and connectivity session use

ure 4-5). The UD_USS (user domain user service session) and the terminal communication session need to interact to establish terminal flow connections. This is shown by the UAP and TCSM interactions in Figure 4-5.

As shown in Figure 4-5, the communication session is established in the service provider's domain. It establishes a user relation with a connectivity session in the connectivity provider's domain. These sessions work together to provide the service level connections. This relation can be related to the user-provider paradigm: the communication session acts as a user of the connectivity session (see Figure 4-4) and represents the service provider domain with in the connection management service. From the computational viewpoint, this is shown by the CSMs establishing a connectivity session via the CCF, and interacting with the CC and FCC (see Figure 4-5).

An alternative way of describing the relation is to consider the overall service level connection management service as being composed of the communication session and connectivity sessions. The communication session calls on the services of the connectivity session to establish the network part of the connections.

The communication session can act on the connectivity session (see Figure 4-4) to add, modify or remove flow connections by modifying the connection graph describing network flow connections within the connectivity sessions. The communication session can also act on the connectivity session to activate, deactivate or delete it.

The terminal part of the communication session has relations with both the user domain part of the service session and the overall communication session (see Figure 4-4). From the computational viewpoint, this is shown by UAP and CSM interaction with the TCSM. The UAP interacts with the TCSM to setup application and terminal specific details of the terminal flow connection. These details are generally not apparent outside the terminal. The CSM interacts with the TCSM to establish a relation between a TFC and both the service level and network level connections. This is described by a "correlation" relation between the communication session and its associated terminal communication sessions.

4.2 Functional Requirements

This section outlines the functional capabilities that are required of the TINA Connection Management functional area.

Connection management in TINA involves management of the following kinds of connectivity resources:

- *Stream flow connections*: A stream flow connection transports information from a source stream flow endpoint to one or more sink stream flow endpoints. The following configurations are possible: point-to-point unidirectional, and point-to-multipoint unidirectional
- *Network flow connections*: A network flow connection transports information across a connectivity layer network between two or more network flow endpoints. The following configurations are possible: point-to-point bidirectional, point-to-point unidirectional, and point-to-multipoint unidirectional.
- *Trails*: A trail transports information across a layer network between two or more trail termination points. The following configurations are possible: point-to-point bidirectional, point-to-point unidirectional, and point-to-multipoint unidirectional.
- *Subnetwork connections*: A subnetwork connection transports information across a subnetwork between two or more edges, where each edge is bound to a network connection termination point. The following configurations are possible: point-to-point bidirectional, point-to-point unidirectional, and point-to-multipoint unidirectional.
- *Link connections*: A link connection transports information over a topological link that interconnects two subnetworks. The endpoints of a link connection are network connection termination points. The following configurations are possible: point-to-point bidirectional, and point-to-point unidirectional.
- *Tandem connections*: A tandem connection transports information across a contiguous portion of a layer network. The endpoints of a tandem connection are network connection termination points or trail termination points. The following configurations are possible: point-to-point bidirectional, point-to-point unidirectional, and point-to-multipoint unidirectional.

For each of the above connectivity resource type, the following capabilities are offered:

- Establishment (i.e., creation)
- Release (i.e., deletion)
- Addition of a branch to a point-to-multipoint connectivity resource
- Removal of a branch from a point-to-multipoint connectivity resource
- Modification of the traffic and/or QoS parameters
- Activation; the activation operation sets the administrative state of the connectivity resource to *unlocked* state; a connectivity resource can transport information only if the administrative state is unlocked. In the case of a point-to-multipoint configuration, activation can be applied to specific branches.

- Deactivation; the deactivation operation sets the administrative state of the connectivity resource to *locked* state; a connectivity resource does not transport information when its administrative state is locked. In the case of a point-to-multipoint configuration, deactivation can be applied to specific branches.

In addition to the above capabilities associated with individual connectivity resources, some grouping or aggregation capabilities are also offered by TINA connection management as described below:

- *Communication Session Management*: A communication session is a context in which one or more stream flow connections can be established. A client of a communication session can establish and manipulate stream flow connections within the session. A stream flow connection must belong to a communication session.
- *Connectivity Session Management*: A connectivity session is a context in which one or more network flow connections can be established. A client of a connectivity session can establish and manipulate network flow connections within the session. A network flow connection must belong to a connectivity session.

For each of the two types of sessions, the following management capabilities are offered in TINA connection management:

- Session creation: at session creation time, one or more component flow connections can also be established.
- Session deletion: when a session is deleted, all component flow connections are released.
- Session activation: when a session is activated, its administrative state is set to unlocked state. This action does not change the administrative state of the component flow connections.
- Session deactivation: when a session is deactivated, the administrative states of the session and all component flow connections are set to locked state.

4.3 Information Viewpoint

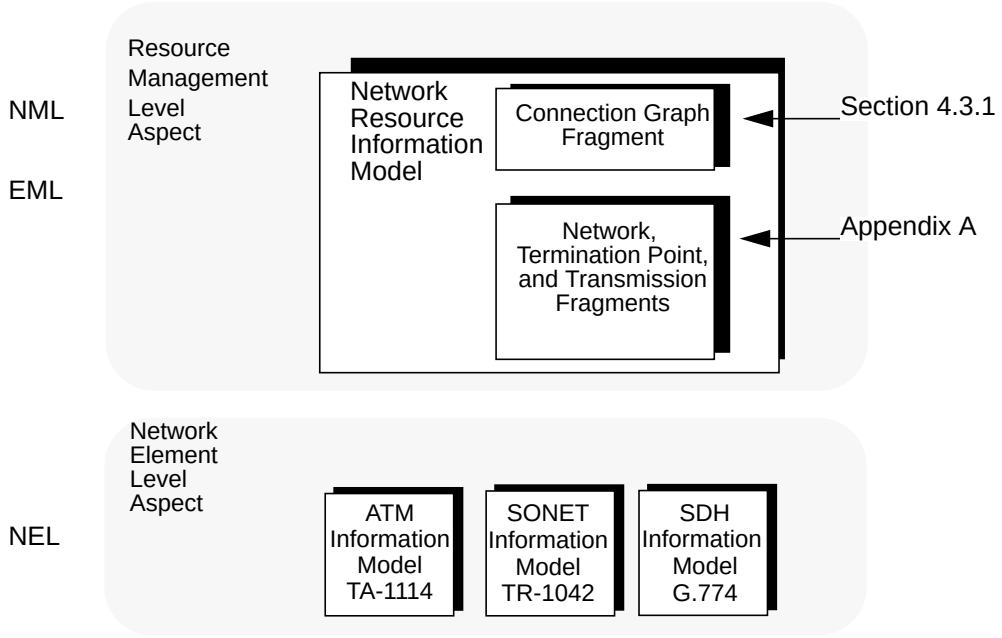


Figure 4-6. CM information Model and TMN Layers

This section describes information specifications of resource abstractions important to Connection Management with illustrative examples. Interested readers are encouraged to read the Network Resource Information Model Specification [1].

Figure 4-6 illustrates two categories of Network Information Models pertaining to Connection Management. Both of them fall in the broader category of Network Resource Information Model. The Connection Graph Fragment described in Section 4.3.1 is a collection of objects that model network connectivity that are tailored for the interface between CM computational objects such as CSM and CC, and the CM clients. Section 4.3.2 describes the session concepts that are introduced to fill the gap between the former network and service architecture.

There are a number of existing information models in the Network Element Level Aspect. This document does not define objects of this layer, its relationship with Network Resource Information Model objects is described in [1].

4.3.1 Connection Graph Fragment

Figure 4-7 depicts the class diagram of the generic Connection Graph (CG). All the classes in this diagram will be specialized according to a particular application (LCG, PCG or NCG).

The CG contains a number of Flow Connections (FCs). FCs are setup between Flow End Points (FEPs). Each FC is related to at least one root FEP and one leaf FEP. The relation between the FC and a leaf FEP is described by a FCBranch. FEPs may be created on the fly or may be pre-provisioned. FEPs will have attributes and relations to other classes (e.g. the physical termination points of connections) according to the specialization level they address.

No restrictions apply to the directionality of both FCs and FEPs. The specializations will deal with the directionality.

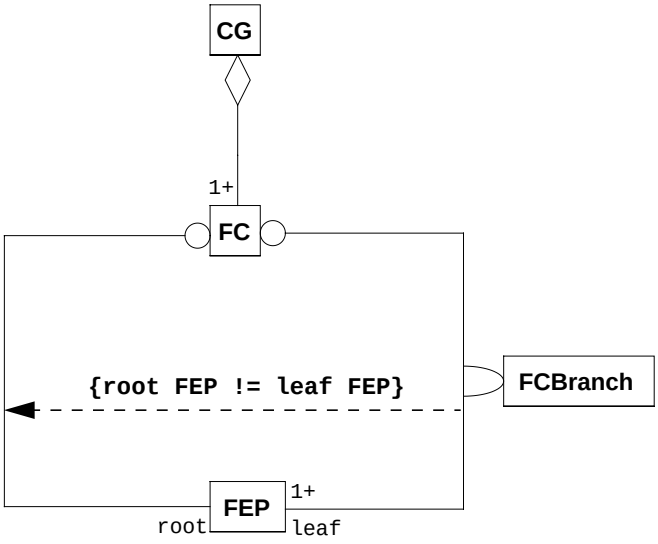


Figure 4-7. Connection Graph

4.3.1.1 Logical Connection Graph (LCG)

The concept of *Logical Connection Graph (LCG)* is used in TINA as an information model of the concept of stream binding defined in the Computational Modelling Concepts [4]. An example of a stream binding is shown in Figure 4-8.

Figure 4-9 shows the class diagram of the LCG. The LCG and all the other classes are specializations of the corresponding classes in the CG. This specialization is not explicitly presented to not overload the figure. The complete specification of the classes can be found in the next version of NRIM [1].

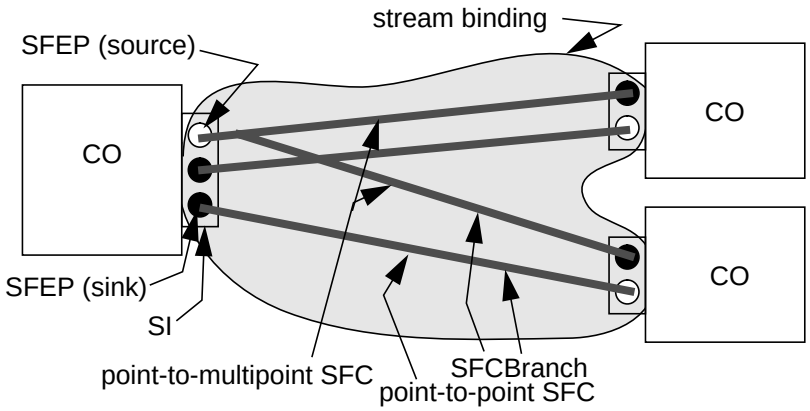


Figure 4-8. Example of stream binding

The LCG contains Stream Flow Connections (SFCs) that directly interconnect Stream Flow End Points (SFEPs). These SFEPs are grouped in Stream Interfaces (SIs), but this is outside the scope of the LCG.

SFEPs are uni-directional as has been defined in Computational Modeling concepts. Consequently SFCs are uni-directional and represent streams.

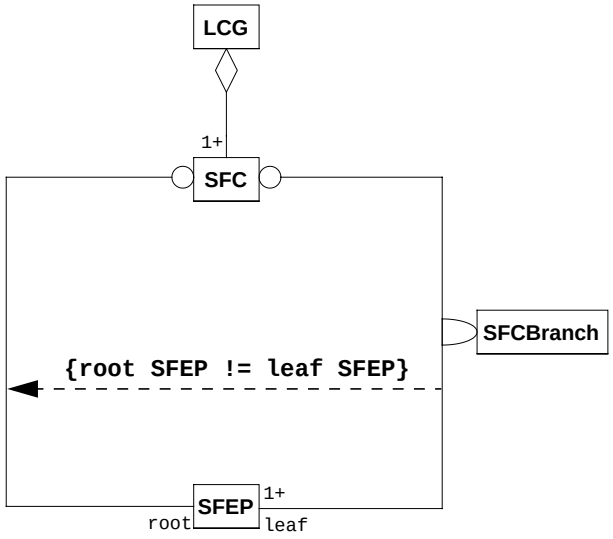


Figure 4-9. Class diagram of the Logical Connection Graph (LCG)

SFEPs are never created as a side effect of setting up a SFC; they already exist before they can be used in a SFC. In this sense they may be regarded as pre-provisioned although the use of this terminology is not appropriate for application level terminations.

The *LogicalConnectionGraph* corresponding to the stream binding example (Figure 4-8) is shown in Figure 4-10.

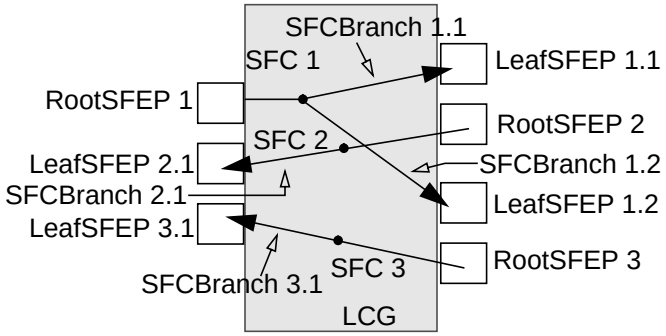


Figure 4-10. Logical Connection Graph for the stream binding of Figure 4-8

4.3.1.2 Physical Connection Graph

Physical Connection Graph (PCG) is used to specify end-to-end network connectivity. The LCG represents an end-to-end connection between computational interfaces (i.e. application level), the PCG between network termination points.

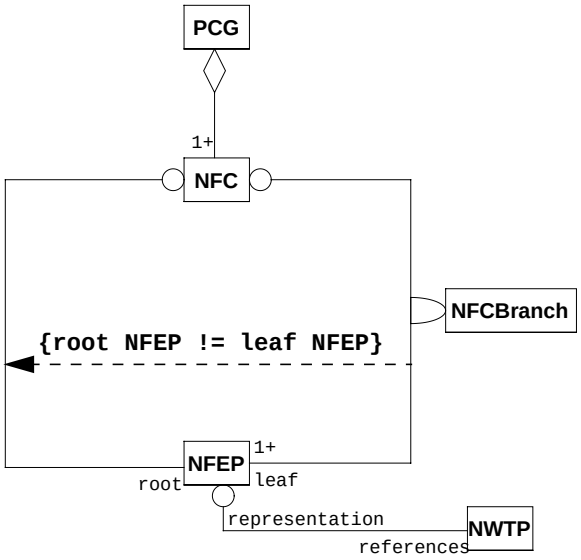


Figure 4-11. Class diagram of the Physical Connection Graph (PCG)

Figure 4-11 shows all the classes involved in a PCG. The Network Flow End Points (NFEPs) reference addressable network termination points (NWTP).

Resources on the network level may be either uni- or bi-directional. Therefore the same applies to Network Flow Connections (NFC) and NFEPs. Certain restrictions apply. A bi-directional NFC may only have one leaf NFEP (multi-point to point is not supported). The NWTPs referenced by the NFEPs in a bi-directional NFC must of course be bi-directional. The NWTPs referenced in a uni-directional NFC have to be either of the exact type being referenced (source for root NFEP and sink for leaf NFEP) or bi-directional. The latter option allows a migration (evolution) of a point to point uni-directional NFC to either a point to point bi-directional or point to multi-point uni-directional NFC and vice versa.

It should be stressed that NFCs and NFEPs are network technology independent while the NWTPs referred to by NFEPs are network technology specific. Hence, NFEPs are a technology independent *representation* of NWTPs.

4.3.1.3 Nodal Connection Graph

The concept of Nodal Connection Graph (NCG) has been defined to specify the connectivity requirements in a node (terminal). The connectivity requirements specified at the node are almost similar to the LCG, but there are some differences with LCG as in the node it should differentiate what bindings are resolved locally in the node, and which have to be associated with a remote node. Note that this implies a heterogenous interconnection of application level and network level terminations

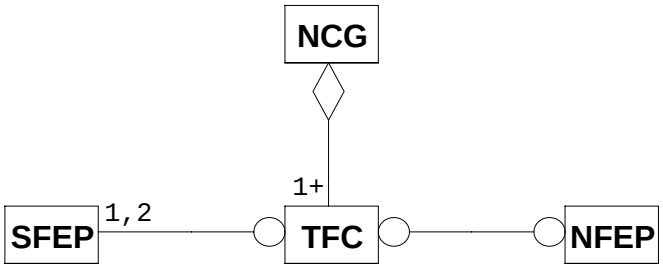


Figure 4-12. Class diagram of the Nodal Connection Graph (NCG)

Figure 4-12 depicts the class diagram of the NCG. The work on the nodal binding is not completed yet. The figure above reflects the current architecture but will most likely be adapted in a next version of this document.

A NCG is a group of one or more Terminal Flow Connections (TFCs). The NCG may be quite complex in its description of each TFC. There may be a need to take into account engineering view concerns and include terminal specific information on terminal devices and operating systems. Further complexity is added if the terminal is itself a distributed entity. However, the NCG needs not to export its complexity outside the node.

TFCs interconnect SFEPs and NFEPs. In the current solution these TFCs only support a point-to-point connection, either between two SFEPs or between a SFEP and a NFEP.

The NCG is used for negotiation between CSM and TCSM for establishing the correspondence between stream interface addresses and network addresses.

4.3.1.4 End Point Fragment

Figure 4-13 depicts all end point classes used in the current CM and their relationship.

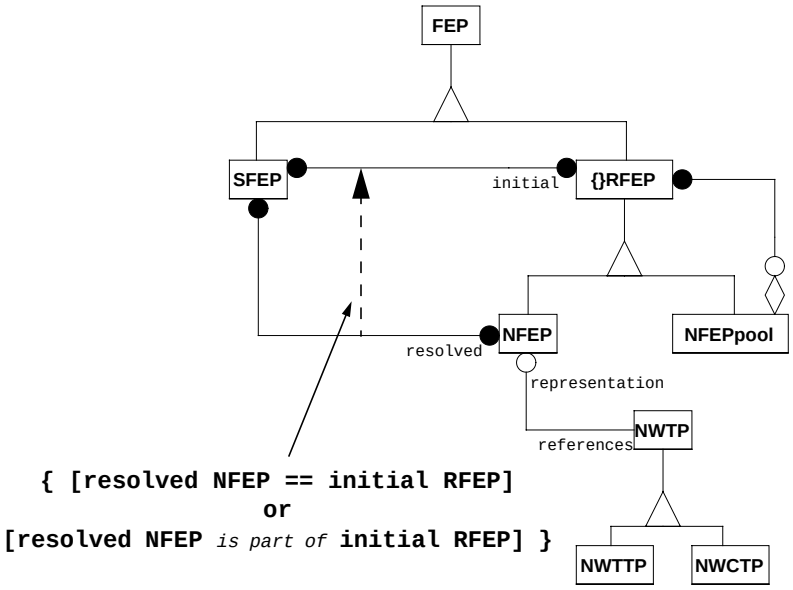


Figure 4-13. End Point Fragment

Both the SFEP and the Resource Flow End Point (RFEP) class are derived from the FEP class, which serves as the generalization of a FC termination. The *abstract*[2] RFEP class is specialized into a NFEP and a NFEPpool class.

A NFEP instance has always a relationship with a Network Termination Point (NWTP). This NWTP class is specialized into a Network Trail Termination Point (NWTTP) and a Network Connection Termination Point (NWCTP) class. NFEPs are a technology independent representation of the technology dependent NWTPs. Future work on this subject may conclude that the NFEP class should be directly related to one or the other derived class, but for now it is saver to keep the relation with the NWTP class.

NFEPpools aggregate RFEPs. This means that a NFEPpool can contain both NFEPs and other NFEPpools. This recursive pool construction is a handy grouping mechanism for managing NFEPs on different levels.

2. So an RFEP can NOT be instantiated.

SFEPs can have an initial relation with RFEPs. This relation will be used to setup TFCs between SFEPs and NFEPs. When the whole process of channel selection and TFC setup is finished, the SFEPs will have resolved relations with NFEPs[3]. The resolved relation is constrained by the initial relation in the sense that a resolved relation can only exist between the SFEP and the initially addressed RFEP or between the SFEP and a (recursively) contained NFEP in the initial addressed RFEP. In the former case the initial RFEP is the resolved NFEP and in the later the initial RFEP is a NFEPpool.

4.3.2 Session Fragment

Session concepts have been introduced in Section 4.1.5. Sessions provide a temporary grouping of objects and resources required to provide an instance of a service. A session can be viewed as providing an “environment” associated with the service instance. In general, this environment includes ownership, state, and can be extended to support concepts such as “user”. For connection management related sessions, the environment includes a view of connections: i.e. the logical (end-to-end service-level connections) or physical (end-to-end network connections) views. In a way, it is similar to the environment provided by a layer network, except that this only exists for the life-time of the session.

This section examines the communication and connectivity sessions. The relations between the sessions, between sessions and connection graph, and some aspects of functionality, including mapping between connection graphs will be considered. Some of these relations will be conveyed by detailed schema’s of information exchanged or held by sessions. For completeness, a discussion on communication aspects of service sessions is included. This should be considered descriptive material included to show the relation between service connection concepts and the CMA.

3. The cardinality of the initial and resolved relation is left open to not limit future work on this subject.

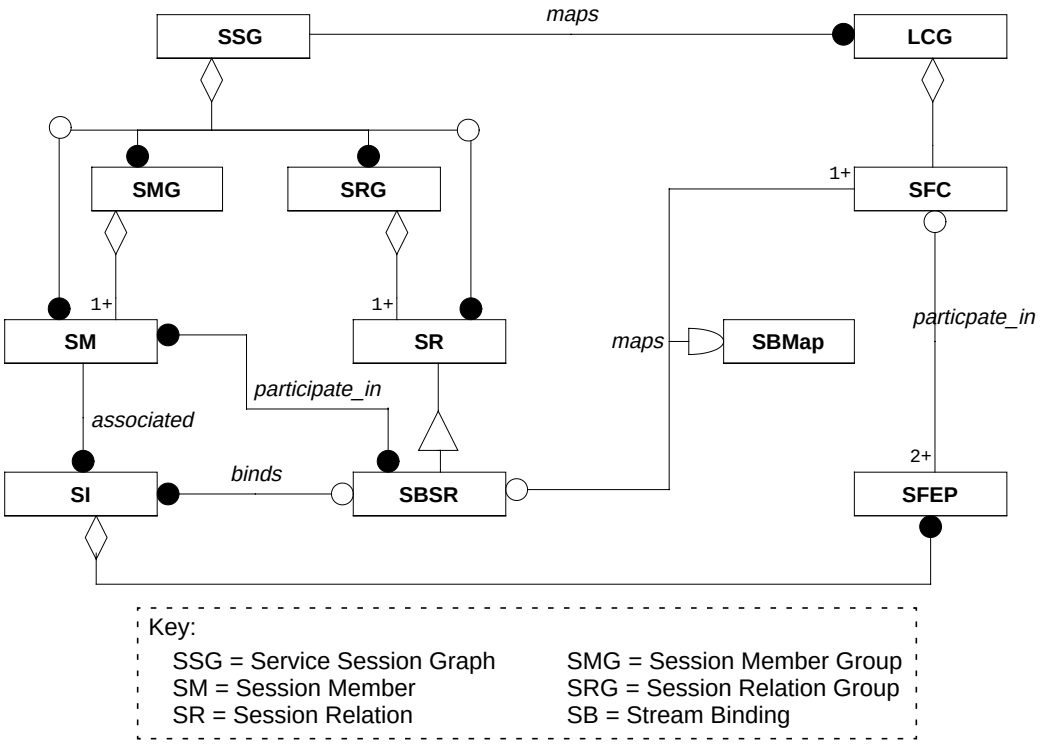


Figure 4-14. Service Session (CM Client) Connection Related Schema

4.3.2.1 Connection aspects of the Service Session

This section has been included to relate service session connection concepts to the service provided by the CMA. It is purely descriptive. Service sessions view connections as a stream binding between stream interfaces. These stream bindings are described by service level information models, such as the service session graph [20], and support multipoint-to-multipoint, multimedia connections.

Figure 4-14 illustrates a service session schema that relates the service session graph to a logical connection graph. The left side of the figure shows a simplified service session graph. This graph is an aggregation of members, such as users, session relations, which can represent any association between session members, and groups of members or relations. In this model, a stream binding is a specialized form of session relation. The stream binding connects the stream interfaces of participating members. Each of these stream interfaces can contain multiple SFEPs.

The service needs to translate its complex service oriented view to one supported by the communication session, whose services are used to establish the stream binding. In TINA, the communication session view is modeled by a LCG (Section 4.3.1.1), part of which is shown on the right side of Figure 4-14. Only the stream binding session relationship and the SIs play a role in this mapping.

4.3.2.1.1. Stream binding SR to Logical Connection Graph Mapping

A stream binding contains a number of stream flows (defined implicitly within the stream binding). The stream binding maps to a number of stream flow connections, see Figure 4-14. This set of stream flow connections may be incorporated in a single logical connection graph. To make a mapping, the appropriate SFEPs associated with the participating SIs in the stream binding need to be determined. This logic is currently part of the service session. (A communication session role in this is for further study, see Section 8.4.)

In the simplest mapping, a stream binding may map to a single stream flow connection, where it expresses a point-to-point or point-to-multipoint unidirectional connection. More generally, a stream binding maps to multiple stream flow connections, and more than one mapping is possible. For instance, a multi-party to multi-party stream binding could be mapped to a number of point-to-multipoint SFCs. Or, a special resource may be needed to perform a binding which may be invisible to binding members (e.g. a bridge or adaptation resource). A number of point-to-point SFCs between the resource and session members could then be used to establish the stream binding.

As well as mapping the stream binding to a set of stream flow connections, it is necessary to map QoS. Each stream flow in the binding needs to have relevant QoS. However, the QoS at the service level should be expressed in terms relevant to the service. For some services, such as a VPN service, the semantics could be connection related, e.g. bandwidth (average and peak), error rate, and delay. However, for other services it may be expressed in terms of video quality: e.g frames per second and jitter. There may be QoS measures associated with stream flow relations, e.g. synchronization between streams.

Service oriented QoS needs to be mapped to connectivity related QoS when a communication service is requested. The QoS mapping could be done either at the stream binding to stream flow connection mapping, or at the stream flow connection to network flow connection mapping (using standard or well known mappings).

4.3.2.1.2. Service session relation to communication service session schema

The service session maintains a user relationship with a communication session (Figure 4-4). This allows it to control the communication session as a whole or manipulate SFCs with in the communication session. The logical connection graph (Figure 4-9) is used to model this relationship. The service session interacts with the communication session to modify or release SFCs by manipulating the logical connection graph shared between them.

The relation may also include controlling the mapping between stream and network flow connections. This is desirable, if for instance, a service session would like two unidirectional point-to-point SFCs mapped to a bidirectional point-to-point NFC. This kind of control may be expressed as a mapping restriction (Section 8.4 outlines further study in this area).

4.3.2.2 Communication Session

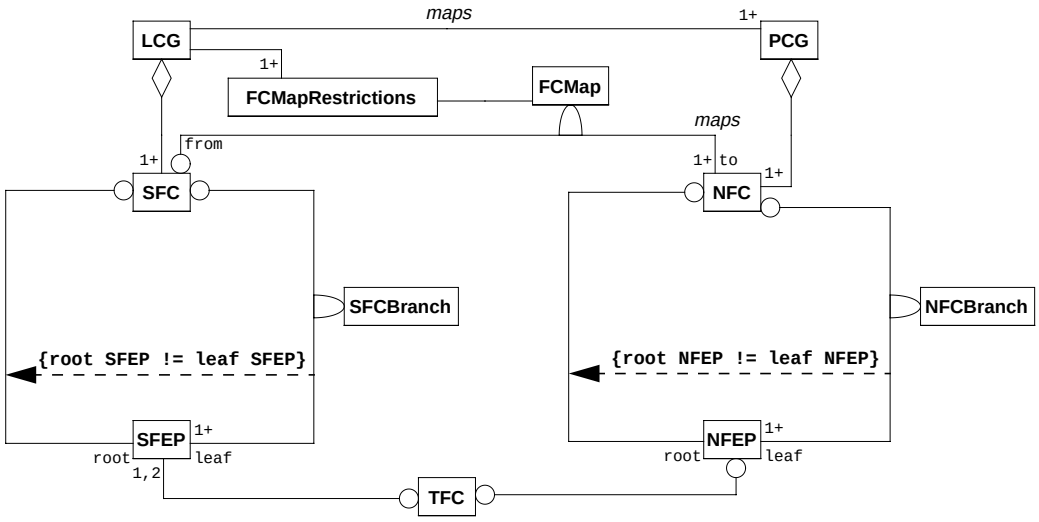


Figure 4-15. Communication session information model

The communication session offers an end-to-end service level connection service. The communication session is responsible for establishing and maintaining SFCs. To support this functionality, the communication session requires terminal components, described by the terminal communication session.

The communication session is responsible for responsible for mapping given SFCs to NFCs and interacting with the connectivity session to establish and control the NFCs. The terminal communication session establishes and maintains the TFC between a SFEP (or an associated terminal device) and a NFEP (or another SFEP). The communication session is responsible for maintaining the relation between individual TFCs and their associated SFCs and NFCs.

To do this, it establishes a coordinating relation with each terminal communication session (Figure 4-4). Each TFC, identified by an SFEP, is associated with a global correlation identifier. The correlation identifier identifies a SFC and an associated NFC. This correlation identifier is used later, either by the communication session or terminal logic, to identify the NFEP to be used in the TFC and hence complete the TFC establishment.

Figure 4-15 shows a schema of the communication session relating the logical connection graph to the physical connection graph. The communication session needs to map the set of SFCs to a set of NFCs, in accordance with its mapping rules and any mapping restrictions supplied by a client. A major part of this mapping is relating SFEPs with a network termination. Initially, each SFEP is associated with a RFEP. These RFEPs are used by the communication session to describe the NFCs which need to be established by the connectivity session. They are resolved to NFEPs by the connectivity session and the layer network controllers. Once the NFEP is known, the TFC connecting the SFEP and NFEP can be completed.

The communication session also needs access to information about connection providers. Connection providers are determined from the RFEPs among which the network flow connections need to be setup. This information must contain connectivity provider selection criteria

4.3.2.2.1. Logical Connection Graph Mapping to Physical Connection Graph Mapping

The mapping from logical connection graph to physical connection graph is reasonably straight forward. Each SFEP is associated with a RFEP, which must be resolved to a NFEP before the mapping can take place. Like stream flow connections, a network flow connection can only be point-to-point or point-to-multipoint. However, there may be more restrictions on modifying a network flow connection: for instance it is difficult to change a point-to-point NFC into a point-to-multipoint NFC later.

Point-to-point unidirectional and point-to-multipoint mappings are generally straight forward: a stream flow connection can be mapped directly to a network flow connection of the same type by finding the appropriate RFEP for each SFEP and determining a connectivity provider that can make the connection.

More complex mappings are possible. A communication session may have the freedom to consider an entire logical connection graph and determine the most economical way to perform the connections. For instance, it could map two uni-directional point-to- point SFCs to a bidirectional NFC. Or, it could find a special resource to support a SFC and make the necessary NFCs between the given SFEPs and the special resource.

The controlling service session may replace restrictions on how the mapping is performed: for instance it could request the CSM to preserve separation for all connections in the LCG, or for point-to-point connections to be established as multipoint connections to allow later additions to a connection.

4.3.2.2.2. Logical Connection Graph Mapping to Nodal Connection Graph Mapping

A SFC normally maps to one or more NFCs and at least two TFCs. From the point of the CSM, each TFC is a simple unidirectional point-to- point connection between a SFEP and either a NFEP or another SFEP. To simplify the logical connection graph to nodal connection graph mapping, each TFC is initially identified with by an SFEP.

4.3.2.2.3. Relation between communication and connectivity session

The communication session maintains a user relationship with a connectivity session. This allows it to control the connectivity session as a whole or manipulate NFCs with in the communication session. The physical connection graph (Figure 4-11) is used to model this relationship. The communication session interacts with the connectivity session to modify or release NFCs by manipulating the physical connection graph shared between them.

This relation may also include some routing restrictions. Routing restrictions can be used to set up specific routing policy: e.g. diverse routes. Other kinds of controls are possible, such as using pre-provisioned resources. However, these require further study and are not currently supported by connectivity session.

4.3.2.2.4. Communication session relation to terminal communication session relation

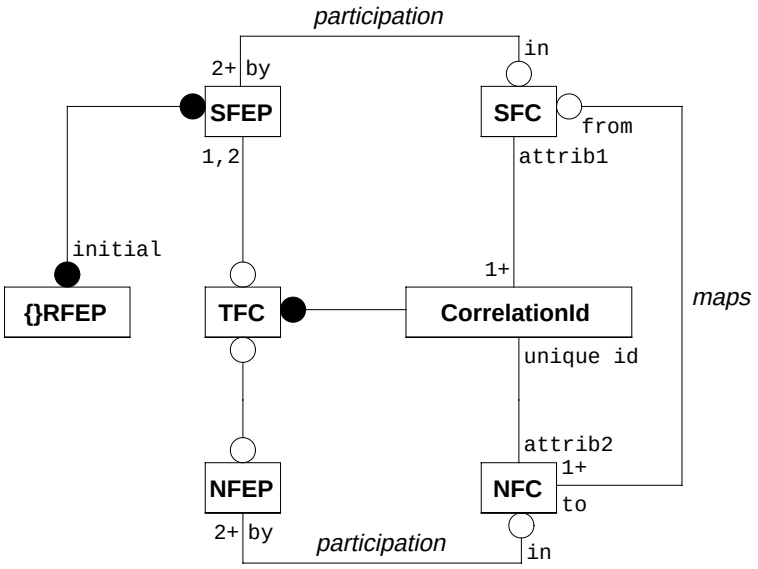


Figure 4-16. Communication session relation to terminal schema

The communication session establishes a coordination relationship with associated terminal communication session. This allows the communication session to control and maintain the relation between an TFC and its associated SFC and NFC.

The communication session sets up a correlation identifier, that identifies a particular mapping from a SFC to a NFC. By associating this correlation identifier with a SFEP within the terminal communication session, it creates a TFC. The correlation identifier can be regarded as an global reference to the particular NFC.

The information model allows a relation of multiple SFEPs with multiple initial RFEPs. In this release, only a point-to-point TFC between a SFEP and NFEP is supported. The former relation prescribes the possible NFEPs that can be associated with a SFEP, while the later describes the completed connection. Connecting multiple SFEPs with a NFEP or visa versa currently requires establishing multiple TFCs. Expanding the TFC concept to support point-to-multipoint as well as point-to-point requires further study.

Figure 4-16 depicts a view on the information model related to the communication session. The participation relation between FCs and FEPs is a generalization of the root and leaf relations. The CorrelationId uniquely identifies the SFC to NFC mapping on a global scale by combining the local name of the NFC and the global name of the related SFC. As well, the CorrelationId identifies all the TFCs associated with a particular SFC to NFC mapping.

As earlier mentioned, the CorrelationId and SFEP are used to create the TFC in the terminal domain[4]. The RFEP and the CorrelationId are passed to the connectivity service to set up the network connection. The CorrelationId will be used to associate the resolved NFEP with the TFC and the corresponding SFEP.

4.3.2.3 Connectivity session

The connectivity session offers an end-to-end network connection service. The connectivity session is responsible for establishing and maintaining NFCs. The connectivity session provides an abstract, technology independent view of network connection to its clients. This abstract view needs to be mapped to technology dependent layer networks.

The connectivity session is responsible for determining which layer networks are needed to support a particular NFC and then mapping each NFC to one or more trails over one or more layer networks. (The number of trails is equal to the number of layer networks involved in establishing a NFC). The connectivity session is responsible for any adaption between layer networks required for trails to inter-connect.

The connectivity session acts on the layer network via layer network coordinator. These can establish trails within their layer network. The connectivity session can manipulate the trail to suit changes to the NFC it supports. The connectivity session uses trail management components to finally resolve the NFEP. Once the NFEP has been resolved, this information can be communicated to the terminal communication session (directly or via the communication session) allowing TFCs and SFC establishment to be completed.

4. It should be stressed that the TFCs are completely unknown outside the terminal domain.

4.4 Computational Viewpoint

From the computational viewpoint [4], connection management functions are described as provided by a set of interacting computational objects, of different types. This allows the distribution of the control of large networks to several processes, satisfying requirements of load, scalability, geographical and administrative partitioning. This chapter describes the types of computational objects of the TINA connection management architecture, and how they can be interrelated.

Figure 4-17 gives an overview of the computational objects associated with the connection management architecture. It groups them according to conceptual levels introduced in Section 4.1.2. We will discuss the objects according to these groupings. Conceptual groupings and associated objects are listed below:

- Communication session related objects
  - Communication Session Manager Factory (CSMF)
  - Communication Session Manager (CSM)
  - Terminal Communication Session Manager (TCSM)
- Connectivity session related objects
  - Connection Coordinator Factory (CCF)
  - Connection Coordinator (CC)
  - Flow Connection Controller (FCC)
- Layer network related objects
  - Layer Network Coordinator (LNC)
  - Trail Manager (TM)
  - Tandem Connection Manager (TCM)
  - Terminal Layer Adaptor (TLA)
- Subnetwork related objects
  - Connection Performer (CP)

At each session related level, there is a factory object, a CSMF or a CCF, to create session control objects because these only exist for the lifetime of a service. These objects are not necessary at resource related levels.

Each grouping has an “environment” related object: the session control objects, such as the CSM and CC, for service-oriented layers, and resource management objects, such as the LNC and CP at resource oriented levels. These objects allow clients to establish connections within the related environments.

Connection control related interfaces are present at each level, but for various reasons, at some levels they have been described as part of connection control objects, while in others they have been associated with the environment related object. An explanation for each of these groupings will be given in the section related to each level.

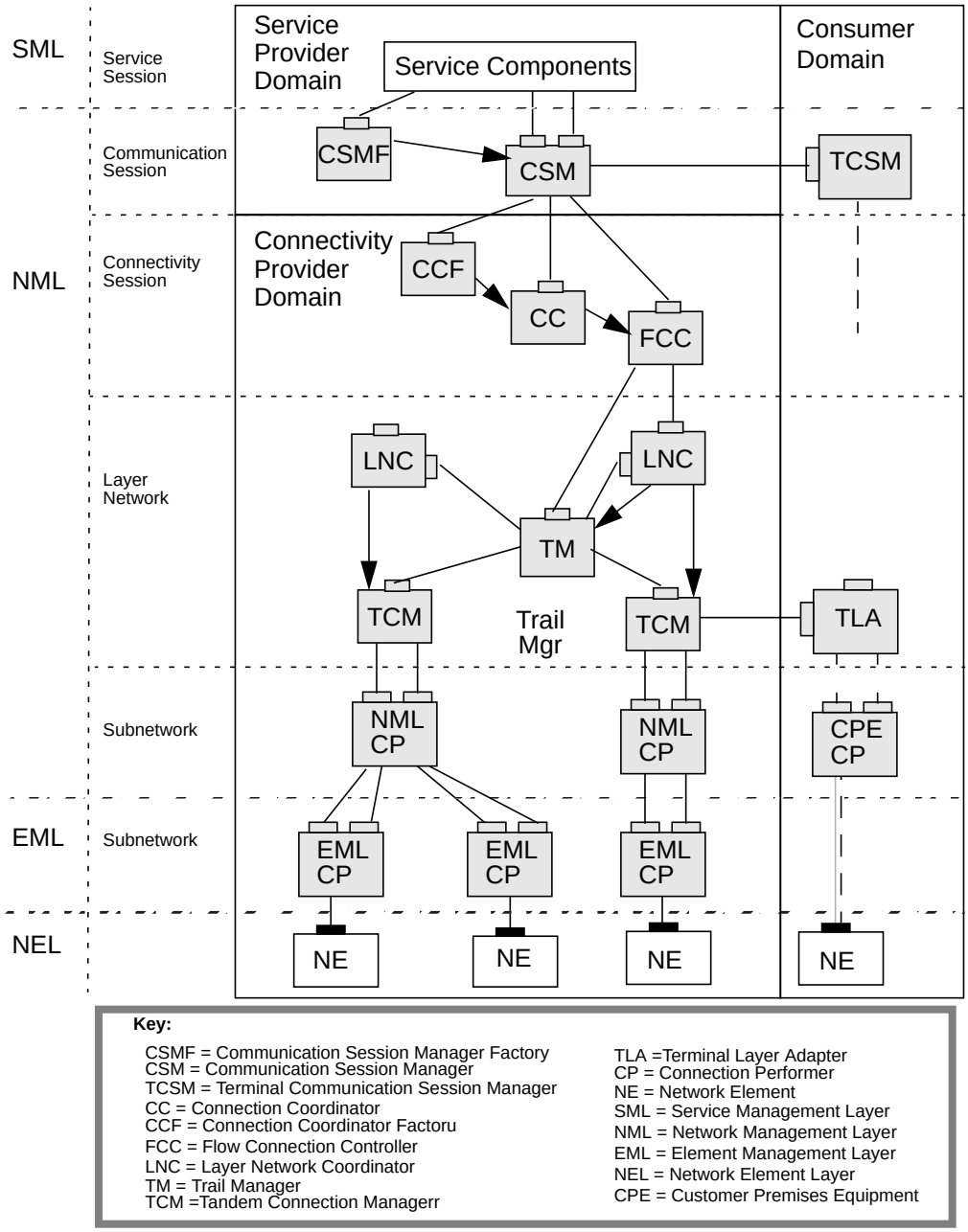


Figure 4-17. Overview of the CMA: the Computational Viewpoint

Object and interface relations are not prescriptive, as long as their dynamic behaviour remains unchanged. In general, interfaces between domains may be considered prescriptive. If functionality to a particular level is provided, then the interfaces visible between that level and the next are prescriptive. Also note that technology dependent interfaces may need to be specialized for that technology. Finally, levels below the LNC are a TINA idealization. They may be replaced to suit a particular technology.

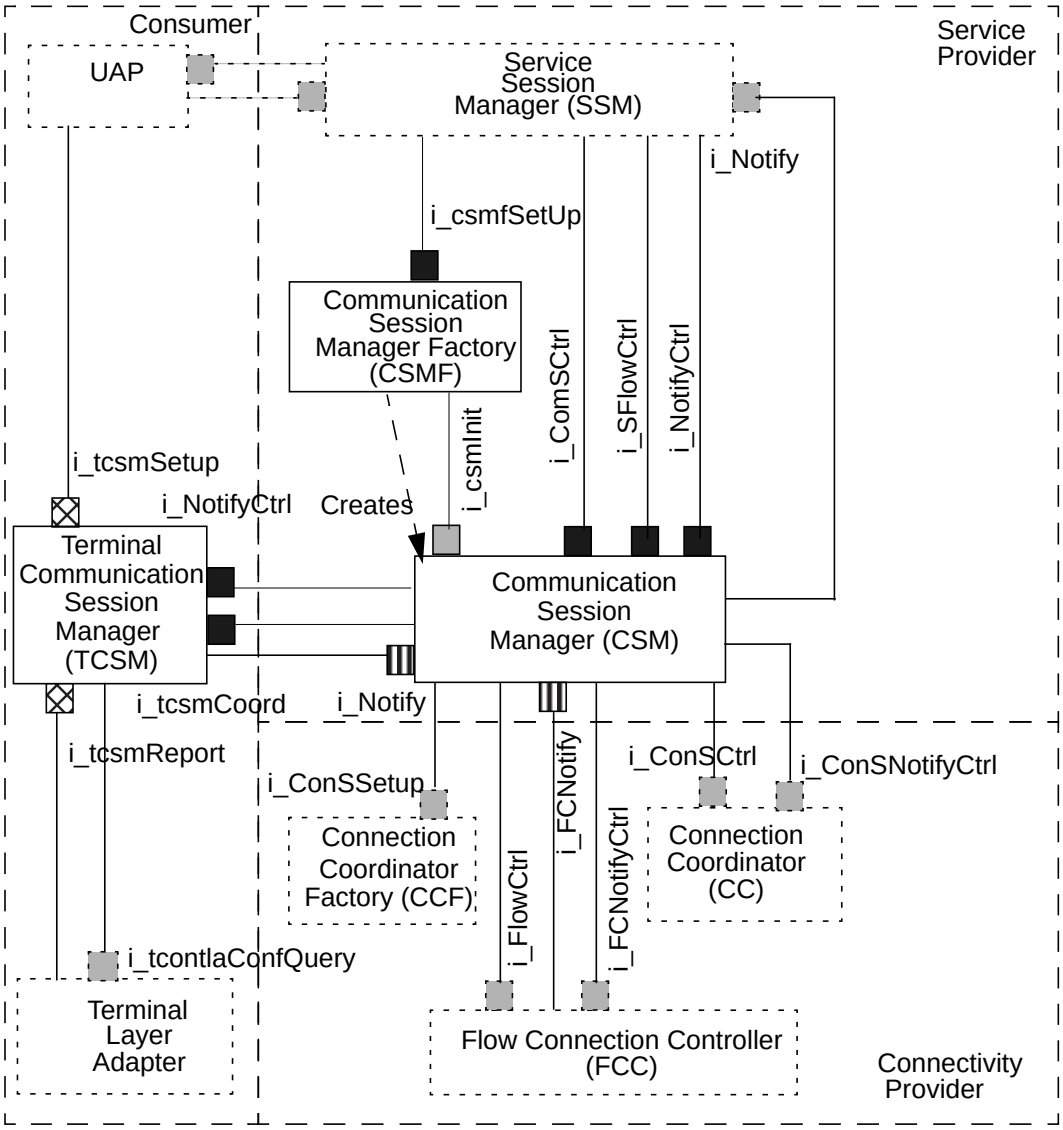


Figure 4-18. Communication session related objects

4.4.1 Communication Session related objects

The communication session is related to a logical view of connections, as seen by service level clients. This logical view, which allows connections between the SFEPs of service components, is mapped to a physical connection view, which provides a technology independent abstraction of network connections. Figure 4-18 introduces the computational objects in this grouping. There are three principal components: the Communication Session Manager Factory (CSMF), the Communication Session Manager (CSM), and the Terminal Communication Session Manager (TCSM). They are shown with objects from other groups to give readers insight in relations between the groupings.

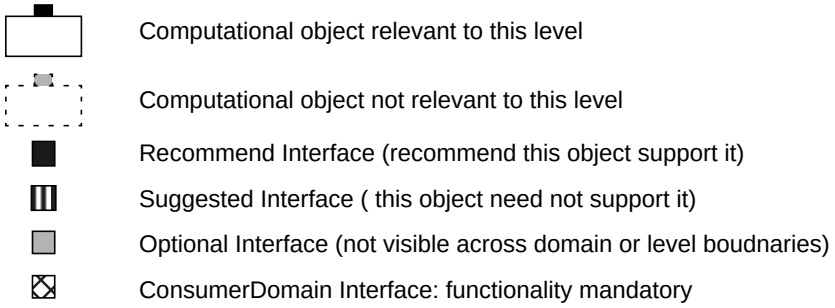


Figure 4-19. General Computational Diagram Key

Figure 4-19 shows a key which explains the meanings of the different interface and object descriptors. This key applies to computational object diagrams throughout Section 4.4. Note that only computational objects related to the conceptual level being discussed are shown with a connected line, all others are shown by a dotted line.

Like all service oriented layers, the communication level includes a factory object, the CSMF, to allow the creation of session control objects. As large numbers of communication session requests are likely, the session control and factory objects have been separated.

The CSM is the session control object. It allows clients to set up stream flow connections within the communication session. In line with earlier service architecture definitions the CSM may control multiple communication sessions. A simple interface to allow this is suggested. However neither the interface or the cardinality is prescriptive.

Unlike the connectivity level COs, there are no separate COs to support connection control. Instead, the CSM also supports stream flow connection control interfaces. This interface type is very similar to the FCC's i_FlowCtrl interface, supporting the same operations but specialized for the LCG. So from an interface and dynamic interaction perspective these levels are equivalent.

The communication level interface grouping and object separations are suggested because the mapping of SFCs to NFCs may not be independent of each other. For instance, two point-to-point unidirectional SFCs may be mapped to the one point-to-point bidirectional NFC (as bidirectional connections are not supported at the communication level). In general, it is perceived that each communication session will support relatively few SFCs, so no great need for distributed management is required. However, this is not mandatory, and connection control may be separated if desired. In this case the grouping of the connectivity level is suggested.

The communication session components interact with connectivity session components and support the interfaces required by the connectivity session components. Note that notification interfaces are listed as optional, as it may be found appropriate to receive and manage notifications separately.

4.4.1.1 Communication Session Manager Factory

A Communication Session Manage Factory (CSMF) coordinates the establishment of communication sessions and controls access to communication session managers.

A request to establish a communication session will result in the creation of a communication session and access to its control interface. CSMs are created in accordance with the service provider's communication management policies. A CSM may manage multiple communications sessions, each controlled by a separate interface. A request to establish a communication session may not necessarily create a new CSM.

A CSMF supports capabilities to:

- Establish a communication session, with a set of SFCs if desired.
- General communication session management: This includes the ability to list sessions, acquire session control interfaces, and delete sessions. These operations should to allow managers to intervene in communication session control if necessary, aid recovery, and allow the transfer of control of sessions.

A CSMF may support optional capabilities to:

- Set communication session management policy
- Explicitly create and manage CSMs (assuming multiple communication sessions are managed by a single CSM).

4.4.1.1.1. i_csmfSetup

This interface allows the establishment of communication sessions and their associated computational objects. It also allows some general communication level management operations. These operations are primarily included to aid management, such as recovery and transfer of communication session control.

- Establish a communication session
  - This operation is used for setting up a communication session. If it is successful, this operation creates a communication session and i_ComSCtrl interface which allows users to manipulate the new session. When a Communication Session setup is requested, multiple SFCs can be established at the same time. A client can indicate if they wish to wait for SFCs to be setup, or have their establishment confirmed later (by notification). Control interfaces associated with each SFC are returned.
- List communication sessions:
  - Return a list of communication session identifiers. Filtering may be used to manage which communication sessions are listed.
- Get a communication sessions:
  - Return the control interface references to one or more communication sessions associated with the given communication session identifier.
- Delete communication sessions:

- Delete the communication session(s) associated with the given identifiers.This is intended for management.

4.4.1.1.2. Required Interfaces

The following interfaces may be required when creating and initializing CSMs. Their use is optional. It is suggested that i_csmInit is used when a CSM controls multiple communication sessions. The initialization process is not visible to users of the CSMF. Object creation depends on the services of the DPE.

i_csmInit: This interface allows the CSMF to establish a new communication session with specified SFCs in a particular CSM, see Section 4.4.1.2.4. This required interface is optional.

i_ComSCtrl: This interface allows the CSMF to initialize an CSM with requested SFCs, if the CSM supports a single communication session, see Section 4.4.1.2.1. This required interface is optional.

4.4.1.2 Communication Session Manager

The Communication Session Manager (CSM) can manage a number of communication sessions. Each communication session in turn can establish and maintain one or more stream flow connections which are described by a logical connection graph.

A CSM client may activate, deactivate, release or query the status of a communication session. The CSM also support requests to set up, modify and delete stream flow connections within a communication session. SFC manipulations are supported by a separate interface for communication session manipulation, following the separation supported by the connectivity session.

The CSM is responsible for mapping stream flow connections to network flow connections. Stream flow connections are unidirectional point-to-point or point-to-multipoint. Network flow connections may be uni- or bi-directional point-to-point or uni-directional point-to-multipoint.

CSM can be given freedom to establish mappings between SFC and NFC, but may get specific requests to: e.g. maintain separation (i.e. don’t amalgamate or split), allow multiparty, multiplex stream flow connections over a physical connection and so on. Mapping restrictions may be made when establishing a communication session of SFC. Restrictions can be modified later. Mappings require further study, see Section 8.4.

The CSM may also discover resources, or be requested to use particular resources, in association with a SFC. For instance, a CSM may need to use a special resource to match stream protocols (e.g. a NSTC to PAL converter). The CSM may be free to find its own converter, or requested to use a particular converter. Associates may be identified by a client when establish a communication session or SFCs. Associates may be added or removed during the communication session lifetime. The use and role of associates needs further study, see Section 8.4.

4.4.1.2.1. i_ComSCtrl

i_ComSCtrl: This interface provides operations to establish stream flow connections and control the communication session.

- Setup flow connections
  - Setup a given list of SFCs. Each SFC is described as a set of SFEPs and QoS parameters. A successful operation causes the creation of a new SFC and its related control interface (i_SFlowCtrl). The client may specify if they wish to wait for successful completion, or be notified of the completion (or failure) of this operation. SFCs can be established in active or inactive state.
- Activate flow connections (or entire communication session)
  - Activate one or more (possibly all) SFCs associated with this communication session. Returns information regarding the success of this operation for all specified SFCs.
- Deactivate flow connections (or entire communication session)
  - Deactivate one or more (possibly all) SFCs associated with this communication session.
- Release flow connections
  - Release one or more (possibly all) SFCs associated with this communication session. This operation does not release the communication session. Releasing all SFCs will end the communication session.
- Get communication session information
  - Return state of communication session, and list of all SFCs by name.
- Get flow connection control interfaces
  - Return the control interfaces of one or more (possibly all) SFCs associated with the communicant session.
- Get notification control interface
  - Return the notification control interface reference associated with this communication session. This interface allows a client to modify notifications, including destinations and notification types of interest.
- Modify mapping:
  - Modify the mapping controls, which specify restrictions on the mapping of SFC to NFCs in this communication session.
- Add and delete associates:
  - Associates represent special resources that may be used by a communication session. These operations let clients specify particular special resources they would like associated with this communication session.

4.4.1.2.2. i_SFlowCtrl

i_SFlowCtrl: This interface provides operations to control a specific stream flow connection. A number of operations are specified as acting on selected branches rather than a flow connection. To act on an entire flow, an all flag may be set in these operations.

- Add flow connection branches
  - Add branches to the SFC associated with this interface. Branches can be added in an active or inactive state. If they are added in an active state, they do not need to be specifically activated later.
- Delete flow connection branches (or entire flow)
  - Delete specified branches of the SFC associated with this interface. All branches can be designated with an “all” flag. This will result in releasing the entire SFC. (Note that the SFC can also be released by the i_ComSCtrl interface.)
- Activate flow connection branches (or entire flow)
  - Activate one or more (possibly all) specified branches of the SFC associated with this interface. By specifying all branches, the entire SFC is activated.
- Deactivate flow connection branches (or entire flow)
  - Deactivate one or more (possibly all) specified branches of the SFC associated with this interface. By specifying all branches, the entire SFC is deactivated.
- Modify flow connection branches (or entire flow)
  - Modify the QoS of one or more (possibly all) specified branches of the SFC associated with this interface. By specifying all branches, the entire SFC is modified.
- Get flow connection information
  - Return a description including overall status, QoS, and the status and QoS of individual branches.
- Modify mapping:
  - Modify the mapping restrictions which specify how this SFC maps to NFCs (may be used to override general session restrictions).
- Add and delete associates:
  - Specify or remove associates to be used in conjunction with this SFC. This allows particular resources to be associated with a particular SFC.

4.4.1.2.3. i_FCNotify

This interface is used in conjunction with the connectivity level COs. It receives notifications from the FCCs controlled by this CSM regarding their associated NFCs. Though we have positioned this interface as part of the CSM, this is only a suggested grouping. Other managers may be found to be more appropriate. It provides the following operation:

- *flow_connection_status_changed*

- This operation is used to notify the a client when the operational state of one or more branches of the flow connection changes.

4.4.1.2.4. i_csmInit

i_csmInit: An initial interface used to control the CSM object. This interface is necessary if a CSM supports multiple communication sessions. Note that operations associated with this interface only relate to communication sessions managed by the CSM.

- Setup communication session:
  - Create a new communication session and return its control interface. A client may specify SFCs to be setup with the communication session if desired.
- List communication sessions:
  - List all communication sessions controlled by this CSM
- Get communication sessions
  - Get interface reference of specified communication sessions.
- Release communication sessions:
  - Release all specified communication sessions and their associated SFCs.
- Quit:
  - Release all remaining communication sessions and delete CSM.

4.4.1.2.5. Generic or Predefined Supported Interfaces

i_NotifyCtrl: The CSM may support a notification control interface to allow the control of notifications associated with this session. See Section 4.4.1.4.2.

i_Notify: The CSM may support a notification interface from which it aquires notificaton from the TCSM. These notifications can be used to indicate changes in the status of an TFC controlled by the TCSM. (Note: the TCSM only has a local view, so it can't change the overall SFC, but it can change the local connection to modify QoS or type or even remove the connection if necessary.) See Section 4.4.1.4.1.

4.4.1.2.6. Required interfaces

The following interfaces are required by the CSM (assuming ConS)[5]:

i_ConSSetup: operations to support the establishment of connectivity sessions, see Section 4.4.2.1.1

i_ConSCtrl: operations to support the control of a connectivity session and the establishment of flow connections, see Section 4.4.2.2.1

i_FlowCtrl: Once the FCC is created, this interface can be used to manipulate NFCs.

5. These interfaces are necessary if the NRA is being supported or a ConS connectivity provider is used.

i_FCNotifyCtrl: operations to support the control of the emission of notifications from the FCC, see Section 4.4.2.3.2

i_Notify: The CSM requires a notification interface, to which it directs notifications associated with the session, see Section 4.4.1.4.1.

i_tcsmCoord: This interface allows the coordination between nodal and physical(network) parts of a logical connection. This interface is located on the TCSM. See Section 4.4.1.3.1.

4.4.1.3 Terminal Communication Session Manager

The Terminal Communication Session Manager (TCSM) is responsible for establishing the nodal part (or TFC) of a SFC. It cooperates with the communication session manager responsible for the stream flow connection. It also interacts with user applications for the relation of SFEPs with particular connections.

We assume that there is one TCSM per terminal, special resource, or any node that terminates a stream flow connection. A TCSM can manage multiple TFCs and terminal communication sessions. The TCSM manages the connection setup for the node: it does not participate in actual connections[6]. The TCSM knows the details of the nodal binding which are hidden from the CSM.

When setting up a stream interface, the associated CO must register all associated SFEPs with the TCSM, with some description of the associated TFC[7]. These SFEPs can be used to locally identify a TFC. Initially, a TFC is known by a TCSM and a SFEP.

Later, when the CSM is establishing a SFC, it needs a way of associating the network part of the connection it is to establish with the terminal part of the connection. To do this, it passes a unique[8] identifier, called a correlation identifier, to each TCSM associated with the SFC. This identifier identifies both the associated SFC and NFC, by including both their names. It is correlated with the TFC of a particular node by associating it with a particular SFEP. At this stage the NFEP has not been resolved.

This correlation identifier is also passed from the communication session to the connectivity session, and from the connectivity session to a trail in a layer network. Once a unique NFEP is either created or selected, during the establishment of the layer network trail, the correlation identifier can be passed to the TLA. The TLA then informs the TCSM of the chosen NFEP, using the correlation identifier to identify the specific TFC. At this point the TCSM can complete the connection. By using a global correlation identifier, the Trail Managers are not required to associate an SFEP with each NFEP.

---

6. This is in line with the stream channel model, though terminology and some operations have changed
7. The exact format of this TFC requires further study. It is likely to be highly platform dependent and refer to devices, such as speakers or cameras with in the terminal, or to particular communication devices within the terminal. For these reasons, the details of the TFC, apart from associate SFEPs are not visible to the network.
8. Unique with in the service provider and network provider domains (at the time!)

To allow the inclusion of non-TINA compliant connectivity providers, TLA to TCSM operations are also available between the CSM and TCSM

4.4.1.3.1. i_tcsmCoord

i_tcsmCoord: This interface allows the coordination between nodal (TFC) and physical (network - NFC) parts of a stream flow l connection.

- Correlate:
  - Relate a TFC, locally identified in the node by an SFEP, to a unique correlation identifier that identifies the network part of the connection with which it is to be associated.
- Resolve:
  - Relate an SFEP (and TFC) to a RFEP (i.e NFEP or NFEPpool). This operation allows the CSM to specify the NFEPpool related to a TFC (locally identified in the node by the SFEP), if the TCSM was not able to resolve this locally.[9]
- Associate:
  - Associate a TFC, specified by a correlation identifier, with a particular NFEP. The TCSM can complete a TFC on receiving this notification.[10]
- Disassociate:
  - Disassociate a nodal binding, specified by a correlation identifier, from a particular NFEP. This allows an SFC to migrate from one NFC to another if required. The CSM may also use this command when release an SFC or SFC branch.
- Query:
  - Report on state of a given TFC (identifier by an SFEP). This allows the CSM to establish the overall state of an SFC if necessary.
- Modify:
  - Modify the QoS or type description of a stream (identified by an SFEP). This allows the CSM to modify the overall SFC QoS if necessary.
- Invalidate:
  - Invalidate a correlation identifier. When a SFC is released, its correlation identifier is no longer valid and must be removed. To do this, the CSM must invalidate the SFC at each associated node. When an SFC branch is released, its correlation identifier is no longer valid for the node terminating that branch. The CSM must then invalidate the correlation identifier with the associated TCSM.
- Activate: Activate a TFC. This allows the CSM to activate the SFC

9. In this case, the SFEP was associated with some kind of terminal device identifier, which could be resolved by the service provider or connectivity provider to an appropriate RFEP).

10. This operation is equivalent to the i_tcsmReport Associate operation. It allows a TFC to be completed at the request of the CSM if necessary. Normally we avoid this due to the overhead involved.

- Deactivate: Deactivate a TFC. This allows the CSM to deactivate the SFC

4.4.1.3.2. i_tcsmSetup

i_tcsmSetup[11]**:** This interface allows the TCSM to interact with a UAP to associate SFEPs with terminal data necessary for stream flow connection establishment and modification. This interface is still under discussion. In particular, operations for local modification of SFEPs and TFCs still need discussion, as some of this function may be provided by the DPE.

- Resolved SFEP Registration:
  - Register a SFEP reference and resolve to an RFEP, or terminal device identifier (if TCSM can not locate an RFEP). Setup should specify a TFC. If no TFC is specified some default will be assumed (e.g associate SFEP and its CO with any available RFEP).
- Unresolved SFEP Registration:
  - Register a SFEP reference. Setup may specify a TFC, but no RFEP or terminal identifier will be returned.
- Modify SFEP:
  - Modify the QoS of a SFEP and any TFC associated with that SFEP: allow the local modification of QoS parameters within a stream binding or SFC.
- Remove SFEP Registration:
  - This will release any associated TFC (if it still exists), and remove the SFEP registration.
- Release TFC:
  - Release a TFC associated with the currently associated with an SFEP. This will not remove the SFEP registration.
- [optional] Modify State:
  - Modify the state of an SFEP (i.e. deactivate or activate): allows the local activation or deactivation of a TFC (and corresponding branch of the SFC)
- [optional] Resolve:
  - Resolve a SFEP reference to an RFEP or terminal device identifier. A TFC may be specified. This may be used to associate a SFEP with a new TFC or to find the RFEP associated with the SFEP (e.g. if SFEP previously setup unresolved)

11. This interface specifies descriptive interactions between the user application and the TCSM.

4.4.1.3.3. i_tcsmReport

i_tcsmReport[12]**:** This interface allows the TCSM to receive information from the TLA notifying it of changes affecting nodal connections it supports. This interface requires further study and the operations associated with it should expand, Section 8.5.

- Associate:
  - Associate a TFC, specified by a correlation identifier, with a particular NFEP. The TCSM can complete a TFC on receiving this notification.
- NFEPStatusChange:
  - notify TCSM that an NFEP associated with a TFC has failed or degraded.

4.4.1.3.4. Generic or Predefined Supported Interfaces

i_NotifyCtrl: The TCSM may support a notification control interface to allow the control of notifications associated with terminal communication sessions, see Section 4.4.1.4.2.

4.4.1.3.5. Required interfaces

The following interfaces are required by the TCSM (assuming ConS)[13]:

i_Notify: The TCSM requires a notification interface, to which it directs notifications associated with the session, see Section 4.4.1.4.1. This allows the TCSM to report changes in SFEP or TFC status and other problems.

i_tcontlaConfQuery: The TCSM can query the TLAs associated with each layer network termination to find the status of NFEPs and NFEPpools, see Section 4.4.3.4.3. The relation between the TCSM and TLAs requires further study, see Section 8.5. Initially, we will reuse the interface specified for the TCon reference point.

4.4.1.4 Generic Management Interfaces

The following interfaces are generic management interfaces. They assume a generic notification format that can be adapted to different semantics. In time, a notification service should be come available. These interfaces should allow graceful migration to that service, as they do not assume particular semantics and allow centralization of notification control if desired. Any object that wishes to use these generic notifications should require the i_Notify interface. Any object that wishes to support notification management should support the i_NotifyCtrl interface.

4.4.1.4.1. i_Notify

This interface receives notifications from any source.

- Events:

12. This interface specifies descriptive interactions between the TCSM and the TLA.
13. These interfaces are necessary if the NRA is being supported or a ConS connectivity provider is used.

- Receive one or more notifications. A generic notification format is assumed allowing notifications from many different sources to be accepted. This generic format includes basic notification type and instance identifier attributes, and an extensible list of attributes of any type.

4.4.1.4.2. i_NotifyCtrl

i_NotifyCtrl: This interface is used to control notifications. It need not directly belong to the object being managed, but could belong to a notification server of object group manager. It is used to enable and disable notifications, and to set destinations of notifications. Multiple destinations for events are supported.

- enable notifications
  - Enable distribution of notifications to a given list of destinations.
- Disable notifications
  - Disable distribution of notifications to a given list of destinations.
- Set notification destinations
  - Associate one or more destinations, specified by an i_Notify interface reference with a set of specified (possibly all) notifications types. This command can also be used to modify the set of notifications desired.
- Remove notification destinations
  - Disable distribution of notifications to a the given list of destinations, identified by registration id, and remove these destinations from further use.

4.4.2 Connectivity Session related objects

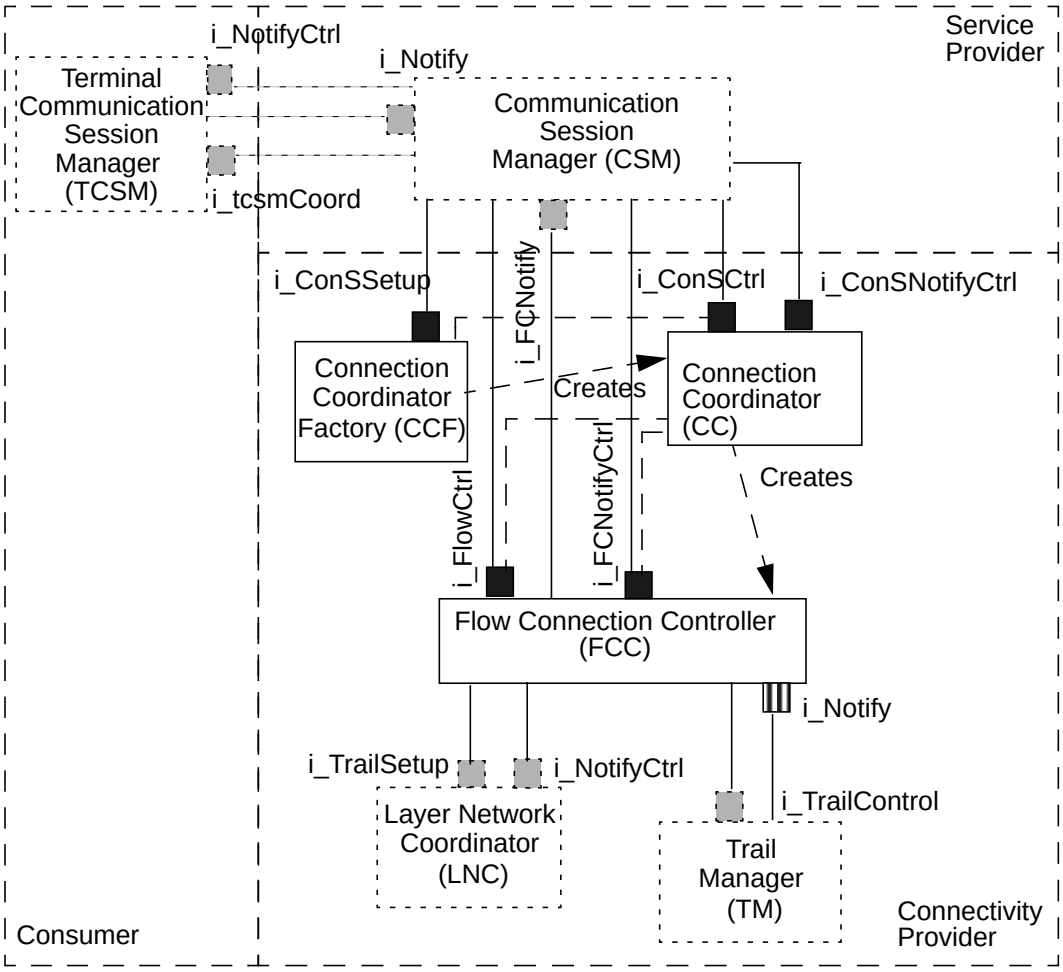


Figure 4-20. Connectivity session related objects

The connectivity session is related to a technology independent abstraction of network connection. This view is used by the communication level to establish the network part of its stream flow connections. This abstracted view must be mapped to a suitable layer network (or networks). Figure 4-20 illustrates the connectivity level computational objects. There are three principal COs: the Connection Coordinator Factory (CCF), the Connection Coordinator (CC) and the Flow Connection Controller (FCC). As before, they are illustrated in association with objects from other groups with which they interact.

The connectivity level includes a factory object, the CCF, to allow the creation of session control objects (CCs). As large numbers of connectivity session requests are likely, the session control and factory objects have been separated, as for the communication level.

The CC is the session control object. It allows clients to setup and release network flow connections within the connectivity session, and to manipulate the connectivity session as a whole. In keeping with the ConS reference point definitions, we recommend a CC be associated with a single connectivity session. However this cardinality is not prescriptive, as long as the interface and interaction dynamics are maintained.

At the connectivity level, session control and connection control have been separated at the object level. Clients can manipulate NFCs through the FCC object. This allows to distribute the management of the connectivity session as flexibly as desired. In the following description a single NFC is associated with one FCC. Neither this cardinality nor the object groupings are prescriptive, as long as the interfaces and interaction dynamics are maintained.

The connectivity session components interact with layer network related COs and support the interfaces required by them. Note that notification interfaces are listed as optional, as it may be found appropriate to receive and manage notifications separately.

4.4.2.1 Connection Coordinator Factory (CCF)

The Connection Coordinator Factory (CCF) serves as the factory object for connectivity sessions. It provides access to the connectivity level, and its technology independent network view. It has one interface, called `i_ConSSetup`, and this interface provides the operation for setting up a connectivity session.

4.4.2.1.1. i_ConSSetup

This interface allows the establishment of connectivity sessions and their associated computation objects. It also allows some general connectivity level management operations. These operations are primarily included to aid management, such as recovery and transfer of connectivity session control.

- Establish a connectivity session
  - This operation is used for setting up a connectivity session. If the invocation is successful, this operation creates a CC object which provides operations for manipulating the newly created connectivity session. When a connectivity session set up is requested, multiple NFCs can be established at the same time. Control interfaces associated with each NFC are returned.
- List connectivity sessions:
  - Return a list of connectivity session identifiers. Some filtering may be used to manipulate which connectivity sessions are listed.
- Get a connectivity session:
  - Return the control interface reference to a connectivity session associated with the given connectivity session identifier.
- Delete a connectivity session:
  - Delete the connectivity session(s) associated with the given identifiers. This operation is intended for management.

4.4.2.1.2. Required Interfaces

The following interface are suggested for initialization of the CCs that support a single connectivity session. The creation of objects depends on DPE services.

i_ConSCtrl: Once the CC is created, the CCF can use this interface to establish any requested NFCs.

4.4.2.2 Connection Coordinator (CC)

The connection coordinator (CC) is created by the CCF. It is recommended that an instance of this object exists for each connectivity session that is setup. A Connection Coordinator object provides operations for manipulating the associated connectivity session including the connectivity session release operation. When a connectivity session is released, the associated Connection Coordinator object is deleted. It allows for NFCs established within the connectivity session, and to be activated, deactivated or released individually or together. A Connection Coordinator object offers the interface described below.

4.4.2.2.1. i_ConSCtrl

This interface provides operations for setup, activation, deactivation, and release of one or more flow connections of the connectivity session associated with the Connection Coordinator object. It also provides a connectivity session query operation and an operation to aquire a notification control interface associated with the connectivity session.

This interface provides the following operations:

- Setup flow connections
  - This operation is used for setting up one or more NFCs within the connectivity session associated with the CC. If the invocation is successful, this operation creates one or more FCC objects which provides operations for manipulating the newly created NFCs. When a NFC set up is requested, the client should specify at least one branch of the NFC. NFCs can be established in an active or inactive state.
- Activate flow connections
  - This operation is used for activating one or more (possibly all) NFCs of the connectivity session associated with the CC. One of the output is List of triples of the form <FC, FEP, Status> where FC is the name of a NFC, FEP is the name of an endpoint (root or leaf) of the NFC referenced by FC, and Status is either “Activated” or “UnableToActivate”.
- Deactivate flow connections
  - This operation is used for deactivating one or more (possibly all) NFCs of the connectivity session associated with the CC.
- Release connectivity session

- This operation is used for releasing the connectivity session associated with the CC. When the connectivity session is released, all component NFCs are also released and the CC is deleted.

- Release flow connections
  - This operation is used for deactivating one or more (possibly all) NFCs of the connectivity session associated with the CC.
- Get notification control interface
  - Return the `i_ConSNotifyCtrl` interface reference associated with this connectivity session. This interface allows a client to modify enable or disable notifications at a session level or change the default destination.
- Get connectivity session info
  - This operation is used for retrieving the connectivity session information. Outputs are State of the connectivity session and List of names of NFCs that are components of the connectivity session.
- Get flow connection control interfaces
  - This operation is used for obtaining the references to the `i_FlowCtrl` and `i_FCNotifyCtrl` interfaces associated with one or more (possibly all) flow connections of the connectivity session associated with the CC.

4.4.2.2.2. i_ConSNotifyCtrl

This is a suggested interface for controlling notifications at the session level. This allows notifications associated with the connectivity session and its associated NFCs to be controlled together or individually, as desired.

- Enable flow connection notification
  - Enables notifications associated with either listed NFCs or the connectivity session (i.e. all NFCs in the connectivity session).
- Disable flow connection notification
  - Suspends sending notifications associated with either listed NFCs or the connectivity session (i.e. all NFCs in the connectivity session).
- Update flow connection notification destination
  - This operation is used to set the default flow connection notification destination parameter for this connectivity session only.

4.4.2.2.3. Required Interfaces

The following interface are suggested for initialization of the FCCs:

i_FlowCtrl: Once the FCC is created, the CC can use this interface to initialize NFCs.

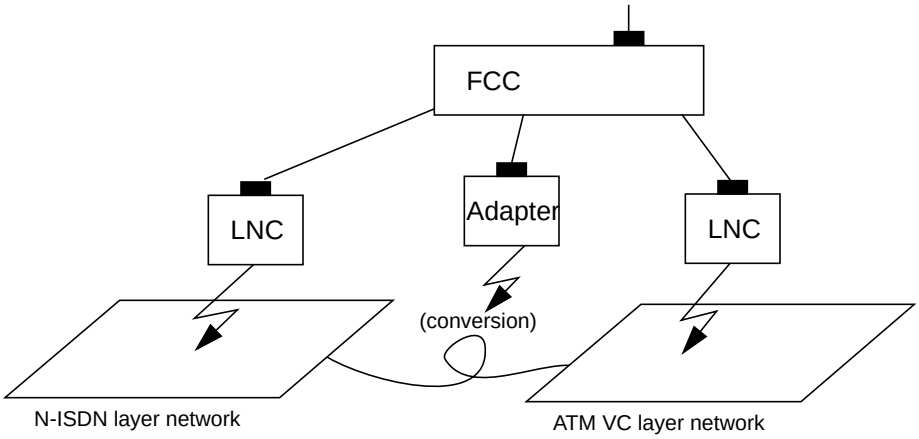


Figure 4-21. Control scenario for layer networks with peer-to-peer

4.4.2.3 Flow Connection Controller

One instance of this object type exists for each Network Flow Connection (NFC) established. The Flow Connection Controller (FCC) is created by the CC associated with the connectivity session of which the NFC is a part. A FCC object provides operations for controlling the associated NFC including the flow connection release operation. When a flow connection is released, the associated FCC object is deleted.

The FCC responsible for translating the NFC, with its technology dependent view to a particular layer network. As part of this translation, the FCC needs to determine which layer network, or networks, should be used. A selected layer network should support the requested QoS and provide suitable termination for at least one branch of the NFC. If such a connection can not be established across a single layer network, then the FCC would have to setup connections across two different layer networks. As shown in Figure 4-21, each LNC represents only a single layer network. Layer network components do not support inter-working (or federation) across different types of layer networks. This means the FCC would have to find a suitable adapter and setup trails across both layer networks.

4.4.2.3.1. i_FlowCtrl

This interface provides operations for addition, removal, modification, activation, and deactivation of one or more branches (possibly all branches) of the NFC associated with the FCC object. It also provides a query operation for obtaining information on the associated NFC.

This interface provides the following operations:

- .Add flow connection branches
  - This operation is used for adding one or more branches to the NFC associated with the FCC object. NFC branches can be added in an active or inactive state.

- Delete flow connection branches
  - This operation is used for removing one or more branches (possibly all branches) from the flow connection associated with the FCC object. A request to delete all branches releases the NFC, and removes the related FCC. (The i_ConSCtrl interface can also be used to release NFCs).
- Activate flow connection branches
  - This operation is used for activating one or more branches (possibly all branches) of the flow connection associated with the FCC object.
- Deactivate flow connection branches
  - This operation is used for deactivating one or more branches (possibly all branches) of the flow connection associated with the FCC object.
- Modify flow connection branches
  - This operation is used for adding one or more branches to the flow connection associated with the FCC object.
- Get flow connection info
  - This operation is used for retrieving the flow connection information.

4.4.2.3.2. i_FCNotifyCtrl

This interface provides operations for controlling the emission of notifications regarding the network flow connection associated with the FCC object.

It provides the following operations:

- Enable flow connection notification
  - This operation is used for instructing the FCC to emit notifications regarding the NFC.
- Disable flow connection notification
  - This operation is used for instructing the FCC to suspend emission of notifications regarding the NFC.
- Set flow connection notification destination
  - This operation is used to communicate the interface reference to the FCC which its client will use to receive flow connection notifications.

4.4.2.3.3. Generic or Predefined Supported Interfaces

The following generic or predefined interfaces are supported by the FCC:

i_Notify: the FCC may support this interface to receive notifications associated with the trails it establishes in the layer network, see Section 4.4.1.4.1.

4.4.2.3.4. Required interfaces

The following interfaces are required by the FCC:

i_FCNotify: The FCC sends notifications regarding its associated flow connection to this interface. See Section 4.4.1.2.3.

i_TrailSetup: The FCC uses this interface to establish a trail, see Section 4.4.3.1.1.

i_TrailControl: The FCC uses this interface to control a trail it uses or has established, see Section 4.4.3.2.1.

i_NotifyCtrl: The FCC may require this interface to acquire notifications associated with trails it controls in the layer network, see Section 4.4.1.4.2.

4.4.3 Layer Network related objects

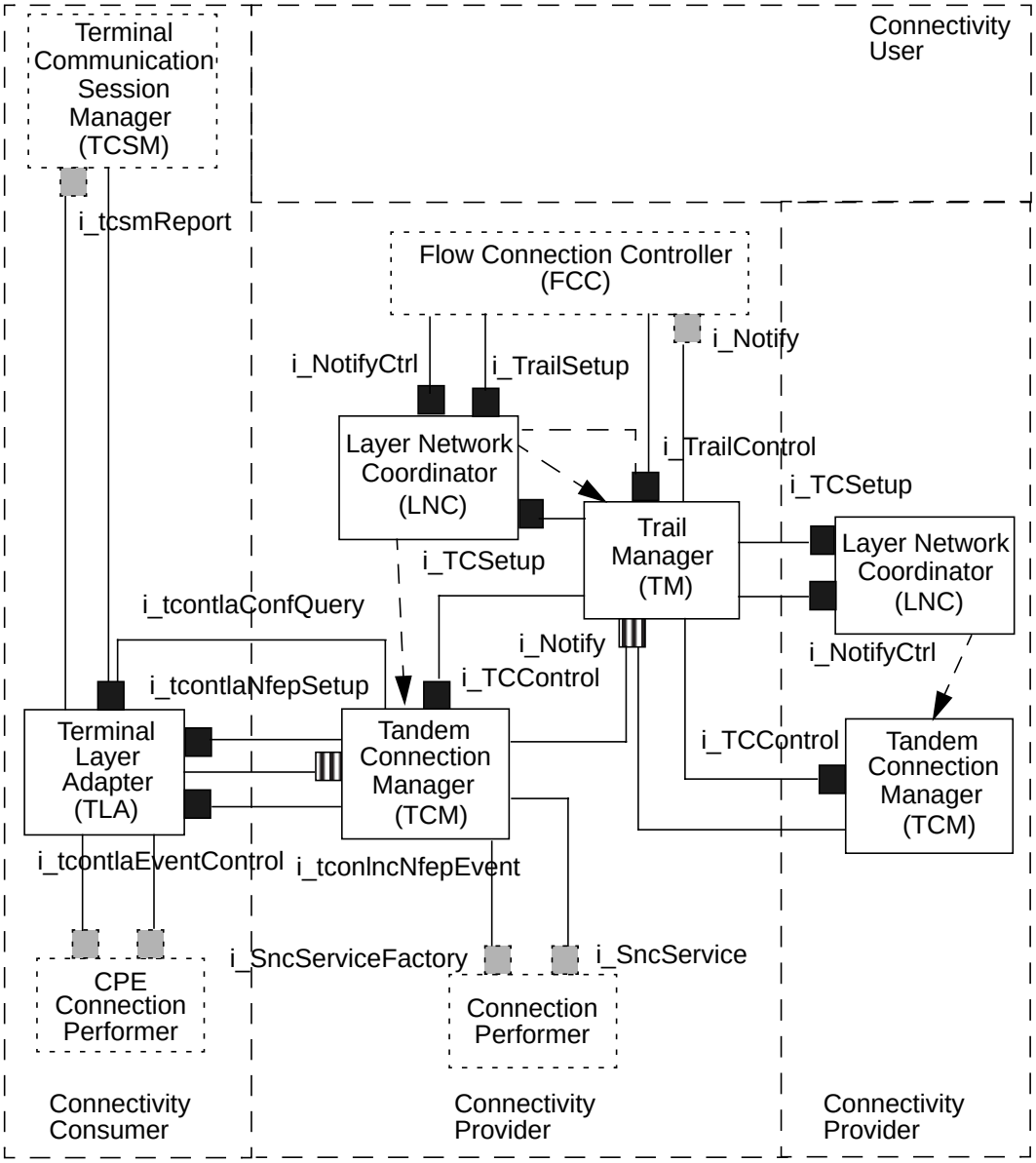


Figure 4-22. Layer network control related objects

Section 4-22 shows the computational objects associated with the layer network. To illustrate the relation between components associated with different domains, the figure includes components from two connectivity provider domains, as well as objects associated with other conceptual groupings.

The layer network coordinator acts on behalf of a layer network domain. The LNC's role is similar to that of the CC or CSM: it allows connections to be setup within the layer network. However, LNCs are concerned with two types of connections: trails and tandem connections. Each connection type has its own set up interface.

The LNC lets connectivity level clients (FCCs) establish trails across the layer network. The LNC is typically associated with large numbers of trails, so a separation between coordinating a layer network and managing the trails has been made. Trails are controlled by a trail manager (TM). For each trail it manages, the TM offers an associate trail control interface that allows a client (the FCC) to manipulate or release the trail.

To setup a trail across multiple domains may include setting up a number of tandem connections in various domains. As tandem connections may exist independent of the trail management in their domain, it is useful to consider tandem and trail connection management separately too. The tandem connection manager (TCM) allows a client (typically another TCM or a TM) to manipulate or release the tandem connection.

Though trail and tandem connection managers are always illustrated as managing a single trail or tandem connection, this is by no means prescriptive. As long as the dynamics of the interfaces are preserved, i.e each trail or tandem connection is associated with a single interface, these objects may control as many trails or tandem connections as desired. Also, in the initial domain, both a tandem connection and a trail always coexist. Implementers may wish to specialize trail managers to support tandem connection management functionality as well.

Interfaces between domains are usually considered prescriptive. However, interfaces related with the TCon reference point (between consumer and connectivity provider) are not yet stable. When these interfaces mature they will become prescriptive. For the moment, operation details must be considered indicative. As well, as the layer network view is technology dependent, various interfaces may need to be specialized to support technology dependent concerns. As long as they maintain the basic functionality of the prescribed interfaces, they will be considered as conforming to the TINA architecture.

Ideally, the layer network related COs interact with subnetwork related COs. Objects here are described as supporting these interfaces. However, this level may be replaced or modified to set a particular network technology.

Finally, note that all interactions described above relate to a a single layer network. Layer network components can not federate or interact with components of different types of layer network. Federation at this level must be performed by the client, typically the FCC. Figure 4-21 illustrates this kind of federation. Do not confuse this type of inter-working with federation (or inter-working) between domains in the same layer network. This is supported as described above, and detailed in the following sections, by COs related to the layer network.

4.4.3.1 Layer Network Coordinator (LNC)

There is one Layer Network Coordinator (LNC) in the connectivity provider's domain for each administrative domain of every layer network. The LNC services a similar role to the CC at the connectivity level, except that instead of creating connections with a session (which is an service abstraction), it establishes connections in relation to a real network.

In essence, the LNC is the single point of access to a domain in a layer network. It allows trails and tandem connections to be established or released with in layer network. It supports queries on the status of the layer network and its associated trails. Similarly, it supports queries about tandem connections established within a domain.

There are two types of clients of the LNC:

- Connectivity type clients: these want to set up a trail across the layer network:
  - Flow connection controllers.
  - Layer Network Topology Configuration Management (LNTCM) of a client layer network, which requests for trails, that will be used as topological links in the client layer network.
- Peer type clients: these want to set up a tandem connection in this domain, to support trail establishment
  - Trail managers.
  - Tandem connection managers.

`Federation.` LN components of different domains in the same layer can federate with each other to establish a trail. Therefore, for a connectivity client, the LNC is a single point of access to the entire layer network. A client may have a choice between various LNCs for one layer network but for every requested trail it only communicates with one LNC. Federation will be elaborated in future dot releases of the NRA.

4.4.3.1.1. i_TrailSetup

This interface allows the establishment of trails and their associated computation objects. It also allows query and trail release operations. These operations are primarily included to aid management, such as recovery and transfer of trail control.

- Setup trail
  - This operation is used to set up one or more trails. If successful, this operation creates a Trail Manager object (if necessary) and returns a trail control interface for each trail established. This interface allows a client to manipulate or release the trail. Trails can be established in active or inactive state.
- Release trails
  - This operation is used to release one or more specified trails within the LNC. It can only be used to release trails established in the domain associated with the LNC. This operation is included for management purposes.

- Get layer network trail info
  - This operation is used for retrieving lists of trails in the layer network. Filtering attributes may be used to manage which trails are listed.
- Get trail control interfaces
  - This operation is used for obtaining the references to specified trail control interfaces.
- Get notification control interface
  - Return the notification control interface reference associated with this LNC. This interface allows a client to modify notifications, including destinations and notification types of interest.

4.4.3.1.2. i_TCSetup

This interface allows the establishment of tandem connections and their associated computation objects. It also allows query and tandem connection release operations. These operations are primarily included to aid management, such as recovery and transfer of tandem connection control.

- Setup tandem connection
  - This operation is used to set up a tandem connection. If successful, this operation creates a Tandem Connection Manager object and returns a tandem connection control interface for each tandem connection established. This interface allows a client to manipulate or release the tandem connection. Tandem connections can be established in an active or inactive state.
- Release
  - This operation is used to release one or more specified tandem connections within the LNC. It can only be used to release tandem connections established in the domain associated with the LNC. This operation is included for management purposes.
- Get layer network tandem connection info
  - This operation is used for retrieving lists of tandem connections in the layer network. Filtering attributes may be used to manage which tandem connections are listed.
- Get tandem connection control interfaces
  - This operation is used for obtaining the references to specified tandem connection control interfaces.
- Get notification control interface
  - Return the notification control interface reference associated with this LNC. This interface allows a client to modify notifications, including destinations and notification types of interest.

4.4.3.1.3. Generic or Predefined Supported Interfaces

i_NotifyCtrl: The LNC may support a notification control interface to allow the direction of notifications associated with this layer network. Requests may be filtered to direct notifications associated with a particular trail or tandem connection. See Section 4.4.1.4.2.

4.4.3.1.4. Required Interfaces

The following interface are suggested for initialization of the TMs and TCMs. Both these suggested interfaces assume a single manager for each trail or tandem connection.

i_TrailControl: Once the TM is created, the LNC can use this interface to initialize trails.

i_TCControl: Once the TCM is created, the LNC can use this interface to initialize tandem connections.

4.4.3.2 Trail Manager

A Trail Manager (TM) is created by the Layer Network Coordinator of the layer network domain in which this trail is initially established. An instance of this object type exists for each trail within the layer network. A trail manager object provides operations for manipulating the associated trail, including the trail release operation. When a trail is released, the associated trail manager object is deleted.

Trails are a component of network flow connections within a layer network domain. The trail manager acts as a single point of contact for control of the trail, even if it extends over multiple domains. The FCC is the usual client of the trail manager.

Note that a trail manager always exists with a tandem connection manager in the initial domain (i.e. the domain of the LNC that receives the set up trail request). The TCM functionality includes terminating tandem connections and trails. We have described these as separate objects, because, apart from this initial domain, TCM functionality is used without the added TM functionality. In practise, the TM and TCM functionality may be merged into a specialization of the TM that supports tandem connection setup with in its domain and supports interactions with the TLA. However, we will not elaborate this possibility here.

`Federation.` An LNC will return the interface to a single trail manager even if the trail requires federation between multiple domains of the layer network to be established. A trail manager may interact with tandem connection managers to establish and manage a trail. The role of trail managers in federation and interactions between them and tandem connection managers will be elaborated in future release of this document.

4.4.3.2.1. i_TrailCtrl

The Trail Manager must support this interface. Through this interface a client can manipulate a trail by adding, modifying, activating, deactivating or releasing branches. An operation can act on the entire trail by specifying all branches. A trail and its branches may be specified in terms of Network Trail Termination Points (NWTTPs) or NWTTP pools. The following operations are supported:

- Add trail branches
  - This operation is used for adding one or more branches to the trail associated with the Trail Manager object. Branches can be added in an active or inactive state.
- Delete trail branches (or entire trail)
  - This operation is used for removing one or more branches (possibly all branches) from the trail associated with the Trail Manager object. Deleting all branches will release the entire trail.
- Activate trail branches (or entire trail)
  - This operation is used for activating one or more branches (possibly all branches) of the trail associated with the Trail Manager object.
- Deactivate trail branches (or entire trail)
  - This operation is used for deactivating one or more branches (possibly all branches) of the trail associated with the Trail Manager object.
- Modify trail branches (or entire trail)
  - This operation is used for modifying one or more branches to the trail associated with the Trail Manager object.
- Get trail info
  - This operation is used for retrieving the trail information. It returns information on routing constraints, traffic type, QoS and termination points.

4.4.3.2.2. Required interfaces

The Trail Manager (TM) requires the following interfaces:

`i_TCSetup` The TM uses this interface to setup the tandem connections required to set-up a trail. The TM can locate the LNC associated with different domains of the layer network and their associated i_TCSetup interfaces. See Section 4.4.3.1.2.

`i_TCControl` The TM uses this interface to manipulate a tandem connection. This interface includes a tandem connection release operation. See Section 4.4.3.1.2.

i_Notify: The TM can notify clients of events associated with the trail it manages using this interface. See Section 4.4.1.4.1.

4.4.3.3 Tandem Connection Manager

A Tandem Connection Manager (TCM) is created by the Layer Network Coordinator of the layer network domain in which this tandem connection exists. One instance of this object type exists for each tandem connection within a layer network domain. A tandem

connection manager object provides operations for manipulating the associated tandem connections, including the tandem connection release operation. When a tandem connection is released, the associated tandem connection manager object is deleted.

A tandem connection is a component of a trail across a layer network domain. Tandem connection managers are responsible for the routing of the tandem connection through the layer network. They provide the a single point of contact for control of the tandem connection. The typical clients of a TCM are either the trail manager or another TCM.

A tandem connection may be terminated by a connection termination point or a trail termination point. To support trail termination, a tandem connection manager can interact with a TLA to manipulate the endpoints of the network flow connection associated with the tandem connection. The tandem connection manager manipulates NFEPs through the TLA's `i_tcontlaNfepControl` interface.

`Federation.` Tandem connection managers may interact to establish a trail across multiple layer network domains. The role of TCMs in federation and interactions between them and trail managers will be elaborated in future dot releases of the NRA. Tandem connections may be setup recursively, in which case a TCM can request an LNC in another domain to establish a new tandem connection.

4.4.3.3.1. i_TCControl

The TCM must support this interface. Through this interface a client can manipulate a tandem connection by adding, modifying, activating, deactivating or releasing branches. An operation can act on the entire tandem connection by specifying all branches. A tandem connection and its branches may be specified in terms of Network Trail Termination Points (NWTTPs), Network Connection Termination Points (NWCTPs), NWTTP pools, or NWCTP pools. The following operations are supported:

- Add tandem connection branches
  - This operation is used for adding one or more branches to the tandem connection associated with the Tandem Connection Manager object. Branches can be added in an active or inactive state.
- Delete tandem connection branches
  - This operation is used for removing one or more branches (possibly all branches) from the tandem connection associated with the Tandem Connection Manager object.
- Activate tandem connection branches
  - This operation is used for activating one or more branches (possibly all branches) of the tandem connection associated with the Tandem Connection Manager object.
- Deactivate tandem connection branches

- This operation is used for deactivating one or more branches (possibly all branches) of the tandem connection associated with the Tandem Connection Manager object.

- Modify tandem connection branches
  - This operation is used for modifying one or more branches to the trail associated with the Tandem Connection Manager object.

- Get tandem connection info
  - This operation is used for retrieving the tandem connection information. It returns information on routing constraints, traffic type, QoS and termination points.

4.4.3.3.2. i_tconlncNfepEvent

The TCM may support this interface. Through this interface the TLA can notify the Tandem Connection Manager (or other layer network manager, e.g. Trail Manager) of changes in the operational state of a NFEP. The following operation is provided:

- NFEP status change
  - This operation is used to notify the connectivity provider when the operational state of a NFEP changes.

4.4.3.3.3. Required interfaces

The TCM requires the following interfaces:

`i_tcontlaNfepSetup` The TCM uses this interface for the dynamic creation of NFEPs. See Section 4.4.3.4.1.

`i_tcontlaNfepControl` The TCM uses this interface to control NFEPs associated with its tandem connections. See Section 4.4.3.4.2.`i_tcontlaNfepControl`

`i_tcontlaConfQuery` The TCM may use this interface to find out about NFEPpools and their status. (Note that this configuration management may also be done by some other management component). See Section 4.4.3.4.3.

`i_tcontlaEventControl` The TCM may use this interface to control events generated by associated TLA's. This is only a suggested placement, more centralized control of TLA events may be more appropriate. See Section 4.4.3.4.4.

i_Notify: The TCM requires a notification interface, to which it directs notifications associated with the trail. See Section 4.4.1.4.1.

`SncServiceFactory` The TCM may require this interface to setup subnetwork connections required to establish the tandem connection. See Section 4.4.4.1.1.

`SncService` The TCM may require this interface to manipulate subnetwork connections the participate in the tandem connection. See Section 4.4.4.1.2.

i_TCSetup The TCM uses this interface to support the recursive set up of tandem connections. The TCM can locate the LNC associated with different domains of the layer network and their associated i_TCSetup interface. See Section 4.4.3.1.2.

4.4.3.4 Terminal Layer Adapter (TLA)

The Terminal Layer Adapter (TLA) object provides functions for setting up (creating), manipulating and deleting network flow endpoints. There is one TLA in the connectivity consumer's domain for each type of layer network. It offers the following interfaces.

4.4.3.4.1. i_tcontlaNfepSetup

NFEPs may be set up dynamically during connection establishment or may be pre-provisioned. This interface provides the following operation:

- Setup NFEP:
  - This operation is used to set-up the NFEP in the connectivity consumer domain. The connectivity consumer will execute the operation and report success or failure. When the operation is performed successfully, the NFEP will exist, be bound to a NWTTP and its collocated NWCTP, and be in the *unlocked* state or the *locked* state, depending on the value of the parameter "Initial Administrative State". Only in the *unlocked* state the NFEP is able to transport information.

4.4.3.4.2. i_tcontlaNfepControl

This interface provides the following operations:

- Activate:
  - This operation is used to activate a NFEP in the connectivity consumer domain. The connectivity consumer will execute the operation and report success or failure. A precondition to this operation is that the NFEP is in the *locked* state. After the operation is performed successfully, the NFEP is in the *unlocked* state.
- Deactivate:
  - This operation is used to de-activate an active NFEP in the connectivity consumer domain. The connectivity consumer will execute the operation and report success or failure. A precondition to this operation is that the NFEP in the *unlocked* state. After the operation is performed successfully, the NFEP will be in the *locked* state.
- Modify:
  - This operation is used to modify the characteristic of the NFEP (e.g. bandwidth, QoS, bearer service type, etc.). The connectivity consumer will execute the operation and report success of failure. This operation may be invoked in the *locked* state or the *unlocked* state. It will not result in a state change.
- Release:

- This operation is used to release a NFEP in the connectivity consumer domain. The connectivity consumer will execute the operation and report success or failure. This operation may be invoked in the *locked* state or in the *unlocked* state. After successful execution of the operation, the NFEP will not be bound to the supporting NWTTP anymore, and the supporting NWTTP will be deleted. If not pre-provisioned, the supporting NWCTP is deleted as well. Similar, the NFEP is deleted if it is not pre-provisioned.

4.4.3.4.3. i_tcontlaConfQuery

This interface allows a client, such as a TCM or TCSM, to query the TLA about available NFEPpools and their status. It provides the following operation:

- Get NFEPpools:
  - This operation is used for retrieving the names of all flow endpoint pools that the connectivity consumer domain supports.

4.4.3.4.4. i_tcontlaEventControl

This interface allows a client, such as a TCM, to control the emission of event notification regarding the operational state of NFEPs by the TLA to the LNC. It provides the following operations:

- Enable:
  - This operation instructs the connectivity consumer (TLA) to emit notifications regarding the NFEPs.
- Disable:
  - This operation instructs the connectivity consumer (TLA) to suspend emission of notifications regarding the NFEPs.
- Set destination:
  - This operation is used to instruct the TLA about the `i_tconlncNfepEvent` interface to which notifications should be sent.

4.4.3.4.5. Required interfaces

The TLA requires the following interfaces:

`i_tcsmReport` The TLA may use this interface to notify the TCSM of the NFEP selections associated with particular connections. It may also use this interface to notify the TCSM of any problems or status changes with NFEPs. See Section 4.4.1.3.3.

`i_SncServiceFactory` The TLA may require this interface to setup subnetwork connections (possibly at an element level) to within the terminal. See Section 4.4.4.1.1.

`i_SncService` The TLA may require this interface to manipulate subnetwork connections within the terminal. See Section 4.4.4.1.2.

4.4.4 Subnetwork related objects

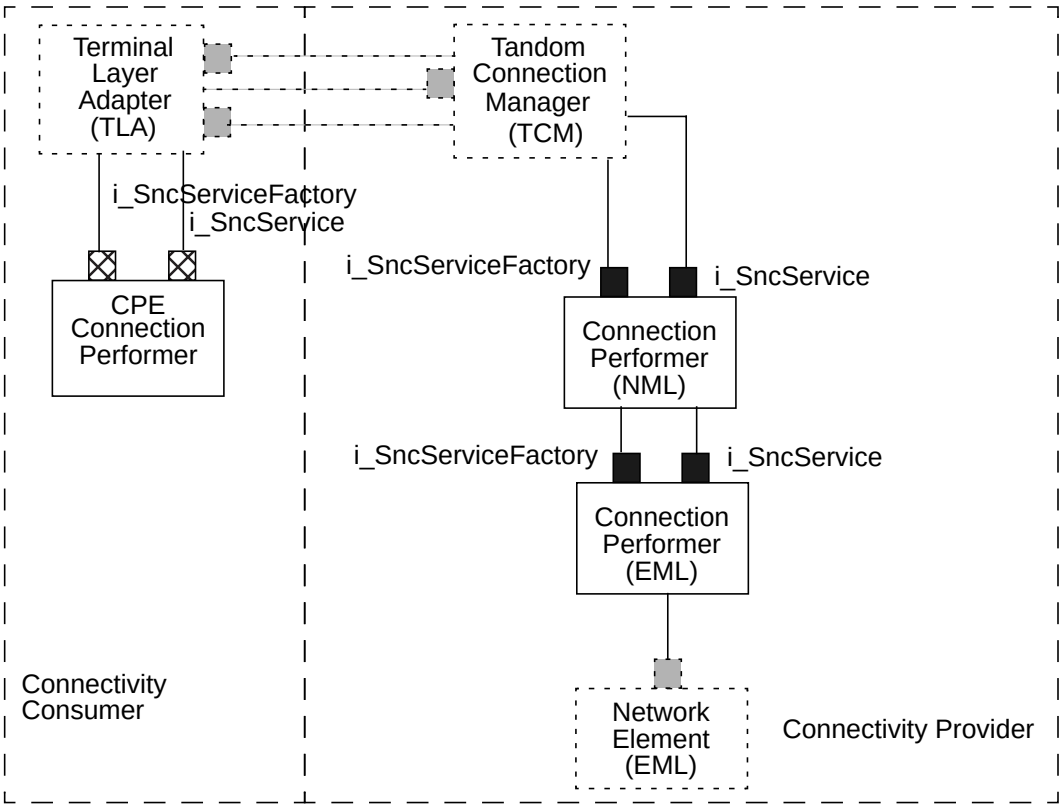


Figure 4-23. Subnetwork control related objects

4.4.4.1 Connection Performer (CP)

Most work in 1995 and 1996 focussed on the interaction of the SA and the CPE with the NRA. Therefore the components inside the layer network did not change at all. They will be updated in a next version. The rest of this section is extracted from the CMA and CMS of 1994.

A Connection Performer (CP) manages SubNetwork Connections (SNCs) across one subnetwork (SNW). The CP is specialized along two dimensions:

- The first dimension is the specialization for the TMN layer that it is used for: Element Management Layer (EML) or Network Management Layer (NML). Any instance of a CP will be either an EML-CP or a NML-CP.
- The second dimension is the technology for which the CP is used as a manager (e.g., for an ATM VC subnetwork or a SDH subnetwork). In principle, the CP definitions given in TINA are generic for different technologies, and as such they

can be used to instantiate NML-CP or EML-CP objects. However, it is possible to make an specialization for a particular technology (this approach is also followed in some standards, e.g., ITU-T M.3100 [29]).

The Connection Performer (CP)

The CP provides interconnections of termination points of a subnetwork. A CP can satisfy requests to establish, modify and release subnetwork connections in its subnetwork. These requests are stated to a CP by a client in terms of *SubNetworkConnections* and *Edges*. (*Edge*s represent termination points of a *SubNetworkConnection*). These *SubNetworkConnection*s can be point-to-point uni- and bidirectional or point-to-multipoint unidirectional. The subnetwork in the scope of a CP will not span resources in more than one administrative domain.

CPs may establish hierarchial relations with each other, reflecting the hierarchial relations between subnetworks. For example, one CP can use another CP to establish a connection (see Figure 4-23), where that CP acts on behalf of a subnetwork within the subnetwork of the controlling CP. In other words, subnetwork connections can be established recursively. However, peer relations between CPs have not been considered. Finally, we do not consider routing algorithms, as this is considered to be outside of the scope of TINA.

Network Management Layer Connection Performer (NML-CP)

The NML-CP controls connections in a partitioned subnetwork of a layer network domain.

Element Management Layer Connection Performer (EML-CP)

The EML-CP has all the characteristics of the CP and adds the following:

- The subnetwork domain in the scope of a EML Connection Performer breaks down to the NE layer. EML-CPs have access to an agent in a Network Element.

Refinement of Connection Performers for technology

CP specification in TINA Connection Management Specifications is technology independent. However, actual implementation may have to be specialized in order to make it suit for individual technology (e.g. Frame Relay, ATM, SDH, N-ISDN).

CP provides following interfaces.

4.4.4.1.1. i_SncServiceFactory

The i_SncServiceFactory interface creates i_SncService interfaces. They are deleted when the connection is released. The operations on i_SncService are applied to the specific instance of subnetwork connection.

This interface provides following operations.

- Create a new Subnetwork Connection

- This operation will create a new SNC and its root edge, returning the new i_SncService interface instance and the root edge identifier.

- Create and setup SNC
  - This operation will create and setup the SNC. This operation is provided for inter-working purposes, and is a shortcut to the entire sequence: [create_snc, create_edges, attach_edge]. The operation will be executed in a all-or-nothing fashion.

4.4.4.1.2. i_SncService

This interface provides following operations.

- Destroy subnetwork connection
  - This operation will destroy a subnetwork connection.
- Create leaf edges
  - This operation creates a number of leaf edges, returning an ordered list of identifiers corresponding to the list passed as an input.
- Delete edge
  - This operation removes a leaf edge from a subnetwork connection.
- Attach edges
  - This operation attaches previously created edges. Only when the edge is attached, traffic is possible.
- Detach edge
  - This operation changes the state of a leaf edge from *busy* to *reserved*
- Migration edge
  - This operation migrates a leaf edge, i.e. modifies the network endpoint the edge is bound to.The conditions under which this operation can be executed, and its behavior are for further study. Its usefulness can be considered for the mobile terminal case.
- Subnetwork Connection Attributes
  - The following operations are used to modify the characteristics of a SNC in terms of its traffic description and QoS requirements.

    modify_traffic_description
    modify_qos

4.5 Dynamic View

This section describes a set of interactions between TINA network components in a series of scenarios. No examples of connection establishment within a layer network (trails, TCs and SNCs) are worked out in this version of the document.

The following assumptions have been taken for the scenarios in this chapter:

- two parties will be involved in the communication session (although some diagrams will only include interactions with one of them),
- their terminals know the NFEPpool they are connected to,
- the application in each terminal is informed of this NFEPpool,
- NFEPpool information can be passed encapsulated in SIs or SFEPs.

Other scenarios are possible if different assumptions are taken.

For each scenario a computational diagram is provided where the interactions between the different computational objects involved are shown; returns are indicated in square brackets. Then a textual explanation is given.

Finally event trace diagrams trace the steps defined in each scenario. Comment boxes denote actions performed by the object as a result of events; these may be internal or external actions, they may involve network resource objects not described in this document, or DPE services (location servers, traders, etc.). Operation returns are denoted by [ ], and those which return a value as [return value].

4.5.1 Setup of a communication session

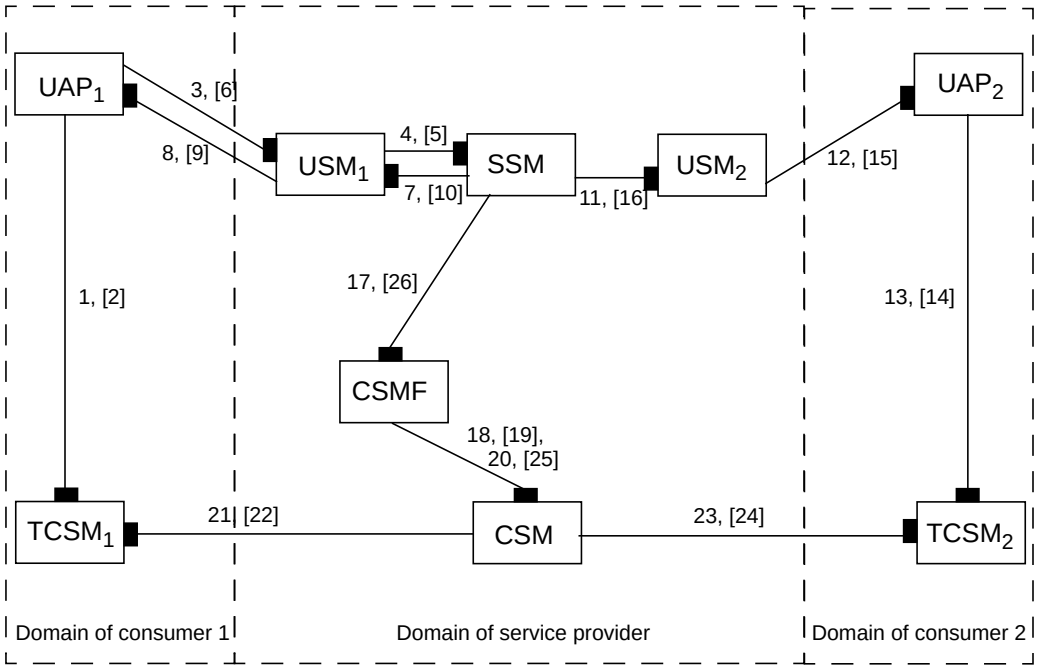


Figure 4-24. CS Setup: Computational Diagram

1. Consumer1's UAP requests his TCSM to setup the application; the aim of this setup is for UAP_1 to get the NFEP(pool)s related to the stream interface used by consumer 1 in this service. To get this information UAP_1 provides $TCSM_1$ with the SFEPs related to this stream interface.
2. $TCSM_1$ returns to UAP_1 the NFEP(pool)s requested; from now on they will be passed encapsulated in SI or SFEP identifiers.
3. UAP_1 requests from USM_1 the setup of a stream binding with consumer 2, providing it with the information obtained in the application setup.

   USM_1 may optionally make the necessary checks to make sure consumer 1 is allowed to setup the requested stream binding.
4. USM_1 forwards the request to the service session's SSM.

   Optionally SSM may check for permission to setup this stream binding; if necessary it may negotiate for permission with the other session parties specified in the session graph.
5. If permission is obtained SSM returns a stream binding identifier, as well as a request identifier for later confirmations.

6. USM_1 forwards this response to UAP_1.

   Now that SSM has been requested to establish the stream binding it will centralize the establishment and treat both parties, the request originator (consumer 1) and the one that is invited (consumer 2) in the same way.

7. SSM requests USM_1 to join the stream binding.
8. USM_1 forwards the request to UAP_1.
9. UAP_1 accepts, and returns this acceptance to USM_1 together with a description of consumer1's terms of participation in the stream binding, as well as a stream interface descriptor.
10. USM_1 forwards this acceptance, and the associated information, to SSM.
11. SSM requests USM_2 to join the stream binding.
12. USM_2 forwards the request to UAP_2.
13. UAP_2 starts an application setup scenario to get the NFEP(pool)s related to the stream interface user by consumer 2 in this service.
14. $TCSM_2$ returns to UAP_2 the NFEP(pool)s requested.
15. UAP_2 accepts joining the stream binding, and returns this acceptance to USM_2 together with a description of consumer1's terms of participation in the stream binding, as well as a stream interface descriptor.
16. USM_2 forwards this acceptance, and the associated information, to SSM.

    SSM may use a trader to find an interface reference to a CSMF through which it can request the establishment of the communication session.

17. SSM requests CSMF to setup a communication session.
18. CSMF creates a new CSM for this new communication session (note that, although it is shown in the diagram for the sake of clarity, this is not an invocation of ODL operation but an interface instantiation by the DPE).
19. A reference to an initial interface of the new CSM is returned to CSMF.
20. CSMF requests CSM the establishment of a new communication session.

    There two different ways to do it: CSMF could request the establishment of an "empty" communication session and return SSM the CSM CS control IR; then SSM would request CSM the SFC setup and get CSM SFC control IR in the operation return.

    The second way, which is the one followed here, is the following: CSMF requests CSM the establishment of both the new CS and associated SFC(s). In the return of the request it will be given both the CSM CS control IR and the associated SFC control IR(s), which it will forward to SSM.

    Besides the object that invokes this operation (CSMF in this case) can choose to specify in the request if it wants to either wait to be informed of the successful

setup completion, or continue and just be notified of its completion or failure when it happens. Here the choice is to wait for the confirmation.

Now CSM will chose a connectivity provider and locate the Actions in the domain of consumers 1 and 2.

21. CSM requests from $TCSM_1$ the correlation of the TFC in the terminal of consumer 1, that is locally identified by the SFEPs related to the stream interface of UAP_1, to a unique (within both the service and connectivity providers' domains) correlation identifier that identifies the network part of the connection with which it is to be associated.
22. $TSCM_1$ agrees to the correlation.
23. CSM requests from $TCSM_2$ the same correlation for the TFC in the terminal of consumer 2.
24. $TCSM_2$ agrees to the correlation.

    For each SFC CSM performs the mapping SFC -> NFC(s) and resolves SFEPs to NFEPs(pools).
25. CSM returns to CSMF a reference to its communication session control interface.
26. CSMF forwards this IR to SSM.

Now a communication session has been established. In the future if new participants want to be added to it SSM will request it from CSM. For each request SSM will decide how the new participant is to be added: either adding a new branch to an existing stream flow connection or adding a new stream flow connection to the communication session.

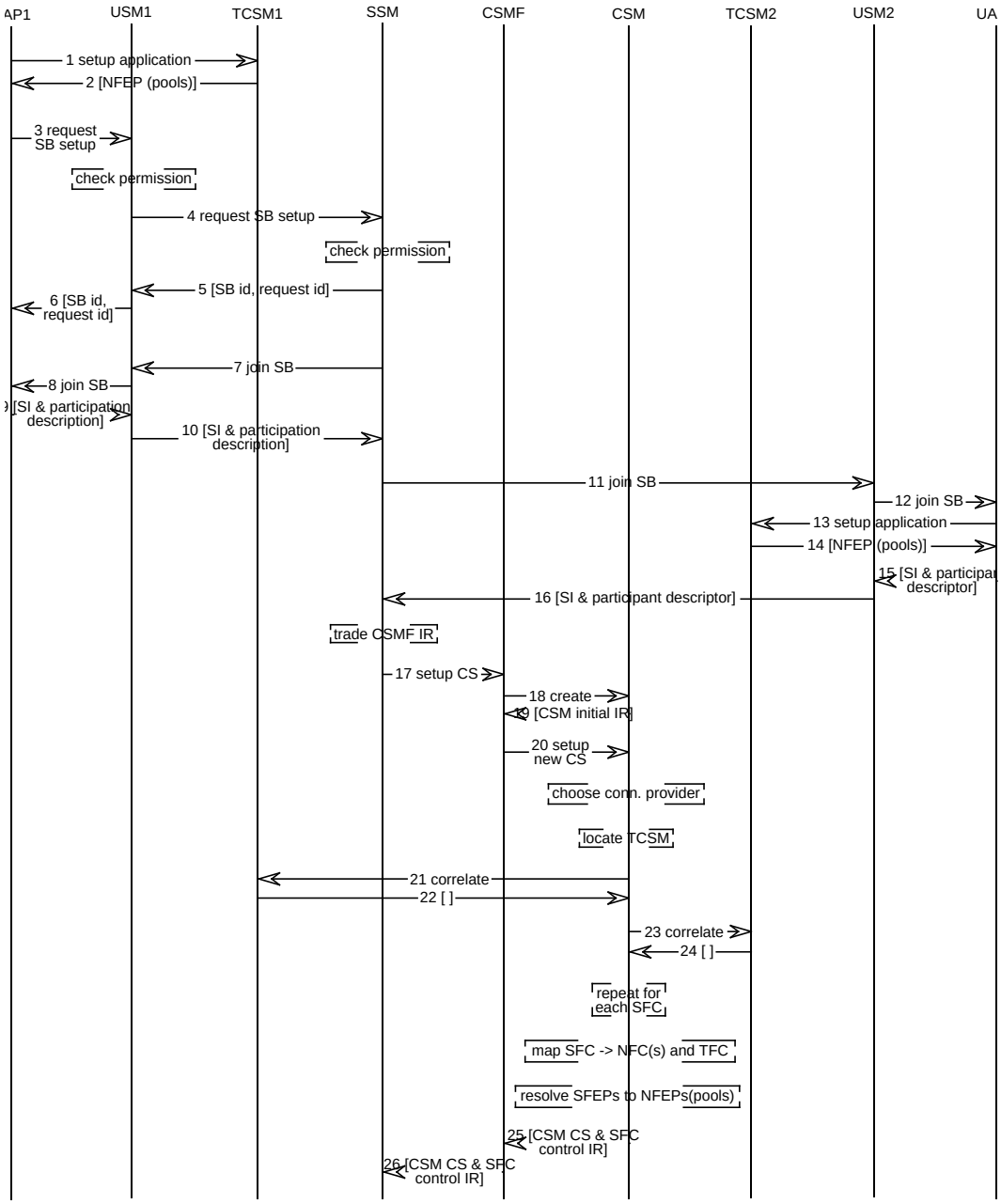


Figure 4-25. CS Setup: Event Traces

4.5.2 Setup of a connectivity session

A connectivity session will be established by request from CSM after a communication session has been setup.

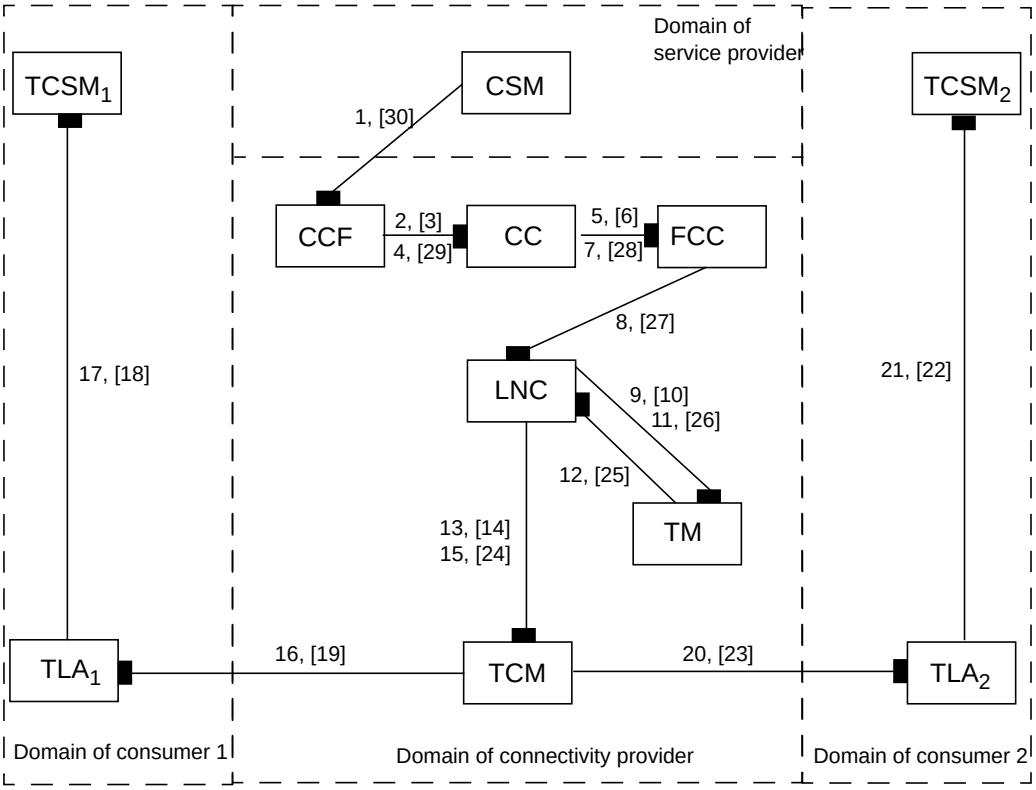


Figure 4-27. conS Setup: Computational Diagram

1. CSM requests the CCF to setup a new connectivity session.
2. CCF creates a new CC for the new connectivity session; although it is shown in the diagram this is really a DPE operation.
3. CCF is informed of the creation and returned a reference to a connectivity session control interface of CC.
4. For each network flow connection in the connectivity session CCF requests from CC its setup.
5. To fulfill this request CC creates an FCC.
6. A reference to a network flow connection control interface of FCC is returned to CC.

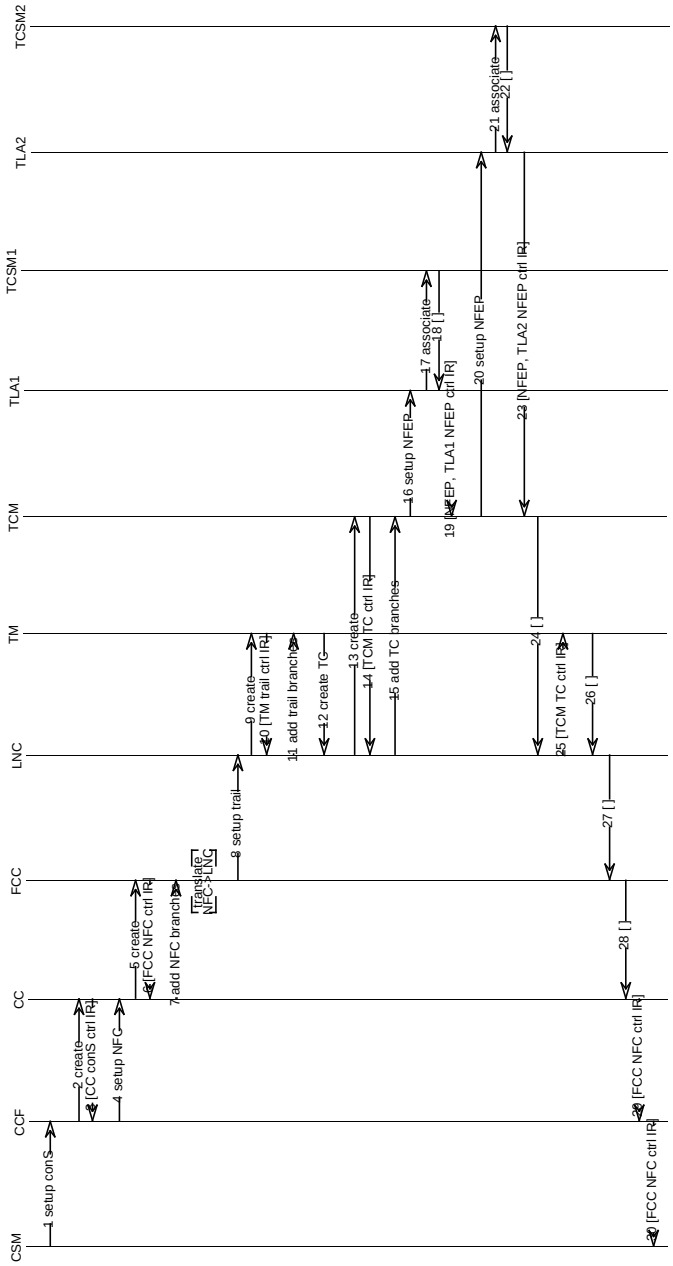


Figure 4-26. ConS Setup: Event Traces

7. CC requests FCC to add branches to the new network flow connection.
8. FCC chooses the necessary layer networks for the new network flow connection. This scenario assumes only one layer network is necessary. FCC locates the LNC for this layer network and request from it a trail setup.
9. LNC creates a TM to manage the new trail. This is again a DPE operation.
10. A reference to a trail control interface of the new TM is returned to LNC upon its instantiation.
11. LNC requests TM to add branches to the new trail.
12. To attend this request TM requests the creation of a tandem connection to LNC.
13. LNC creates a TCM for the new tandem connection (DPE operation).
14. LNC gets a reference to the tandem connection control interface of the new TCM.
15. LNC requests TCM to add branches to the new tandem connection.
16. TCM requests the setup of NFEPs from the TLA in consumer1's domain, including in the request the unique correlation identifier corresponding to the network part of the connection to which the TFC in consumer1's terminal is associated.
17. TLA_1 selects or creates (depending on whether they are pre-provisioned or not) the corresponding NFEPs and requests from $TCSM_1$ the association of the TFC, specified by the correlation identifier, with the established NFEPs.
18. $TCSM_1$ performs the association requested, which completes the connection[14].
19. TLA_1 returns to TCM the name of the chosen (and established) NFEP(s) and a reference to its NFEP control interface.
20. Now TCM requests the NFEP setup from TLA_2.
21. TLA_2 performs the NFEP creation or selection and requests the completion of the connection to $TCSM_2$.
22. $TCSM_2$ performs the association that completes the connection.
23. TLA_2 returns to TCM the name of the chosen (and established) NFEP(s) and a reference to its NFEP control interface.
24. Return of Step 15.
25. Return of Step 12, that includes a reference to a tandem connection control interface of TCM.
26. Return of Step 11.
27. Return of Step 8, that includes a reference to a trail control interface of TM.
28. Return of Step 7.

14. It is assumed that the nodal binding has been established: interactions internal to the terminal are not described here.

29. Return of Step 4, that includes a reference to a network flow connection control interface of FCC.
30. Return of Step 1, that forwards the previous reference to a network flow connection control interface of FCC.

4.5.3 Stream Binding Setup Confirmation

Any participant in a stream binding can request, when responding to a request to join the stream binding, to get a confirmation when the stream binding has been setup. Although this section has considered this is not the case here we present a scenario that shows how it happens when both consumer 1 and consumer 2 participating in the stream binding request this confirmation

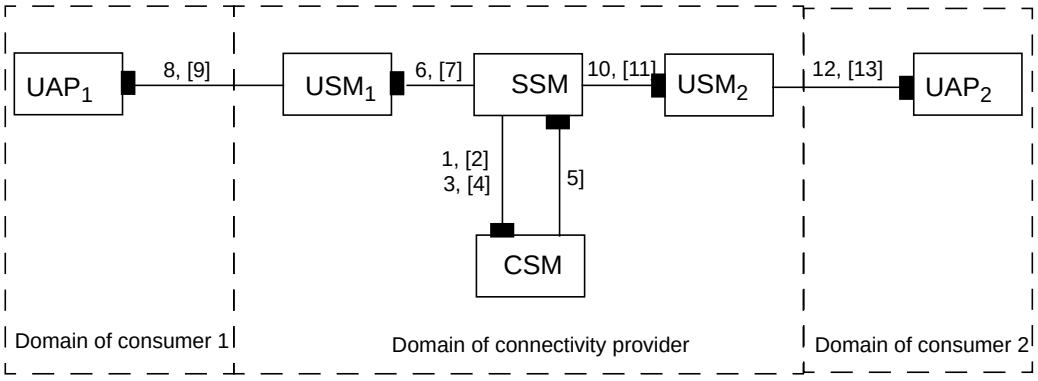


Figure 4-28. SB Setup Confirmation: Computational Diagram

1. SSM registers for notifications of connection setup or session establishment.
2. CSM agrees to notify SSM.
3. SSM enables notifications from CSM.
4. CSM agrees.
5. CSM notifies the connection setup to SSM.
6. SSM sends the requested confirmation to USM_1, including in it details of consumer1's participation.
7. Return of 6.
8. USM_1 analyzes this confirmation and forwards it to UAP_1.
9. Return of 8.
10. SSM sends the requested confirmation to USM_2, including in it details of consumer2's participation.

11. Return of 10.
12. USM_2 analyzes this confirmation and forwards it to UAP_2.
13. Return of 12.

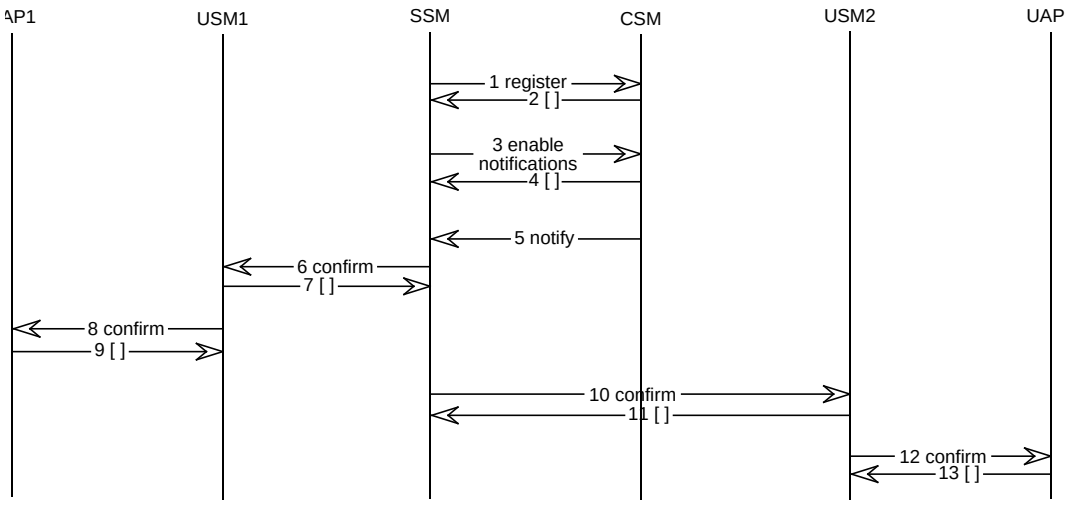


Figure 4-29. SB Setup Confirmation: Event Traces

4.5.4 Stream Binding (and Whole Communication Session) Release

These interactions are triggered by a request from one of the participants to release the service session. This request will reach the service session's SSM, which will request the participants' USMs to leave the service session. These steps correspond to the Service Architecture and will not be detailed here. Then the SSM will request the CSM to release the communication session; the CSM will transform this request in actions in related to every participant. For the sake of simplicity this event trace will only show the interactions with consumer1's components.

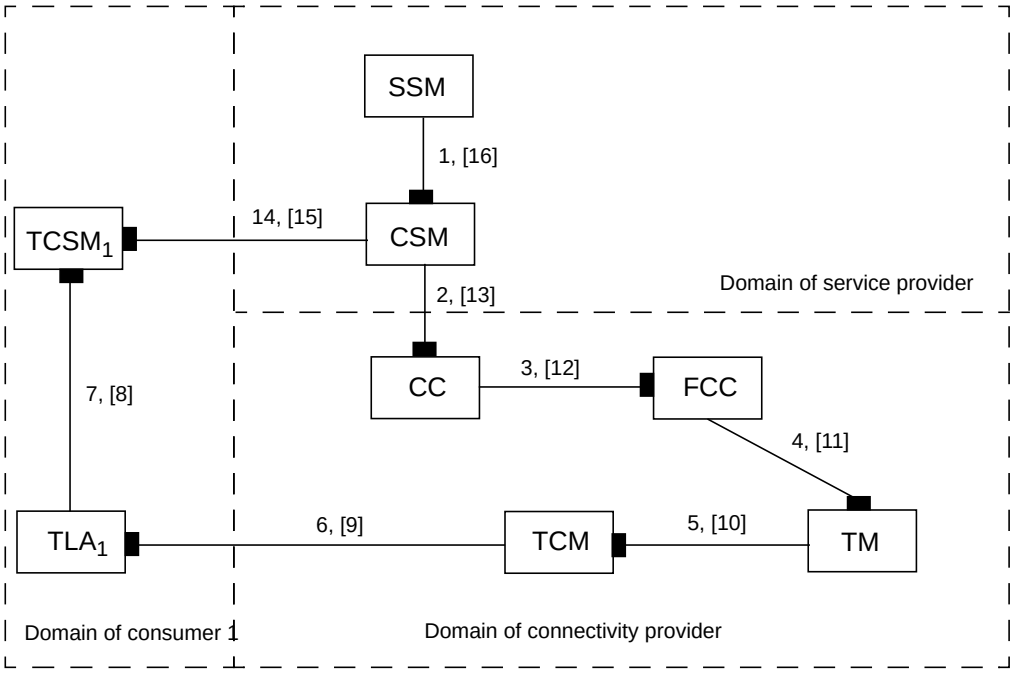


Figure 4-30. CS Release: Computational Diagram

1. SSM requests the release of the communication session from CSM.

   To attend this request CSM must release all the SFCs currently controlled by the communication session and then end it. Since SFCs are managed through a control interface of the CSM it makes an internal request to an instance of this interface, for each SFC in the communication session, to delete its branches specifying “all branches” in the request which is understood as a request to delete the entire SFC.

2. In order to end the communication session CSM requests CC to end the connectivity session.

   CC must release first all the NFCs the connectivity session consists of, then inform CC and finally delete itself.

   The following ten steps will be repeated for every network flow connection in the connectivity session:

3. CC requests the flow connection controller in charge of the network flow connection to delete all the branches in the NFC, which is understood as a request for FCC to delete the NFC, inform CC and delete itself.

4. FCC requests from the trail manager in charge of the trail that implements the NFC the deletion of the whole trail through a “delete all trail branches” request.
5. TM forwards the request to TCM in terms of deleting the whole tandem connection the trail consists of.
6. TCM requests TLA_1 to delete the NFEP(s) that had been setup in the domain of consumer 1. TLA_1 attends the request and as a result of that the binding to supporting NWTTPs is destroyed, the NWTTPs are deleted and so are the NFEP(s) if they were not pre-provisioned.
7. For every NFEP TLA_1 requests $TCSM_1$ to disassociate the nodal binding (that was the part of the connection in the domain of consumer 1, and was specified by a unique correlation identifier) with the corresponding NFEP with which the TFC was completed.
8. $TCSM_1$ informs TLA_1 of the completion of the disassociation.
9. TLA_1 informs TCM of the deletion of NFEP(s).

   The previous four steps are repeated for every participant in the communication session. Only one participant is shown here.
10. TCM informs TM of the deletion of the tandem connection, and then deletes itself.
11. TM informs FCC of the deletion of the trail and then deletes itself.
12. FCC informs CC of the deletion of the network flow connection and then deletes itself.
13. When all its requests to delete NFCs in the connectivity session have succeeded CC informs CSM of the deletion of the connectivity session and deletes itself.
14. For every participant in the communication session (only one is shown here) CSM requests its corresponding TCSM to invalidate the correlation identifier that specified the nodal part of the connection in that node.
15. The corresponding TCSM informs CSM of the invalidation of the correlation identifier.
16. CSM informs SSM of the deletion of the communication session and then deletes itself.

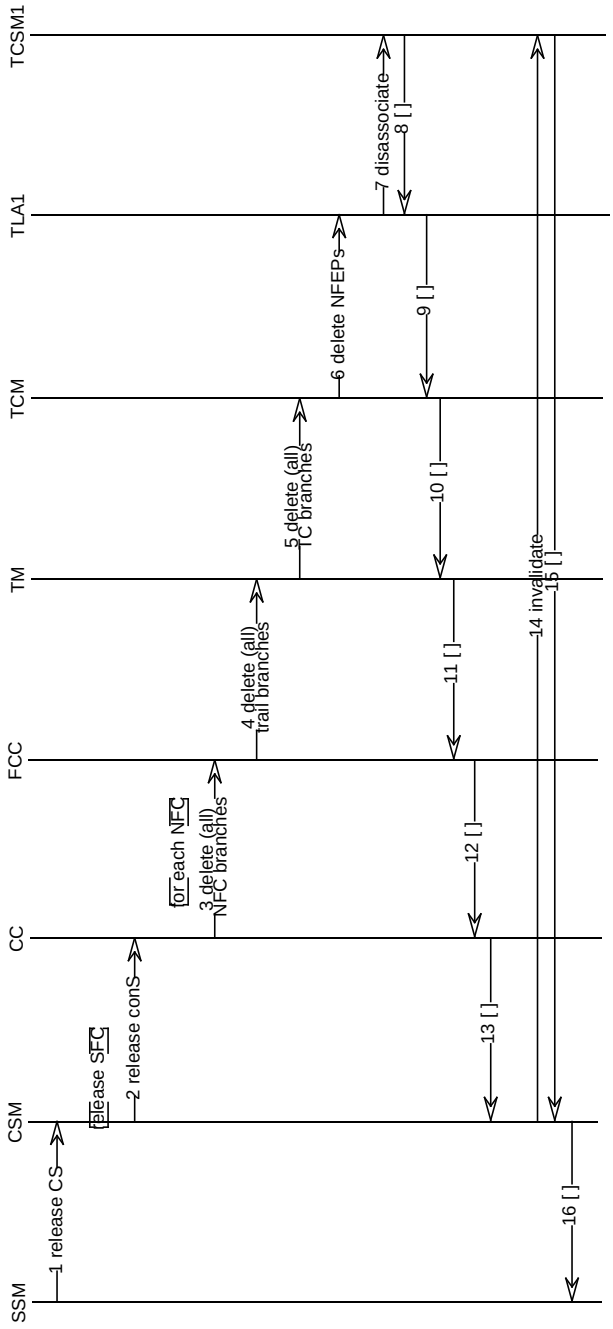


Figure 4-31. CS Release: Event Traces

5. Network Topology Configuration Management

5.1 Scope

Configuration management identifies, exercises control over, collect data from and provides data to systems[1] for the purpose of preparing for, initializing, starting, providing for the continuous operation of, and terminating interconnection services.

TINA generic configuration management domain can be classified into following 3 types of architectural configuration management domains.

- Service configuration management domain
- Network(resource) configuration management domain
- Computing configuration management domain

And, the network configuration management domain has been specialized into two functional areas, Connection Management in the connectivity resource aspect and Network Topology Configuration Management (NTCM) in the topology resource aspect. Figure 5-1 illustrates this classification of configuration management domains.

This chapter focuses on the NTCM. And, it introduces some aspects of connectivity resource configuration management (connection management) in order to help understanding the distinction of topology resource configuration management and connectivity resource configuration management.

Following this section, the functional requirements of NTCM is described in Section 5.2 and the information viewpoint is also described in Section 5.3. Finally, Section 5.4 describes the computational viewpoint.

5.2 Functional requirements

The main goal of the Network Topology Configuration Management (NTCM) is to administrate the life cycle of the installed network topology resources (creation/deletion of the resources, status change of the resources, etc.). The functions that should be provided by NTCM are categorized as follows:

- Provisioning
- Status and control
- Installation support

1. In TMN it means NE (Network Element) but in this document the term ‘systems’ means any set of interacting components, not implying necessarily an underlying physical implementation.

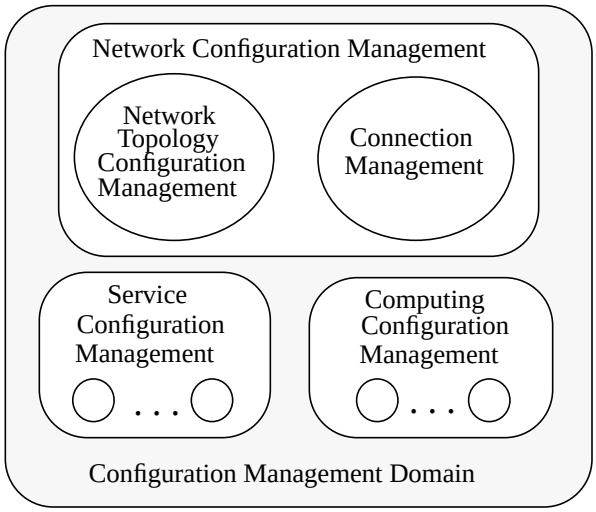


Figure 5-1. Configuration Management Domain

5.2.1 Provisioning

Provisioning consists of procedures which are necessary to bring a network resource (it can be a physical/logical resource of a network element level or network level) into service, not including installation. Once the network topological resource is ready for service, network topology configuration provisioning function initializes the network topology resource.

The state of the topological resource, e.g. administrative state, operational state, reserved state and selected parameters may also be controlled by network topology provisioning functions.

Activation shall initialize the network topology resource, and change the operational state to enabled. Deactivation shall change the operational state to disabled.

Reservation shall assign the resource to a service and change its usage state to reserved.

Locking of the network topology resource shall have it prohibited from providing service and unlocking of the network resource shall have it permitted to provide service.

Over the spectrum of network element, the use of the provisioning functions can vary widely. For small transmission elements, these functions are used once and rarely again. Digital switching and cross-connect equipment may require frequent use of these functions as circuits are put up and dropped.

5.2.1.1 EML Functions[2]

- Request configuration: NTCM request that NE agent reports the current configuration of each network resource.
- Configuration report: NE agent reports configuration information of each network resource, e.g. capacity of network resource, type of network resource, version information and optional parameters.
- Grow: NTCM notifies NE agent of the presence of a newly installed network resource.
- Prune: NTCM notifies NE agent of the disconnection of an network resource.
- Restore: NTCM notifies NE agent to begin monitoring the newly installed network resource.
- Assign: NTCM notifies NE agent that a previously unequipped network resource is now equipped.
- Delete: NTCM notifies NE agent that a previously equipped network resource is no longer equipped.
- Set service state: NTCM directs NE agent to place the specified network resource in one of following states- enabled, disabled, locked, unlocked, shutting down, reserved.
- Request assignment: NTCM requests that NE agent reports the identity of each assigned network resource. The request may be for a specified network resource or for all assigned network resources.
- Assignment reports: NE agent reports the identity of each assigned network resource or for a specified network resource.

5.2.1.2 NML Functions

- Request configuration: NTCM request that CM object group reports the current configuration of layer network or NML or EML (topology information).
- Configuration report: CM object group reports the current configuration of layer network or NML or EML (topology information).
- NTCM notifies CM object group of the presence of a newly installed network resource (subnetwork, topological link, TP, etc.).
- NTCM notifies CM object group of the disconnection of an network resource.
- NTCM configures CM object group.
- NTCM gives network topology information to CM object group.

2. In this section, the role of network topology configuration function is limited to the NRIM configurable objects.

- NTCM maintains configuration data (ConfigData) which contains management information of each topology configurable resource. It consists of each topology information among resources and resource state information, etc. These functions may include the functions such as reviewing routing information, routing control and verifying resource information.

5.2.2 Status and Control

Status and control provides the capability to monitor and control certain aspects of the network resource. Examples include checking or changing the state of network resource (operational state, administrative state, usage state) and initiating diagnostics tests of network resource. Normally, a status check is provided in conjunction with each control function in order to verify that the resulting action has taken place. When associated with failure conditions, these functions are corrective in nature (network resource restoration).

5.2.2.1 Generic network resource status and control functions

- Request status: NTCM requests NE agent or CM object group to send current status information of Network resource.
- Status report: NE agent or CM object group reports to NTCM the value of a monitored parameter of network resource.
- Network resource maintenance: NTCM coordinates with other management domains (e.g., fault or performance management domain) in order to maintain the status of network resource with best operational condition. It may rearrange network resource or re-route traffic according to the network resource status.

5.2.2.2 Status change log data control

- Request status change log data: NTCM requests NE agent to transmit the status change log data of network resource.
- Status change log data report: NE agent sends the status change log data of network resource to NTCM.
- NTCM maintains the status change log data of network resource.

5.2.3 Installation support

Installation support function can support the installation and removal of network resource. It covers also the extension or reduction of a system. Some NEs call for the initial exchange of data between themselves and NTCM. An example of another function is the installation of programs into NEs from data base systems. In addition, administrative data can be exchanged between NEs and NTCM. But in this document installation support assumes that

- Physical resources shall already have access to a communication infrastructure, have been powered on, have access to kernel software, and have been assigned a physical address.
- The software and data associated with a logical resource shall already be stored on an accessible device.

- Acceptance testing shall be carried out by fault management at the request of network resource configuration.

Installation support shall provide support for self configuring devices. It shall be responsible for handling notifications generated by these devices when they are physically installed or removed. NTCM shall update the resource configuration data when this occurs.

When it proves impossible to complete an installation or removal without manual intervention, NTCM shall generate an appropriate notification.

5.3 Information Viewpoint

5.3.1 Object types and relationship types

The object types and relationship types identified in the network topology configuration fragment are described briefly in this section and are listed in Table 5-1 and Table 5-2.

Table 5-1. Object Type Descriptions

Object types	Description
Entity	The top class in TINA
Manageable	Represents the requirements to the objects which are assigned to a domain.
Configurable	This class is a super class for all configurable classes
NetworkTopologyConfiguration-Management Domain	Represents a set of resource information objects to which a set of network topology configuration management policies are applied
AttributeValueChangeRecord	This class is used to define the information stored in the log as a result of receiving attribute value change notifications or attribute value change event reports
ObjectCreationRecord	The class is used to define the information stored in the log as a result of receiving object creation notifications or object creation event reports
ObjectDeletionRecord	This class is used to define the information stored in the log as a result of receiving object deletion notifications or object deletion event reports
StateChangeRecord	The stateChangeRecord is used to define the information stored in the log as a result of receiving state change notifications or state change event reports

Table 5-1. Object Type Descriptions

Object types	Description
ConfigData	This object shall provide all the attributes required by topology resource configuration to track a resource throughout its life-cycle, i.e. installation, operation, and removal. Configuration data object shall contain attributes that represent resources that are to be tracked by Network topology Configuration

Table 5-2. Relationship Type Descriptions

Relationships types	Description
reportStatusChangeTo	The reportStatusChangeTo relationship is a type of relationships which represents the relationships between a resource which reports a status change report(i.e. attributes value change, object creation/deletion and state change information, etc.)and a domain which manages the status change report
isAssignedTo	The isAssignedTo relationship is a type of relationships which represents the relationships between a network topology configuration management domain and the objects assigned to the domain. A network topology configuration management domain object take the set role, and the associated objects take the element roles

Under the entity object in the inheritance hierarchy, the *configurable* object is defined which is the superclass of whatever being configured. Attributes of configurable are administrativeState(X.731), operationalState(X.731) and usageState(X.731). Basic operations and notifications relevant to configuration are also defined for configurable(e.g. lock, unlock, state change notification).

Subclasses of configurable includes dynamic connectivity objects such as SNC and trail which are configured by Connection Management, and topology objects such as NWTTP and Topological Link that are configured by Network Topology Configuration components. For those objects that are configured by the Network Topology Configuration components, support objects are maintained. They are called ConfigData.

5.3.2 Network Topology Configuration Fragment

Figure 5-4 illustrates the OMT graphical notation of Network Topology Configuration Fragment.

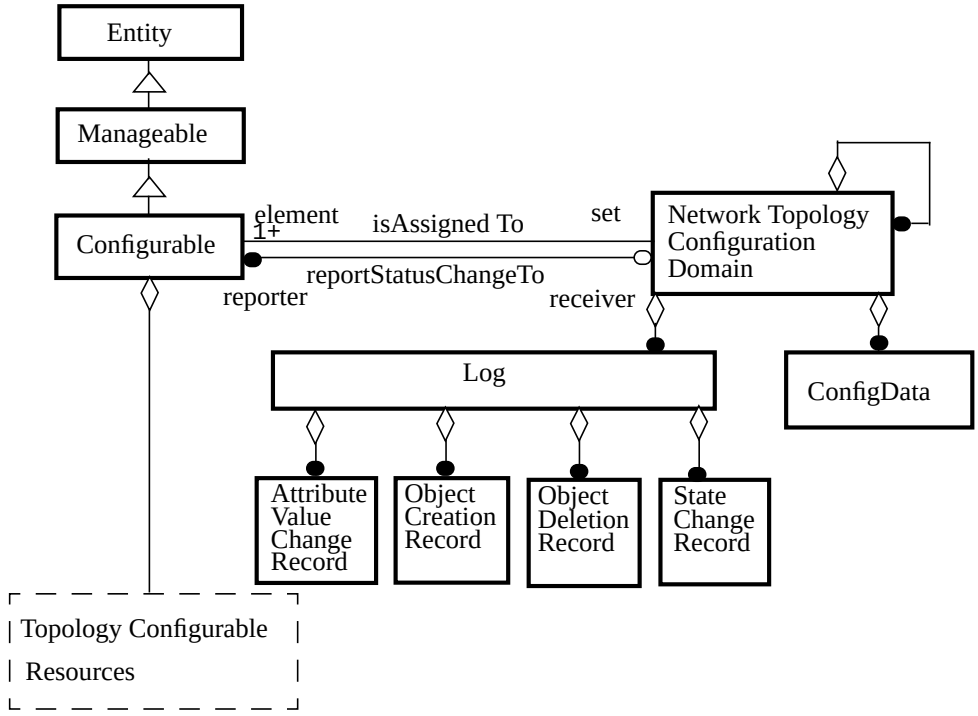


Figure 5-2. Network Topology Configuration Fragment

The topology configurable resources currently defined are shown in Figure 5-3[3].

A layer network modeled by these information elements may be decomposed into subnetworks. In order to manage these subnetworks efficiently, the hierarchical concept of TMN is applied. Figure 5-4 shows this hierarchical relationship between these resources and NTCM.

5.4 Computational Viewpoint

In this section the computational objects and interfaces associated with NTCM are described. Interfaces among NTCM and other management domains(e.g. connection management domain, fault management domain, network administrator, etc.) are considered and computational objects within NTCM (internal/external relationships among computational objects) are defined.

3. These information elements are defined in NRIM [1] which will be updated. So, this figure may be considered as an example.

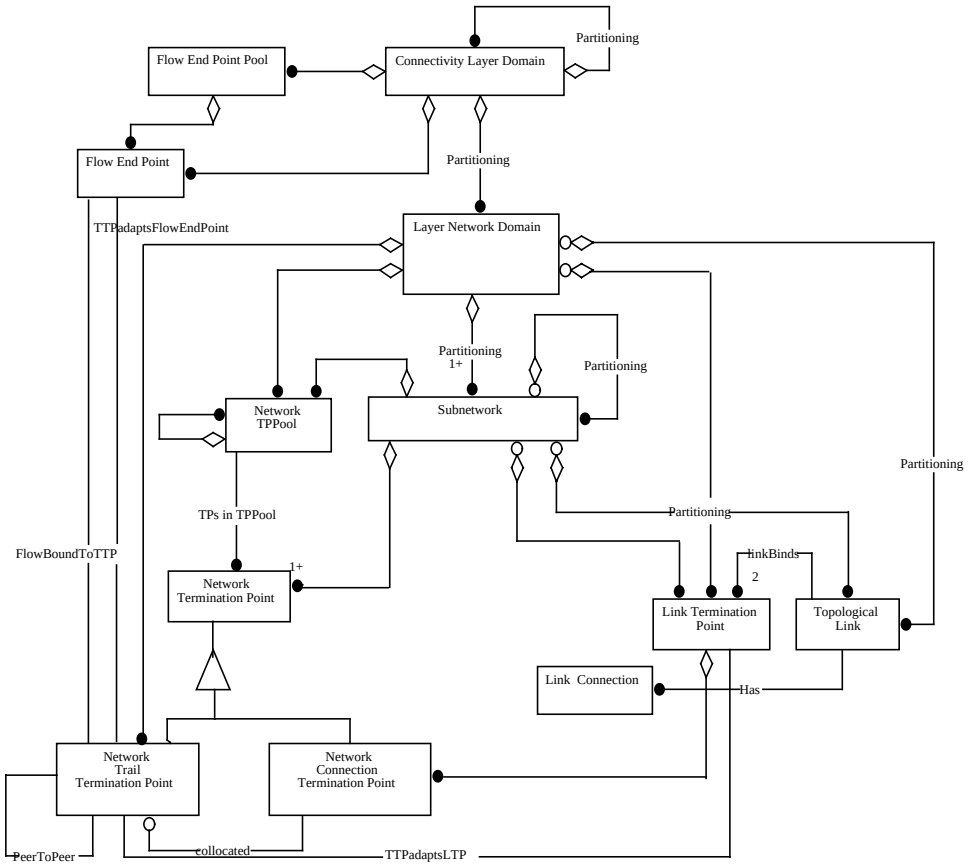


Figure 5-3. Examples of Topology Configurable Resources

The computational objects (COs) defined in NTCM are:

- Layer Network Topology Configurator (LNTC)
- Network Management Layer Topology Configurator (NML-TC)
- Element Management Layer Topology Configurator (EML-TC)

Figure 5-5 shows the role of these COs and the relations between them.

5.4.1 Layer Network Topology Configurator

Layer Network Topology Configuration (LNTC) configures the layer network level topology resources within a corresponding layer network. In other words, LNTC is responsible for creation/deletion, provisioning and status/control of the layer network topology resources. The related resources are network trail termination points.

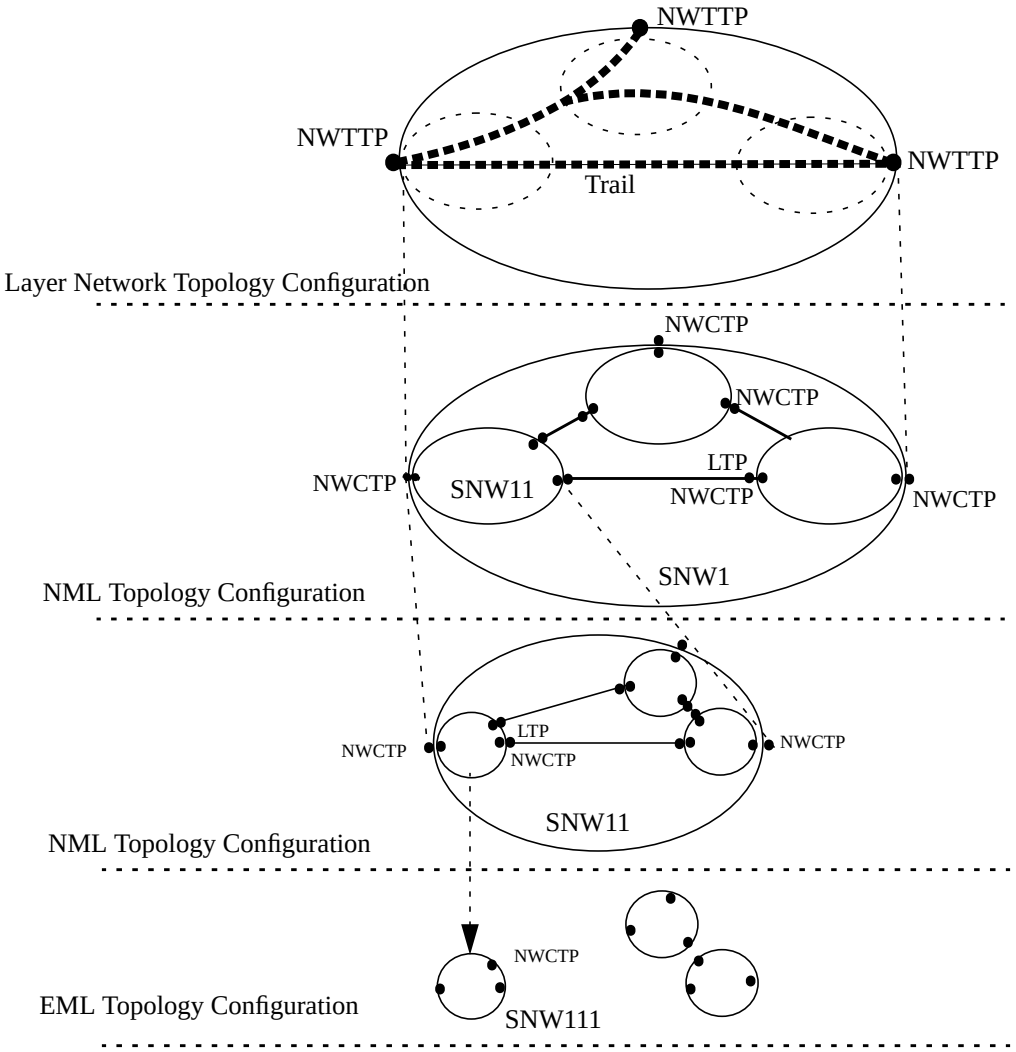


Figure 5-4. Hierarchical Concept of Topology Configurable Resources

LNTC has topology configuration data of the layer network topology resources. LNTC knows network topology information between layer network and lower level network management layer to create a trail.

It has several kinds of interfaces among the different computational objects such as NML-TC and different management domain computational objects, etc. It also has a federation interface with neighboring layer network.

The functions of LNTC are as follows:

- Provisioning

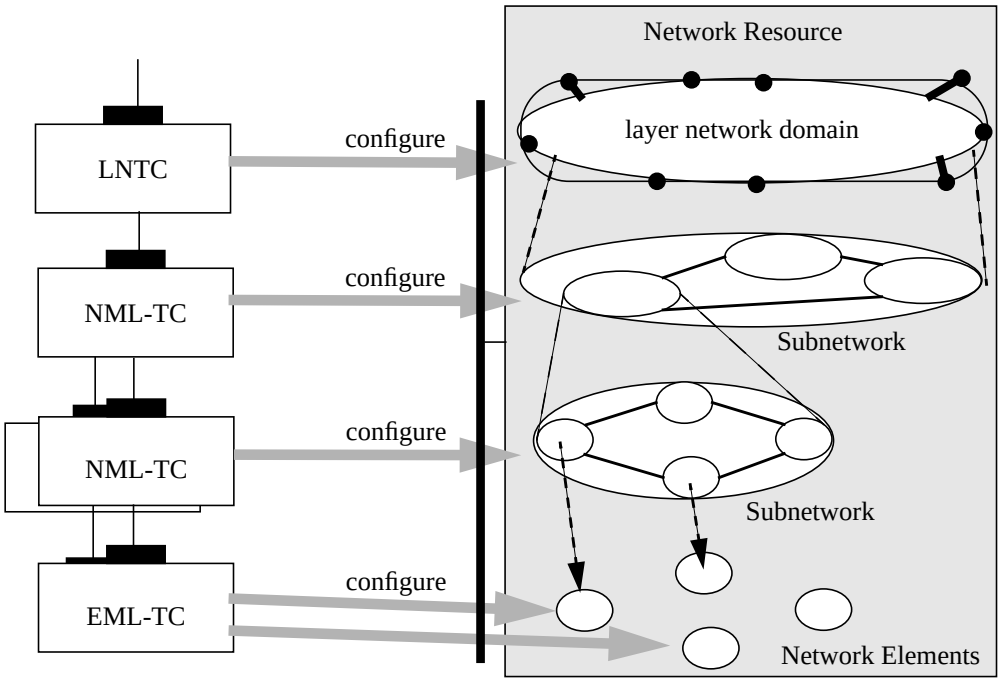


Figure 5-5. Computational Objects in NTCM

- LNTC controls LNC to begin or end its operation.
- When there is any change in layer network topology resources, LNTC sends the changed information to LNC.

• Status and control
- LNTC monitors and controls the status of layer network topology resources.
- LNTC monitors and controls the status of LNC.

• Installation support
- LNTC creates a LNC when a new layer network has been configured.
- When a new trail has been setup by CM domain, LNTC creates the corresponding Network Train Termination Points (NWTTPs) and updates its layer network configuration information.

5.4.2 NML Topology Configurator

Network Management Layer Topology Configurator (NML-TC) configures NML topology resources. The examples of these resources are network connection termination point, link termination point, network termination point pool.

NML-TC has topology view information between associated management layer network topological resources to create a subnetwork connection including a link connection.

It has also several kinds of interfaces among the different computational objects such as LNTC, EML-TC, NML-TC and management domains, etc.

The functions of NML-TC are as follows:

- Provisioning
  - NML-TC controls NML-CP to begin or end its operation.
  - When there is any change in NML topology resources, NML-TC sends the changed information to NML-CP.
- Status and control
  - NML-TC monitors and controls the status of NML topology resources.
  - NML-TC monitors and controls the status of NML-CP.
- Installation support
  - NML-TC creates a NML-CP when a new subnetwork has been configured.
  - When a Network Termination Point Pool (NWTpPool) or Network Connection Termination Points have been created as a result of the setup of a connection in a server layer, NML-TC updates its NML topology configuration and inform NML-CP. Figure 5-6 shown this server-client relationship between layer networks.
  - When a topological link have been created, NML-TC creates a set of Link Termination Points (LTPs) and updates its NML topology configuration.

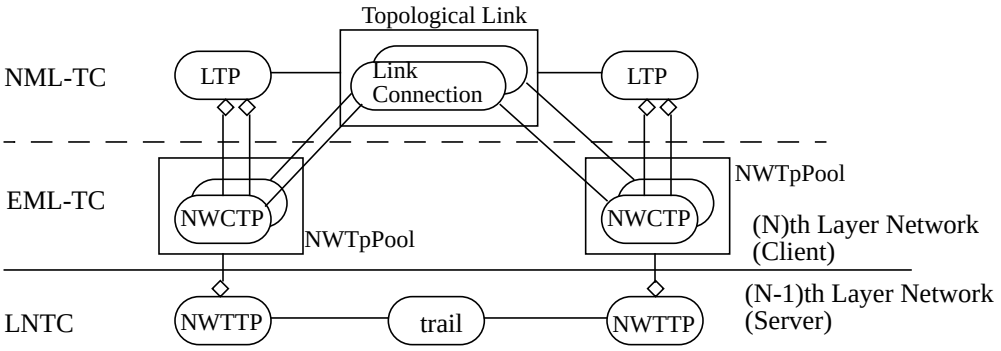


Figure 5-6. Server-Client Relationship in NTCM Domain

5.4.3 EML Topology Configurator

Element Management Layer Topology Configurator (EML-TC) configures EML topology resources. The related resources are network connection point, network termination point pool.

It has topological information between higher level NML topological resources and its own topological resources to create a subnetwork connection.

It has also several kinds of interfaces among the different computational objects such as NML-TC and management domains, etc.

The functions of EML-TC are as follows:

- Provisioning
  - EML-TC controls EML-CP to begin or end its operation.
  - When there is any change in EML topology resources, EML-TC sends the changed information to EML-CP.
- Status and control
  - EML-TC monitors and controls the status of EML topology resources.
  - EML-TC monitors and controls the status of EML-CP.
- Installation support
  - EML-TC creates a EML-CP when a new network element has been configured.
  - When a Network Termination Point Pool (NWTpPool) or Network Connection Termination Points have been created as a result of the setup of a connection in a server layer, EML-TC updates its EML topology configuration and inform EML-CP.

5.4.4 Relation to Other Domains

5.4.4.1 Connection management Domain

There are three point concerning the relationship between CM and NTCM:

- NTCM configures CM object group
- NTCM gives network topology information to CP and LNC
- NTCM uses CM services when it configures a topological link.

5.4.4.2 Other management domains

NTCM Domain also coordinates with other management functions (fault management function or performance management function, etc.).

1. Fault management domain interface
   - Each layer NTCM CO will enable the exclusion of faulty network topological resource from operation and as a result it may rearrange network resource or reroute traffic.
   - Each layer NTCM CO will initiate routine maintenance when executed automatically or a scheduled periodic basis.

2. Performance management domain interface

   - Each layer NTCM CO coordinates with performance management domain to change routing path according to the load status of network resources.

5.4.4.3 Network Administrator interface

Network administrator is responsible for configuration (installation/provisioning/modification/remove, etc.) of network resources.

Network administrator interfaces NTCM domain by the workstation functions. It translates information from network administrator to an adequate format of NTCM domain, vice versa.

Network administrator can interact with each NTCM CO respectively or can access NTCM COs through intra NTCM interface.

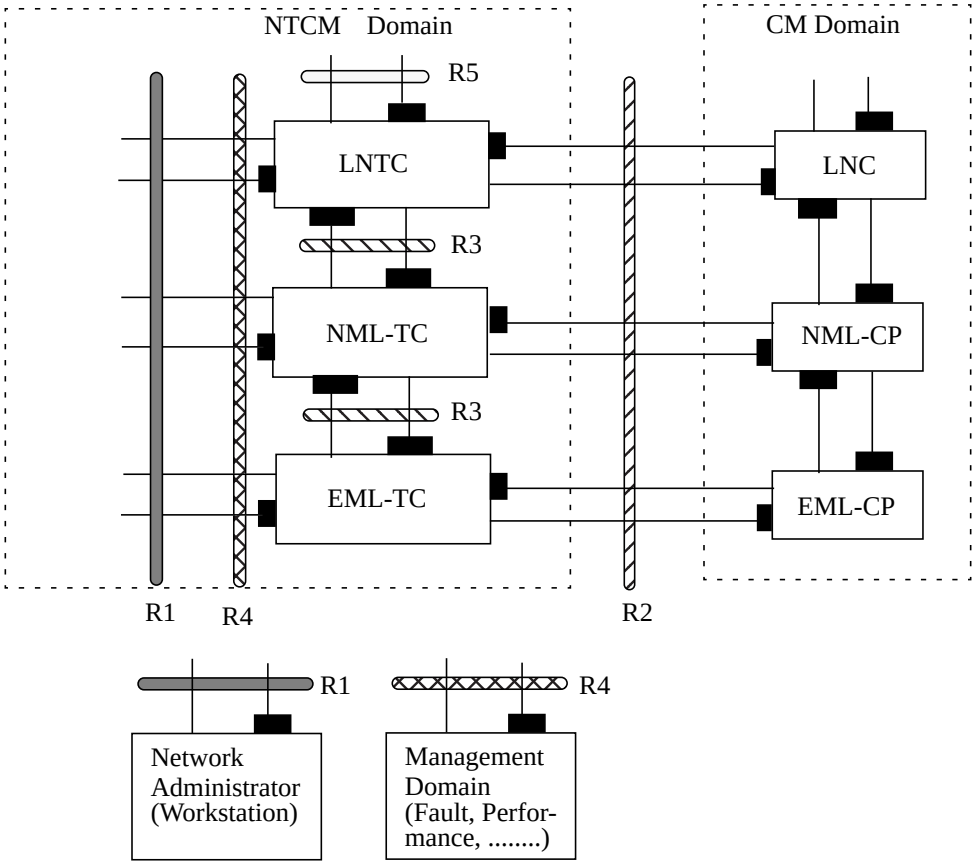


Figure 5-7. Computational Interfaces of NTCM COs

5.4.5 Computational Interfaces of NTCM COs

In this section, basic operation and behavior executed through different interfaces are briefly described. There can be other operation and behavior executed through each interfaces according to the detailed function requirements.

Figure 5-7 shows the computational interfaces supported by NTCM COs.

5.4.5.1 Network Administrator interface (R1)

- Installation support
  - Creation/Deletion of topological resource (object)
  - Creation/Deletion/Modification of topological information (configuration data) within NTCM
- Provisioning
  - Begin/End of Connection Management Object Group
  - Lock, Shutdown, Unlock (CM computational object and topological resource)
  - Assign topological resource to (LNC, NML-CP, EML-CP, NWTPPool)
- Status control
  - Notification result report (creation, deletion, state change)
  - Request/report status (administrative, usage, operational state)
- Topological information request/report
  - Each layer topological resource (LNC, NML, EML)
  - Routing information between associated networks
  - Bandwidth of topological resource

5.4.5.2 Connection Management interface (R2)

- Provisioning
  - Begin/End of Connection Management Object Group
  - Lock, Shutdown, Unlock (CM computational object and topological resource)
- Topological information request/report
  - Each layer topological resource (LNC, NML, EML)
  - Routing information between associated networks
  - Bandwidth of topological resource

5.4.5.3 Intra NTCM interface (R3)

- Topological information change report
- For network administrator to be able to access specific NTCM COs

5.4.5.4 Management Domain interface (R4)

Operation and behavior of this interface will be needed to further works.

- Fault management domain interface
- Performance management domain interface

5.4.5.5 LNTC federation interface (R5)

This is a federation interface needed to exchange topology information between neighboring layer network NTCMs.

Operation and behavior of this interface will be needed to further works, especially according to the results from network federation works.

- topological information request/report

6. Fault Management

6.1 Scope

Fault Management is concerned with detection, localization and correction of abnormal behavior of the telecommunication network and its environment. Within the context of TINA, fault management is related to the following areas:

- within service management: recognize and handle faults related to telecommunication services
- within network resource management: management of faults within NEs and communication facilities between them
- within management of DPE: management of the DPE and the application which use it

In this document, the management of fault in the context of network resource management is specifically dealt. For simplicity, the term “fault management” within this document will refer to fault management in network resource management unless noted otherwise.

The contents of this chapter include the following:

- Definition of Fault Management activities in a network system
- Description of Fault Management aspects of the Network Resource Information Model
- Description of a set of Computational Objects and their computational interfaces in the network system

6.1.1 Architecture Overview

Figure 6-1 provides an overview of the current fault management architecture. In the figure, resources in layer networks are abstracted by information models. The information models in the network system are categorized into three aspects according to the TINA functional layering concept, e.g., network element layer (NEL) aspect, resource management layer (RML) aspect and service management layer (SML) aspect. The TMN Operations Functional Hierarchy is applied to the layering of the computational model. The TMN management service hierarchy for particular functions is specified as a computational interface.

SML functions are considered to be users of the Network Resource Management Service. The Network Resource Information Model provides a uniform view of network resources. Actual network element (NE) level information models may vary depending on the transmission technology, but are not visible from RML. Network management layer (NML) application mainly manages networking of the subnetworks. A smallest subnetwork in the NRIM represents a domain of NE level resources. NE level resources are considered to be represented by various existing information model standards. An element management layer (EML) application is responsible for the NE resource management of a particular type of NE information model. EML applications have the function to represent NE resources as a subnetwork of the NRIM.

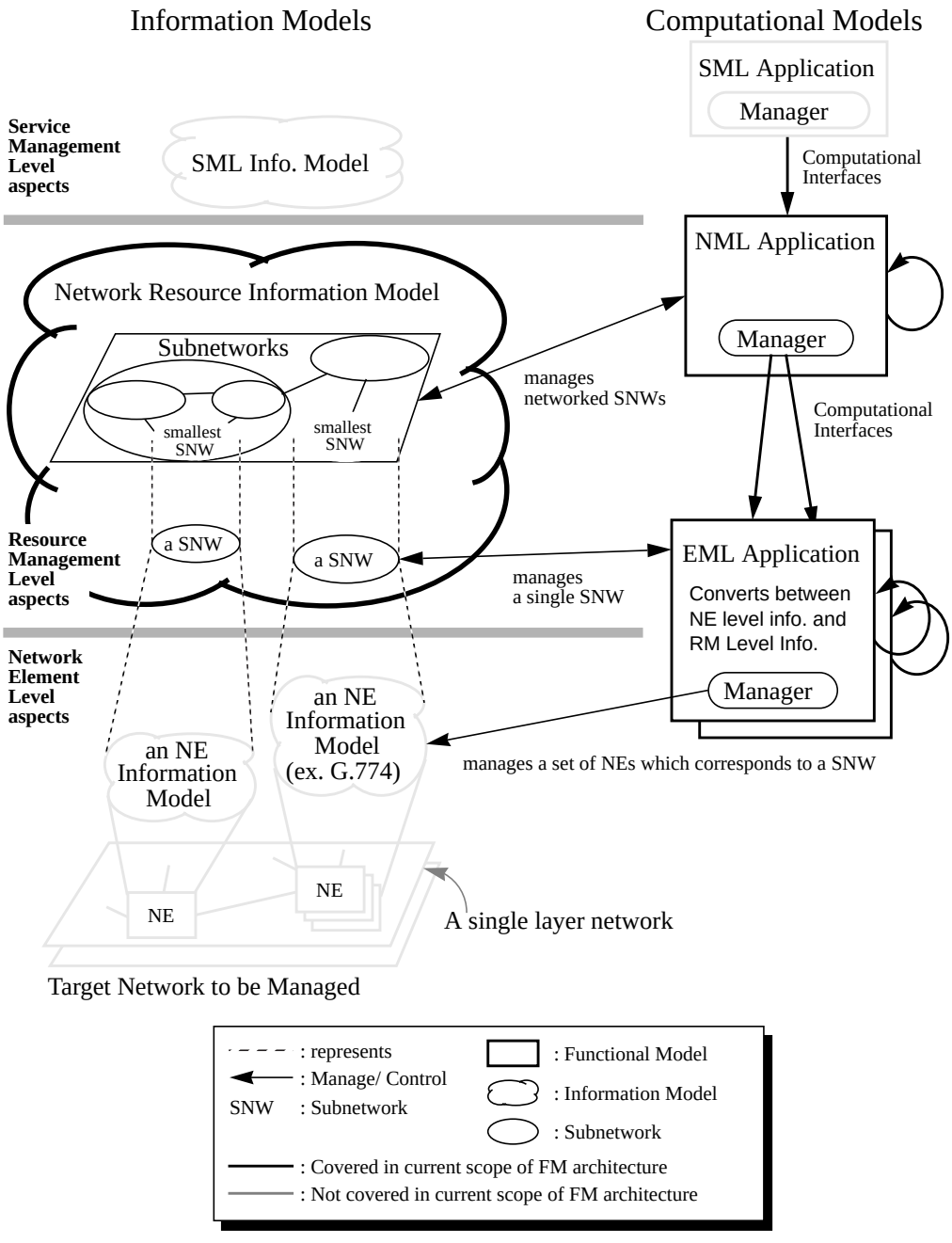


Figure 6-1. Fault Management Architecture Overview

6.1.2 Fault Management Activities

Fault management activities are performed through interactions between the fault management service user and the fault management functions. This section describes the fault management activities in terms of operations done within fault management functions and interactions between fault management functions and fault management service users.

Depending on the case in which fault management is active, the following five different activities are identified [47].

- alarm surveillance
- testing
- fault localization
- fault correction
- trouble administration

Alarm surveillance permits monitoring of resources and makes information about fault status available outside the resource itself.

Fault localization identifies the specific resources that are responsible for improper behavior within the network. The fault management functions perform internal fault localization procedures through interrogation of previous alarm records and testing of a set of possibly related resources. Also, the fault management service user may perform fault localization through issuing diagnostic requests for a set of resources.

Fault correction refers to those activities performed by fault management functions for the restoration of the functions of resources currently in fault condition.

In general, depending on the situation, a fault within a network can be handled in three different levels: network resource level, fault management level, and fault management service user level. In network resource level, network resources perform fault correction through automatic protection and restoration for failed resources[1]. In this level, a notification of the automatic fault correction is sent along with the information on failed resource. Fault correction in fault management level refers to those activities done by fault management functions for the restoration of functions of failed resource[2]. In this case, if the failed resource is associated with a set of backup resources, the fault management functions perform automatic restoration procedure by activating one of the backup resources. Fault correction in fault management service user level refers to those activities of fault management service users for the correction of faults reported by fault management functions. In this case, for the correction of a fault from the network, the user may request alternate resource

1. A typical example can be automatic protection switching (APS) found in SONET.
2. This situation assumes that the network resources do not have automatic fault correction capabilities.

setup to network resource management functions. However, this request is likely to be sent to resource configuration or connection management rather than fault management functions.

Testing/diagnostics is associated with testing and analysis of circuits, paths and transmission media and the supporting resources in NEs. Testing/diagnostics can be carried out either within one NE or through multiple subnetworks or networks.

Trouble administration is an activity between manager and agent which enables troubles to be reported and their status tracked. Trouble administration services include request trouble report format, enter trouble report, add trouble information, cancel trouble report, request trouble report status, review trouble history, add trouble information, attribute value change notification, object creation/deletion, verify trouble repair completion, and modify trouble administration information.

Procedures within fault management functions and interactions with fault management service user for trouble administration are for further study.

6.2 Functional Requirements

This section describes the functional requirements of Fault Management. Scope of the functional requirements are for providing information and computational specification which confirm the TINA-C telecommunications management.

Fault management shall provide a set of functions which enables the detection, isolation and correction of abnormal operation of the telecommunication network and its environment.

6.2.1 Alarm Surveillance

- A fault within a network shall be detected by fault management functions by receiving alarms from managed objects (MOs) in the network.
- There can be multiple alarms which are related to the same fault. Fault management shall create alarm records which can be used for activities such as fault localization, fault correction, and alarm summary reporting.
- Detected faults shall be filtered and, after fault localization procedure within the fault management functions, reported to the fault management service user.
- In NML
  - network fault event correlation and filtering: using the network level information, remove redundancy and narrow the range of possible root causes of a fault
- In EML
  - alarm reporting: reports and controls the reporting of alarms and related information.

- alarm summary: reports and controls the reporting of a summary of the current alarm conditions of specified resources. Reports are available on a scheduled and on a demand basis.
- alarm event criteria: manages the criteria used to determine when a certain condition is to be considered an alarm.
- alarm indication management: controls audible and visual alarm indications.
- log control: control of the logging and the retrieval of alarm history for an NE.
- alarm correlation and filtering: alarm notifications are sent to report alarms that are not redundant within the scope of an NE or group of NEs.

6.2.2 Fault Localization

- A fault within a network may result in multiple alarms from MOs directly or indirectly related to the fault. The fault management functions shall include fault localization capabilities, which determine the root cause of the alarms received.
- In NML
  - network fault localization: analyzes filtered alarms, diagnostics and other symptoms of a fault to identify the root cause. Based on this analysis, network fault localization determines if service is affected by the fault. If service has been affected, then status information is updated.
- In EML
  - NE(s) fault localization: select and schedule diagnostic, exercise, audit, etc. If a root cause is found within an NE or group of NEs, that fact is reported.

6.2.3 Fault Correction

- Fault correction is responsible for the repair of a fault and for the control of procedures that use redundant resources to replace equipment or facilities that have failed.
- In NML
  - scheduling and dispatch administration of repair force: determines what analysis, testing, or repair activity is to be performed and assigns it to another functionality. Scheduling and dispatch administration of repair force also monitors the efficiency and timeliness of repair.
- In EML
  - NE(s) fault correction: manages redundant units (hot-standby) or isolates a unit with a fault. Reporting of automatic restoration processes carried out within an NE or groups of NEs working together. A report is made of a successful restoration or an unsuccessful attempt at restoration.

6.2.4 Testing/Diagnostic

- Testing is an activity within fault management functions for simply verifying the functionality of resources in the system. Diagnostic represents a service from fault management functions which is invoked by user. Diagnostic includes not only the reporting of test results of a set of MOs but also analysis of testing results to find out the main cause of abnormal behavior within the network.
- The fault management functions shall include capabilities of testing/diagnosing one or more MOs in the network for various purposes including fault identification and periodic maintenance.
- In NML
  - connection selection, test correlation and fault location: systematically tests the segments of a connection referred for test to determine which segment or segments contain a fault or degraded component.
  - selection of test suite: receives requests that a specific part of a connection be tested, selects an appropriate suite of tests, receives results, and returns result to the requestor.
- In EML
  - test access configuration: manage test access arrangements
  - test connection configuration: manage the test configuration of a connection.
  - NE(s) test control: control the performance of a test or a suite of tests
  - results and status reporting: report results of tests and status information
  - test access path management: manage test access path resources

6.2.5 Trouble Administration

- Fault management shall provide an activity between manager and agent which enables troubles to be reported and their status tracked.
- It coordinates action to investigate and clear the trouble and provides access to the status of services and the progress in clearing each trouble.
- In NML
  - trouble ticket administration: provides for a clearing house for all trouble reports, whether originated by customers, internal staff, or analysis of alarms or performance monitoring exceptions (reports from analysis are called trouble tickets). The progress of fault correction and the clearing of trouble reports and trouble tickets is tracked.
- In EML
  - no EML functionality has been identified.

6.3 Information Viewpoint

6.3.1 Scope

The Network Resource Information Model (NRIM) specifies an information model for network resources and their management support functions using the ODP information viewpoint. This section addresses the fault management aspect of the NRIM. Readers of this section are expected to be familiar with the Network Resource Information Model Specification [1]. This version of the document only addresses high level abstract model of information objects and their detailed specification will be provided either in a future version or in a separate document.

The information related to the Fault management may include the following:

- Alarm management information,
- Test management information,
- Trouble administration information,
- Generic object management information.

Basic concept for the fault management information modeling architecture is applying the OSI systems management framework to the TMN functional hierarchy, to take advantage of the several existing fault management information models in the TINA resource management.

This version of the document mainly covers the generic fault management information modeling concepts, and conceptual level description of the alarm management information. Specification of the alarm management conceptual objects can be found in the fault management fragment of the Network Resource Information Model Specification [1]. Other information models regarding fault management are for further study.

6.3.2 Modeling Concepts

6.3.2.1 Information Level Aspects

Figure 6-2 illustrates the separation in information models. Several existing information models for various transmission technology abstracts Network Element Level resources. The Resource Management Level aspect focuses on management of NE level resources of various technology as well as network level resources. A purpose of the NRIM is to provide a uniform view of the network resources. The network resource management does not cover the Service Management Level aspects. The semantics of each aspect is described in the TINA-C Management Architecture [7].

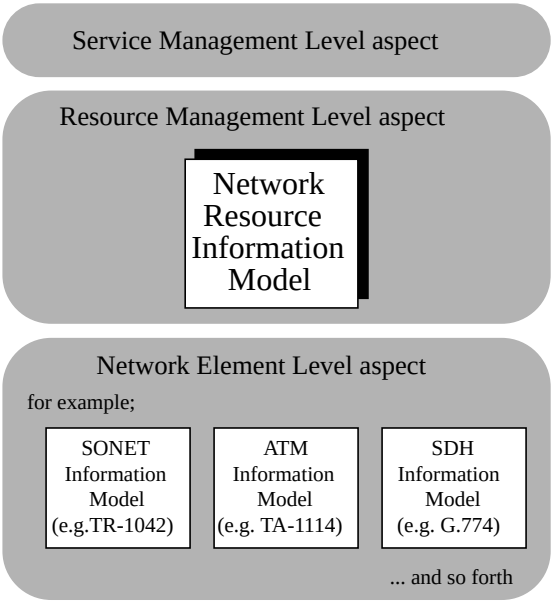


Figure 6-2. Information Model Separation

Figure 6-3 illustrates the relationship between NE level information models and NRIM. The fault management aspect of the NRIM provides information that can be used for management and control of Resource Management Level resources in terms of fault management.

A group of NE level objects can be seen at the Resource Management Level as a single smallest subnetwork (SNW) object. Resource Management Level information focuses on networked subnetwork resources as well as network level resources such as network topology.

The smallest partition of the Resource Management Level subnetworks represents a group of NE level objects. A subnetwork may consist of objects which represent resources in one or more NEs. At the Network Element Management Level, those detail level objects can be recognized. Various types of NE information models, such as a SDH and an ATM, may be uniformly abstracted at the Resource Management Level using NRIM. The smallest subnetwork of a recursive descent of a given top NRIM subnetwork, maps 1:1 with a NE fabric Managed Object. Thus a focused fault origin can be determined at NRIM level.

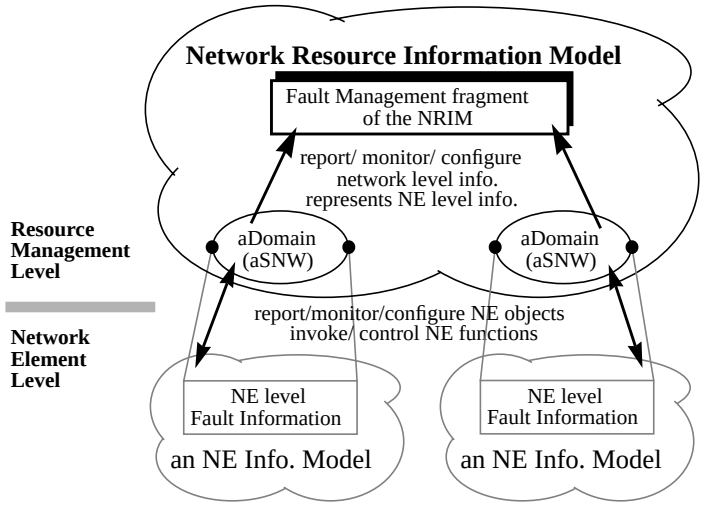


Figure 6-3. Fault Information Models

6.3.2.2 Related Standards

Several information models including fault management are available as international or industrial recommendations and standards. Those models mainly aims at the Network Element management level, but their logical architecture is applicable to the Resource Management level. The fault management aspect of the NRIM is based on existing standards and other sources, and the challenge of this section is to incorporate those models into network level fault management.

6.3.3 Alarm Management Information

An alarm is a notification of a specific event which is defined in the alarm management system. The specific event may include not only errors but normal behaviors of a managed object, and therefore an alarm may or may not represent an error. An error is a deviation of an object from normal behavior, while a fault is the physical or algorithmic cause of a malfunction and manifest itself as an error of the corresponding managed object.

An alarm report is a specific type of event report used to convey alarm information, and its form may be defined as an interface for a computational object.

The alarm management aspect of the NRIM provides information that is necessary for representing an alarm, monitoring alarm conditions, logging alarm for later analysis, and configuring the alarm management information. The alarm management capability may be included in objects at the Resource Management Level as well as objects at the Network Element Level. It is expected that the NE information models have a basic alarm management capability which can be found in current international and industry standards. This section describes the Resource Management Level alarm information.

Alarm Management information may be included in two types of information objects; objects to be managed by the alarm manager, and objects which support the alarm manager. Characteristics of the former objects are described in Section 6.3.3.1, and the latter objects are described in Section 6.3.3.2. All objects described in this section are Resource Management Level objects.

6.3.3.1 Resource Objects with Alarm Management Capability

The alarm management service may be applicable to any resource object that supports the necessary information for alarm management such as alarm reporting and perceived severity assignment. The resource object may include service resource objects and DPE resource objects as well as network resource objects. This section calls those object types Alarmable Resource Object (ARO) as a general term, and describes information for AROs that are recognized so far. An ARO may include some of or all of the information to be described. Additional information may be necessary for a particular alarm management service to be added, but it is for further study. Detailed specification of AROs will be included in actual resource objects such as connections and termination points.

6.3.3.1.1. Attribute Types

The following attribute types that are defined in current standards are necessary for ARO.

- Alarm Status (X.721)
- Alarm Severity Assignment Profile Pointer (M.3100)

6.3.3.1.2. Notification Types

Alarm notification is one of the notification types defined in CCITT Rec.X.721[43]. The following alarm notification types are defined in the recommendation.

- Communications alarm
- Environmental alarm
- Equipment alarm
- Processing error alarm
- Quality of service alarm

NE level network resource objects may also emit Equipment Alarms. Network resource management concerns fault status of equipments, but it is assumed that an Equipment Alarm notification at an NE level object is considered as a local alarm notification at the corresponding Resource Management Level object, and the alarm will be reported as a Communication Alarm of the Resource Management Level object if necessary.

6.3.3.2 Support Objects for Alarm Management

The following managed object classes are defined in NRIM, and support the Resource Management Level alarm management.

- Alarm Record

- Log

The following objects are defined in existing standards, and support the Resource Management Level alarm management.

- Discriminator (X.721)
- Event Forwarding Discriminator (X.721)
- Event Log Record (X.721)
- Log Record (X.721)
- Alarm Severity Assignment Profile (M.3100)
- Current Alarm Summary Control (X.821)
- Management Operations Schedule (X.821)

6.3.3.3 Information Object Types and Relationship Types

OMT notation of the fault management fragment information object types and relationship types on the alarm management aspect are illustrated in Figure 6-4.

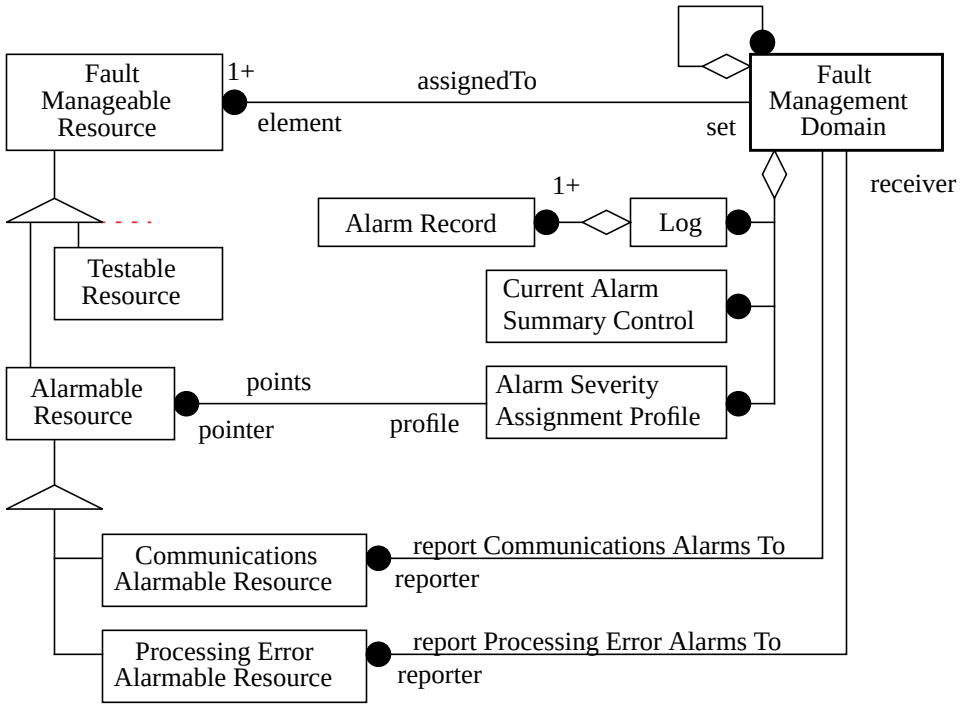


Figure 6-4. Information Object and Relationship Types of Fault Management Fragment

Objects and key attributes mentioned in this section are listed in Table 6-1. See each reference for complete specification. Objects marked as NRIM are defined in the Network Resource Information Model Specification.

Table 6-1. Managed Object Classes

Managed Object Classes	Managed Object Class Description	Key Attribute types	Key Attribute type Description
Alarm Severity Assignment Profile (M.3100)	Specifies the alarm severity assignment for managed objects. (Instances of this object are referenced by the alarm Severity Assignment Profile Pointer attribute in the managed objects.)	Alarm Severity Assignment List (M.3100)	An attribute type whose value provides a listing of all abnormal conditions that may exist in instances of an managed object class, and shows the assigned alarm severity information (minor, major etc.) for each condition.
Current Alarm Summary Control (Q.821)	Provides the criteria for generation of current alarm summary reports.	Alarm Status List	Consists of a set of possible Alarm Status. An Object with matching Alarm Status attribute value(s) is to be included in a current alarm summary report.
		Object List	Set of object instances to be included in a current alarm summary report.
		Perceived Severity List	Consists of a set of possible Perceived Severities. An Object with alarmOutstanding Alarm status attribute value that matches one of the Perceived Severity in the list is to be included in a current alarm summary report.
		Probable Cause List	Consists of a set of possible Probable Causes. An Object with alarmOutstanding Alarm status attribute value that matches one of the Probable Cause in the list is to be included in a current alarm summary report.
Log (NRIM)	Keeps a history of events for a particular entity.		

Table 6-1. Managed Object Classes

Managed Object Classes	Managed Object Class Description	Key Attribute types	Key Attribute type Description
Alarm Record (NRIM)	Used to define the information stored in the Entity log as a result of receiving alarm notifications or alarm reports.	Probable Cause (X.721)	Defines qualification as to the probable cause of the alarm.
		Perceived Severity (X.721)	Indicates how it is perceived that the capability of the managed object has been affected. Following severity levels are defied in X.733. - cleared - indeterminate - critical - major - minor - warning
		Alarm Status (X.721)	See description of ARO
Alarmable Resource (NRIM)	Alarmable Resource Object (ARO) is a general term which is used in this document to represent any NRIM Objects with capability to be managed by the Alarm Manager.	Alarm Status (X.721)	Alarm Status attribute can have zero or more of the following values. See X.731 for semantics. - underRepair - critical - major - minor - alarmOutstanding
		Alarm Severity Assignment Profile Pointer (M.3100)	This attribute identifies an Alarm Severity Assignment Profile object. An object may contain this attribute if it support both Communication Alarm and alarm configuration services.

6.3.4 Test Management Information

Test management provides information that is necessary for invoking the test capability of objects, controlling the testing process, monitoring the testing process, and configuring the testing information. It may also include logging of test results for later analysis. The testing capability may be included in objects at Resource Management Level as well as ones at Network Element Level. It is expected that NE info models have a basic test management capability such as one to be defined in the OSI test management function [46].

The support managed objects that are defined in CCITT Rec. X.745 may also support resource management level test management.

6.3.5 Trouble Administration Information

The trouble administration information may include the information that is necessary for tracking various transactions regarding the fault management, such as trouble reports, proceeding of the restoration of the trouble, re-configuration requests, an engineer dispatching for hardware re-configuration, and so forth.

6.4 Computational Viewpoint

This section describes the computational viewpoint of the network resource fault management architecture. The main objective of this section is to identify computational objects and the interaction between them for fault management activities. The identification of computational objects follows the resource management architecture design process described in [58].

This section is structured as follows: first, the fault management computational architecture overview is described in Section 6.4.1. And the detailed descriptions for each Computational Objects are followed in Section 6.4.2 for Alarm Manager, in Section 6.4.3 for Fault Coordinator, and in Section 6.4.4 for Testing/Diagnostic Server, correspondingly. The interfaces between this COs are not described in this document, and will be described in the Component Specification document.

6.4.1 Computational Architecture Overview

The network resource fault management services are provided by the interaction of the Computational Objects (COs) inside and outside of the fault management area. COs identified for network resource fault management are as follows[3]:

- Alarm Manager (AM)

  The Alarm Manager (AM) receives fault-related alarm from MOs and performs relevant procedures for alarm correlation, alarm filtering, forwarding the alarm to fault coordinator or fault management service user and for alarm record management. Each AM has its own discriminating criteria through which incoming alarms are logged and forwarded to relevant computational objects in the system.

- Fault Coordinator (FC)

  The Fault Coordinator (FC) includes capabilities to internally analyze alarms received from multiple MOs to determine next possible step for fault localization/ correction. For this purpose, the FC correlates all available information to refine information concerning the root cause of the event in question. During the analysis, the TDS can be invoked to run tests as appropriate.

- Testing/Diagnostic Server (TDS)

3. The COs defined in this document are initial set. Other COs (e.g., concerning trouble administration) can be further identified as needed. Also, the objects defined in this section may be further decomposed into a set of sub-level objects.

The Testing/Diagnostic Server (TDS) is concerned with testing of MOs for the purpose of service and function verification of MOs. From fault management's view, the TDS is invoked by either fault coordinator or fault management service user. However, it is also possible that the TDS can be invoked by other computational objects in the system, e.g., resource configuration and connection management objects or scheduler[4].

The basic computational architecture for fault management is derived from these COs. Figure 6-5 is a diagram which shows the interactions among COs in fault management functions. The dotted round rectangle in Figure 6-5 shows the functions of fault management and interfaces for fault management services and activities.

Figure 6-5 can be interpreted as those interactions that are provided by fault management COs that manage the network at various levels. For example, the same mechanism can be applied to either the bottom level subnetwork (which is definitely one or a set of network elements) or the top level subnetwork (which can be the whole network operated by a network operator or a set of those networks). Figure 6-6 describes the basic architecture for fault management computational model incorporating this idea.

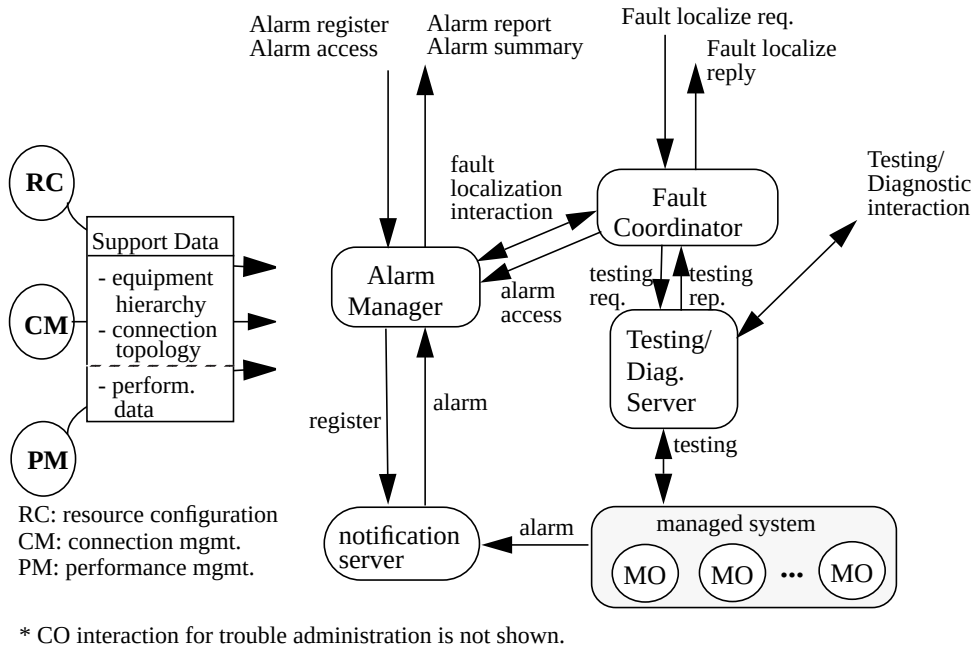
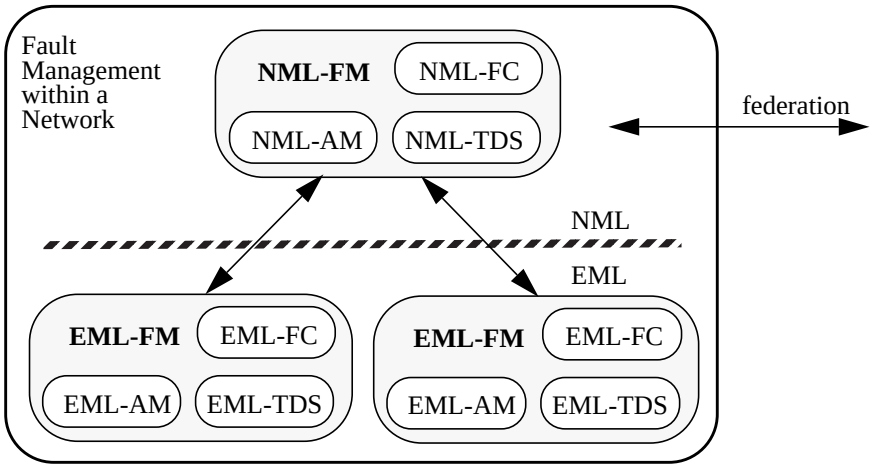


Figure 6-5. CO interactions in fault management

4. Currently, the scheduler is not included in fault management computational objects. Whether to include the scheduler in fault management area or not is for further study.



AM: Alarm Manager, **FC**: Fault Coordinator, **TDS**: Testing/Diagnostic Server

Figure 6-6. Basic computational architecture

6.4.2 Alarm Manager

6.4.2.1 EML Functions

General description

An event alarm, which is generated by an MO, is first received by EML Alarm Manager (ELM-AM). The EML-AM first makes the corresponding event log record and filter the alarm. In case the alarm passes filtering, EML-AM prepares the corresponding alarm record which can be referred for further alarm analysis and reporting purposes. The filtered alarm passes the alarm correlation function which provides redundant alarm removal and initiates fault localization procedure, if necessary. The alarm is finally passed to NML Alarm Manager (NML-AM) through alarm forwarding functions. Figure 6-7[5] shows the functional architecture for EML-AM.

Functions of EML-AM

- log functions

  The log functions receive event alarms from MOs and prepares an event log record if the alarm satisfies the criteria defined within the log. The log function passes the event alarm to alarm filtering functions.

5. This figure shows the required functions and their relations in EML-AM and does not prescribe the internal structure of the object. And, the same statements are applied for the following figures about the functiona architecture of COs in this chapter.

- alarm filtering functions

  The alarm filtering functions receive event alarms from log functions, perform alarm filtering, and prepare the alarm record for alarms which passed filter. For alarm filtering, the alarm filtering functions check local data to determine whether reporting of such events as alarms are inhibited, or whether full supporting information is inhibited.

- alarm correlation functions

  The alarm correlation functions receive alarms from alarm filtering functions and perform alarm correlation analysis using equipment hierarchy and connection topology as supporting data. In this phase, any alarm which is a duplicate or a subset of previously reported alarms is discarded.

  Alarm correlation functions interact with EML-FC for fault localization and reported fault corrective interaction. Fault localization interaction is initiated by alarm correlation functions for the determination of root cause of a set of related alarms. The fault localization results are then reported to the NML-AM using alarm forwarding functions. In case a fault is identified in alarm correlation functions, the fault correction interaction occurs with a EML-FC. Note that the fault correction within fault management performs automatic restoration of faulty resources which have back-up resources.

- alarm forwarding functions

  The alarm forwarding functions forward alarms, received from an EML-AM, to destination(s) registered to received alarms from the EML-AM. A typical example of destination is NML-AM where network-wide alarm management actions take place.

Interactions with other COs

- interaction with EML-FC
  - fault localization interaction: requested by an EML-AM and root cause alarm analysis results are returned.
  - fault correction interaction: requested by an EML-AM and results of automatic fault restoration activity within the EML-FC are returned.
  - alarm record access by the EML-FC
- interaction with higher layer COs
  - alarm forwarding destination registration request from higher layer COs, e.g., NML-AM
  - alarm forwarding to higher layer COs
- interaction with EML-Network Topology Configuration Manager(NTCM)
  - get the equipment hierarchy and connection topology data from EML-NTCM

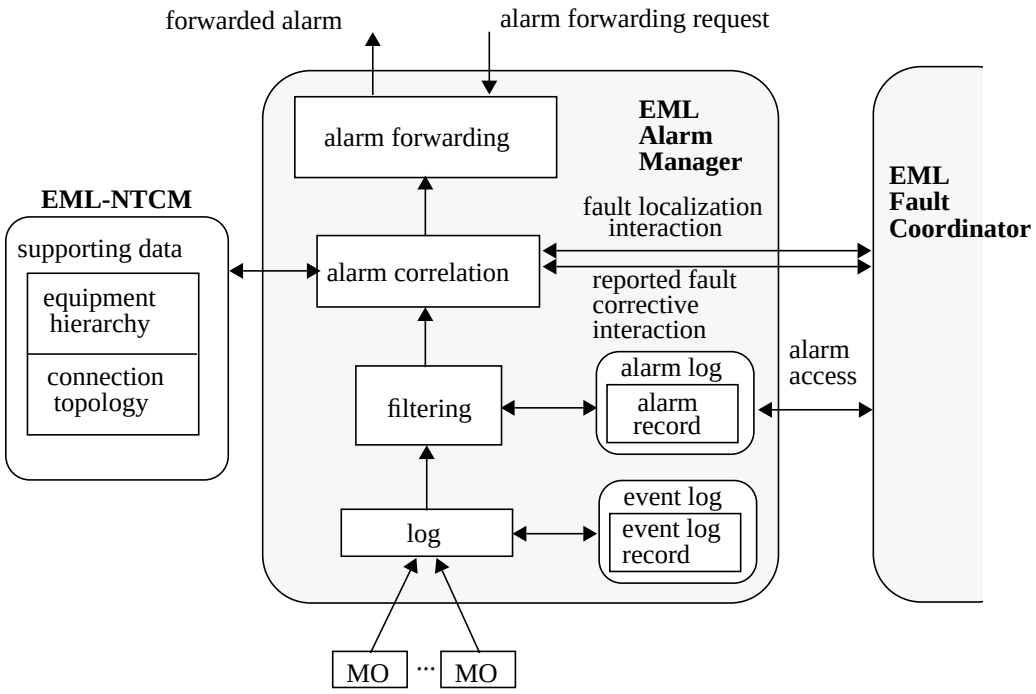


Figure 6-7. EML Alarm Manager functional architecture

6.4.2.2 NML Functions

General description

The structure and functions for NML-AM are similar to EML-AM except that the log function does not exist in the NML-AM. NML-AM receive alarm reports from corresponding EML-AM. The alarm reports from an EML-AM may or may not specify the root cause of the alarm report. For example, when there is a fault in a resource which connects two subnetworks, the fault can be handled in NML-AM. As with the EML case, the NML-AM interacts with the NML-FC to identify the root cause of alarms and forwards alarm records to users[6]. NML-AM makes use of connection topology information between subnetworks as supporting data. Figure 6-8 shows the functional architecture for NML-AM.

Functions of NML-AM

- alarm filtering functions

  The alarm filtering functions receive event alarms from EML-AM, performs alarm filtering, and prepares alarm record for alarms which passed filter.

6. Users in this context may be fault administration functions or another NML alarm management functions.

- alarm correlation functions

  The alarm correlation functions receive alarms from alarm filtering functions and perform alarm correlation analysis using subnetwork connection topology as supporting data which can be obtained from NML-Network Topology Configuration Manager (NTCM). Alarm correlation functions interact with NML-FC for fault localization. Fault localization interaction is initiated by alarm correlation functions for the determination of root cause of a set of related alarms. The fault localization results are then reported to users using alarm forwarding functions.

- alarm forwarding functions

  The alarm forwarding functions forward alarms, received from alarm correlation functions, to destination(s) registered to received alarm from NML-AM.

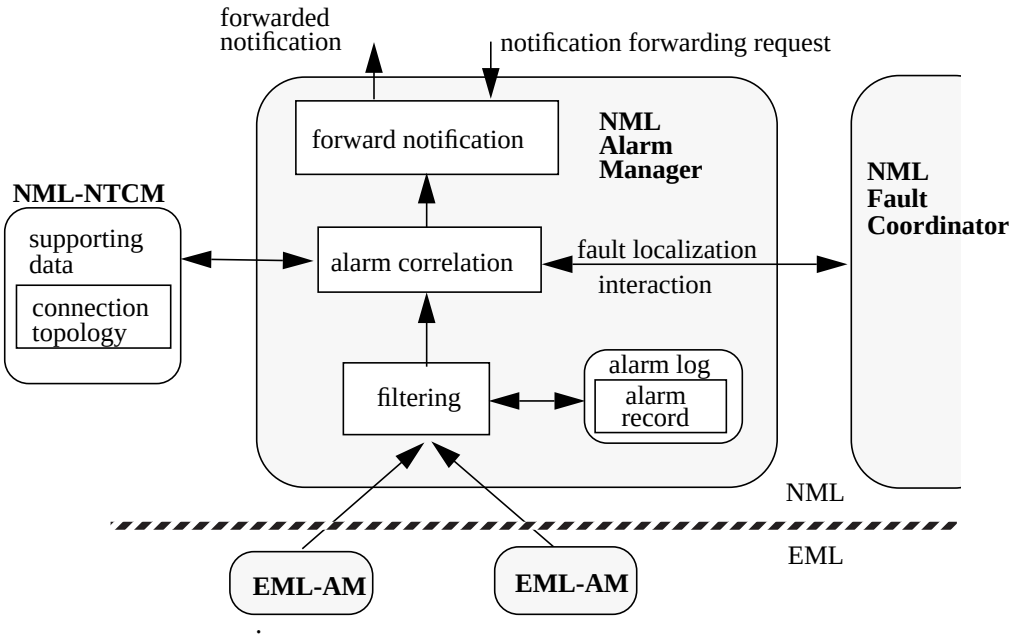


Figure 6-8. NML Alarm Manager functional architecture

Interactions with other COs

- interaction with NML-FC
  - fault localization interaction: requested by NML-AM and root cause alarm analysis results are returned
  - alarm record access by NML-FC
- interaction with users

- alarm forwarding destination registration request from users
- alarm forwarding to users

- interaction with NML-NTCM
  - supporting data access to obtain topological information by NML-AM

6.4.3 Fault Coordinator

6.4.3.1 EML Functions

General description

The functions of EML Fault Coordinator (EML-FC) are alarm analysis and fault correction. The alarm analysis function, upon receiving fault localization request, performs analysis of current alarm records to determine the root cause of a set of related alarms. During this phase, the EML-FC interact with EML-TDS for testing/diagnostic over a set of related resources. The EML-FC performs automatic restoration by activating a back-up resource of a faulty resource. Figure 6-9 shows the functional architecture for the EML-FC.

Functions of EML-FC

- alarm analysis functions

  The alarm analysis functions are activated by a request from EML-AM. They use information about the equipment hierarchy, connection topology, current alarm records, and results of testing on a set of related resources to determine the root cause of a set of related alarms. The identified root causes of alarms are informed to EML-AM for further processing.

- fault correction functions

  The fault correction functions are activated by EML-AM or NML-FC when a resource is identified to be faulty. The EML-FC performs automatic restoration procedures for a faulty resource which has a list of back-up resources by activating one of its back-up resources.

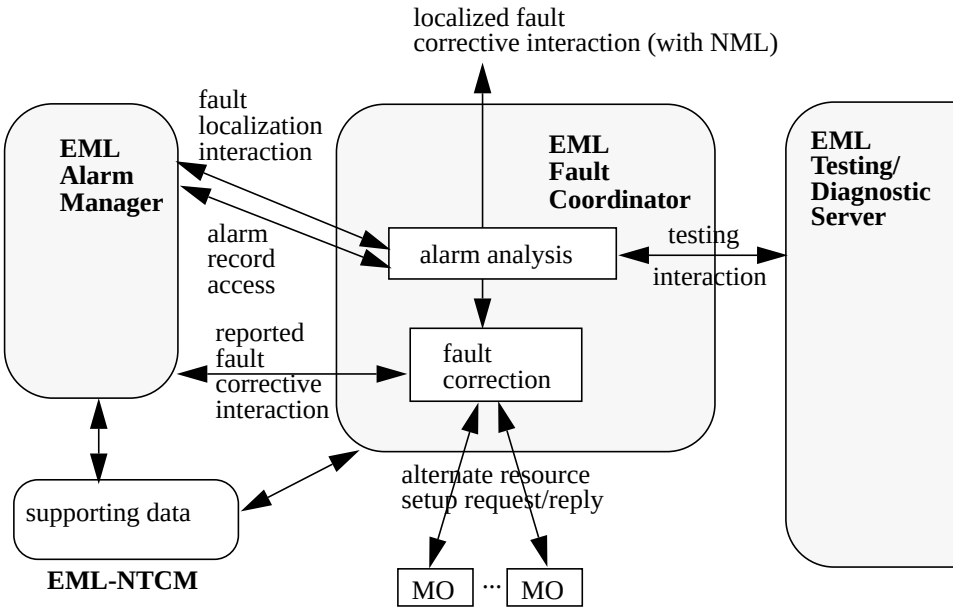


Figure 6-9. EML Fault Coordinator functional architecture

Interactions with other COs

- interaction with EML-AM
  - fault localization interaction
  - alarm record access interaction
- interaction with EML-TDS
  - testing/diagnostic of a set of resources related to current alarms
- interaction with higher layer COs
  - interaction for automatic restoration of fault identified at NML level
- interaction with EML-NTCM
  - get the equipment hierarchy and connection topology data from EML-NTCM
  - report any changes of configuration as a result of fault correction

6.4.3.2 NML Functions

General description

NML Fault Coordinator (NML-FC) performs alarm analysis for the current alarm records and interact with NML-TDS to determine the root cause of a set of related alarms. For the localization of faults which span multiple networks, the NML-FC interacts with NML-FCs in

other networks through federation. During alarm analysis, connection topology between subnetworks is used as supporting data. Figure 6-10 shows the functional architecture for NML-FC.

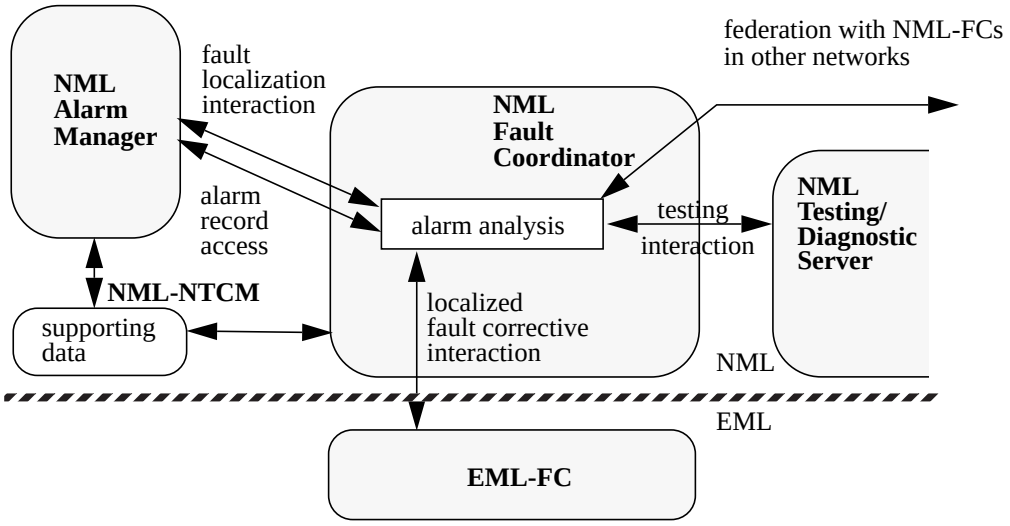


Figure 6-10. NML Fault Coordinator functional architecture

Functions of NML-FC

- alarm analysis functions

  The alarm analysis functions are activated by a request from NML-AM. It uses information about the subnetwork connection topology, current alarm records, and results of testing on a set of related resources to determine the root cause of a set of related alarms. For locating faults which span multiple networks, federation is used for cooperation with other NML-FCs. The identified root cause of alarms is informed to NML-AM for further processing. In case the faulty resource can be automatically restored, interaction between alarm analysis and corresponding EML-FC occur.

Interactions with other COs

- interaction with NML-AM
  - fault localization interaction
  - alarm record access interaction
- interaction with NML-TDS
- interaction with NML-FCs in other networks: federation
- interaction with EML layer COs

- interaction for automatic restoration of fault identified at NML level

• interaction with NML-NTCM
  - supporting data access to obtain topological information by NML-AM
  - report the changes of topological information as a result of fault correction

6.4.4 Testing/Diagnostic Server

6.4.4.1 EML Functions

General description

The EML Testing/Diagnostic Server (EML-TDS) provides capabilities for testing/diagnostic of a set of resources. Testing and diagnostic have different meanings in that a testing is a simple function verification of a set of resources and diagnostic includes a more complex procedure for analyzing the results of testing to find out the main cause of abnormal behavior within the network, if any. Figure 6-11 shows the functional architecture for EML-TDS. The EML-TDS is activated by EML-FC, NML fault management COs, scheduler[7], and other FM clients.

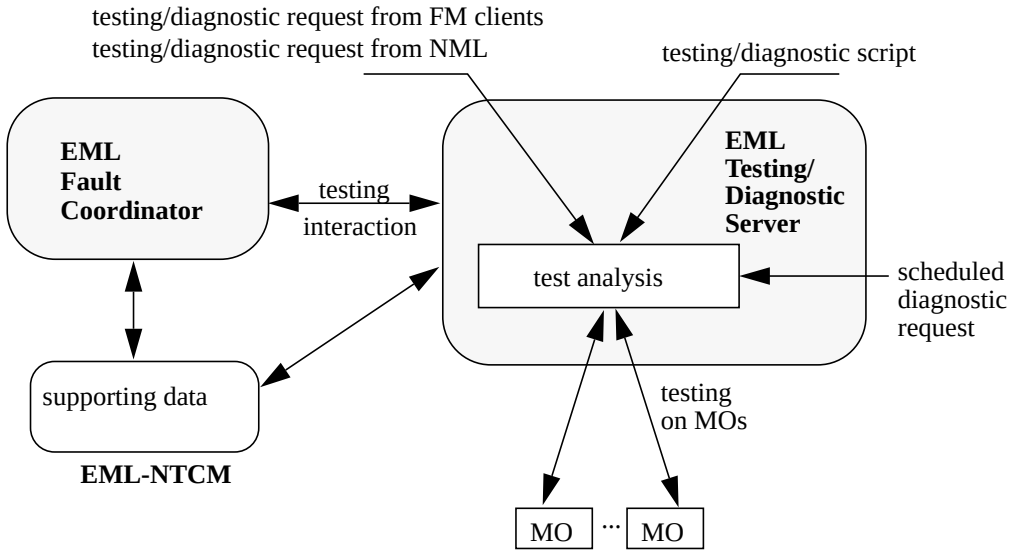


Figure 6-11. EML Testing/Diagnostic Server functional architecture

Functions of EML-TDS

7. The scheduler has not been defined in this document. This document assumes the existence of a scheduler which can be controlled to invoke TDS periodically.

- test analysis functions

  The test analysis functions perform testing on a set of resources which are predefined or specified at the time of invocation. The diagnostic results, if requested, are sent to client. During the analysis, it interacts with EML-NTCM to get the equipment hierarchy and connection topology information as supporting data.

Interactions with other COs

- interaction with EML-FC
  - testing/diagnostic request from alarm analysis function in EML-FC
- interaction with NML COs
  - testing/diagnostic request from NML fault management COs
- interaction with COs in other functional areas
  - request for testing/diagnostic of a set of resources from other functional areas, e.g., resource configuration and connection management.
- interaction with EML-NTCM
  - get the equipment hierarchy and connection topology data from EML-NTCM

6.4.4.2 NML Functions

General description

The NML Testing/Diagnostic Server (NML-TDS) provides capabilities for testing/diagnostic of a set of resources which span multiple subnetwork or multiple networks[8]. Figure 6-12 shows the functional architecture for NML-TDS. The NML-TDS is activated by NML-FC, NML fault management users, scheduler, NML-TDSs in other networks, and other FM clients including resource configuration and connection management.

Functions of NML-TDS

- test analysis functions

  The test analysis functions perform testing on a set of resources which are predefined or specified at the time of invocation. The diagnostic results, if requested, are sent to client. During the analysis, these functions use subnetwork connection topology information as supporting data and, if necessary, interact with NML-TDSs in other networks for testing resources which span multiple networks.

Interactions with other COs

- interaction with NML-FC
  - testing/diagnostic request from alarm analysis function in NML-FC

8. A typical resource which spans multiple networks is a trail (or connection). These logical resources are defined in the network resource information model.

- interaction with users
  - testing/diagnostic request from users (fault administration function or higher layer NML fault management COs)
- interaction with COs in other functional areas
  - request for testing/diagnostic of a set of resources from other functional areas, e.g., resource configuration and connection management.
- interaction with NML-TDSs in other networks: federation
- interaction with EML-TDS
  - networkwide testing/diagnostic can be subdivided into a set of subnetwork level testing/diagnostics. NML-TDS interacts with corresponding EML-TDSs in subnetworks under its control to test or diagnose a network level resources.
- interaction with NML-NTCM
  - supporting data access to obtain topological information by NML-AM

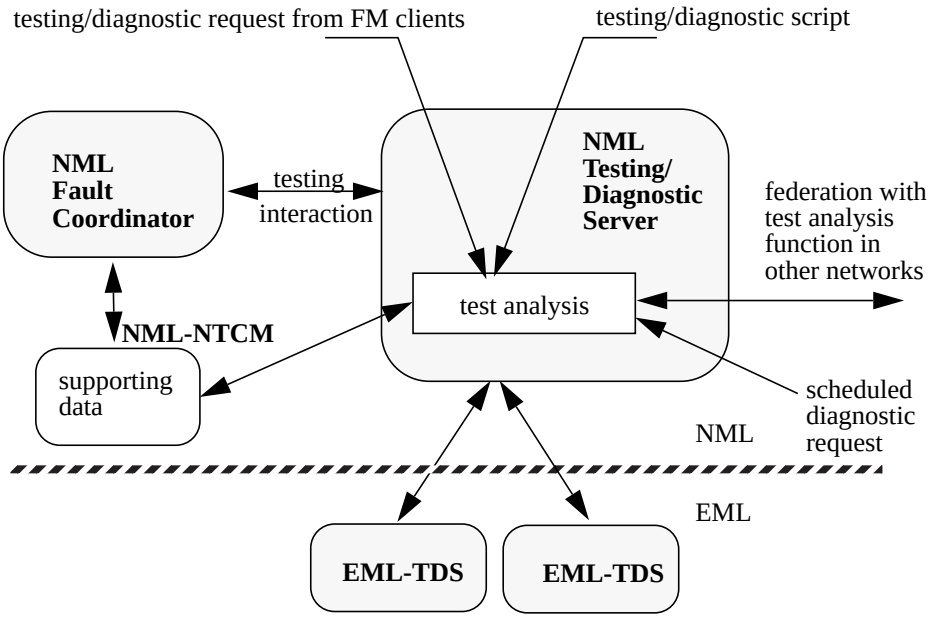


Figure 6-12. NML Testing/Diagnostic Server functional architecture

7. Accounting Management

7.1 Scope

The scope of this section is to describe the resource level accounting management from the following three viewpoints.

- *Information Viewpoint*: the information viewpoint explains the basic concepts and their relationships from the high-level functional point of view.
- *Computational Viewpoint*: the computational viewpoint explains the accounting management components, which supports the basic accounting concepts. They are designed as computational objects, and their specifications in English are given.
- *Engineering Viewpoint*: the engineering viewpoint addresses most practical issues in the accounting management, such as deployment, interworking with and migration from legacy accounting management systems.

In addition to the above, a short introduction is provided to give readers background on the TINA service management, which is expected to be the client of the resource level accounting management functions. At the end of this section, a few accounting management examples are provided, to illustrate some practical insights.

The scope of this document is broad, and the engineering issues in particular are mostly still waiting for serious validation. Therefore the reader should be aware that the examples throughout this document are prepared to illustrate the concepts and the principles, not to emphasize any particular implementation preference.

7.2 Introduction

Accounting Management in TINA has the following major two objectives.

- Accounting as one of FCAPS service management functions: accounting and billing requirements of TINA applications must be satisfied, and they need to be mapped onto the TINA resource layer seamlessly and in the most efficient manner.
- Resource level accounting: usage of resources of the various types such as network resource, DPE resource, software resource etc. need to be measured efficiently, and the measurement be recorded if necessary.

These two objectives are like two sides of a coin, in much the same way both sides need to be addressed and understood to get the whole picture. In this part of the NRA, our focus is mostly on the latter side, i.e. the resource level accounting. Among all the available resources in TINA, our focus in this section is primarily on the network resources. However, the same principle will generally apply to other types of resources. In the following parts of the introduction, we briefly summarize the requisite materials from the service management perspective.

7.2.1 TINA Service Management Principle

Figure 7-1 illustrates the TINA service management principle. The figure illustrates TINA service management approach and its correspondence to the TMN model. TINA service management can be equated with SML within TMN. A major difference however, is that the TINA service management resides on top of the object-oriented TINA resource layer, in contrast to the TMN's layered approach.

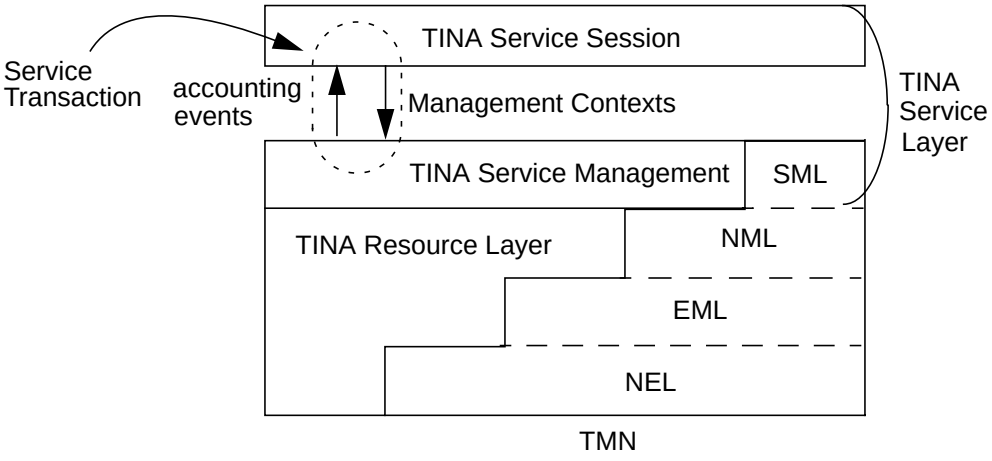


Figure 7-1. TINA Service Management Principle

There are two main architectural components in the TINA service management.

- *Management context*: the FCAPS service management requirements of the TINA service layer are represented by a set of management contexts. The context is configured using a usage assistance service in the service layer, and is then bound to a service session, when the service session is executed. The management contexts are passed to the TINA service management, which subsequently interprets the contexts such that they are implemented in the TINA resource layer.
- *Service transaction*: the service transaction is a construct, which carries the whole service management process. Its objective is to guarantee the integrity of the service and service management requirements. In particular, the billing is based on the fulfillment of the service requirements, which can be measured by the performance monitoring.

During the lifetime of the service transaction, the TINA service management generates events and notifications, reporting to the TINA service session and concerned parties. For example, accounting events may be reported to the end-user, enabling the on-line billing.

The relationship between TINA and TMN are actually several-fold, and more words are necessary to describe it fully.

- *TMN layering*: TINA adopted TMN principle, particularly at its resource layer, and its layering principle as well. TINA however uses TMN layers more of a reference model, rather than a rigid framework for layer-by-layer construction of network management functions. For example, a computational object in TINA resource layer may be positioned at NML, by virtue of its function and its role within a particular management function, such as the connection management. The same computational object, however may be reused at EML for a similar purpose. In particular, the accountable object is a generic interface, such that it can be reused at NML, EML, or even at NEL.
- *Computational architecture*: the computational architecture of TINA, i. e. DPE, supports extra freedom which was not expected in TMN. For example, concepts such as domain, object grouping, and emerging mobile-agent paradigm allows the user of the management services (i. e. TINA service session) to see the network resources in a more flexible, virtual point of view, which can be customized for a particular management needs. These advantages in the computational architecture can be left unused, or less accessible, if the strict TMN layering principle has to be imposed on the TINA resource layer.
- *Interworking with TMN*: TINA has to interwork with TMN. Due to the fact that many TMN based has been installed and are being installed up to NML, the need for interworking is more likely to occur at NML, progressing downward to NEL, not the other way around. Although the NRA would not address this issue up to the level (NEL), the same management approach based on DPE can be extended, and may eventually prevail.

7.2.2 Basic Accounting Cycle

The accounting management consists of four cycles, namely metering, classifying, charging, and tariffing, as it was identified by [59].

- *Metering*: the tracking and recording of usage of resources ([59], p.8). Metering is the first step and the basis of all the following accounting activities. Metering is mostly a resource level accounting problems.
- *Classification*: classification of metering information into a set of classes based on usage of service, resources being used, zone information, which corresponds to the distance between the caller and callee, and so on. A major objective of the classification is to categorize and to reduce the amount of metering information such that tariffing is performed with ease.
- *Tariffing*: a step to calculate charging (billing) information from the classification obtained in the previous cycle and the tariff structure. The tariff structure is often represented by a table of costs for each service category. It may change over time, and its structure usually depends on service provider.
- *Billing*: a process of charging (billing) information being stored and then sent to a customer (an entity the service is being provided). The billing schedule, either monthly, daily, or hourly, can depend on the agreement (contract) between the customer and the service provider.

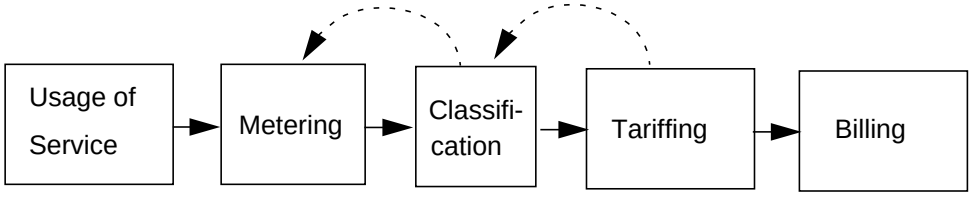


Figure 7-2. Basic Accounting Cycle

Figure 7-2 illustrates the basic accounting cycle. Solid lines represent the flow of information whereas dotted lines represent control of focus, or information flow control. The tariffing structure directly/indirectly influences on how the metering information is categorized at the classification cycle, and the classification categories directly/indirectly influences the metering cycle on how the metering information is collected from usage of service.

7.2.3 Accounting Management Context

Accounting management context (AccMgmtCtxt) specifies the details necessary for the accounting management to perform at each respective stage of the basic accounting cycle. The AccMgmtCtxt contains the following information objects:

- *EventManagementConfiguration*: EventManagementConfigurations gives definitions such as delivery timing, whether it is provider-driven, user-driven, and the type of event channel to be used for accounting events delivery, whether they are *Synchronous, Asynchronous, NotificationServer*, etc.
- *TariffStructure*: TariffStructure is essentially a function. Provider calculates charge/charging-rate from a given accounting state, whether the user is using a video phone or a voice conference. On the other hand, the user tends to look at a service as a sequence of accounting events, selecting a menu or clicking a mouse button, and so on. Therefore we conclude that provider's view is mostly state-based whereas user's view is more of event-based. For this reason, this interface may support the two views. The behavior of the function itself is provider-specific, therefore no common tariffing scheme is assumed. The purpose of this interface is to provide a common object-oriented interface for each provider-specific tariffing scheme.
- *BillingConfiguration*: BillingConfiguration contains types of billing options accepted in the service transaction e.g. *Online, SharedBilling, ThirdParty, Credit-Debit*. Some options are mutually compatible e.g. Online and SharedBilling. For each billing option, more detailed spec. follow, e.g. the name of the third party the bill is to be ascribed to, in the case of ThirdParty billing.
- *RecoveryConfiguration*: RecoveryConfiguration specifies recovery actions to be taken at the wrap-up phase of the service transaction, if necessary. Options such as *PartialBackup, AllBackup, ChargingCompensation* are specified, and the causes that triggers recovery actions are specified, e.g. *UnsatisfactoryPerformance* (coming from performance management), *LinkFailure* (coming from fault management) etc.

7.2.4 Relevance to Other TINA-C Documents

The overall management principle is described in Management Principles Architecture [19]. The service management aspect of accounting is also described in the respective section of TINA service architecture (SA96 Annex 1, Ch 5 [20]).

7.3 Functional Requirements

The following issues need to be addressed, and solved in accordance with overall principles of TINA NRA.

- *Event-driven accounting management*: as it is with TMN, the accounting management architecture should be event-driven.[1] It is believed that event-driven approach is well suited to the distributed style of management, where asynchronous nature and inherent parallelism in accounting event management can be taken into account. With some exceptions at EML level and NE level objects, which may require polling by the nature of the underlying hardware, we assume that the event-driven approach is generally applicable.
- *Multiple accounting domains*: as it is clear from the TINA business model, TINA is a multi-stakeholder world. Multiple providers being involved in a service session need to exchange accounting information to get the billing right. Although this problem of multiple accounting domains is mostly dealt at the service management level (SA96, Annex 1 [20]), there are some important cases that multiple accounting domains need to cooperate at the resource level, i.e. two domains are federated.
- *Event as a common call record*: since events are to be exchanged at the resource level when two domains are federated, and events are to cross the business model boundaries when on-line billing option is used, accounting events can be understood as a common call record. When events are logged in a central call server, they are in fact equivalent to call records.
- *Technology dependent metrics*: although TINA network resources are meant to be technology independent, some metrics can make sense only within certain technology. Since TINA resource layer encompasses all the TMN layers below NML (Figure 7-1), technology specific metrics need to be addressed as well. Due to their technical importance for TINA applications, ATM specific and SDH specific metrics should be studied.
- *Policy-based accounting control*: a policy is a set of rules to be applied to the objects in a domain. In contrast to a management context, which is only applied to the objects in a service session, a policy can be applied to all the objects in an accounting domain. In other words, a management context is more dynamic and service-oriented whereas a policy is more static and resource-oriented. An accounting management policy can be updated, or switched to a new policy. Two policies may be applied to a group of objects at the same time, when two domains are federated.
- *Support for flexible service management: flexibility*: resource level accounting should provide requisites for flexible service management such as security audit trail, on-line billing, and so on. Flexibility in TINA accounting management can come mostly in two ways, from the accounting management context and

1. In TMN, an MO (seemingly) spontaneously generates a notification when a certain attribute value changes, and the notification may subsequently trigger actions on the receiver (manager) side. In this manner, certain actions are driven and activated by the occurrence of notifications (events).

the accounting management policy of the domain. The accounting management context can be interpreted, or even executable by a mobile-agent like management functional entity, providing a service session specific, a most flexible mechanism. Resource layer objects are much more stable and also slower to change, but certain things can be changed by updating the accounting management policy.

- *Interworking with legacy accounting systems*: TINA has to interoperate with legacy networks at various levels. at service level, primarily with IN and Internet. At resource level, primarily with TMN. Although some interworking scenarios are already worked out by P508, an additional work needs to be done with more focus on accounting management problems.

7.4 Information Viewpoint: Basic Concepts in Accounting Management

In this section, we illustrate basic concepts in accounting management. Most of these concepts have been known, and have also been explained in various parts of TINA-C documents in different contexts. Our objective here is to give them a set of consistent and coherent meanings, particularly in the light of accounting management.

7.4.1 Accounting Management Domain

In the accounting management fragment part of NRIM, The accounting management domain is explained as follows:

> The *accounting management domain* is a type of information object which represents a set of resource objects that are under the purview of an accounting management function and thus are governed by the same accounting management policy. The resource objects have a is-assigned-to relationship[2] with an accounting management domain. An accounting management domain may have a recursive structure, i.e., an accounting management domain may contain other accounting management domains. The assignment of resources to domains should be consistent with the network resource topology, e.g., a domain boundary should coincide with a subnetwork boundary.

In general, a domain is an information object associated with certain management functionality such as accounting, security, or DPE management etc. In a way it is similar to the object grouping concept, as far as they both represent a set of objects. Unlike object grouping however, a domain does not usually provide explicit membership operations, since domain boundaries are usually based upon natural affinities between objects, e.g., the network resource topology, business stakeholder, geographical area, etc.

2. In the following part of this section, this “is-assigned-to” relationship is taken as equivalent to “consists-of”. In the accounting management domain information model (Figure 7-3), an association is replaced by an aggregation, accordingly.

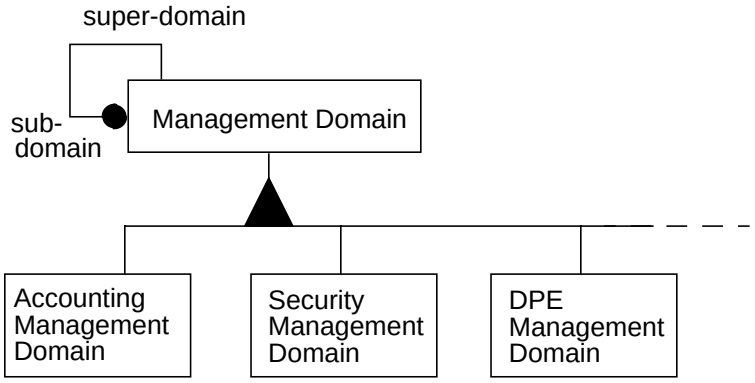


Figure 7-3. Management Domain Information Model

Figure 7-3 illustrates the management domain information model. By nature, the domain concept allows a hierarchical composition of domains. The generic management domain can be specialized with a set of management functions (policies) to become a specific management domain, e.g., an accounting management domain. Domains are by no means disjoint. Not only that an object may belong to two or three different management domains, these domains may also relate to each other in different ways. For example, two accounting management domains may belong to two respectively different security management domains, but they both may belong to a single DPE management domain.

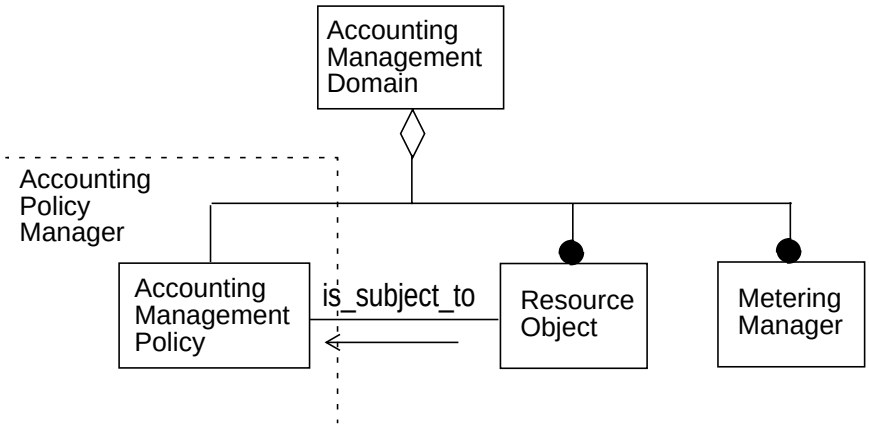


Figure 7-4. Accounting Management Domain Information Model[3]

Figure 7-4 illustrates the accounting management domain information model. The figure shows that an accounting management domain consists of an accounting management policy, resource objects, and possibly many metering managers. The accounting policy itself is maintained by an accounting policy manager, from the computational viewpoint.

7.4.2 Accounting Management Policy

The accounting management policy is a set of common accounting rules, to which all the resource objects under the given accounting domain would follow. In the resource level accounting, the policy primarily deals with the metering and the event management. The accounting management policy should contain the following information.

- *Metering manager*: the metering manager is in charge of controlling the logging of accounting events and management of the logging records. An accounting management domain needs at least one metering manager, or possibly a few of them.
- *Event management*: the event management dictates acceptable event management options, e. g., translation, forwarding, duplication, etc., in the domain. It is possible that a certain accounting management domain may prohibit some options from security or performance concerns, even though the underlying DPE supports them. Availability of event management options may limit interoperability and federation between different accounting management domains.
- Fault tolerant features: accounting events are usually logged, and can be logged in two separate locations to eliminate a single point of failure. Since accounting records can be utilized for various fault recovery actions, ranging from fault recovery to charge compensation, they deserve special protection separately from other network resources. The fault tolerant features of the accounting management policy dictates the degree and extent of the protection of accounting events against faults.
- *Policy applicability rules*: the policy is a set of rules. As such, their applicability may vary depending on how the accounting management domain is being operated. Each rule is associated with one of three different applicability values; *mandatory (hard)*, *optional (soft)*, or *negotiable*. These applicability values are also used to resolve conflicts when two accounting management domains overlap, as it may be the case with federated domains.

There are additional requirements. In particular, security requirements are the most relevant and the most important.

- *Security features*: since accounting events are the base of billing calculation, they can become targets of attacks, aiming at toll fraud or denial of service. Security features have not always been required in legacy accounting management systems. In fact many of legacy accounting management systems do not have necessary security provisions. In TINA, however, since the entire re-

3. Accounting management context is not shown in this figure, since the management context is not an intrinsic part of the management domain. Please see Section 7.6.8.

source layer is to be built upon an open and distributed platform, security features are almost mandatory. Accounting events can be made non-reputable by using one of security options. Non-reputable events can help resolving toll disputes, in particular when a third-party is involved in the accounting process (e. g. federation of accounting domains).

7.4.3 Accounting Metrics

The accounting metrics are the quantities to be measured in the accounting. They can be physical, electrical, computational or informational, depending on the nature of the measurement. In the NRA, we are naturally most interested in the measurement of network resources. Some metrics are technology independent (creation, deletion, bit error rate etc.), others are technology dependent (peak cell rate, cell loss rate etc.). Though the TINA connection management architecture is meant to be mostly technology independent, some technology dependent metrics need to be addressed, in particular ATM, due to its significance for TINA applications. In general, an accounting data has the following three fields.

- Metric type: peak cell rate, cell loss rate, etc.
- Metric unit: number of cells, frames, bit/s, etc.
- Measurement time: time of the measurement in a standard notation, or period of the measurement, etc.

.

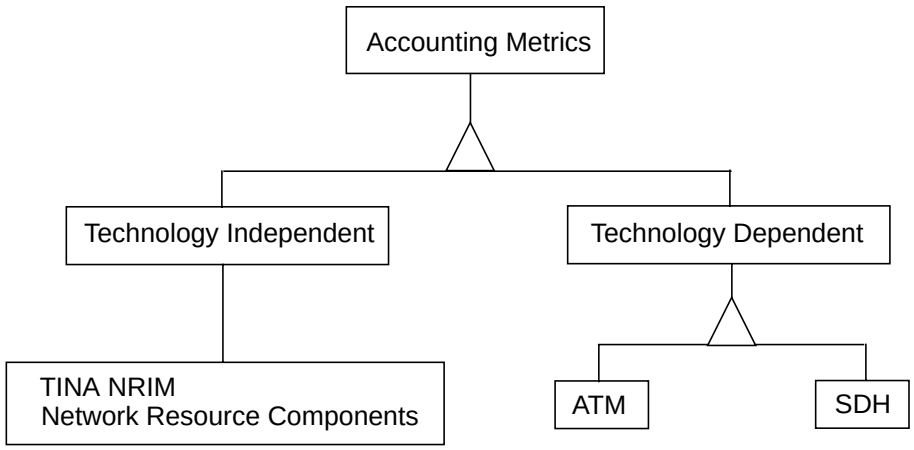


Figure 7-5. Accounting Metrics Information Model

Figure 7-5 illustrates the accounting metrics information model. Accounting metrics are categorized into two groups.

- *Technology independent*: TINA network resource components defined in NRIM offers technology independent view of network connectivity. As such, only technology independent metrics such as *bit_per_sec*, *byte_per_sec* are offered by these components. Technology independent metrics correspond to the connectivity network layer in the TINA network hierarchy (Section 3.1).
- *Technology dependent*: technology independent metrics are actually derived from, or accounted for, by the underlying technology specific metrics. It is in much the same way the TINA connectivity layer is a virtual network, which consists of technology specific layer networks (Section 3.1).

Figure 7-6 illustrates the accounting data information model.

.

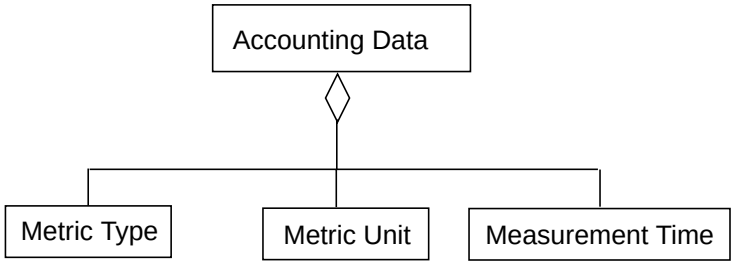


Figure 7-6. Accounting Data Information Model

For more details of accounting metrics of specific technology, examples are given in Appendix A (Section 7.9).

7.4.4 Accountable Object

The accountable object is a type of resource object, which can be subject to accounting management. The accountable object provides an accounting management interface, which is in essence equivalent to X.742 AccountableObject [42]. The accountable object is capable of generating accounting events. The granularity and the verbosity of accounting events depend on the nature of the resource which the accountable object represents, but they may be controllable through the accounting management interface.

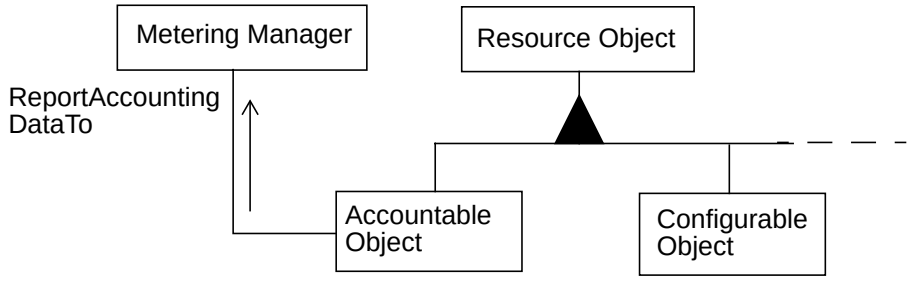


Figure 7-7. Accountable Object Information Model

Figure 7-7 illustrates the accountable object information model. An accountable can be also a configurable object, and can be another type of manageable object at the same time, as many as it can provide requisite management interfaces. The accountable object can have a ReportAccountingDataTo relationship with a metering manager, or possibly with many of them. The accountable object notifies accounting events to the related metering managers.

7.4.5 Accounting Event

The accounting event is an event with an accounting data. An accounting event may be encrypted, message-integrity-check (MIC)'ed, or non-repudiated, following the security policy of the domain. An accounting event usually carry one accounting data, but multiple of accounting data may be piggybacked in one accounting event, for the sake of efficiency. The accountable event is produced at an accountable object, and may be carried by DPE event management mechanisms such as CORBA event service or DPE notification service, and the event is consumed by an metering manager or a by notification interface of another accountable object.

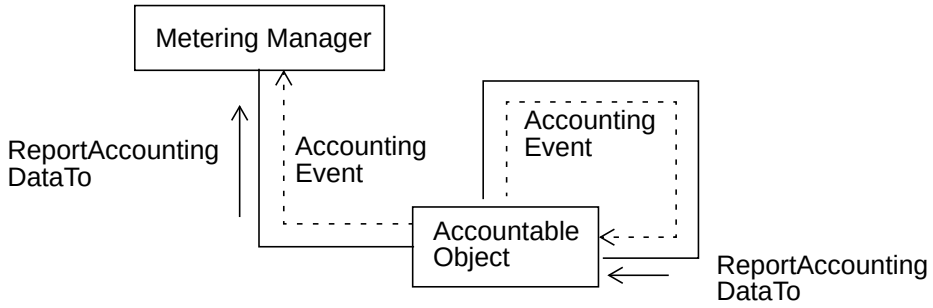


Figure 7-8. Accounting Event Information Model

Figure 7-8 illustrates the accounting event information model. Each accountable object reports accounting events either to a metering manager which is responsible for the domain, or to another accountable object in the event management ladder, which the accountable object belongs to. For the details of the event management ladder, see the corresponding part of Section 7.7.3.

7.5 Computational Viewpoint: Accounting Management Components

The accounting management components are the key players in the accounting management. In this section, we describe more of their computational aspects without using the ODL.[4] The following descriptions should be understood as the English version of the components specifications.

7.5.1 Accountable Object

- Description: the accountable object is a TINA resource object with an accounting management interface. In other words, the resource object must support *i_AccountableObject* interface in its ODL specification, to be accountable.
- Behavior: the metering of the accountable object can be controlled through the management interface, *i_AccountableObject*.
- Operations: the following operations should be supported.
  - Control operations:
    start
    stop
    suspend
    resume
    set_state
  - Notification control operations:
    suspend_notification
    resume_notification
    set_verbosity_level
    set_Notification_Destination
    reset_Notification_Destination

7.5.2 Accounting Policy Manager

- Description: the accounting policy manager maintains the accounting policy. For example, when a new resource object is created in the domain, the object obtains the accounting policy from the accounting policy manager. The accounting policy manager provides *i_AccountingPolicyManager* interface.
- Behavior: the accounting policy manager serves as a policy server, from which the objects in the domain can obtain the policy. When a new policy is installed or an old policy is updated, the policy needs to be notified to all the accountable objects in the domain, with the help of the network resource management of the domain.[5]
- Operations:
  start
  stop

4. ODL/IDL specifications are to be provided as a part of the network resource components specification.
5. The network resource manager (to be studied) must know all the management interfaces of the resource objects in the domain, particularly the accounting management interfaces.

suspend
resume
update_policy
delete_policy
propagate_policy

Besides the above control interfaces, the accounting policy manager provides a management interface to maintain the policies (see Section 7.6.9).

7.5.3 Metering Manager

The metering manager is responsible for the metering of the accountable objects in the domain. Although the metering manager may take the full responsibility of the accounting process in the entire accounting domain, e. g., starting or stopping accounting of all the accountable objects, the role of the metering manager in NRA is limited to event filtering and logging. There are two reasons for this decision at this point of writing:

- Need for integrated resource management: the need for an integrated resource management based on a solid naming and addressing principle is still waited. For example, control aspects of accounting, security audit trail, and configuration management are similar, and they should be based on a common architecture.
- Due to the recursive structure in the TINA connection management and the accounting management domain, it is often more convenient to view an accountable object represent management and control aspects of accounting in a subdomain. For example, a *Subnetwork* accountable object actually represents accounting activities related to the communication service in the accounting subdomain.[6]

Here are the specification of the metering manager.

- Description: the metering manager is responsible for event filtering and logging of accounting events in the accounting domain.
- Behavior: the metering manager provides a notification interface *i_MeteringManager*, to which accounting events are to be sent. It performs logging and filtering. The metering manager may also perform pre-processing or classification, for the sake of billing efficiency.
- Operations:
  - Control operation:
    start
    stop
    suspend
    resume
    set_Log

6. For more details of this concept, please see Section 7.6 and Section 7.7, in particular Section 7.6.6.

reset_Log
set_Filtering
reset_Filtering

- Notification operation:
  event_notify

7.5.4 Log Manager

The log manager is equivalent to the *log* managed object of X.721 [43]. The log manager does not play a major role in the generic TINA accounting management, but it exists for the purpose of easing the interworking with TMN based accounting management system.

- Description: the log manager performs logging.
- Behavior: the same as X.721 log managed object.
- Operations:
  - Control operation:
    start
    stop
    suspend
    resume
    set_log_attributes
    reset_log_attributes
  - Notification operation:
    event_notify

7.5.5 Event Forwarding Discriminator

The event forwarding discriminator (EFD) is equivalent to the *eventForwardingDiscriminator* managed object of X.721 [43]. The EFD does not play a major role in the generic TINA accounting management, but it exists for the purpose of easing the interworking with TMN based accounting management system.

- Description: the EFD performs event forwarding discrimination. It may also perform filtering as part of the forwarding process.
- Behavior: the same as X.721 eventForwardingDiscriminator managed object.
- Operations:
  - Control operation:
    start
    stop
    suspend
    resume
    set_EFD_attributes
    reset_EFD_attributes
  - Notification control:
    set_Notification_Destination

- Notification operation:
  event_notify

7.6 the Management of the Accounting Management Domain

It is clear that the accounting management domain itself needs to be managed. To study the management of the accounting management domain, two sides of the coin, i. e., its structure and semantics need to be understood. In short, the structural aspects of the accounting management domain such as the recursive composition is inherited from the generic management domain concept (Figure 7-3), whereas the semantic aspects of it are given by the associated accounting management policy. In the first half of the following part, the structural aspects are explained. In the latter half, the semantic aspects, i. e., the management of the accounting management policy, are explained. Lastly, we illustrate how the federation between accounting management domains can be treated using these concepts.

7.6.1 Operations on Accounting Management Domain

As an information object, an accounting management domain has its own life cycle. A new accounting management domain may be created by the following operations:

- *Create*: *create* operation creates a new domain.
- *Divide*: *divide* operation divides a domain into two new domains. As a result, a new smaller domain is created.

An accounting management domain may be deleted by the following operations:

- *Delete*: *delete* operation deletes a domain.
- *Merge*: *merge* operation merges two domains into a new domain, and at least on of the two original domains is normally deleted.

There are other operations which do not create nor delete domains, but change they way the domains relate to each other.

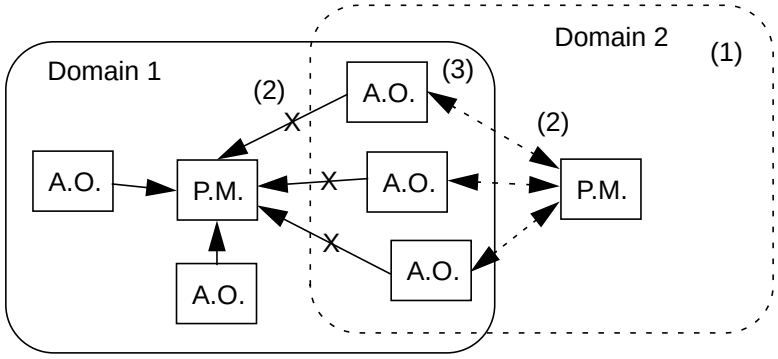
- *Overlay*: two or more domains are overlaid. The original domains do not change, but the way accounting management policies are applied to the objects within can be changed.
- *Mount*: an accounting management domain is mounted on an accountable object, in such a way that accounting management is acted upon the accounting domain as if it is acted upon the accountable object, and the accounting events from the accounting domain come out as if they are from the accountable object. In short, an accounting sub-domain is represented by an accountable object in the parent domain.

7.6.2 Create and Delete

Create and delete are operations themselves, but they may be used as a step within other operations such as divide and merge. In any case, creation of a new accounting domain is performed in the following three steps:

1. The network resource manager selects a set of objects in the original domain, and creates a new group.[7]

2. The objects in the new domain are associated with the new policy manager, and are then dissociated from the original policy manager.
3. The network resource manager triggers the new policy be propagated to the objects in the new domain from the new policy manager.



A.O.: Accountable Object

P.M.: Policy Manager

Figure 7-9. Creation of Accounting Management Domain

Figure 7-9 illustrates the creation of a new accounting domain, Domain 2, from Domain 1. Although the new domain exists at the end of Step 1, the accountable objects in the domain do not realize its existence until the end of Step 3, when the new accounting management policy is propagated. At this point, it is rightly asked, whether the accounting process in the domain 1 needs to be disrupted due to the creation of Domain 2. The answer may vary, depending on the engineering options available for the domain management. In general, however, the followings are expected:

- *Domain invariance*: for the accountable objects staying with Domain 1, their accounting process should not be disrupted by, or not even affected by the creation of Domain 2. Accounting process in the domains other than Domain 2 should stay invariant regardless of the creation or deletion of Domain 2.
- *Non-interruption of service (session)*: not all the policy changes cause interruption of service, or more accurately, a session associated with the accounting process. In fact, most of them would not cause any interruption on the session. For example, adding another metering manager for the sake of additional fault tolerance would not affect any running accounting process. On the other hand,

7. It is likely that the object grouping is to be used by the network resource manager at this stage of domain creation. Details however still awaits further study.

a policy change may bring about a conflict between the policy and the management context of the running accounting process. For example, a policy change may prohibit on-line billing in the domain, whereas some of the accounting processes may be running with on-line billing, in accordance with the previous policy before it was changed.

To address the second issue, a few engineering options are available, and an administrative decision would dictate which option is to be taken in a particular case.

- *Delayed action*: application of the new policy to the already running accounting process is delayed, until the running session naturally terminates. In other words, the users started with on-line billing is allowed to receive the same accounting service until they finish it. This is the most lenient policy, but it may not always desirable from the operational point of view.
- *Policy migration*: accounting process may be adjusted to the new policy, without interrupting the running accounting process. Adding a metering manager of switching between metering managers may be such an example. This migration, however, can not be expected to be possible for all the policy rules.
- *Session termination*: the most drastic, but the sure-to-work measure. The session (actually its accounting process) is terminated upon the detection of conflicts between the management policy and the running accounting process. This measure may be necessary as well as justified, only when the conflicts are expected to lead the network into a serious situation. Otherwise the user is most likely to be very much dissatisfied.

Delete operation proceeds in rougly the same steps as creation operation, except that the order of the steps are reversed. Using Figure 7-9, deletion of an accounting domain, Domain 2, is performed in the following three steps.

1. The accountable objects in Domain 2 are dissociated from the policy manager in Domain 2. The accountable objects are then associated with the Domain 1 policy manager.
2. The resource manager resolves the group of objects corresponding to Domain 2, and then merge them with those corresponding to Domain 1.
3. The accounting management policy of the Domain 1 policy manager is propagated to the new members of Domain 1.

7.6.3 Divide

Divide operation is similar to creation, except that the new domain is separated from the original domain, and the size of the original domain is reduced. At the end of divide operation, there are two smaller accounting management domains.

7.6.4 Merge

Merge operation is similar to deletion, except that the resultant domain consumes the original two domains. The resultant domain may have the same policy manager as one of the original ones. In this case, it is more correct to say that the surviving domain absorbed the other one. In fact, the deletion example in Section 7.6.2 shows such a case, where Domain 1 absorbs Domain 2.

7.6.5 Overlay

A new domain can be created in such a way that its members have a joint membership with other domains. In other words, the superimpose operation results in a situation where two or more number of domains co-exist, overlapping each other in their members.

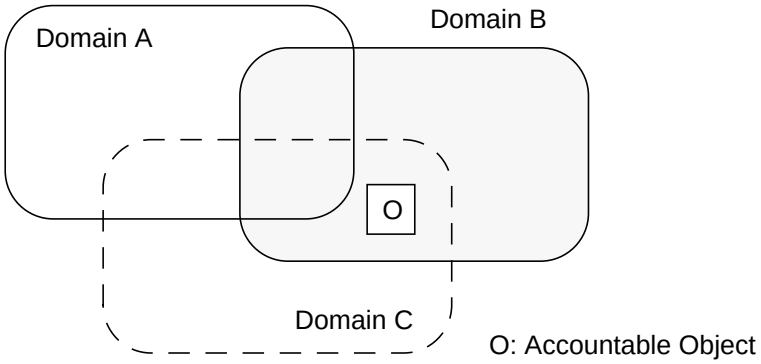


Figure 7-10. Overlay of Accounting Management Domains

Figure 7-10 illustrates the overlay of accounting management domains. In the figure, an accountable object O belongs both to Domain B and Domain C, thus the object O is subject to two different accounting management policies. In other words, two management policies are superposed, and being applied to the overlaid area of domains. There are three possible cases:

- *No conflict*: two policies do not conflict. Thus the object O is able to follow two policies at the same time.
- *Solvable conflict*: two policies do conflict, but the difference can be resolved by adjusting the negotiable part of the policies, by negotiating with the policy managers.
- *Unsolvable conflict*: two policies do conflict, an they can not resolved by negotiation. If two rules in the mandatory part of the two policies conflict, they are clearly unsolvable.

In the first two cases the two policies are called solvable; in the last case they are called unsolvable. It is clear that two domains can be overlaid, if and only if the two management policies associated with the domains are solvable. For more details of the resolution process, please see Section 7.6.7.

A specially important case of the overlay is the domain hierarchy, i. e., domains are contained by one another in a hierarchical manner.

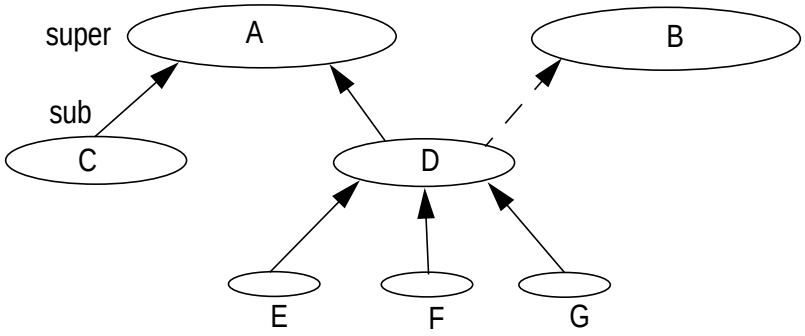


Figure 7-11. Management Domain Hierarchy

Figure 7-11 illustrates the management domain hierarchy. In a purely hierarchical system, each domain can be given a unique path name from the root, e. g., /A/D/E or /A/C. Since domains are contained, and thus overlaid, the accounting management policy being applied to an domain is the superposition of the management policies of its super domains and of its own. For example, the management policy to be applied to domain e is the superposition of the policies of domains A, D, and E, which are the domains on its path name /A/D/E up to the root.

Suppose if domain D becomes a sub-domain of B, the entire system is no longer purely hierarchical, though it still partially is. The principle of superposition still applies, that is that the management policy of domain B is being applied to domain E, in addition to the policies of A, D, and E itself.[8]

8. However, there is a possibility that policy B and policy E are unsolvable, thus B and E may not be overlaid, although policy B and policy D are still solvable. The purpose of this section is mainly to explain the concept, not to discuss the applicability of the concept in reality.

7.6.6 Mount

An accounting domain can be represented by an accountable object, in such a way that the accounting activities related a communication service would look as if they are coming from the accountable object. For example, a Subnetwork accountable object in the TINA network resource architecture actually represents related accounting activities in the subnetwork, which is usually an independent and separate accounting domain.

This relationship between the accounting management domain and the accountable object is similar to mount operation in UNIX file system. In analogy with the hierarchical file system, an accountable object representing an accounting management domain is called a mount point, and the accounting management domain is called being mounted on the accountable object.

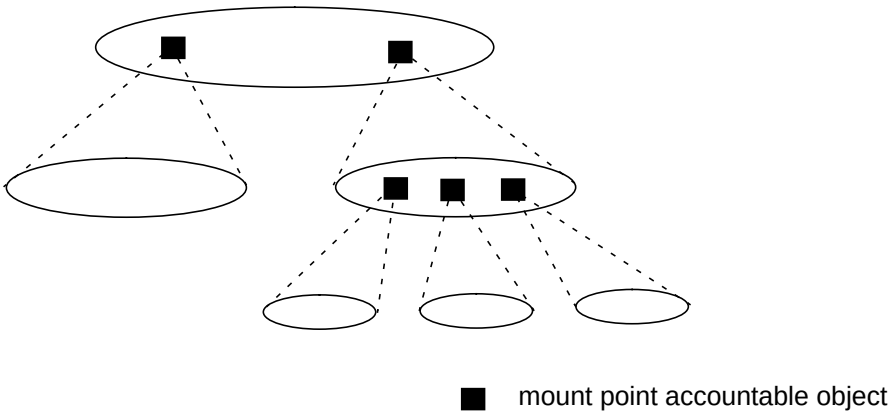


Figure 7-12. Mounting of Accounting Management Domains

Figure 7-12 illustrates mounting of accounting management domains. The entire system looks hierarchical, but domains are disjoint, not overlapping. Unlike containment hierarchy of domains (Figure 7-11), domains are not overlaid, thus the management policies are independent and separate, not interfering with each other. As such, even though the domain policies are unsolvable, they can still be mounted.

Mount operation is more convenient than overlaid containment hierarchy, since there is no need for the policy resolution process. Mount point is also created just for the duration of the service, or more precisely the service session, which carries the accounting management context (and other service management contexts are well) to the server domain, i. e. the mounted domain.

In fact, almost all the cases appear in the TINA connection management are the mounted domain. Other than a few cases which may appear in the federation, the overlaid hierarchy has been rarely used.[9]

7.6.7 Resolution of Accounting Management Policy

When two policies do have conflicts, they need to be resolved. The conflicts can be solvable or unsolvable, depending on the applicability of conflicting rules and the negotiation capability of the concerned policy managers. To illustrate the point, we give a few examples.

- Two policies conflict each other in their mandatory rules. There is no possibility of resolution, therefore two rules are unsolvable.
- Two policies conflict each other in their negotiable rules. Both of the policy servers are capable of negotiation, and the conflicts are eventually resolved.
- Two policies conflict each other in their negotiable rules. Either of the policy servers however is not capable of negotiation, and the policies remain unsolvable.
- Two policies conflict in their optional rules. By definition of the optional rules, either party can resolve the conflicts at will.

Table 7-1 shows the resolution scheme for the conflicting policies.

Table 7-1. Conflict Resolution Scheme of Policy Rules

	Mandatory	*Negotiable*	*Optional*
Mandatory	Unsolvable	Negotiation	Solvable
Negotiable	Negotiation	Negotiation	Solvable
Optional	Solvable	Solvable	Solvable

The row and the column shows the applicability values of the two conflicting policy rules. Apparently, the most interesting case is the resolution by negotiation.

The negotiation in general is dependent on the semantics of the policy rules. It is possible, however, for the policy manager to provide a highly semantics-independent negotiation interface, as it was designed for the general context configuration manager [23]. In particular, when the rules are represented by a fixed set of attribute-value pairs, the negotiation process are described as follows. Let us call the set of negotiable rules as the negotiable set.

1. The user presents the negotiable set to the provider, setting the values of the attributes as the user wishes.

9. It is for this particular reason that the overlaid hierarchy is studied in this part of NRA, since the issues in the federation can not be fully resolved without understanding the meaning of the management domain overlay.

2. The provider receives the negotiable set. If the values are acceptable, the provider confirms the set and the negotiation ends successfully. Otherwise the provider changes the unacceptable values of the attributes to the values the provider wishes, and send the modified negotiable set back to the user.
3. The steps 1. and 2. repeats until one of the following three things happens:
   - Two parties agree on the negotiable set. The negotiation successfully ends at step 2.
   - Two parties do not agree on the negotiable set. The repeat count of the steps 1. and 2. exceeds a pre-defined limit. The negotiation fails.
   - Two parties do not agree on the negotiable set. The negotiable set repeats itself to one of the previously negotiated, unsuccessful one. Unless the two negotiators are highly intelligent such that they can change their negotiation strategy for the apparent stalemate, whole the negotiation process will repeat itself. The negotiation fails.

In any case, the resolution of conflicting policies and the solvability of two accounting management policies can be performed in a practical time-frame.

7.6.8 Scoping Issues between Management Policy and Management Context

The scope of an accounting management is a set of objects, which the policy applies. In the usual case, it is nothing but the domain that the policy is associated with. When two accounting management domains are overlaid, the two policies must be solvable. The respective scopes of the two policies have an overlap, which is the overlaid area.

The accounting management context is similar to the accounting management policy, in the sense that they both dictate accounting or billing should be done over a set of accountable objects. There are however the following differences.

- *Flexibility*: the management context is associated with a service session. To be more precise, the context is associated with a service transaction, which performs a portion of the service session in the domain. On the other hand, the management policy is associated with the domain, which is not specific to any particular session or service. It follows that the management context can be more service specific and flexible than the management context.
- *Lifetime*: since the lifetime of a service session is normally shorter than the lifetime of a domain, it follows that the lifetime of the scope a management context is normally shorter than that of a management policy.[10]

Figure 7-13 illustrates the scoping between the context and the domain. The accountable objects in the scope of the context is only those objects involved in the service transaction, which the context is associated with. The scope of the context is a subset of the scope of the policy.

10. The lifetime of the scope a management context is meant to be the lifetime of the context within a particular service session. The context itself may have a longer lifetime than the service session, since the same management context can be re-used for many different service sessions.

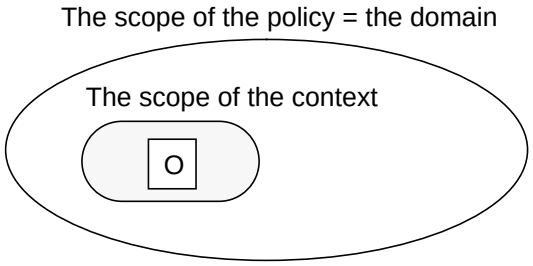


Figure 7-13. Scoping between the Context and the Domain

In the area where the two scopes overlap, both the context and the policy are being applied to the objects. It is clear that the context and the policy must be solvable, such that the objects in the dotted area of overlapping scopes do not fall into conflicting behaviors. In other words, a service transaction can not start in the domain, unless its context and the domain policy are solvable.

The management context and the management policy do not usually conflict, since the former mostly represents the service management requirements whereas the latter mostly represents the resource management requirements. They may, however, occasionally conflict, particularly when the domain explicitly prohibits certain service level accounting, e. g. On-line billing.

7.6.9 Maintenance of Accounting Management Policy

The accounting management policy needs to be maintained, in such a way that addition of new rules, modification or deletion of existing rules would not fall into an inconsistent policy. The accounting policy manager provides an management interface, which is responsible for maintaining the consistency of the policy rules. It will provide the following operations:

- *List_all_rules*: the list_all_rules operation is more of a query operation rather than a maintenance operation. This operation lists all the current rules in the policy.
- *Add*: the add operation adds a new rule to the policy. The rule is added to the current set, but it will not be in effect unless the new rule is committed.
- *Delete*: the delete operation deletes an existing rule from the current set. The deletion does not become effective unless the change is committed.
- *Modify*: the modify operation modifies an existing rule with a new value, new parameters, etc. The modification does not become effective unless the change is committed.
- *Check*: the check operation checks the consistency of the new (added, deleted, or modified) rule set. It also checks the solvability with the overlaid policies.

- *Commit*: the commit operation reflects the changes to the new rule set to the existing one, such that the new rule set becomes effective.

7.6.10 Federation in Accounting Management

The federation has a simple meaning from the management point of view. If two management domains are federated, they have an overlay, and objects in the respective domains are allowed to operate in the objects in the overlaid area, in accordance with the pre-defined authorization list of operations and the roles they are assigned in the scenario. In a very rough analogy with the connection management, the federation can be considered as an extension and generalization of the third-party control.

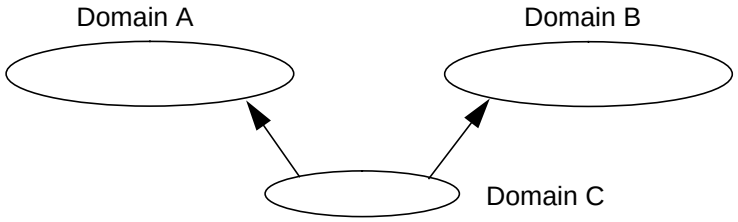


Figure 7-14. An Example of Federated Domains

Figure 7-14 illustrates an example of federated domains. In the figure, Domain C is a sub-domain of both domain A and domain B, therefore domain A and domain B is overlaid on domain C. By definition of the overlay, the management policies of Domain A and Domain B both applies to Domain C, though some of the optional rules in the respective policies might have been turned off. The level of federation can vary, depending on the business relationship between the respective stakeholders administering the domains A and B. The followings are some sample scenarios. To be concise, we will denote the stakeholder of the respective domains simply as A and B.

- *Scenario 1*: the business relationship between A and B are very close, and the domains are federated semi-permanently. Most of the management interfaces up from the NEL in domain C are exposed both to A and B, and both controls and manages resource objects in domain C as well. Operations from A and B may be performed in parallel in domain C, however only with the condition that the two parallel transactions do not fall into a deadlock or in a inconsistent state - a quite familiar situation well-known in the data-base management. The accounting management can be a separate issue, however it is most natural to imagine that the accounting management is federated as well in the federated domains. For example, accounting events up from the NEL are reported to the metering manager of A, or to that of B, or possibly to the both.

- *Scenario 2*: the business relationship between A and B is less close, and the domains are federated by a contract basis. For example, A may want to rent a part of its network (=domain C) to B, earning a revenue while relieving itself of certain network management tasks. From B's perspective, B may be an independent network management service company, whose task is delegated from A, to manage domain C. B may be allowed to access some management interfaces, e. g. the NEL and the EML, and up to the NML, whereas A still maintains management interfaces at the SML. In the accounting management, B may be responsible for the overall resource level accounting, whereas A is responsible for the service level accounting and billing.
- *Scenario 3*: the business relationship between A and B is temporary, and the federation is maintained only for the duration of a particular service. For example, B may be only responsible for the billing only, and the rest of the resource management is taken care of by A. B may be allowed to access very limited group of management interfaces, which enables B to perform the billing management.

The following observations can be made from the above analysis of scenarios.

- *Management scenario*: the degree of federation and what interfaces to be exposed to the federated partner depends on the management scenario and the business relationship between the two partners. The idea is somewhat similar to the intra-domain reference point, in the sense that they both specify a set of interfaces based on the business roles of the two parties. The reference point, however do not usually focus on management issues, nor it does not specify a fine-grain access control (authorization control), which seems necessary to deal with the federation between the management domains.
- *Role-based authorization*: the management scenario identifies the roles of the partners to be played within the federation. The role, in turn determines the authorization necessary to perform the scenario, i. e. a set of management interfaces and permitted operations upon them. This type of role-based authorization has been studied [Sloman], and it still awaits to be developed as a part of TINA security management architecture. The security framework section of NRA is to address the role-based authorization in view of the resource management.
- *Object grouping*: from the computational viewpoint, it seems desirable if a set of interfaces are grouped together, and the combined set is added an authorization control interface, which enables the role-based authorization. The group need to be dynamic, not always static, since a different management scenario would lead to a different management role, which may require a different combination of interfaces. TINA DPE already provides a suitable mechanism for this purpose, which is the object grouping. Although DPE object allows multiple interfaces, the supported interface set is fixed and static. The object group, on the other hand, allows multiple objects to be grouped dynamically through the control of the group manager. The object grouping, therefore, is an essential mechanism for the federation between management domains.

7.6.11 Accounting Management Federation Example

Since a different management scenario can lead to a different form of federation, there are virtually infinite variations of accounting management federation. The purpose of this section is to illustrate the overall picture of the accounting management federation through an example. In the following, we use the same example of Figure 7-14. We assume that domain C belongs to domain A, but is not a sub-domain of domain B at the beginning. We also assume that each domain is operated through a respective domain resource management system.[11] The accounting management federation proceeds as follows:

1. The resource management system of domain A (the domain A manager) provides a federation interface, which allows to be accessed by the manager B.
2. The manager B requests a federation of domain C to the manager A through the federation interface. If the condition is met, the manager A requests if domain C can be federated, requesting the respective policy managers (the policy manager of domain C and domain A) to check if domain C can be overlaid with domain B, to become its sub-domain. The policy managers check the solvability of the policies, possibly involved in negotiation with the domain B policy manager. If the two policies are solvable, the federation is possible, and the policy manager returns a positive answer to the requesting party (the manager B).
3. The newly federated policy of domain C is imposed on the resource objects in the domain.
4. The role-based authorization is given to the manager B, w. r. t. the objects in domain C.

11. A TINA resource management system, which is to be explained in Management Principles Architecture.

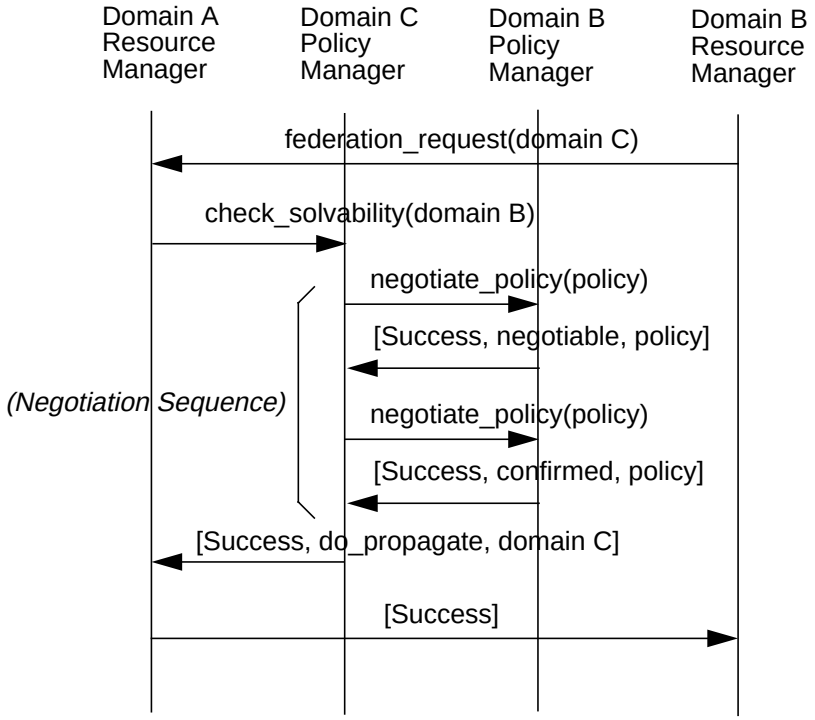


Figure 7-15. Federation between Management Domains

Figure 7-15 illustrates a high-level event sequence of the federation between two accounting management domains. At the end of negotiation sequence, the federated policy is propagated to domain C, by the resource manager of domain A. At the end of the federation sequence, the resource manager of domain B is entitled a role-based authorization, which may be delegated to service transactions originates in domain B.

The benefits of the federation are several. Some of the most notable benefits may be that the federation allows the domain managers to share the management resources and the management responsibility as they are demanded by service or management scenarios. In the following part of this section, we give a few examples of the federation.

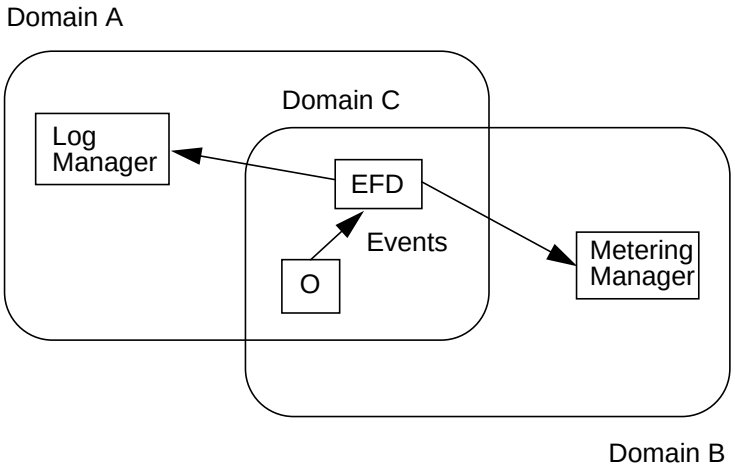


Figure 7-16. Sharing of Management Responsibility

Figure 7-16 illustrates the basic concept of the sharing of management responsibility. In the figure, domain C is a federated management domain between domain A and domain B. It is assumed that domain A is predominantly TMN-based, whereas domain B is predominantly TINA NRA based. An accountable object O generates accounting events, which are handled by EFD, to be re-forwarded to the log manager in domain A and the metering manager in domain B. Accounting events are recorded in X.721 compliant forms in the domain A log manager where as the events are recorded in TINA NRCM compliant forms in the domain B metering manager. This example also shows an interworking between TMN-based and TINA NRA-based accounting management systems. Although the resources (logging facility) are duplicated in domains A and B, the management responsibility is shared between the two managers.

Suppose if the domain A log manager fails. The accounting management of domain C can still be performed by using the domain B metering manager, offering a basic-level of fault tolerance.

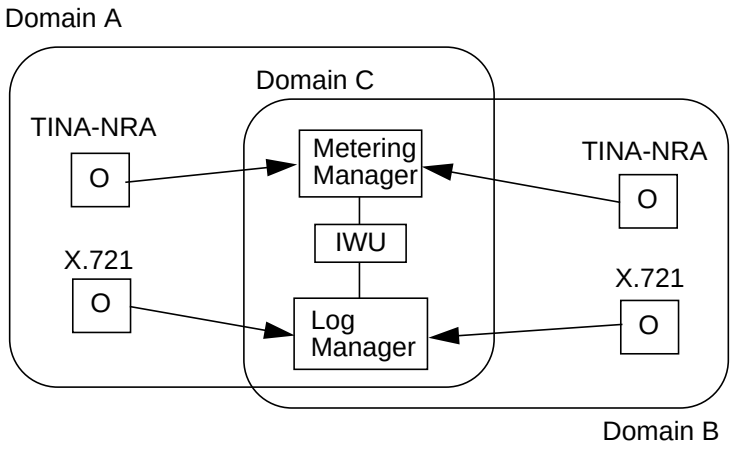


Figure 7-17. Sharing of Management Resources

Figure 7-17 illustrates the basic concept of the sharing of management resources. Domain C is again a federated domain between domain A and B. It is assumed that both domain A and domain B contains X.721 compliant and TINA-NRA compliant accountable objects. In this case, domain C contains the metering manager and the log manager interworking through an IWU, providing an interworking facility for all the domains. Management resources (metering manager, log manager, IWU) are shared between the federated domains, contributing to the reduction of operational cost.

7.7 Accounting Event Management

Once an accounting event is generated at an accountable object, it needs to be delivered to a right recipient in a right format. Although the event management is a common issue to many of the resource level management issues in NRA, the accounting event management probably offers one of the most notable case, since most of the resource components in NRA are considered accountable, thus generating accounting events. The accounting event management is up to the following challenges.

- *DPE event management service*: we assume that the DPE event management services such as the notification service would guarantee the delivery of accounting events.
- *Interworking Issues*: TINA NRA accounting event management should be able to interwork with X.734 [44] compliant TMN accounting event management. This requirement is a part of the interworking and the migration issues in the large, but adding this requirement at the event management level would allow flexibility and versatility in the migration scenario.
- *Service-management-scope preserving accounting event management*: as it can be seen in Figure 7-13, a service management context sets up a scope of objects for the duration of the service transaction which the context is associated with. This scope of object, which we call service management scope, designates a set of objects which share a common service management interests. It is therefore more natural that accounting events are collected within the scope, such that the event correlation and the diagnosis can be made more efficient, since the events are expected to be semantically more relevant.
- *Federation*: the federation in the accounting event management is a part of the federation issues in the large. This level of federation, however can be done without assuming a complete federation of the accounting management domains. For example, when an accounting management domain is mounted on an accountable object (Section 7.6.6), the two domains are not really federated, in the sense that respective management policies of the domains remain independent. It is however possible that the two accounting event management systems are federated, in such a way that the relevant accounting events can cross the management domain boundary without interruption.

7.7.1 DPE Event Management Facilities

The underlying event management mechanisms is to be provided by DPE. DPE event management facilities will be based on the following specifications.

- CORBA COS Event Service
- X/Open Notification Service
- DPE Notification Service
- CORBA Notification Service

Details of the respective specifications will be provided in due time.

7.7.2 TMN-style Event Management Facilities

TMN provides event management facilities as a set of managed object classes.

- *eventForwardingDiscriminator*: the eventForwardingDiscriminator (EFD) is defined in X.734/ISO 10164-5 [44]. The EFD discriminates events, following the filtering conditions being given, in such a way that the EFD decides which events are to be forwarded to a particular destination as real events, and which are not to be forwarded (and eventually discarded).
- *log-control function*: the log-control function is defined in X.735/ISO 10164-6 [45]. The log-control function enables the user of the log managed object to control its logging activities and to update its logging records.
- *log*: the log managed object is defined in x.721/ISO 10165-2 [43]. In simple words, the log managed object is a container of logging records with a management interface.

Although EFD supports filtering as an option, it is not widely used in practice, due to its implementation difficulty and of its limited usage.[12] TINA accounting management system supports TMN-style event management by way of the following two computational objects.

- *EFD*: the EFD is mostly the same as the eventForwardingDiscriminator, except that the EFD does not support extensive filtering options of X.734 counterpart.
- *Log Manager*. the log manager is the same as X.721 log, except that its operational interface is given in ODL, not in GDMO.[13]

In either case, TINA NRA components only support minimum necessary functionality of their TMN counterparts, reducing the optional supports. There are two purposes in introducing TMN-style event management facilities.

- *Fine-grain interworking/migration*: the details of the migration issue are to be discussed in a next release, but in general the migration can be characterized from a coarse-grain to a fine-grain, or from a loosely-coupled to a tightly coupled [LaPierre]. In the fine-grain interworking, two accounting systems (TINA and TMN) are mixed, while two systems are interworking with each other using these minimum TMN-compliant components.
- *Service-management-scope independent accounting*: TMN-style event management is service-management-scope independent. In other words, an EFD or a Log Manager is set-up for a set of resource objects in a particular domain, not for a particular service session. This operational assumption is more correct for the resource-oriented activities such as fault report or diagnosis, but it is less correct for the service-oriented activities such as billing and its underlying accounting activities.

12. For example, parameters given to filtering are not expressive enough to be used for event correlation.
13. It is possible to see X.721 log as an information model of TINA NRA Log Manager.

7.7.3 Event Management Ladder

Since some of the resource level accounting activities are triggered by a service transaction, and are sustained only for the duration of the service transaction, it is natural that the context for the accounting event management be established for those service-oriented accounting management. The context is to be established at the beginning of the service transaction, and is to be resolved at the end of the service transaction.

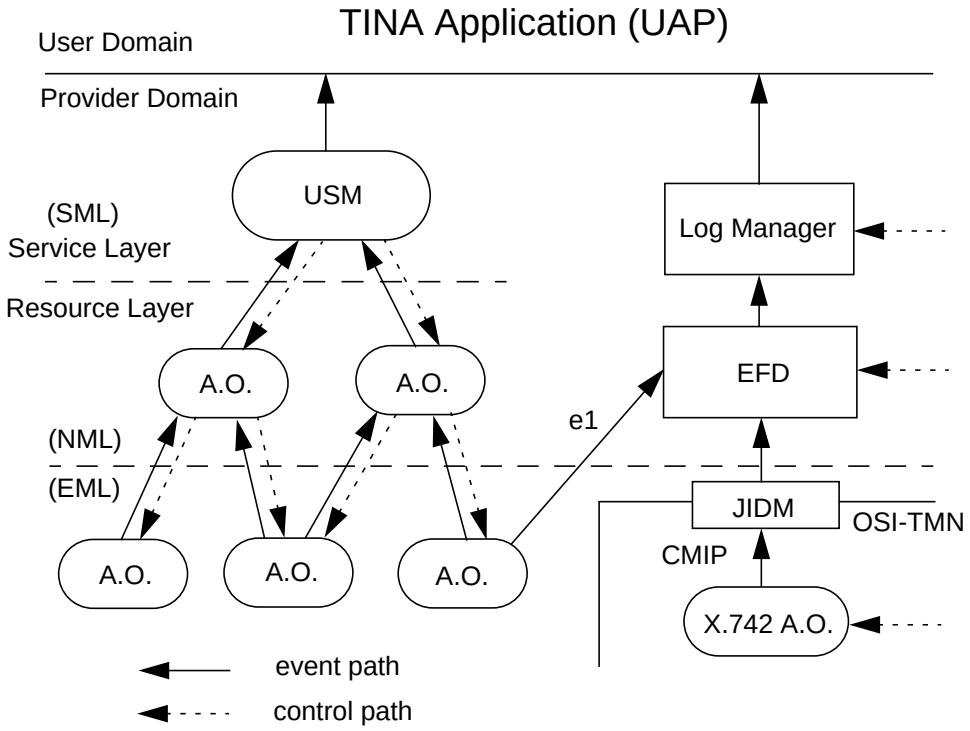


Figure 7-18. Event Management Ladder

Figure 7-18 illustrates the event management ladder concept. The ladder represents a context for the accounting event management, encompassing the accountable objects being involved in the service session. The control messages are passed from the top of the ladder, which is USM within the service layer. The accounting events are accumulated from the bottom at the EML of the resource layer, to the service layer up in the ladder, synthesizing service level billing information from the resource level accounting events.

In contrast to the TMN-style accounting management system shown on the right side of the figure, the ladder is more service-oriented, more dynamic than the TMN-style accounting management. In particular, the added flexibility offered by the ladder systems may be more attractive to the customizable accounting management, e. g. On-line billing.

The two accounting systems can interwork. For example, an accountable object in the ladder can send an event to the EFD, as it is shown as *e1* in the figure. The interworking between the two accounting systems is the key of the fine-grain migration from the TMN-style accounting management to a more TINA generic accounting management system based on the event management ladder.

7.7.4 Event Management Interface *i_eventManagement*

Every accountable object in a ladder provides an event management interface, such that the object can receive accounting events from other accountable objects.[14] The received events may be filtered, translated, and then forwarded or duplicated to be sent to other accountable objects.

The event management interface *i_eventManagement* is similar to EFD in its functionality, except it allows extra freedom in its filtering, event correlation, and diagnosis. Figure 7-19 illustrates the engineering model of the event management component. The component contains major three parts, the filtering, the correlation, and the forwarding functions. Of the three functional component inside, the filtering and the correlation functions are optional, which are attached to the forwarding function at the configuration time.

- *Filtering function*: the filtering function selects from the incoming event stream those events which match certain filtering conditions. Events can be translated into other sets of events, if necessary, since recipients of the outgoing event stream may be expecting a different event format. For example, a set of lower-level accounting events need to be transformed into a service-level billing event (information) at some point in the event management ladder. Filtering function can be utilized to perform *event translation* in general.
- *Correlation function*: the correlation function[15] detects certain correlation patterns within the incoming event stream, looking up the event records which are previously received from the stream. The correlation function can be utilized to perform diagnosis, which detects a higher level incident from a collection of lower-level accounting events.
- *Forwarding function*: the forwarding function forwards the output event stream to a set of designated recipients. Events may be duplicated, if necessary, when there are multiple recipients.

14. An accountable object may be representing a domain, or it may actually be an object group.

15. Although the filtering and the correlation are separated and distinguished here, their operations can be functionally quite similar, since both they transform a set of incoming events into an output event. The distinction mostly lie in their usage in the traditional accounting management systems.

Any of these three functions can be optionally schedulable, if the eventManagement box is configured to allow scheduling option at the configuration time. For example, if the filtering function is schedulable, the user of the eventManagement box can specify the time and the duration of the period that the filtering is applied to the incoming event stream.

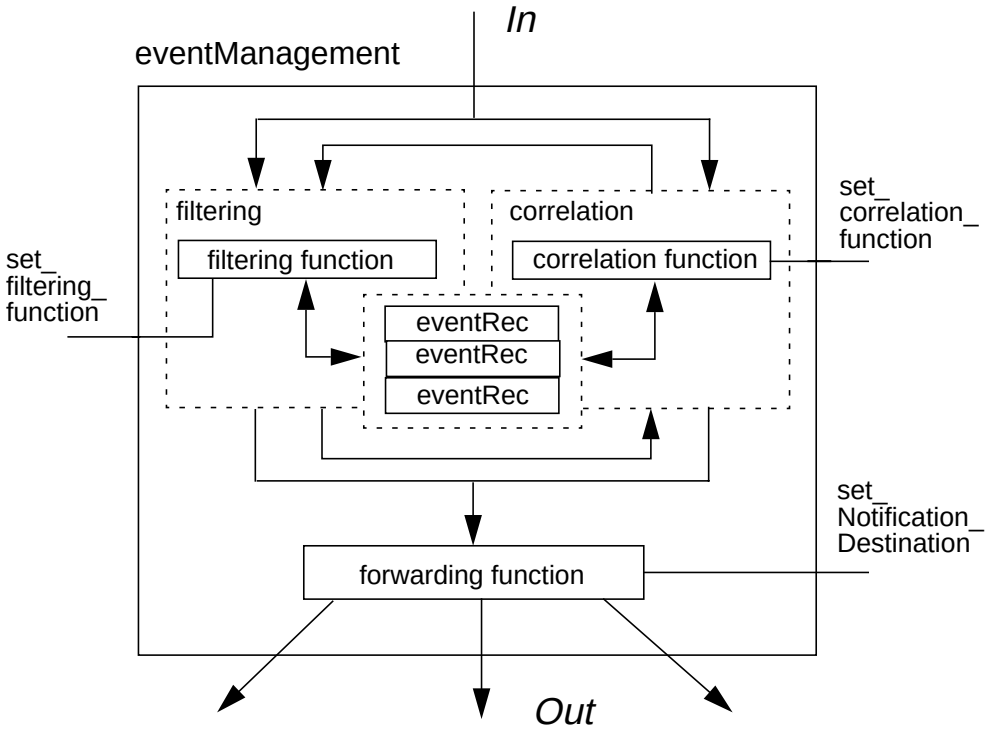


Figure 7-19. Event Management Component Engineering Model

It should be noted that our definition of the filtering function and the correlation function are independent from any particular format, i. e. their syntax as well as semantics are engineering view dependent.; the eventManagement box may accept X.734-style filtering parameters, or it may even accept an executable mobile-agent applet, which can perform more complex functions than can be expressed by a fixed set of X.734-style parameters.

Here are the computational specifications of the event management interface, *i_eventManagement*.

- Description: the eventManagement performs event forwarding discrimination. It may also perform filtering, translation, event correlation and diagnosis, as part of the forwarding process.

- Behavior: an input event stream is transformed into one or more output event streams.
- Operations:
  - Control operation:
    start
    stop
    suspend
    resume
    set_eventManagement_attributes
    reset_eventManagement_attributes
    configure
    reconfigure
  - Filtering function control:
    set_correlation_function
    reset_correlation_function
    query_supported_format
  - Correlation function control:
    set_correlation_function
    reset_correlation_function
    query_supported_format
  - Notification control:
    set_Notification_Destination
    reset_Notification_Destination
    add_Notification_Destination
    delete_Notification_Destination
  - Notification operation:
    event_notify

7.7.5 Federation in Accounting Event Management

We discussed the federation between accounting management domains already (Section 7.6.10). Even though federation between the management domains are not in effect or not possible, federation in accounting event management can be possible, still achieving many of the goals of the interoperability between different accounting management domains. We define the federation between two accounting event management systems as the following:

Accounting Event Management Federation (AEM-federation): two systems are federated, if and only if any one of the output event stream of one system can be fed into any one of the input event streams of the other system, and vice versa.

The above definition is fairly abstract, and can be explained only through an appropriate example.

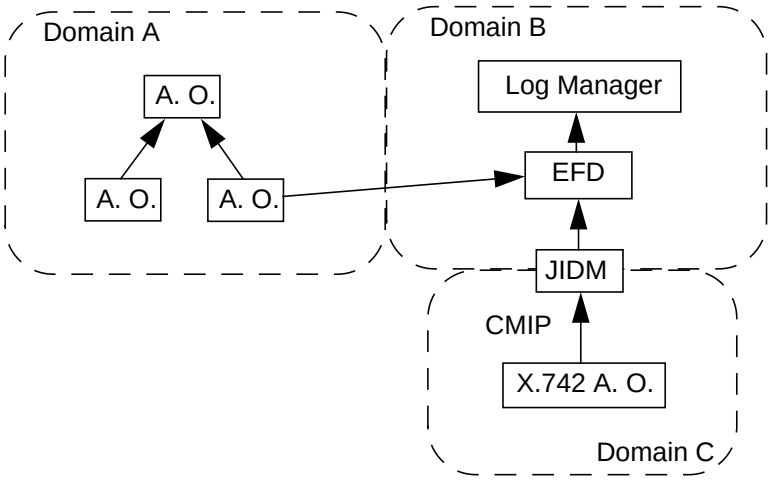


Figure 7-20. Federation in Accounting Event Management

Figure 7-20 illustrates the federation in accounting event management. In the figure, three domains are independent, and they are not federated as management domains. Domain A and B are federated in the accounting event management, since accounting events can cross the domain boundary, and an output event stream of an A. O. in domain A can be fed into the EFD in domain B, as it is shown in the figure. Similarly, domain B and domain C are federated in terms of the accounting event management, since events can cross the domain boundary via the JIDM translation module. Domain A and domain C are not federated in any sense, however, since event path from Domain A to domain C are separated by domain B.

In simple words, two accounting event management domains are federated, if and only if accounting events generated in the domains can travel freely, or via a specified gateway, between the two domains, with the condition that necessary security precautions are taken at the domain boundary. In this type of federation, accounting events can cross the boundary, and the necessary notification interfaces need to be exposed to each other, but the control interfaces are not exposed. In other words, the objects in domain A may receive events from the objects in domain B, but they do not accept control from the domain B objects.

This level of federation is apparently a limited case of the federation discussed in Section 7.6.10, where even the control or management interfaces can be exposed to each other. From the interoperability viewpoint of the accounting management systems, this level of federation is what is mostly needed, since the subsequent accounting operations and eventually the billing can be properly done, as long as the accounting events are correctly received.

7.8 Engineering Viewpoint

In the following part of this accounting management description of NRA, we focus on the issues important from the engineering viewpoint.

- *Deployment*: the deployment deals with the deployment of TINA accounting management system. Service life cycle issues are also discussed, along with requisite service groups and service templates.
- *Interworking*: Interworking issues with legacy accounting management systems are discussed.
- *Migration*: migration issues are discussed.

7.9 Accounting Metrics

7.9.1 Resource Independent Metrics

The resource independent metrics are those common to any accountable objects. As such, the metrics are exclusively those related to the control status and the life cycle of the accountable object. For example, metric types such as *start, stop, resume, create*, and *delete* belong to this metrics group.

7.9.2 TINA Network Resource Components

The TINA resource layer is meant to be technology independent, therefore the TINA network resource components do not assume any particular traffic or protocol characteristics. As a result, the traffic metrics generic to the TINA network resource components can be no more than the amount of traffic flow. For example, a metric type *traffic_flow* with various metric units such as *bit_per_sec*, *byte_per_sec*, etc., will suffice.

The TINA Network Resource Components include the following information objects [NRIM]:

- Subnetwork (G.803)
- LayerNetwork (G.803)
- Topological Link
- Link Termination Point
- Connection (G.803, M.3100)
- NWCTP (G.803, M.3100)
- NWTTP (G.803, M.3100)
- NWTPPool (M.3100)
- Subnetwork Connection (G.803)
- Tandem Connection (G.803)
- Edge (INA)
- Trail (G.803, M.3100)

As is seen from the list above, much of the components have been imported from respective TMN or INA specifications. Of the above components, *Subnetwork* and *LayerNetwork* need special attention, since they usually represent a separate accounting management domain. More details of this issue are explained in Section 7.6 on the accounting domain management.

7.9.3 ATM Specific Metrics

The TINA NRIM inherited its ATM information model from Bellcore TA-1114 [53]. It mentions the following information objects:

- VCCE (Virtual Channel Connection Endpoint)

- VC link
- VC connecting point
- VPCE (Virtual Path Connection Endpoint)
- VP link
- VP connecting point

For the above objects, the ATM traffic metrics applies, such as those defined in Bellcore GR-1110 ([54]], section 10, Usage Information to Support Billing). The following metrics are proposed:

- Ingress Total Cells
- Ingress High Priority Cells
- Egress Total Cells
- Egress High Priority Cells

For the ease of processing accounting data, it is also allowed to include traffic parameters as a part of accounting data.[16] For the ATM PVC, the following parameters shall be available:

- Ingress Peak Cell Rate (CLP=0)
- Ingress Peak Cell Rate (CLP=0+1)
- Ingress Sustained Cell Rate (CLP=0)
- Ingress Sustained Cell Rate (CLP=0+1)
- Ingress QoS Class
- Egress Peak Cell Rate (CLP=0)
- Egress Peak Cell Rate (CLP=0+1)
- Egress Sustained Cell Rate (CLP=0)
- Egress Sustained Cell Rate (CLP=0+1)
- Egress QoS Class

For the ATM SVC, the following parameters shall be available:

- Forward Peak Cell Rate (CLP=0+1)
- Forward Peak Cell Rate (CLP=0)
- Forward Sustainable Cell Rate (CLP=0+1)
- Forward Sustainable Cell Rate (CLP=0)
- Forward Maximum Burst Size (CLP=0+1)

16. The assumption is that the accounting data are to be processed by the billing system. Since the billing belongs to the service management, it is not treated in this document. In TINA, we expect that these traffic parameters are retained differently, possibly within a service management context. It still awaits further study, in conjunction with the mapping issues of QoS.

- Backward Maximum Burst size (CLP=0)
- Backward Peak Cell Rate (CLP=0+1)
- Backward Peak Cell Rate (CLP=0)
- Backward Sustainable Cell Rate (CLP=0+1)
- Backward Sustainable Cell Rate (CLP=0)
- Backward Maximum Burst Size (CLP=0+1)
- Backward Maximum Burst size (CLP=0)

7.9.4 SDH Specific Metrics

To be studied.

8. Open issues

The following issues are subjects for a future release of this document. Probably not all of these issues will be covered in the next release. The priority will be determined by the CTC (Consortium Technical Committee) and the TAB (TINA Architectural Board).

8.1 Federation

Federation is possible at a number of different levels. Most urgently, federation at the LNC level requires further study. However, relations at the connectivity and communication level also needs consideration. Federation may be confined simply to setting up connections between different domains or extended to considering implications for management services in general.

8.2 kTN

The relation of the Kernel Transport Network (kTN) and Network Resource Architecture needs to be worked out. E.g. Is the CMA used to establish (parts of) the kTN? How does bootstrapping effect the kTN and the link to the NRA.

8.3 Security

The focus of the security management in NRA is to:

(1) explain QoP: Quality of Protection represents the service level requirements on the security management. The issue involves what needs to be protected, and how in the resource architecture.

(2) show computational mechanisms: to achieve the above mentioned QoP. It is assumed that DPE would provide some basic mechanisms such as authentication and encryption, but there are other items necessary to address the issue. Besides, DPE security is not likely to be incoorported in the planned release of the document, which means NRA has to stand on its own.

8.4 Stream flow connections and communication session

The relation of between SFCs and stream bindings and expanded communication session capabilities will be subject of a more elaborated study. The capabilities requiring further investigation include mappings restrictions, the use of special resources at this level, and support for stream bindings mappings.

8.5 Terminal/CPE domain setup of connections and the introduction of Private or Consumer Domain Networks

This topic will contain a more in depth study of setting up connections in the Terminal/CPE domain as well as the introduction of private or consumer domain networks. This should also address the relations between the TCSM and TLA in more detail.

8.6 Connectionless Networks

This issue covers the introduction of connectionless networks, in all their perspectives, in the NRA using internet as an example.

8.7 Generic Management functionality

Functionality common to all the management areas, will be separated from each of the individual management chapters and covered in a separate chapter. The role of access session needs to be considered.

8.8 Prescriptive and descriptive parts

Component specifications and validation work will be used to determine in more detail which parts of the NRA need to be prescriptive. This labeling will most likely be done in function of a certain level of interoperability. These levels may be formalized to TINA-C compliancy standards, which will be suitable for interworking.

9. Acknowledgment

The authors would like to thank the authors of the Connection Management Architecture document (CMA) which was the basis of this document:
1994: Motoharu Kawanish, Hiroshi Oshigiri, Juan Pavón, Mike Schenk
1993: Jack Bloem, Glenn Flinchbaugh and Hiroshi Oshigiri.

Finally, this document has benefited from lots of suggestions and comments from many persons in the core team and in the TINA-C member companies. They have helped to improve the technical content and the presentation of this document. The authors would like to thank particulary:

- The external reviewers: Nathalie Charton (Alcatel), Uwe Herzog (Deutsche Telekom), Peter Komisarczuk (Ericsson), Tero Koskivirta (Nokia), Osamu Miyagishi (NTT), Tom Handegard (Telenor), Leif Byström (Telia)
- The internal reviewers: Jarno Rajahalme (Nokia), Harm Mulder (KPN) and Magnus Hedberg (Telia).

Chelo Abarca
Alcatel Telecom, Madrid
Spain

Jan Forslow
Ericsson Telecom
Sweden

Takeo Hamada
Fujitsu Laboratories
Japan

Stephanie Hogg
Telstra Corp.
Australia

Hong Beom Jeon
Korea Telecom
Korea

Dae Seok Kim
Samsung Electronics co., ltd
Korea

Hahn Young Lee
Korea Telecom
Korea

Narayanan Natarajan
Bellcore
USA

Frank Steegmans
Alcatel Telecom, Antwerp
Belgium

Appendix A: Network, Termination Point, and Transmission Fragments

This appendix is extracted from the information model of the 1994 CMA and only here for completeness. in the next version of this document the contents will be updated and reinserted in the main body. Lack of time restrained us from doing this in this version. Although a number of the concepts are outdated the basics are still essential for a good understanding of the Connection Management.

The information specification of Network, Termination Point, and Transmission fragments represents network resources. They are mostly straightforward representations of concepts specified by G.803 [28].

From the perspective of Connection Management, the Network, Termination Point, and Transmission Fragments are classified into the following two categories:

1. Base Managed Objects: These are managed objects that exist before connection setup activities begin and remain after connections are cleared down. They describe the actual network infrastructure. Major ones are:
   - *subnetwork*
   - *topologicalLink*
   - *linkTerminationPoint*
   - *connection*
   - *nWCTP (network connection termination point)*
   - *nWTTP (network trail termination point)*
   - *nWTpPool (network termination point pool)*
2. Dynamic Connectivity Managed Objects: These are managed objects that are created in the course of connection setup activities and destroyed when the related connections are cleared down. The following are major ones:
   - *subNetworkConnection*
   - *tandemConnection*
   - *edge*
   - *trail*

Section A.1 and Section A.2 illustrate both categories by using an example network. Both sections will do this in terms of the G.803 functional model and object instance diagrams. Definitions of each managed object are not given in this document and readers are referred to Network Resource Information Specification[1].

An assortment of ideas about dynamic connectivity objects are described in Appendix B for interested readers.

Table A-1 shows the origins of the managed objects used by Connection Management functions and their relationships with G.803 concepts.

Table A-1. Origin of Managed Objects

Managed Object	Related G.803 concepts	Origin
subnetwork	sub-network	G.803
topologicalLink	link	G.803
linkTermination-Point	none	
connection	link connection	G.803, M.3100
nWCTP	connection termination point termination connection point	G.803, M.3100
nWTTP	trail termination point	G.803, M.3100
nWTpPool	none	M.3100
subNetworkCon-nection	sub-network connection	G.803
tandemConnection	tandem connection	G.803
edge	none	INA
trail	trail	G.803

A.1 Base Managed Objects

Figure A-1 illustrates an example layer network. *SNW3* represents a portion of the layer network. *SNW4* and *SNW5* represent partitions of *SNW3*.

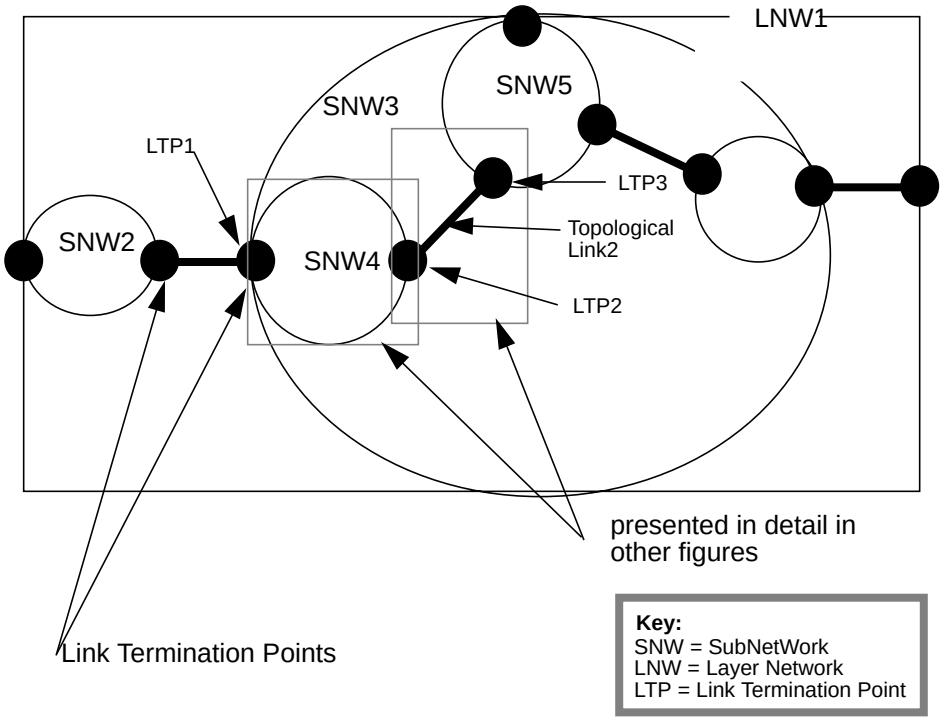


Figure A-1. Example Layer Network (in terms of G.803 concepts)

Figure A-2 provides a high level view of composite relationship of managed objects that represent these *Subnetworks*. This figure presents the whole view of the example network, but each CM domain(i.e. SNW3, and SNW2) does not have the whole view. For example, SNW2 only has a “picture of” SNW3 and is not aware that there are SNW4 and SNW5.

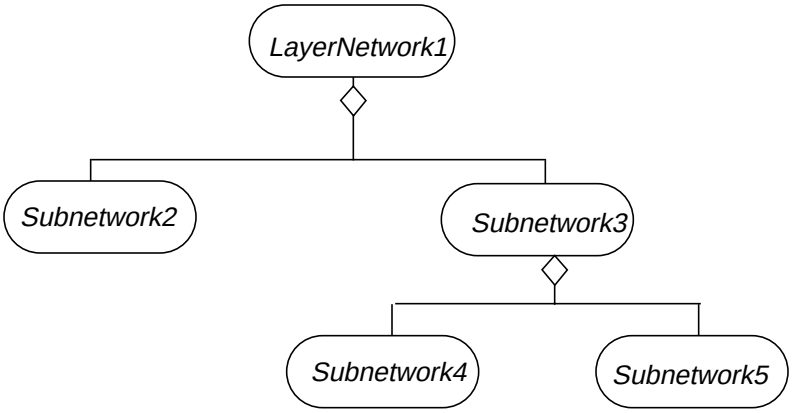
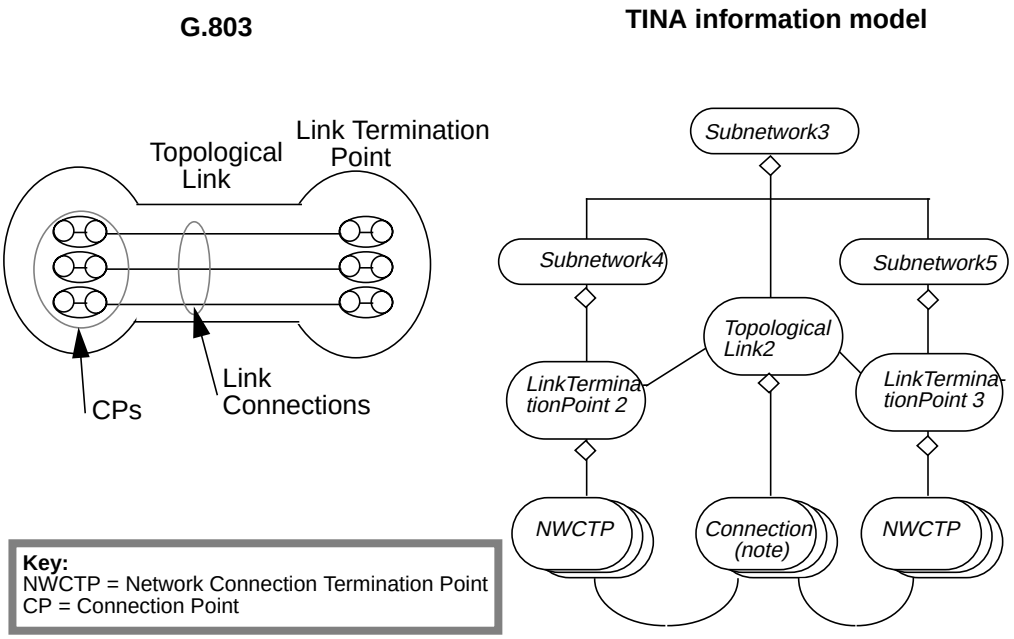


Figure A-2. Composite Relationship of TINA SNW Objects



Note: *A connection* managed object represents a link connection of G.803

Figure A-3. Topological Link and Related Managed Objects

Figure A-3 illustrates the enlargement of the Topological Link 2 in Figure A-1. A *TopologicalLink* is a collection of all the connections that are served by a trail of the server *LayerNetwork*. A *LinkTerminationPoint* is an end point of a *TopologicalLink*.

SNW4 of Figure A-1 is enlarged in Figure A-4 to give an example of *NWTpPools*. A *NWTpPool* represents a collection of termination points. *NWTpPools* are used for the purpose of routing.[1]

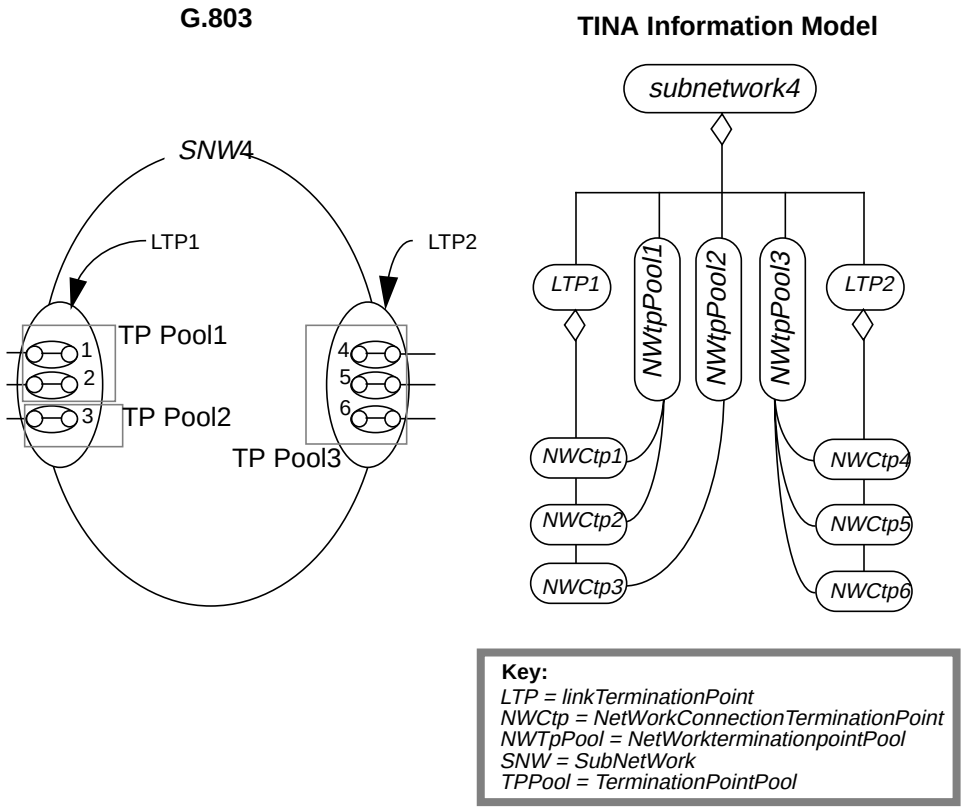


Figure A-4. Network TpPools

A.2 Dynamic Connectivity Managed Objects

Dynamic connectivity objects include *subnetworkConnection, tandemConnection*, and *trail*, each of them having *edges* as subordinates.

1. Typical examples of *NWTpPool* represents a group of trunks that lead to the same node or group of VCIs on the same UNI of ATM access interface.

A.2.1 Subnetwork Connection

A SubnetworkConnection represents a point-to-point unidirectional, point-to-point bidirectional, or point-to-multipoint unidirectional connectivity that runs across a Subnetwork.

Figure A-5 shows an example of a network and a point-to-point subnetwork connection that runs through the network. SNC3 (SubNetwork Connection 3) is a subnetwork connection whose termination points are on the rim of subnetwork3 (SNW3). SNC3 is composed of SNC4 and SNC5.

Figure A-6 is the relationship diagram of information objects that represents this example of subnetwork connection. SNC3 is decomposed into SNC4 and SNC5 using the partitioning relationship. For each *SNC* there are *edge* managed objects as its subordinates. The prime role of an *edge* managed object is the binding between a *SNC* and a termination point. Within a *SubnetworkConnection*, there is one root *Edge*. This is the *Edge* created at the same time as the creation of the *SubnetworkConnection*. If the *SubnetworkConnection* is unidirectional, then the root *Edge* represents the source of the stream. Other *Edges* are called leaf *Edges*.[2] If two *edges* has the *peerEdge* relationship(e.g. *edge42* and *edge 51*), the two *edges* are or will be attached to the endpoint of the same *connection*.

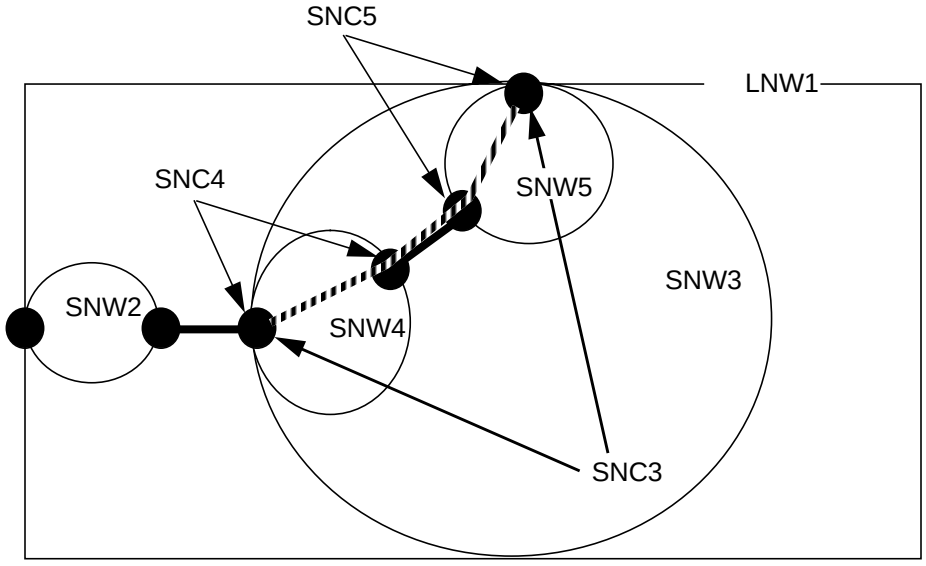
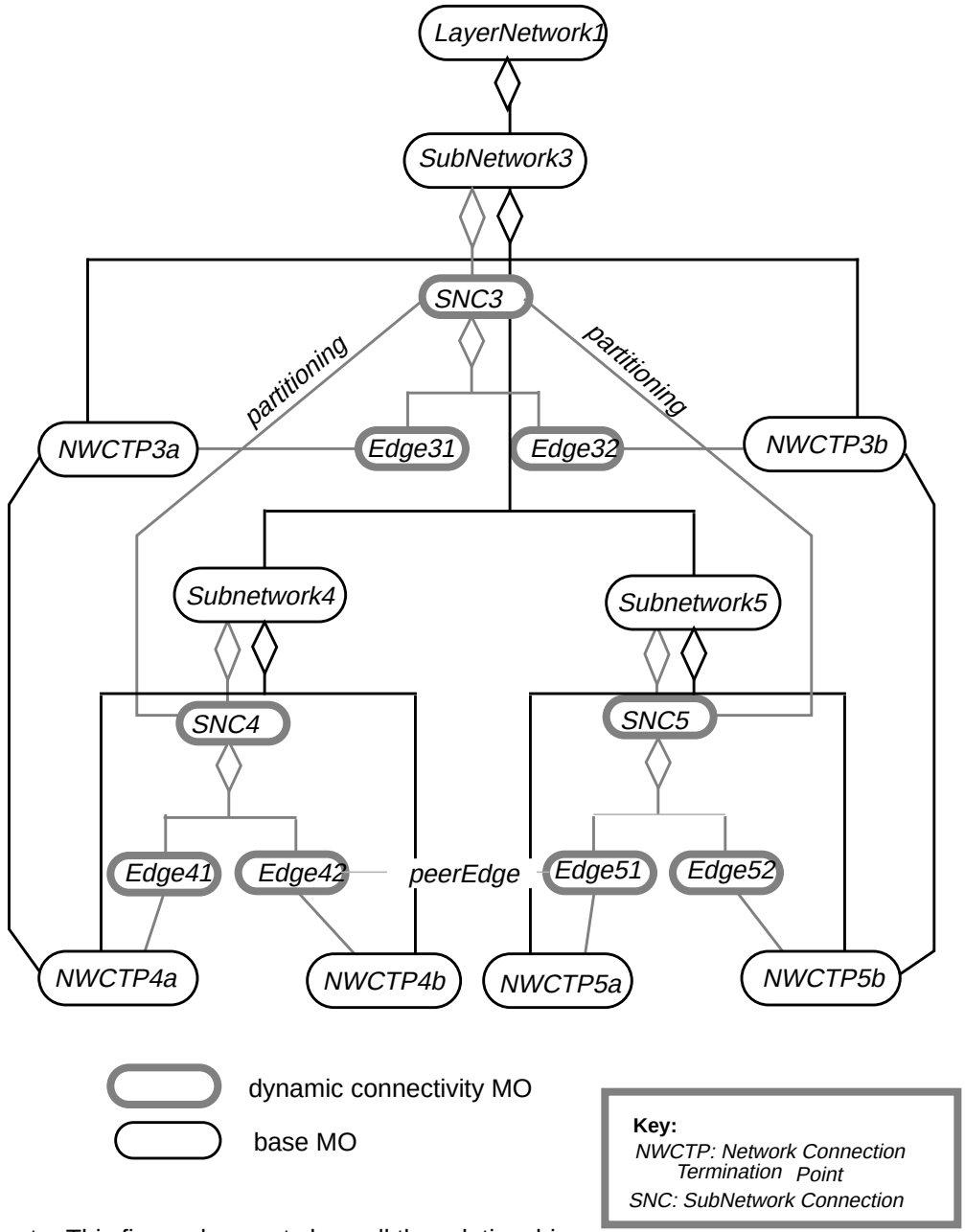


Figure A-5. Example Point-to-Point Subnetwork Connection (in terms of G.803 concepts)

2. Since questions on necessity of the edge object are frequently asked, the answer is given in Appendix B.

SubnetworkConnection is used for operation definition of the CP computational object; more casually, one can interpret that the *subnetworkConnection* is the view offered by the CP object to its clients in order for them to control the connectivity.



note: This figure does not show all the relationships.

Figure A-6. Point-to-Point *SubnetworkConnection*

Figure A-7 depicts a point-to-multipoint subnetwork connection whose object instance relation diagram is given in Figure A-8.

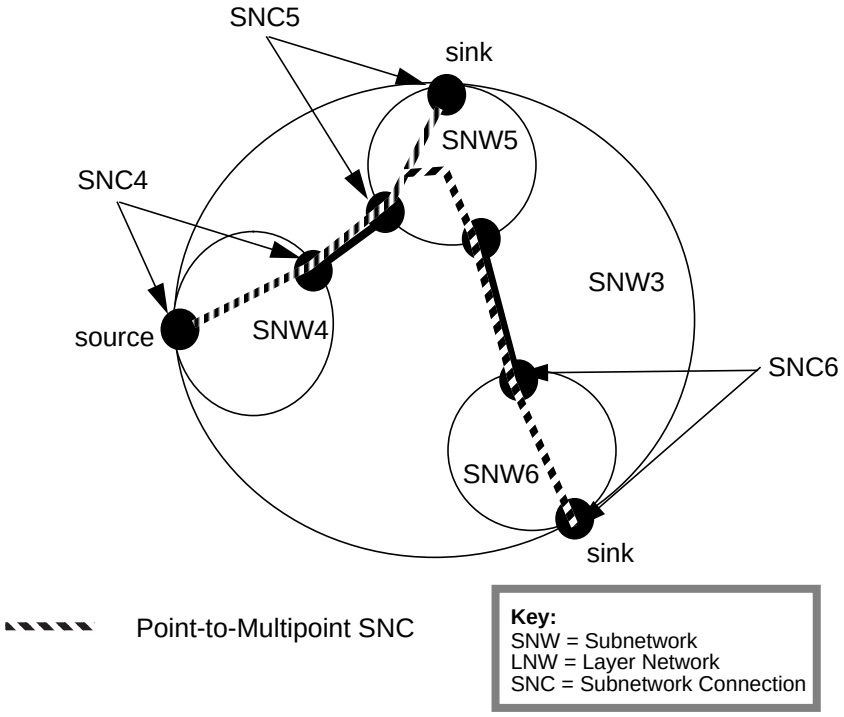


Figure A-7. Example Point-to-Multipoint Subnetwork Connection

In Figure A-8, base MOs are not shown for simplicity. Instead, complete relationships among SNCs and edges are shown. *SNC3* is decomposed into *SNC4, SNC5*, and *SNC6.* This decomposition is represented by *partitioning* relationship. *Edge31* and *edge41* are at different partitioning levels but they represent physically the same endpoint. This is what is meant by the *delegate* relationship.

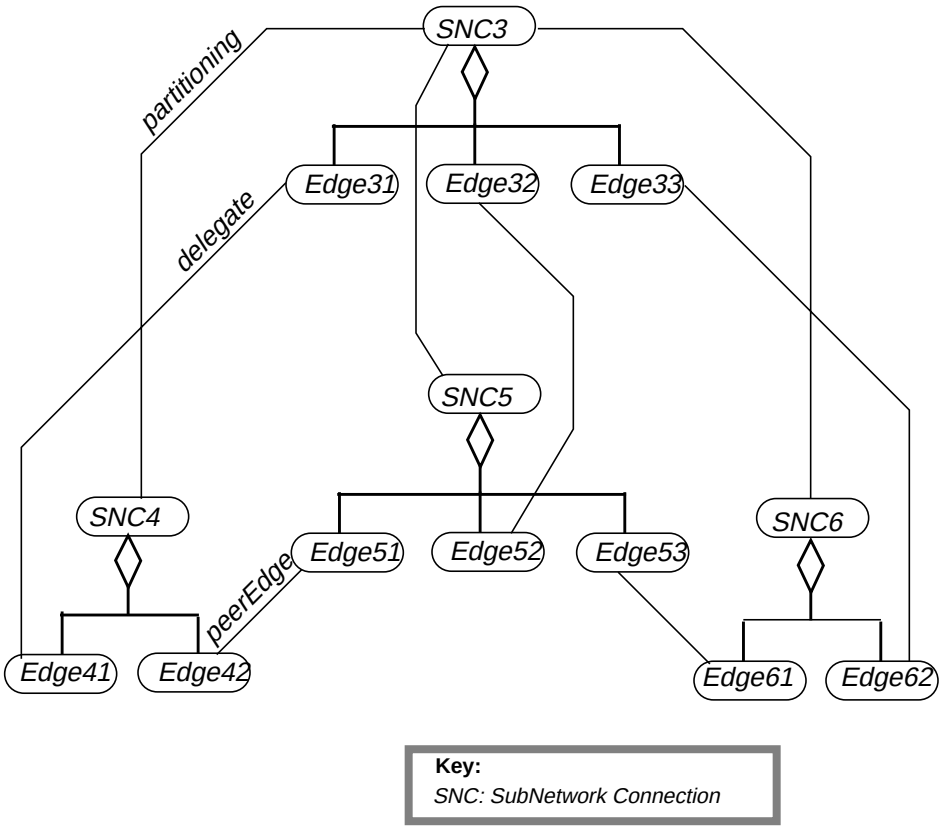


Figure A-8. Point-to-Multipoint Subnetwork Connection

A.2.2 Tandem Connection and Trail

A *tandemConnection* object represents a series of *connections* and *SNCs*. A *tandemConnection* may be composed of only one *SNC* and no *connections*, in which case it coincides with the *SNC*. Similarly, a *tandemConnection* can coincide with a *connection*. If a *tandemConnection* spans between *nWCTP*s on the rim of the *layerNetwork* (or more precisely, *nWCTP*s co-located with *nWTTP*s), then it virtually represents the same connectivity as a *trail.*

In order to explain how *trail* and *tandemConnection* objects are used in connection management activities, the CM domain concept needs to be mentioned. A CM domain is a SNW and a set of CM computational objects responsible for connectivity objects in the SNW. A domain has a complete view of its own SNW and overview of other SNWs. Consider the case where a client requests a CM domain to set up a trail that spans multiple domains. IN this case, the first CM domain needs to cooperate with other CM domains to provide the trail. Basically, there are two options for federation. The first CM domain can either pass the request to a supra CM domain and leave the whole control to it (joint federation model), or

try to set the trail up by having a peer-to-peer interaction with a neighboring CM domain (peer-to-peer federation model). It is the latter case where the relationship between *trail, tandemConnection*, and *SNC* objects is important. In this case the CM domain that has been requested a *trail* decomposes the *trail* object into a *subnetworkConnection* of its own domain and a *tandemConnection* that spans other domains. Then the first CM domain sets up the *SNC* in its own domain while requesting one of the neighboring domains to setup the *tandemConnection*. That *tandemConnection* is again decomposed into an *SNC* and a *tandemConnection* in the neighboring domain. In this manner, a *tandemConnection* setup request is recursively cascaded through the layer network. It is noted that the *trail* object is only instantiated in the contact domain. Other domains are not aware that the requested *tandemConnection* is part of a *trail*.

Now we illustrate what has been explained using a point-to-point trail as an example. Figure A-9 illustrates the decomposition of a trail. The trail is decomposed into SNC1 of domain 1 and TC2 that spans domain 2 and domain 3. This decomposition happens if domain 1 is the contact domain for the trail. A request for setting up TC2 is made to domain 2, where TC2 is decomposed into a SNC of domain 2 and another TC.

Figure A-10 depicts the object instance diagram held in domain 1 and domain 2. Note that the *trail* is instantiated only in domain 1. In domain 1, *trail1* is mapped to *TC1* which represents the virtually equivalent connectivity as *trail1*. *TC1* is decomposed into *SNC1* and *TC2*. In domain 2, *TC2'* which represents the same connectivity as *TC2* is created which is decomposed into *SNC2* and *TC3*.

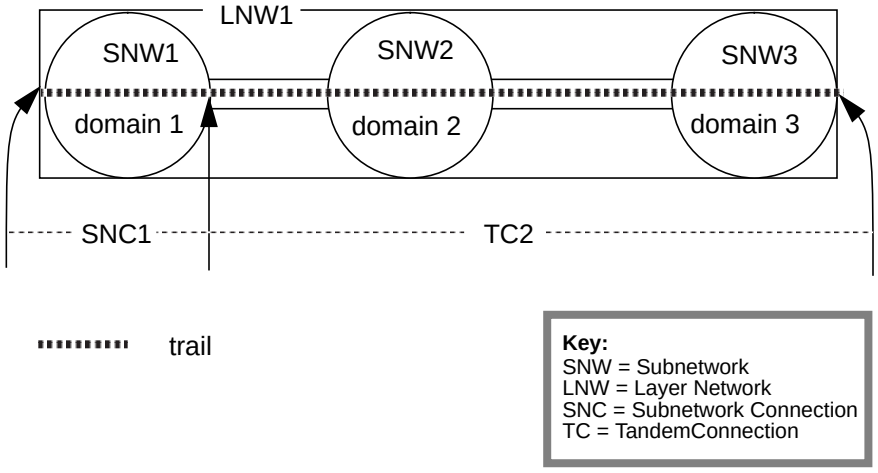


Figure A-9. Trail and Tandem Connection Example

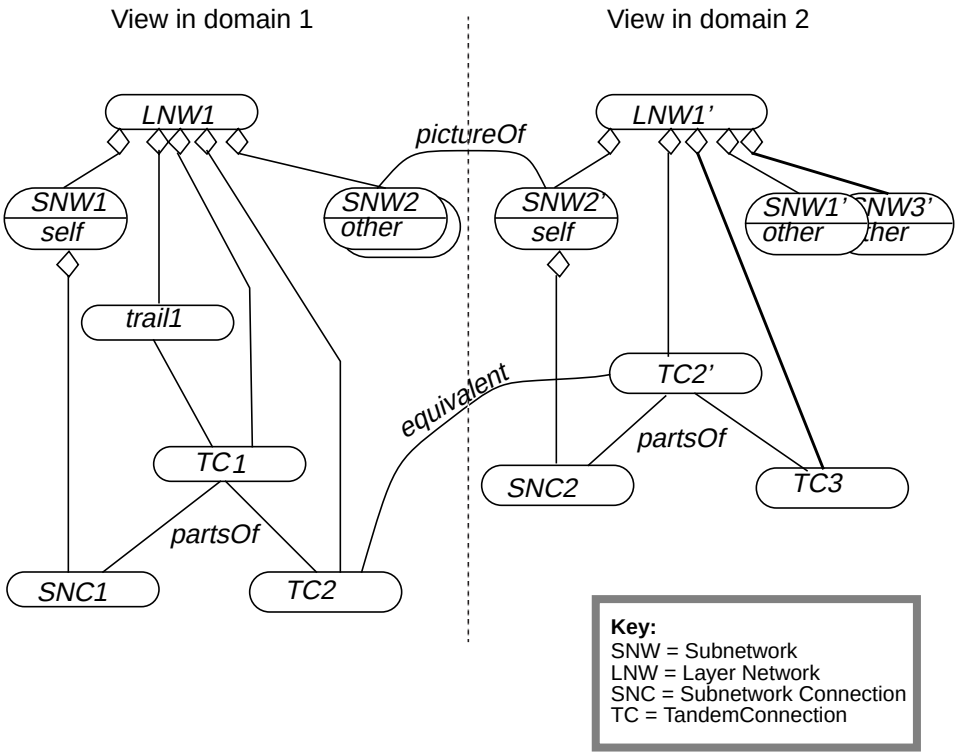


Figure A-10. Tandem Connection

A.2.3 Connectivity Group

In order to represent a group of *trails, tandemConnections*, or *SNC*, an object *connectivityGroup* is defined. A set of connectivity objects (i.e. *trails, tandemConnections*, or *SNCs*) that are associated with the same *connectivityGroup* are synchronized in some way. The following attributes are defined for c*onnectivityGroup* object in order to indicate the nature of the group. The attribute types are all boolean(i.e. yes or no).

- **same route**: if yes, the set of connectivity are routed through the same links and nodes.
- **different route**: if yes, the set of connectivity runs through different routes. This is used for redundant connectivity for security.
- **same life**: if yes, then all the connectivity of the group are destroyed at the same time.

Appendix B: Some Insight on Connectivity Objects

B.1 Edge

The *subNetworkConnection, tandemConnection,and trail* managed object represents connectivity between two or more *nWTP*s (*nWCTPs or nWTTPs)*. Therefore, the important information about them is which *nWCTPs* are connected by them. This is where *edge* comes in. An *Edge* managed object is a subordinate of these connectivity objects and it acts as a pointer to a *nWTP*. If a *SubNetworkConnection* binds three *nWTPs*, for example, there are three *Edges* under it.

Other than this main role as a pointer, it also contains information about the characteristics of the role played by the *nWTP* in the context of the association that is represented by the *SubNetworkConnection*. An important one is root/leaf indicator. When a SubNetworkConnection is created, one *Edge* is also created under it. This *edge* is called the root *edge*, and continues to exist until the superior *SubNeworkConnection* is deleted. During the life time of a *SubNetworkConnection*, other *Edges* are dynamically created and dropped. These *Edges* are called leaf *Edges*. Other attributes of *edge* include hold status, etc.

The reader may wonder why “edge” is represented by a subordinate managed object of a *connectivity* objects rather than by an attribute of them. The objective of “edge” can also be achieved by this alternative way of modelling it. However as a design decision, we chose to represent “edge’ as a managed object for a number of reasons indicated below.

1. There is a good deal of information associated with each “edge”.
2. We want to facilitate for the occasion where a *nWTP* associated with to a *connectivity* is replaced with another *nWTP* taking over the predecessor’s role.
3. *Edges* are added and dropped dynamically during the life time of the superior *connectivity*.
4. relationships between edges (peerEdge, delegate) enables the description of complete information structure of connectivity Objects (trail, tandemConnection, SNC). The usefulness is significant in the case of point-to-multipoint connections.

B.2 Operations on Subnetwork Connection and Edge

There is some difference in operation definitions between the information view point and computational viewpoint. Here we describe the principle that we followed when we made the operation definitions from these viewpoints. The things said in this section apply equally to trail and tandemConnection.

B.2.1 Information Viewpoint

In the information viewpoint, the effect of operation should basically within the object to which the operation is defined. Whenever other objects are affected, it is through relationship definition made between the objects. With this in mind, and considering the case of making a subnetworkConneciton binding two nWCTPs, the operation sequence becomes like as follows.

1. createSNC
2. createEdge (for source)
3. createEdge (for sink)
4. attachEdge (for source)
5. attachEdge (for sink)

Readers are referred to Network Resource Information Model Specification [1] for specification detail.

B.2.2 Computational Viewpoint

In the computational viewpoint, a (set of) information object instance corresponds to an interface instance of a computational viewpoint. In the case of SNC, the computational object is CP. An SNC and its subordinate edges correspond to a CP interface instance.

From the computational viewpoint, we defined operations that have the combined effect of number of operations defined in the information viewpoint. The principle we followed in defining these operations are to provide a simple set of operations that can satisfies the needs. Here operations should also be dynamically simple; i.e. small number of operations should be sent for establishing an SNC.

There are three operations involved in the creation of a *SubnetworkConnection*; namely,

- ***initiateSNC***: This operation is defined on the factory interface of CP. This operation instantiates an interface that is dedicated to the control of one SNC. This operation also has the combined effect of createSNC, createEdge (root) , and attachEdge (root) that are defined from the information viewpoint. Optionally, this operation can also create and attach leafEdge(s).
- **createEdge**: This operation is defined on the interface instantiated by initiateSNC. The effect is basically the same as createEdge of information viewpoint. This operation can optionally have the effect of attachEdge as well.
- ***attachEdge***: This operation is defined on the same interface as createEdge. The effect is the same as attachEdge of information viewpoint.

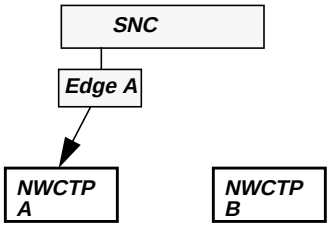
These operations are sent in a fixed order. Figure B-1 shows the order and how managed objects are configured as the result of each operations.

1.Precondition

Termination points exist. (Other objects such as *a SubNetwork* are not shown in the figure).

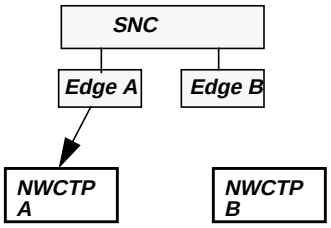
2. *initiateSNC:*

This action creates a *SubNetworkConnection* and an *Edge* Object for A and attaches it to the *NWCTP* for A.The *Edge* created by this action is called a root *Edge.*



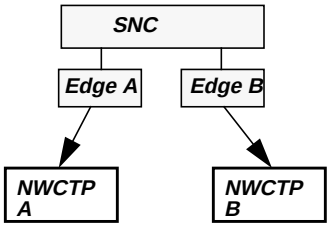
3. *create (edge)*

This operation creates an *Edge* for B. An *Edge* created by this action is called a leaf *Edge.*



4. attachEdge

This action attaches the *Edge* of B to a *NWCTP*).



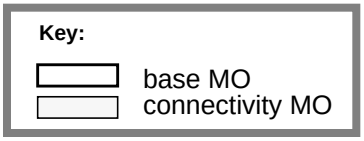


Figure B-1. *SNC* creation procedure

B.3 Nodes other than a Switch or a Cross Connect (CPE, processing

node, etc.)

Any node (e.g. CPE) connected to the public transport network is either modelled to have a *Subnetwork*[1] or not to have one. Figure B-2 illustrates how a CPE is modelled in terms of Network, Termination Point, and Transmission fragments. In the figure, SNW1 represents a SubNetwork within a CPE node. SNW2 represents a public network domain.

.

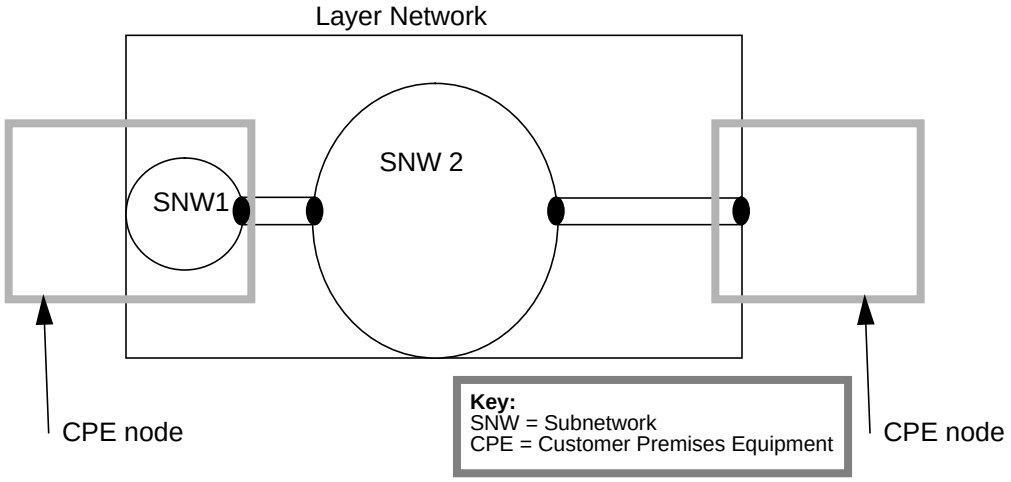


Figure B-2. CPE and SubNetwork

B.4 Special Resources

Special resources here are devices such as video bridges and transcoders. The common characteristic of special resources is that they take one or more input streams, process it, and inject the processed information into output streams. We defined a managed object class called *adaptor* to model special resources with adaptation functions between layer networks.

A node having a special resource is modelled to have a subNetwork as well. The node can be a switch node or a CPE having special resources. Figure B-3 shows an example of a three point bidirectional connection topology. It is modelled by three tandem connections connected between three CPEs and one special resource node.

As the diagram shows, the adaptor resides outside of the Layer Network. CM functions cannot directly control an adaptor. Different Computational Objects are to be defined for controlling different types of Information Converters.

1. This decision does not mean that internal aspects of a CPE become visible from the public network.

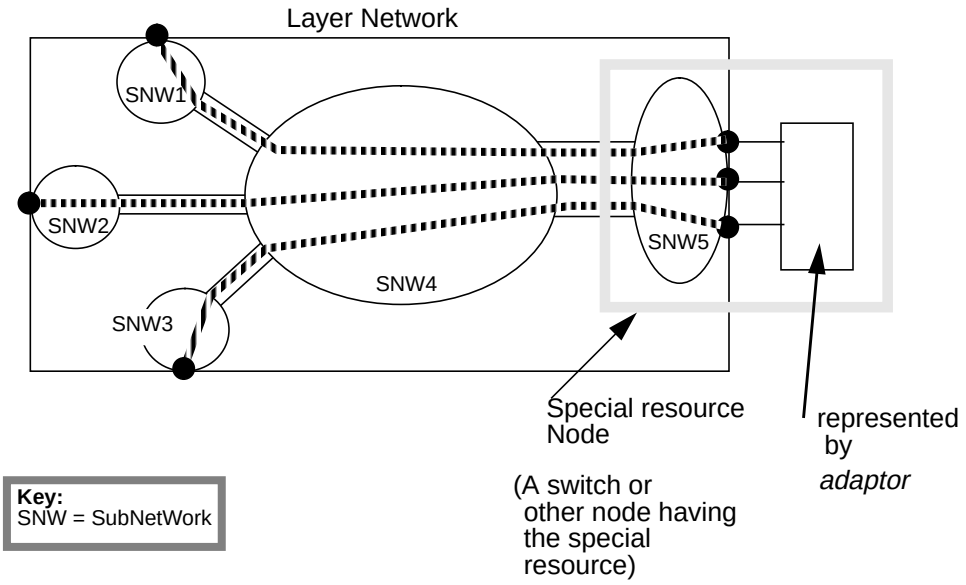


Figure B-3. Special Resource

B.5 Connection and NWCTP as Base Objects

The reader may feel that a connection is something created by the Connection Performer (i.e. Connectivity Managed Objects). But what a Connection Performer does is to create *SubnetworkConnections* that bind the termination points of *connections*. Therefore Connections must preexist as base managed objects.[2] In the case of an ATM network, preexistence of a *connection* merely means preexistence of the VCI (or VPI) value on both sides of the *Connection*. Other elements of a *Connectio*n such as bandwidth are assigned every time it is used(i.e. when a *SubnetworkConnection* is attached to it).

The same applies to *nWCTPs*.

2. Here, “preexist” means that just “exist before SNC is created”. It may be created only just a second before SNC is created, as explained in the following paragraphs.

Appendix C: Naming in the Resource Architecture

This chapter on naming is placed in an appendix in this release of the network resource architecture as it is a new topic and, therefore, lacks stability. The intention is to evolve the text into a stand-alone prescriptive chapter in the main body of the next release of the network resource architecture baseline.

C.1 Scope

This chapter specifies naming systems to be used within the network resource architecture.

C.1.1 Names

Names are distinguished from addresses in the sense that names are used as title identifiers, whereas addresses are used as location identifiers for particular entities. In the TINA DPE environment, application entities use location independent names to relate to each other, while addresses are provided by the DPE itself as part of the allocated object references. This distinction applies for the current TINA network resource architecture.

The embedding of any progressive kTN signalling in the network resource architecture would violate the assumption above and make permanent addressing information visible in the application layer of TINA in order to perform routing. This aspect is not addressed in the current version of the document.

C.1.2 Naming Systems

The naming domains addressed in this chapter are information and computational types and instances in the network resource architecture. The naming systems for information objects and computational objects are separated in order to allow flexibility in the mapping between them.

Two name sets are distinguished for each naming domain.

- One name set is restricted to unique (distinguished) symbols used to unambiguously denote an entity during its lifetime (also called identifiers in [17]).
- Another name set includes all nick names that can be used to denote an entity. Different entities may be assigned the same nick name (homonym name) and an entity may be assigned several nick names (absolute names). Nick names may be re-assigned during the lifetime of the entity.

A name can either be simple (relative) or qualified. A qualified name includes bindings between naming contexts. The qualification may be repeated to form a multi-level name with TINA system global scope.

C.1.3 Name Resolution

Two types of name resolutions are required in the network resource architecture: to resolve naming contexts of a qualified name in a naming network and to resolve the resulting simple name to a computational object reference (embedding address information).

The name bindings between names and naming contexts as well as names and object references are managed by a name resolution mechanism. The resolution mechanism can be implemented as a local mapping in the computational object or as a distributed directory (naming or trading service). A name binding maintained by a directory is normally cached to the local computational object after the first lookup for later usage. The two name resolution mechanisms are, therefore, used in combination.

The name resolution mechanisms are not defined in this document. The reader is referred to [17] for further definitions of name resolution mechanisms.

C.2 Functional Requirements

C.2.1 Names

It shall be possible to assign technology dependent names to information objects in order to simplify the visibility of network resources to persons managing the network, e.g. to use temporarily assigned VPI/VCI values as names for network connection termination points.

C.2.2 Naming Systems

The network resource architecture specific part of the naming systems in Section C.1.2 relates to the information viewpoint (i.e. the definition of naming rules for information objects like subnetworks, termination points, etc.). Naming of computational entities (like a computational interface CC::Connection_Session_Setup) shall be based on the general TINA naming conventions defined in [17].

The TINA network architecture is based on the TMN information model [41]. The naming systems for the information view are, therefore, required to be based on TMN but adapted to TINA quasi-GDMO/GRM. Specifically, an information object instance needs to have one attribute value assertion that denotes its Relative Distinguished Name (RDN). The RDN is then an attribute value assertion in which a particular attribute has a particular value used to identify the object out of all those immediately subordinate to a given information object instance. The RDN shall be used as a component of a Distinguished Name (DN). The DN is the name of an information object instance formed from the sequence of the RDNs of the information object instance and each of its superior objects. One naming network shall cover all information object instances in the network resource architecture. The naming network (the relations between RDNs) shall be defined by name bindings. The name binding in the network resource architecture needs to visualize containment relationships between information object instances. The name binding shall be static during the lifetime of the involved entities in order to achieve a stable naming network for distinguished names that scales to large global TINA systems.

It shall be possible to determine the instance name of an information object to be created in two ways:

- The name may be completely and explicitly specified by the information object requesting its creation, as a parameter in a create operation.
- The information object instance, receiving the create operation, may assign the name if no explicit naming information is given in the create request.

The first alternative will be needed for bootstrapping and to allow network managers to enforce specific names when requesting the creation of an information object instance. In order to keep a consistent naming network, the creating information object must be able to refuse creation if the name request is not in line with the naming network.

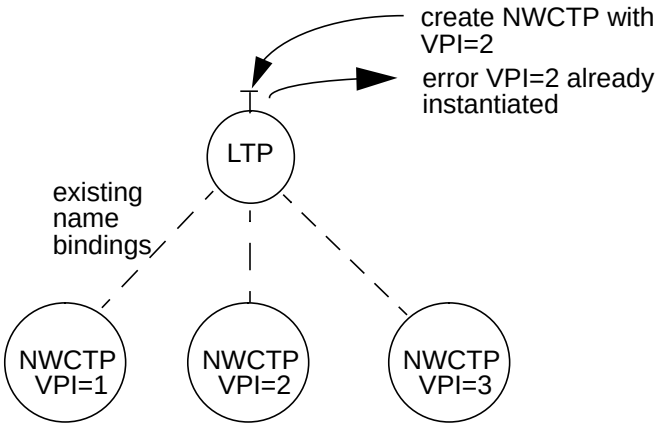


Figure C-1. Example of Create Request and Error Response.

C.2.3 Name Resolution

As stated in Section C.2.2, the naming network for information object instances must make containment relationships between information object instances visible, so that name resolution can be used to resolve containment relationships in the architecture. The naming network must also make scope and filtering information hierarchies visible, in order to allow name resolution as a mechanism for scoping and filtering of information.

However, name binding is only one way to make information object instance relationships visible. A relationship hierarchy can also be implemented using pointers or via an additional relationship object.

- A name binding is appropriate for visualizing 1:n relationships which are not dynamic in nature as the named information object instances are existence dependent of the naming object.

- The use of pointers allows a flexible dynamic representation of relationships at the expense of less efficient navigation of the relationships.
- The use of a relationship object allows other attributes of the relationship to be captured, and can allow for dynamic changes of relationships at the expense of increasing the number of information object instances and lowering the systems performance.

It is a specification decision as to whether a particular relationship in the architecture is made visible as name binding, pointers or via a relationship object.

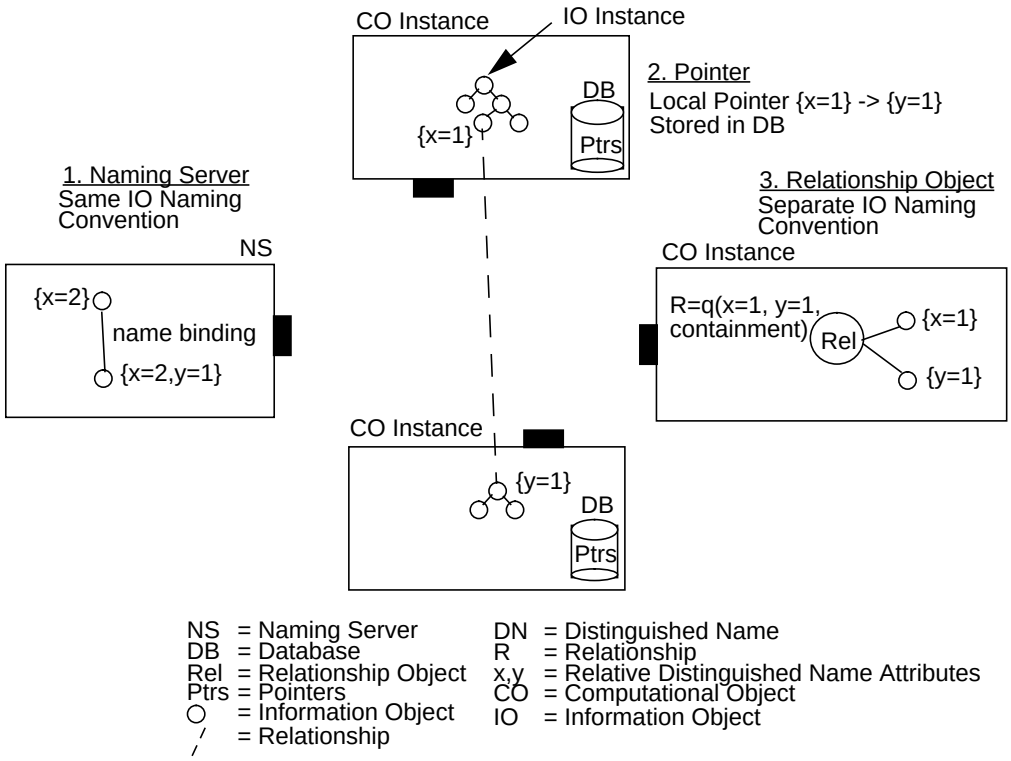


Figure C-2. Three Alternatives for Maintaining an Information Object Instance Relationship Hierarchy.

The following examples give an indication of the specification decisions to be taken with regards to naming in the network resource architecture:

- Entities which are independent of the subnetwork partitioning structure, and are only existence dependent on the layer network for their existence, such as trail and network trail termination point, should be named by the layer network domain.

- Subnetworks could be related by naming. Relating subnetworks by naming is very effective for partitioned subnetworks, but gives restrictions when it comes to reconfiguring subnetwork structures.
- Network termination point pools should have a pointer relationship to network termination points. This structure allows network termination point pools to be flexibly formed and reformed dependent on how the managing system wishes to group network termination points. If a name binding were to be used, the network termination points would need to be deleted if the network termination point pool were to be changed.

C.3 Information Viewpoint

The following chapter defines naming systems for information objects that shall be applied in the network resource architecture based on the naming model defined in [17]. It also gives examples of how to utilize the naming framework in the network resource architecture.

The naming domain for information entities is decomposed into subdomains for information object instances and types respectively. The name sets are divided into unambiguous identifiers and nick names respectively. The proposed naming systems are shown in Figure C-3.

Naming Domain / Name Set	Information Object Type	Information Object Instance
Identifier	Object Identifier	Relative Distinguished Name/Distinguished Name
Nick Name	Object Type Label	Attribute Names

Figure C-3. Naming Systems Defined for Information Objects.

- Object Identifier: Used as unique object template identifier when storing type descriptions in type repository;
- Object Type Label: Used to denote object types and visualize inheritance relationships between object types within the object template;
- Relative Distinguished Name (RDN): Used as simple, unambiguous names for object instances (including syntax and matching constraint definitions).

Distinguished Name (DN): Used as a qualified unique name of an object instance based on a TINA system wide naming network; and

- Attribute Names: Any other name assigned to the object instance within the object template (including syntax and matching constraint definitions).

The quasi-GDMO/GRM object type template structure is given for reference below:

```
<object-type-label> OBJECT TYPE
  [DERIVED FROM <object-type-label> [,<object-type-label>]*;]
  CHARACTERIZED BY <package-label> PACKAGE
  BEHAVIOUR <behavior-definition-label> BEHAVIOUR DEFINED AS
     “
     [COMMENTS: <informal explanations that may ease the understanding of
     the specification>;]
     [INVARIANT: <invariant properties of the object>;]
     [object-constraints]
     [[attribute-comments][attribute-operations]]*
     [operation-spec]*
     “;
  ATTRIBUTES
     [attribute-declaration [, attribute-declaration]*];
     [rdn-attribute-declaration [, rdn-attribute-declaration]*];
     [dn-attribute-declaration];
  ACTIONS
     [action-declaration [, action-declaration]*];
  NOTIFICATIONS
     [notification-declaration [, notification-declaration]*];

REGISTERED AS <object-identifier>;
```

The attribute names may be expressed in any attribute-declaration and are defined within the scope of the template.

C.3.1 Object Identifier

The <object-identifier> is defined in ITU-T Rec. X.680 [40] and is a globally unique identifier used for registering the object template at a naming authority. The top naming authority is the ISO registration authority. The ISO registration authority subdelegates name set sub-graphs to other organizations. These naming authorities are given names and are used as naming contexts in the naming network. In the lower part of the naming network, naming contexts based on document number, template type, object type, version can be used.

At the moment, TINA-C has not requested and been allocated a separate object identifier subtree from ISO and/or ITU-T. TINA should apply for this by following the registration mechanism outlined in [39] and register TINA standardized information objects under this subtree.

In order to allow localization of object definitions, TINA vendors and operators shall use their own allocated subtrees from ISO/ITU-T.

The naming convention for an object identifier is a value notation given as a non-negative integer [see clause 29 in [40]]. It is recommended that an <object-descriptor> is assigned along with the <object-identifier>. The object descriptor is defined as a human-readable GraphicString [see clause 41 in [40]]. The text is not an unambiguous identification of the object, but identical text for different objects is intended to be uncommon. The <object identifier> shall be encoded in a string as a sequence of integers with periods in-between them.

For objects defined within a TINA-C standard, they should be separated into Network Resource Architecture (NRA) and Service Architecture (SA).

{tinac(1) nra(1)}
{tinac(1) sa(2)}

In TINA, the object identifiers are assigned to information object types, relationship types and role bindings only [10]. Attribute and action types are embedded in the other types. The template types shall be separated by a suffix in the naming network.

{tinac(1) nra(1) objectType (1)}
{tinac(1) nra(1) relType (2)}
{tinac(1) nra(1) roleBinding (3)}

The defined information object templates are given object identifier suffixes and version numbers.

{tinac(1) nra(1) objectType(1) actualType() version1(1)}

C.3.1.1 Example

An example for the first version of the Link Termination Point (LTP) template, specified by the core-team, would be:

{tinac(1) nra(1) objectType(1) ltp(1) version1(1)}

C.3.2 Object Type Label

The `<object-type-label>` is a user-friendly name only valid (used) within the scope of the template. The naming authority is the entity specifying the template. The `<object-type-label>` defines a type name which is locally unambiguous within the context of the `<object-type-label>` that the object type is derived from, its naming context.

The naming network is based on the inheritance relationship between objects types. An object type that is specialized from another object type is known as a subtype of that type (its supertype). The supertype name is added in the derived from clause of the object template.

One object type, called top, is designated as the ultimate supertype in the type hierarchy. Top is an abstract object type, i.e. uninstantiable. The type label of top is defined as being the null sequence. Every object type is a subclass of top. Top may be omitted in the derived

from clause, when defining a subordinate object type. In all other cases the derived from clause is mandatory. The kind of an object type (abstract or concrete) should be specified as object constraints in the object template.

The `<object-type-label>` naming system is also used for relationship types and role bindings in TINA quasi-GDMO/GRM.

The naming convention (syntax) is defined within the scope of the object template and is not prescribed in this document. An informal notation for delimiters of type labels in a type hierarchy is "::", as for CORBA type identifiers. In case of multiple inheritance, the super-types are separated by ",".

C.3.2.1 Example

The following is an example of a type naming network based on inheritance relationships within the network resource architecture.

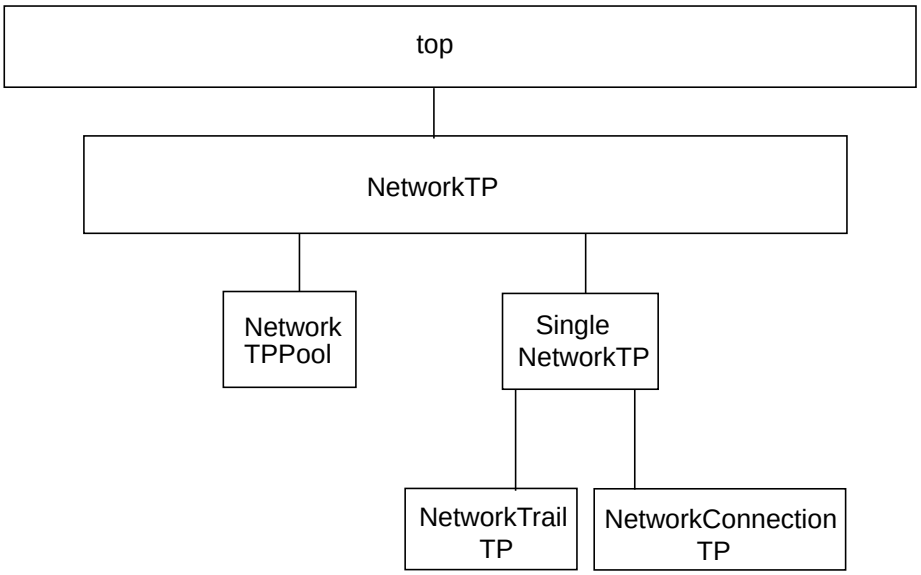


Figure C-4. Partial Example of Type Naming Network based on Inheritance Relationship in the Network Resource Architecture.

The tree can be further specialized into technology specific NWTPs, e.g.:

```
{::NWTP::Single_NWTP::NWCTP::NWCTP_ATM_VP}
```

C.3.3 Distinguished Name

C.3.3.1 Relative Distinguished Name Attribute Declaration

Each information object instance is identified within a naming context by means of an attribute value assertion (AVA) that at least one mandatory attribute has a specified value. When used for naming in this way, the attribute value assertion is called a Relative Distinguished Name (RDN), and must have the property of unambiguously identifying a single information object instance within the scope of the naming context. The RDN shall be location independent.

In the network resource architecture, naming context is based on the containment relationship between information object instances. An information object instance is contained within one and only one containing information object instance. The containing information object instance may in turn be contained in another information object instance.

The naming convention for RDNs is given, using the following attribute declaration.

```
rdn-attribute-declaration-> <rdn-attribute>
                            [PERMITTED VALUES: <attribute-syntax>]
                            [EQUALITY MATCHING CONSTRAINTS: <matching-
                            rule>]
                            [ORDERING MATCHING CONSTRAINTS: <matching-
                            rule>]
                            [SUBSTRINGS MATCHING CONSTRAINTS: <matching-
                            rule>]
                            [GET]
```

Several `<rdn-attribute-declaration>` may be nested to allow several fields with different syntax in one RDN.

C.3.3.1.1. Attribute Syntax

The `<attribute-syntax>` gives the naming convention for the RDN/DN. Specifying an `<attribute-syntax>` shall be done by directly specifying the ASN.1 data type from the Attribute-ASN1Mobule [40]. Initially, the character string type GeneralString[1] is recommended without any restrictions on length. Specifically, the notation of an attribute used for the relative distinguished name shall not be one of the following ASN.1 types:

- the real-type;
- the set-type;
- the set-of-type;
- the any type;
- a choice type that includes any of the above types as a choice;

1. The GeneralString type covers the US-ASCII character set. The character set may later be extended to ISO LATIN-1 if needed, and still be covered by the current OMG IDL string definition. Such an extension may, however, not be supported by all operating systems used by TINA DPEs.

- a type derived from any of the above types by tagging, by subtyping or by use of selection type.

C.3.3.1.2. Matching Constraints

Matching constraints are used to evaluate attribute value assertions of attributes. For relative distinguished names, only the equality matching constraint shall be applied. Several matching rules can be used for naming, e.g. caseIgnoreMatch and caseExactMatch (see [38]). The matching rule used for RDNs in the network resource architecture shall be caseExactMatch.

C.3.3.1.3. Valid Operations

One operation (Get) is defined for the <rdn-attribute>, in order to allow name queries. The <rdn-attribute> value cannot be changed after creation. If get-replace operations on the RDN is required for a specific information object type, this shall be declared as an explicit action in the specification taking into account the effects on the subclasses of the information object. When an information object instance is deleted, the value assigned to its naming attribute becomes available for reuse to identify subsequent information object instances created within the same superior naming context.

C.3.3.1.4. Example

The following is a definition of one attribute declaration used for RDNs in the network resource architecture:

```
Attributes
    commonRdnIdentifier
         PERMITTED VALUES: GeneralString
         EQUALITY MATCHING CONSTRAINTS: caseExactMatch
         GET,
```

The <rdn-attribute declaration> is inherited into all subordinate information objects. In the network resource architecture, the <commonRdnIdentifier> can take on values ranging from VPI values for network connection termination point instances to general descriptive names for naming domain instances.

The definition of other <rdn-attribute>-<attribute-syntax> pairs may be defined and used as relative distinguished names. This is achieved by overloading <rdn-attribute> from superior to subordinate information object type. As overloading is allowed in the information model, but not supported by the computational model (and ODL), it is up to the component specification to define the mapping.

C.3.3.2 Distinguished Name Attribute Declaration

The Distinguished Name (DN) is an unambiguous name of an information object instance formed from the sequence of the RDNs of the information object instance and each of its superior naming contexts. Cyclic naming networks are not allowed for information object instances.

The network resource architecture uses containment for establishing the naming network. This means that the creating information object instance always has knowledge of all of the superior RDNs and can, therefore, form the DN for the created information object instance by appending the RDN of the new information object to its own distinguished name. The <dn-attribute-declaration> can, therefore, be embedded within the information object type template.

The naming convention for distinguished names is given using the following attribute declaration.

```
dn-attribute-declaration-> <dn-attribute>
                           [PERMITTED VALUES: sequence <rdn-attribute>]
                           [EQUALITY MATCHING CONSTRAINTS: <matching-
                           rule>]
                           [ORDERING MATCHING CONSTRAINTS: <matching-
                           rule>]
                           [SUBSTRINGS MATCHING CONSTRAINTS: <matching-
                           rule>]
                           [GET]
```

The <dn-attribute-declaration> uses the equality matching constraint only, with a matching rule: distinguishedName. This means that a presented distinguished name value matches a target distinguished name value if and only if all of the following are true:

- the number of RDNs in each is the same;
- corresponding RDNs have the same number of attribute value assertions;
- corresponding attribute value assertions (i.e. those in corresponding RDNs and with identical attribute types) have attribute values which match for equality (in such a match, the attribute values take the same roles - i.e. as presented or target value - as the distinguished name which contains them in the overall match).

C.3.3.2.1. Naming Network

A top naming context is defined with the <rdn-attribute> being the null sequence. Every name is within this naming context.

Naming subdomains can be identified, named (agreed among TINA member companies) and added as part of a naming context (subordinate <rdn-attribute>) to separate instance naming of different object types or DPE domains based on object administration.

If several naming authorities are associated to one naming domain, a naming authority identifier can be added as part of the name of a naming context. The underlying naming network is provided by the subgraph naming authority.

If support of multiple naming conventions within one DN naming network is required, an additional naming convention identifier and version number is recommended to be included in the beginning of the subgraph where the new naming convention and/or version is needed. The version and convension shall be introduced as additional intermediate `<rdn-attribute>`, e.g:

```
{../namingConvention=IETF_URL/version=RFC1738/}
```

written in an informal notation where {} equals sequence bounds, / is an RDN delimiter, and attribute value assertions are denoted as `attribute id = attribute value`.

This approach is favored compared to having version handling for each individual information object instance in the subtree.

C.3.3.2.2. Example

The following figure gives an example of an instance naming network in the network resource architecture.

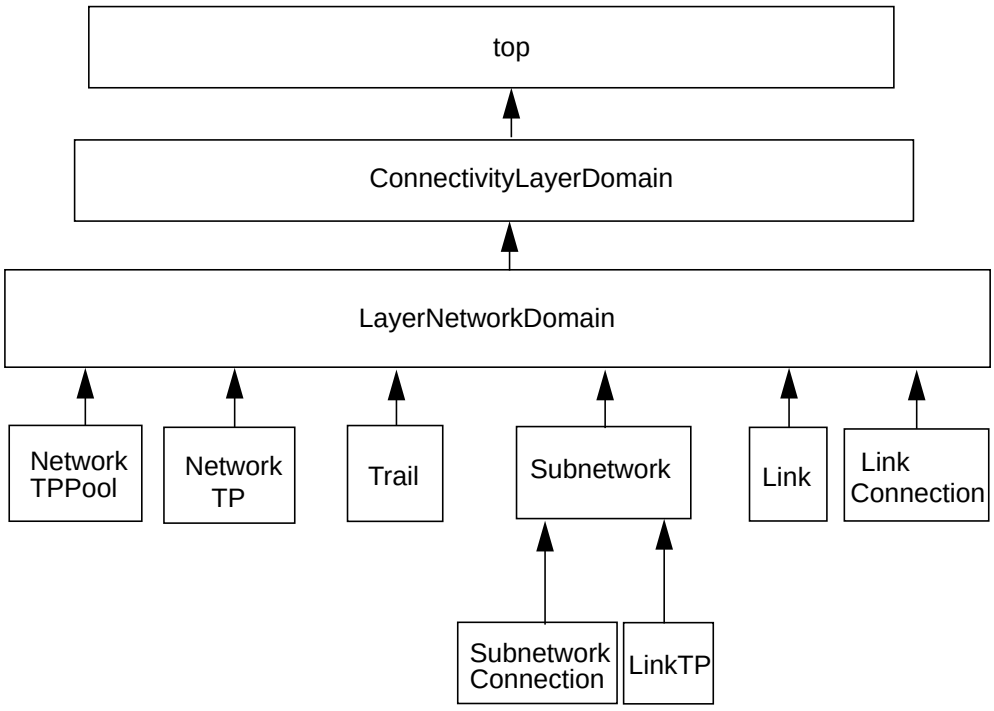


Figure C-5. Partial Example of Instance Naming Network based on Containment Relationship in the Network Resource Architecture.

The following is an example of a <dn-attribute-declaration> for the network resource architecture:

```
Attributes
    commonDnIdentifier
         PERMITTED VALUES: SEQUENCE OF GeneralString
         EQUALITY MATCHING CONSTRAINTS: distinguishedNameMatch
         GET,
```

The same remarks regarding overloading in Section C.3.3.1.4 applies for <dn-attribute> as well.

C.3.4 Attribute Name

Attribute names are any other names used to denote the information object apart from its RDN/DN. The same attribute declaration can be used for attribute names as for RDN/DNs. However, it is up to the application to define which <attribute-syntax> and <matching constraint> to apply for the attribute name.

C.3.5 String Representation Syntax

A string representation is needed when a distinguished name is communicated between two objects. Several attribute string syntaxes can be defined. A common attribute syntax to be used for RDN/DNs in The TINA Trial is defined in [17].

C.4 Computational Viewpoint

C.4.1 Computational Entities

The naming system of computational entities follows the conventions defined in the TINA Naming Framework [17]. No additional conventions are applied to the network resource architecture.

C.4.2 Representing Information Object Names in the Computational View

Multiple information object instance names are possible to represent within the same computational object interface. An information object instance name can either be expressed as an attribute or as an argument in an operation. If an information object <rdn-attribute> is directly mapped to a CO_interface_type attribute, the component specifier should take care of any overloading of <rdn-attribute> made in the information model.

```
//ODL
interface CO_interface_type {
   readonly attribute Name n;

   void operation_type (
       in Name n
   )
};
```

The `operation_type` represents an operation type of the information object assigned to the computational object interface.

C.4.3 Name Resolution Mechanisms

It is recommended that a combination of a naming service and local mapping (as defined in [17]) is used for name resolution in the network resource architecture.

Appendix D: Access session and the NRA

The use of access session in the NRA requires further study. The access session is conceived as setting up a secure link between different stakeholder domains, allowing one to access the services of the other. In NRA, this could relate to interactions between retailer or service provider domains and the connectivity domains, between consumer and connectivity provider domains, and between separate connectivity domains. Note that uses of access session could vary on the reference point in question.

Currently the NRA focuses on usage aspects of the CMA, using FCAPS to scope the areas of interest. Less emphasis has been placed on how these various management services work together. Access session could be used as a way of coordinating the various management services.

When considering access session, careful thought needs to be given to how it is appropriate use it. In particular, how involved should access session be in the use of particular management services. In the example below, a decision was made that the setup of connectivity sessions should be classified as part of the usage of the connection management service, rather than requiring involvement of the access session.

D.1 Example: How to use access session in the ConSRP

In the core-team response to the ConS reference point, the access session is conceived as usually controlling various contractual and longer term management relationships. These management relationships are supported by various services, (such as our “fast connection setup service”) and need to allow this interface to handle large volume requests that usually require fast response, with out usually requiring much negotiation.

For instance: A connectivity user wants to acquire interfaces (i.e. setup) to some instances of a “fast connection setup service” (represented by the CCF object) that can supply connections that match an agreed management context, which sets charging levels, fault management and so on. All services in this scenario are outlined as examples of how access session could work and relate to various services offered over the ConSRP. Note the negotiation is optional, and allows a Connectivity User (CU) to setup a number of previously negotiated management guarantees for the connections it uses.

1. A CU management service locates the Provider Agent (PA) through which it can contact the CP it wishes to use
2. (optional) The CU and CP negotiates a management context
   - 2.1 The CU management service invokes a context negotiation service via the PA
   - 2.2 The PA passes the request on to the UA in the CP’s domain
   - 2.3 The UA in the CP creates the negotiation service
   - 2.4 The CU and CP negotiate an acceptable context for connectivity usage, and save that context
   - 2.5 The context negotiation service is quit

3. The CU management requests a fast connection setup service that uses an agreed (or default) connectivity management context via the PA
4. The PA passes the request on to the UA in the CP's domain
5. The UA in the CP creates (or grants access to) the "fast connection setup" service
6. The UA returns the interface(s) to this service to the CU management service (via the PA)

While this could be complex, but it is not carried out as part of standard connection setup. The standard connection setup uses a fast connection setup service to establish connections.

References

TINA-C

[1] *Network Resource Information Model,* Document No. TB_LRS.011_2.0_94 TINA-C, December 1994.

[2] *Connection Management Specifications*, Document No. TP_NAD.001_0.3_95, TINA-C, March 1995

[3] *Overall Concepts and Principles of TINA*, Document No. TB_MDC.018_1.0_94, TINA-C, December 1994.

[4] *Computational Modelling Concepts*, Document No. TB_NAT.002_3.1_94, TINA-C, December 1994.

[5] *Computational Modelling Concepts*, Document No. TP_HC.012_3.0_95, TINA-C, Aug. 1995

[6] E*ngineering Modelling Concepts (DPE Architecture)*, Document No. TB_NS.005_2.0_94, TINA-C, December 1994.

[7] *Management Architecture*, Document No. TB_GN.010_2.0_94, TINA-C, December 1994

[8] *Resource Configuration Architecture*, Document No. TB_C3.AMB.005_1.0_93, TINA-C, December 1993.

[9] *Definition of Service Architecture*, Document No. TB_MDC.012_2.0_94, TINA-C, December 1994.

[10] *Information Modelling Concepts*, Document Number TB_EAC.001_3.0_94, TINA Consortium, December 1994.

[11] *Analysis of Existing Architectures via Service Examples*, Document No. TB_B.SMO.001_1.0_93, December 1993.

[12] *TINA-C Documentation Roadmap*, Document No. TB_G.RM.003_1.0_93, December 1993.

[13] *Connection Management Architecture*, No. TB_JJB.005_1.5_94, TINA-C, Mar. 1995.

[14] *Connectivity Layers & Connection Graphs,* No. EN_JP.011_2.1_95, TINA-C, Sept. 1995.

[15] Unification of Connection/Session Graphs, Stream Interfaces and Channel Model, No. TR_PL.001_1.3_95, TINA-C, Apr. 1996

[16] *TINA Reference Points*, Version 3.1, TINA-C, Jun. 1996

[17] *TINA Naming Framework*, Ver 0.1, TINA-C, December 1996.

[18] Fault Management Architecture, Document No. TB_C3.DL.002_1.0_93, TINA-C, December, 1993.

[19] Management Principles Architecture, TINA-C, Mar. 1997.

[20] Service Architecture Ver 4.0, TINA-C, Nov. 1996.

[21] Cons-RP RFR/S response, TINA-C core-team, Nov. 1996

[22] Tcon-RP RFR/S response, TINA-C core-team, Nov. 1996

[23] Ret-RP RFR/S response, TINA-C core-team, Nov. 1996

[24] Cons-RP RFR/S unified response, TINA-C review panel, Jan. 1997

INA

[25] Bellcore SR-NWT-002660, *INA Cycle 1 Communications Management Architecture,* Issue 1, April 1993.

[26] *INA Cycle 1 Service Management Architecture (Issue 1)*, No. TM-NWT-020240, Bellcore, Dec. 1991.

[27] *INA Cycle 1 Service Management Architecture (Issue 2)*, No. TM-TSV-021902, Bellcore, Dec. 1992.

ITU-TS

[28] CCITT Draft Recommendation G.803, *Architectures of Transport Networks Based on the Synchronous Digital Hierarchy (SDH),* 1992.

[29] CCITT Draft Recommendation M.3100, *Generic Network Information Model*, 1992

[30] CCITT Draft Recommendation G.774, *Synchronous Digital Hierarchy (SDH) Management Information Model,* November 1991.

[31] ITU-T Draft Recommendation Q.2761: *Broadband Integrated Services Digital Network User Part (BISUP)*, 1993

[32] ITU-T Recommendation Q.2762: *General Functions of Messages and Signals*, 1993

[33] ITU-T Recommendation Q.2763: *B-ISDN User Part Formats and Codes*, 1993

[34] ITU-T Recommendation BQ.764: *Signalling Procedures*, 1993

[35] ITU-T Draft Recommendation Q.93B: *Digital Subscriber System No. 2 (DSS2)*, 1993

[36] CCITT Recommendation I.311: *B-ISDN General Network Aspects*, 1991

[37] *CS 1 Refinements, Section 3.3 of Q.1218. Output of the Madrid meeting, Intelligent Network Capability Sets*, ITU-T SG 11, Jan. 1995.

[38] ITU-T Recommendation X.520 | ISO/IEC 9594-6, “Information Technology OSI The Directory: Selected Attribute Types”, 1993.

[39] ITU-T Recommendation X.660 | ISO/IEC 9834-1, “Information Technology OSI Procedures for the Operation of OSI Registration Authorities - Part 1: General Procedures”.

[40] ITU-T, Recommendation X.680, Information Technology OSI Abstract Syntax Notation

One (ASN.1) Specification of Basic Notation", July 1994.

[41] ITU-T, Recommendation X.720, "Information Technology OSI Structure of Management Information: Management Information Model", January 1992.

[42] ITU-T, Recommendation X.742, "Information Technology OSI Structure of Management Information: Accounting Metering Function", Feb. 1993.

[43] ITU-T, Recommendation X.721, "Information Technology OSI Structure of Management Information: Structure of Management Information: Management Information Model", 1992.

[44] ITU-T, Recommendation X.734, "Information Technology OSI Structure of Management Information: Event Report Management Function", 1992.

[45] ITU-T, Recommendation X.735, "Information Technology OSI Structure of Management Information: Log-control Management Function", 1992.

[46] ITU-T, Recommendation X.745, *Information Technology - Open Systems Interconnection - Systems Management - Part 12: Test Management Function,* COM VII-224-E, 1992.

[47] ITU-T, Recommendation X.903: *Basic Reference Model of Open Distributed Processing - Part 3: Prescriptive Model,* 1993.

ODP

[48] CCITT Recommendation X.901, *Basic Reference Model of Open Distributed Processing - Part 1: Overview and Guide to Use,* June 93.

Eurescom

[49] P103, Project Internal Report 2.3 'Network Resource Model'.

[50] P108 final report, 'TMN prestudy', January 1992

[51] P203 deliverable D4, 'Organisational Models for management of European wide services'.

Other

[52] Network Management Forum, *Transmission Resource Management - Reconfigurable Circuit Service Ensemble - Issue 1.0*, 1992

[53] Bellcore, TA-NWT-001114, Draft Issue 1, *Generic Requirements for Operations Interfaces Using OSI Tools: Broadband ATM Network Operations*, December 1992.

[54] Bellcore, GR-1110-CORE, Issue 1, *Broadband Switching System (BSS) Generic Requirements*, Sept. 1994.

[55] J.M. Cornilly et al., "Experimenting with ODP Concepts for SDH Transmission Network Management Specification", *TINA'93 - The Fourth Telecommunications Information Networking Architecture Workshop* (L'Aquila Italy: September 27-30, 1993)

Proceedings Vol 1 pp 67 - 86.

[56] Daniel Minoli, George Dobrowski, Principles of Signaling for Cell Relay and Frame Relay, 1995.

[57] W. Stallings, *SNMP, SNMP V2 and CMIP: The Practical Guide to Network-Management Standards*, Addison-Wesley, Reading, Mass. USA, 1993.

[58] Andrew Bridge, Lars Richter, “An Object-Oriented Design Process for the TINA Management Architecture,” *TINA’93 - The Fourth Telecommunications Information Networking Architecture Workshop*, Proceedings Vol I pp 103-120, L’Aquila Italy, September 27-30, 1993.

Accounting

[59] *A Proposal for Accounting Management*, Document No. EN_MMH.001_0.9.1_94, TINA-C, Sept. 1994.

[60] *Definition of Service Architecture*, No. TA_MDC.012_2.0_94, TINA-C, Dec. 1994.

[61] *Service Component Specifications*, No. TA_HK.002_1.0_94, TINA-C, Dec. 1994.

[62] *Service Management*, No. TP_JS.003_0.4_95, TINA-C, Aug. 1995.

[63] *Specification of Multimedia Conferencing Services based on the TINA Service Life-Cycle model*, P. Leydekkers, P. Leever, A. Melisse, PTT Research NL, Oct. 1994.

[64] *Information technology - Open Systems Interconnection - Systems management - Part 10: Accounting metering function,* Draft International Standard X.742 | ISO/IEC DIS 10164-10, Feb. 1993.

[65] *Chapter 10, CFS H408 Accounting Management*, NETMAN Deliverable 11.

[66] *Network and Distributed Systems Management ed. by M. Sloman, Chap. 14 User Administration and Accounting*, B. Varley, Addison-Wesley, 1992.

[67] *Domain types and basic Reference Points in TINA*, No. EN_RIC.020_0.5_95, TINA-C, May 1995.

[68] *Session Control Specifications*, No. TR_MJM.005_1.0_95, TINA-C, Sept. 1995.

[69] *Agent concepts in TINA*, No. EN_JS.001_1.0_95, TINA-C, Sept. 1995.

[70] *The QOS Broker*, K. Nahrstedt and J. Smith, IEEE Multimedia, pp. 53-67, Spring 1995.

Acronyms

- ATM: Asynchronous Transfer Mode
- B-ISDN: Broadband Integrated Services Digital Network
- BISUP: Broadband ISDN User Part
- CC: Connection Coordinator
- CG: Connection Graph
- CGBO: ConnectionGraph Binding Object
- CM: Connection Management
- CMA: Connection Management Architecture
- CMIP: Common Management Information Protocol
- CMISE: Common Management Information Service Element
- CMT: Connection Management Team
- CP: Connection Performer
- CPE: Customer Premises Equipment
- CSM: Communication Session Manager
- CTP: Connection Termination Point
- CU: Connectivity User
- DPE: Distributed Processing Environment
- FC: Flow Connection
- FCBranch: Flow Connection Branch
- FEP: Flow End Point
- EML: Element Management Layer
- ETSI: European Telecommunication Standards Institute
- INA: Information Networking Architecture
- ITU-TSS: International Telecommunication Union - Telecommunication Standardization Sector
- KTN/kTN: Kernel Transport Network
- LCG: Logical Connection Graph
- LNC: Layer Network Coordinator
- LNW: Layer NetWork
- LTP: Link Termination Point
- MSC: Management Service Component
- MO: Managed Object
- NCG: Nodal Connection Graph

- NE: Network Element
- NFC: Network Flow Connection
- NFCBranch: Network Flow Connection Branch
- NFEP: Network Flow End Point
- N-ISDN: Narrowband Integrated Services Digital Network
- NNI: Network Network Interface
- NML: Network Management Layer
- NRIM: Network Resource Information Model
- NWCTP: Network Connection Termination Point
- NWTP: Network Termination Point
- NWTTP: Network Trail Termination Point
- NWTpPool: Network Termination Point Pool
- ODP: Open Distributed Processing
- PCG: Physical Connection Graph
- POTS: Plain Old Telephony Service
- QoS: Quality of Service
- RC: Resource Configuration
- RFEP: Resource Flow End Point
- SDH: Synchronous Digital Hierarchy
- SFC: Stream Flow Connection
- SFCBranch: Stream Flow Connection Branch
- SFEP: Stream Flow End Point
- SI: Stream Interface
- SML: Service Management Layer
- SNC: Subnetwork Connection
- SNW : Subnetwork
- SR: Special Resource
- SSM: Service Session Manager
- TFC: Terminal Flow Connection
- TINA: Telecommunications Information Networking Architecture
- TINA-C: Telecommunications Information Networking Architecture Consortium
- TMN: Telecommunications Management Network
- TPPool: Termination Point Pool

- TTP: Trail Termination Point
- UNI: User Network Interface
- VC: Virtual Circuit
- VP: Virtual Path

Glossary

- **Action (2)[1]:** An operation on a managed object, the semantics of which are defined as part of the managed object class definition.
- **Agent:** Functions accessible through a management interface that enable access to, operations on, and notifications from a collection of managed objects associated with the interface.
- **Application**: A software product with a well-defined functionality. In the context of service architecture, this term means a software product or object which provides a telecommunication service.
- **Attribute**: Information of a particular type concerning an object.
- **Behavior (2):** (Of a managed object:) The way in which managed objects, name bindings, attributes, notifications and actions interact with the actual resources they model and with each other.
- **Binding Object**: A computational object that represents a binding; provides operations for controlling the binding; encapsulates the mechanisms required for controlling the binding.
- **Channel:** The engineering realization of a computational binding. Composed of stub, binder and protocol objects.
- **Communication Session Manager:** A computational object in the connection management functional area. It provides clients with the service of interconnection of computational stream interfaces.
- **Computational Interface**: An abstraction that provides access to a subset of capabilities provided by a computational object.
- **Computational Object:** An abstraction that encapsulates data and processing; provides a set of capabilities that can be used by other objects
- **Connection:** A basic abstract concept in most communication models. A connection is an association between two or more end points which is used to convey information between these end points. The term connection can be used recursively, so connections are usually composed of sub-connections, each of which can be managed independently. Examples of end points are sinks/sources in stream interfaces, sink/source ports in a connectiongraph and termination points of a subnetwork. The term “connection” should be used with other clarifying adjectives in order to have a specific meaning.
- **Connection Coordinator:** A computational object in the connection management functional area. It provides clients with the service of interconnection of addressable termination points, multipoint-to-multipoint bidirectional. It hides from clients the concepts of layering and partitioning of transmission networks. The interface specification is based on the connectiongraph concept.

1. The (2) means here that in TINA-C there are two definitions used for this term. See the TINA-C global glossary for the definition of Action (1). Only Action (2) is used in the Connection Management Architecture deliverable.

- **Connectiongraph:** An object type used in the computational interface specification of a Communication Session Manager and a Connection Coordinator. It is used as a container object for other object classes to model transport abstractions.
- **Connection Management:** One of the six TINA-C network management functional areas. Functions in this category are responsible for establishing, modifying and releasing connections in response to client requests.
- **Connection Management Configurator:** A computational object in the connection management functional area. The Connection Managment Configurator provides an interface to Resource Configuration functions that must configure Connection Management functions (such as a Communication Session Manager, a Connection Coordinator, a Connection Performer) that are co-located in one building block.
- **Connection Performer:** A computational object in the connection management functional area. It provides clients with the service of interconnecting termination points of a subnetwork. Every subnetwork is managed by one Connection Performer.
- **CorrelationId:** Identifies the mapping between a SFC and a NFC.
- **Distributed Processing Environment:** The Distributed Processing Environment (DPE) provides the infrastructure for computationally specified applications on top of Native Computation and Communication Environments (NCCEs). It enables the interworking of these applications residing on different, possibly heterogeneous, NCCEs in a distribution transparent way. The DPE consists of a DPE runtime and a DPE development system.
- **Edge:** A Managed Object that represents association between a subnetwork-Connection and a NWCTP or a NWTTP.
- **Element Management Layer:** A sublayer of resource management functions defined in TMN standards that consists of functions that manage individual network elements or subsets of network elements (which may be viewed by network management layer functions as subnetworks).
- **End User:** A stakeholder role who utilizes a telecommunications service, provided by a service provider and subscribed by a subscriber.
- **Federation:** An organizational structure involving two or more autonomous administrations in which the member administrations have an agreement on how they will interwork with each other including the extent to which the resources of one member can be shared by other members.
- **Flow Connection (FC):** Abstract class that models transport between flow end points.
- **Flow End Point (FEP):** Abstract class that models the termination of a flow connection.
- **Functional Area (1):** A task-specific grouping of required network management functions. The OSI defines five management functional areas. The TINA-C architecture defines six TINA functional areas by dividing the OSI Configura-

tion Management functional area into Resource Configuration and Connection Management. The six TINA functional areas are: Accounting Management, Connection Management, Fault Management, Performance Management, Resource Configuration Management, and Security Management.

- **Layer Network:** A set of transport functions which support the transfer of information of a characteristic type. Generally, a layer network is closely tied to a specific type of network transmission and/or switching technology, e.g., SDH/ SONET VC-4, ATM virtual channel (ATM VC) or ATM virtual path (ATM VP).
- **Layer Network Coordinator:** A computational object responsible for providing trails in a layer network. It is associated with one domain in the layer network and federates with other Layer Network Coordinators to provide a trail across domain boundaries.
- **Link Connection:** An connectivity which runs between a pair of Subnetwork.
- **Link Termination Point:** A termination point of a Link.
- **Managed Object:** An abstract representation of a resource that can be supervised and controlled by other objects.
- **Network Element Layer:** The category of functions defined in TMN standards that are linked to the technology or architecture of the network resources that provide the basic telecommunications services. These functions may be accessed by the element management layer functions using standard or open information specifications that may hide vendor-specific functions within network resources.
- **Network Flow Connection (NFC):** A point-to-point uni, point-to-point bi- or a point-to-multipoint uni-directional flow connection between network flow end points.
- **Network Flow End Point (NFEP):** A network level termination point of a network flow connection, that is always related to a network termination point.
- **Network Flow End Point pool (NFEPpool):** An aggregation of resource flow end points.
- **Network Management Layer:** A sublayer of network resource management functions defined in TMN standards that have the responsibility for the management of all the network elements, as presented by the element management layer. It is not concerned with how a particular network element provides service internally. Complete visibility of the whole network is typical, and a vendor independent view will need to be maintained. The functions in this layer interact with the service management layer on end-to-end connections, performance, faults, etc. across the network.
- **Notification:** A management operation initiated by a managed object for the purpose of communicating the occurrence of some significant event within the managed object.
- **Q.93B:** The Q.931 signalling protocol defined for narrowband ISDN, but modified for use in B-ISDN (broadband ISDN).

- **Resource Flow End Point (RFEP):** Abstract class from which network flow end points and network flow end point pools are derived.
- **Router:** A computational object in the connection management functional area. It provides a list of possible paths between two end points of a network.
- **Server**: Defined relative to an operational interface; the object that provides an operational interface is the server of the interface. (See “Client”)
- **Service Management Layer:** The category of functions defined in the TMN standards that provide end-user service specific functions including service logic and service management.
- **Stream Flow Connection (SFC):** A uni-directional point-to-point or point-to-multi-point flow connection between stream flow end points.
- **Stream Flow End Point**: A uni-directional (source or sink) application level end point of a stream flow connection, which is always aggregated in a stream interface
- **Stream Interface**: An abstraction that represents a communication endpoint that may be a source for some stream flows and a sink for some stream flows.
- **Subclass (1):** One class is a subclass of another class precisely when it is a subset of the other class. (See “Superclass (1)”)
- **Subclass (2):** A object class which inherits the template of another class. (See “Superclass (2)”)
- **Subnetwork:** A subset of the network resources such that the resources, having common operations properties (e.g., manufacturer, common function, or common geographical location) cooperate to support some aspect or portion of one or more telecommunications services. It may contain resources of different suppliers, and may consist of several nodes that are operated as a cohesive entity. In the context of Connection Management the subnetwork is used as a topological component to effect routing and management. It can be partitioned into interconnected subnetworks and connections.
- **Subnetwork Connection:** A transport entity formed by a connection across a subnetwork between termination points.
- **Subordinate:** A Managed Object instance which is placed below its Superior Managed Object in the Containment relationship tree.
- **Superclass (1):** One class is a superclass of another class precisely when the other class is a subset of it. (See “Subclass (1)”.)
- **Superclass (2):** A object class whose template is inherited by another class. (See “Subclass (2)”.)
- **Superior:** A Managed Object instance which is placed above its Subordinate Managed Object.
- **Terminal Agent:** An object which represents a terminal attached to a network, or is accessible from a network (such as a mobile phone).

- **Terminal Flow Connection:** A point-to-point uni-directional flow connection between a stream flow end point and another stream flow end point or a network flow endpoint.
- **Topological Link:** Collection of Link Connections which are served by a trail of a server layer network.
- **TP Pool:** A collection of termination points that is used for some management purpose such as routing.
- **Trail:** A transport entity which spans across a Layer Network.
- **Trail Termination Point:** A termination point of a Trail.