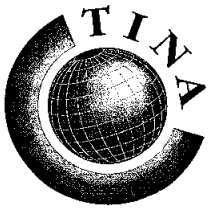

Telecommunications
Information
Networking
Architecture
Consortium

TINA-C Request for Refinements and Solutions

Issue Status: Publicly available

Unified response to Request for Refinements and Solutions for

The TCon Reference Point

Version:1.0

Abstract: This document contains the specification of the Terminal Connection (TCon) Reference Point, that is a result of unifing the two responses (Ericsson and core-team) to the initial RFR/S. This specification includes a detailed requirements specification, an information specification and a computational specification, including definitions of objects and their interfaces in ODL, and event traces.

Affiliation: **TINA-C Core Team**

Contact person: **Frank Steegmans**

Novermber 14, 1996

1. Context of the answer

1.1 Areas of non-compliance

This specification of TCon is fully compliant with the TINA architecture.

1.2 Inter-relationships with other reference points

The TCon reference point does not require any other TINA reference points to be implemented. However, there is an interrelation with the Ret and ConS reference point in case those are implemented as well. This interrelationship is described in the following two subsections.

1.2.1 Inter-relationship with ConS

The scope of ConS is the management of network flow connections. The end-points of these network flow connections are managed through TCon. Manipulations of network flow connections at ConS will require/result in manipulations of network flow endpoints at TCon. For instance, the set-up of a network flow connection at ConS will result in the set-up of a network flow endpoint at TCon.

1.2.2 Inter-relationship with Ret

The Ret reference point has many purposes. One is related to the local (within the consumer's domain) binding between a Stream Flow EndPoint (SFEP) and a Network Flow EndPoint (NFEP), the latter being managed at TCon. Actions at Ret and actions at TCon have to be related. TCon takes this into account, and introduces a special parameter for this purpose.

1.3 Main assumptions

The inter-object communication across the TCon reference point may be supported by a DPE. Possibly other ways may be envisaged as well, but these have not been investigated for this version of TCon.

1.4 Project/prototyping experience

The specification of TCon in this document will be validated by the Core Team.

2. Additional Business Requirements

2.1 Overall functionality and scope of the reference point:

The goal of the Terminal Connectivity (TCon) reference point is to provide functions for the management of network flow endpoints (NFEPs), the endpoints of network flow connections[1]. This happens in a co-operation between connectivity provider and "connectivity consumer". The term connectivity consumer is introduced in this document, and clarified as follows:

> **Connectivity consumer**
> The TCon reference point may exist between the administrative domain of a connectivity provider, and the administrative domain of a consumer, retailer or third party service provider[2]. In this document, consumer, retailer and third party service provider are referred to as *connectivity consumer*.

2.1.1 Illustration of concepts

This section, by means of Figure 2-1, illustrates the concepts used in this document. This section is included just for clarification and is not prescriptive.

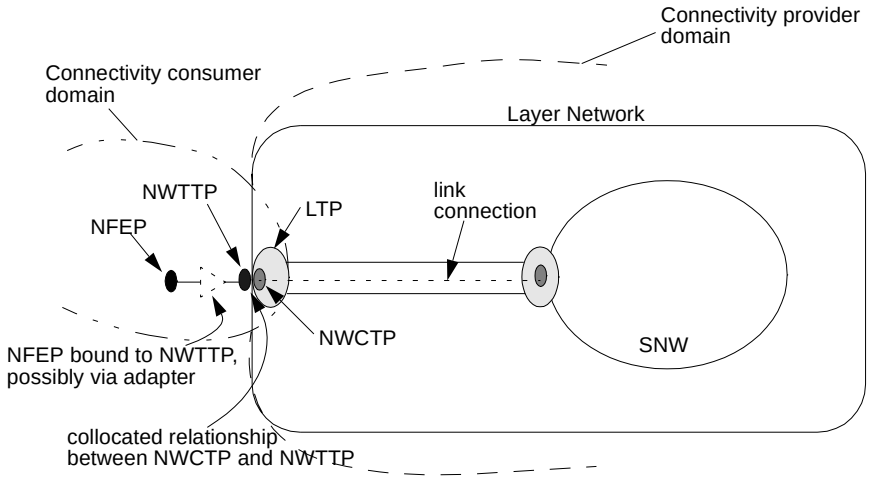

Figure 2-1. Illustration of relationship between NFEP, NWTTP and NWCTP

The scope of the TCon reference point is the manipulation of Network Flow EndPoints (NFEPs) and the Network Trail Termination Points (NWTTPs) and Network Connection Termination Points (NWCTPs) associated with those NFEPs.

The NFEP is the endpoint of a network flow connection that may span multiple layer networks. The manipulation of network flow connections is within the scope of the ConS reference point. Within the layer network where the network flow connection terminates in the

1. The management of network flow connections is in the scope of the ConS reference point, described in [2].
2. "Consumer", "retailer", "third party service provider" and "connectivity provider" should be interpreted as defined in [1].

connectivity consumer's domain, the network flow connection is supported by a trail, the endpoint of which is the Network Trail Termination Point (NWTTP). The NFEP may either be directly associated (bound) to a NWTTP, or indirectly by an adapter.

On the topological link between connectivity provider and connectivity consumer, the trail is supported by a link connection, the endpoint of which is the Network Connection Termination Point (NWCTP). The NWCTP is collocated with the NWTTP.

The connectivity provider and connectivity consumer are connected by means of a topological link, which has Link Termination Points (LTPs) as its endpoint. An LTP contains NWCTPs.

The set-up of a NFEP at TCon includes the creation (or selection in case of pre-provisioned NFEPs) of the NFEP and the creation of the NWTTP. The NWCTP is also created, or in case of a pre-provisioned link, it is selected. As the NWCTP is identified by the link identity and the channel on that link, creation or selection of an NWCTP includes channel selection.

The concepts shown in Figure 2-1 are explained in more detail in NRIM [3].

2.2 Functional requirements

2.2.1 Common requirements:

None identified.

2.2.2 Access requirements:

This section describes the requirements on the access part of TCon. One of the goals of the access part is to enable the setup of a secured binding between connectivity consumer and connectivity provider. It may be possible for this to rely on the setup of a secured binding between two other parties, for instance the retailer and the consumer. This possibility is for further study.

Context modification

TCON-A-1 The connectivity provider shall provide a function for modification of context information, regarding the relationship between connectivity user and connectivity provider (for instance: user preferences).

Security context

TCON-A-2 The security procedure to be used should include the following options:

- Degree of security;
- Degree of trust;
- Authentication checking: authentication can be required on a periodic basis (i.e., time or number of interactions) to reinforce security.

Authentication

TCON-A-3 The connectivity provider and the connectivity consumer shall have the possibility to authenticate each other.

Usage interface request

TCON-A-4 Both the connectivity provider and the connectivity consumer shall provide functions for their counterpart to address usage interface references in order to perform usage part related operations. One implementation of this is for each party to provide functions allowing the explicit request of a usage interface reference by the other party.

2.2.3 Usage requirements:

2.2.3.1 Connection Management Requirements

Connectivity environment

TCON-U-1 The following parameters are used in the requirements:

- *Correlation identifier.* The TCon reference point shall provide a possibility for the connectivity consumer to correlate the actions performed at this reference point with the actions performed at the Ret reference point. More specifically, this correlation shall allow to relate the Network Flow End-Point (NFEP) that is established at TCon level to the Stream Flow End-Point (SFEP) that is established at Ret level.
- *NFEP Name*. This parameter is used to identity a network flow endpoint.
- *NFEPpool Name*. This parameter is used to identity a network flow endpoint pool.
- *NFEPReference*. This parameter identifies an NFEP or an NFEPpool.
- *Type* of the NFEP. The NFEP can either be a *root* endpoint, a *leaf* endpoint, or can be *bi-directional*.
- *NWTTP*. This parameter identifies the NWTTP.
- *NWCTP*. This parameter identifies a NWCTP (this includes identification of the channel used on the link connection between connectivity provider and connectivity consumer).
- *QoS description*. The details of this parameter should be elaborated on a technology specific basis. It will at least consist of the following elements:
  - *Bandwidth description*. How the bandwidth is described depends on the technology. For instance, for ATM *peak bandwidth* and *average bandwidth* could be included (depending on the traffic type).
  - *Delay description.* Again, the details of this element depend on the technology. For ATM, for instance, the *maximum delay* and *maximum variation of delay* are included.
  - *Error rate description.*
  - *Reliability class* which can be one of the following: ReleaseOnFailure, and HoldOnFailure. If the class is ReleaseOnFailure, the NFEP is released when it fails. If the class is HoldOnFailure, the NFEP is not released when it fails, but information flow is suspended

  - *Additional information.* For instance, for ATM the *Traffic type* should be included (Constant Bit Rate (CBR), Variable Bit Rate (VBR), Available Bit Rate (ABR), or Unspecified Bit Rate (UBR)).

- *Initial Administrative State*: this parameter is used to specify whether or not the NFEP should be automatically activated upon setup. This parameter specifies the initial administrative state of the NFEP upon setup, and is either Unlocked or Locked.
- *Application information*: This parameter allows application information to be transferred over the TCon reference point. For instance, this parameter may be used to signal to the connectivity consumer domain to bind the Stream Flow EndPoint (SFEP) immediately upon set-up of the NFEP.

The following failure codes shall be supported:

- *NotAuthenticated*: This failure code is used if authentication is required, but the other end of the communication has not (successfully) performed authentication.
- *NotAuthorised*: This failure code is used if the requester is not authorised to invoke the requested operation.
- *InsufficientResources*: If there is insufficiency of resources.
- *InsufficientBandwidth*: If there is no sufficient bandwidth available.
- *QoSCannotBeMet*: If the requested QoS cannot be provided.
- *NonExistentEndPoint*: If the identified endpoint does not exist.
- *EndpointAlreadyInUse*: If the endpoint is already in use in another context.
- *EndpointAlreadyActive*: If the endpoint is requested to be activated, but is already active.
- *EndpointNotActive*: If the endpoint is requested to be deactivated, but is not active.
- *EquipmentError*: If the request cannot be satisfied for an undefined reason.

NFEP set-up

TCON-U-2 The connectivity consumer shall provide functions to set-up the Network Flow End Point (NFEP). The meaning of this depends on whether the link connection between connectivity provider and connectivity consumer is pre-provisioned. If it is *not* pre-provisioned, set-up of the NFEP means that both the network trail termination point (NWTTP) and the network connection termination point (NWCTP) are created, and attached to each other. The NFEP is then bound to the NWTTP. If it is pre-provisioned, it means that the NWTTP is created, and attached to an existing NWCTP. Again, the NFEP is then bound to the NWTTP. Two cases should be catered for, i.e. either the connectivity provider or the connectivity consumer may choose the NWCTP. Choosing the NWCTP includes selecting the channel, and allocating the bandwidth. Also the NFEP itself may be created upon set-up, or an existing NFEP may be used. Finally, the connectivity provider may set the Initial Administrative State of the NFEP to “Unlocked” or “Locked”.

TCON-U-3 Upon receipt of a NFEP setup request, the connectivity consumer shall first check the validity of the request. If the request is invalid, the connectivity consumer shall send a response to the connectivity provider indicating the invalidity of the request. Note that if a request is invalid, the connectivity consumer shall not setup the NFEP specified.

TCON-U-4 If a NFEP setup request is valid, the connectivity consumer shall setup the NFEP point in accordance with the parameters specified in the request.

TCON-U-5 If the NFEP is setup successfully, the connectivity consumer shall indicate this in its response to the connectivity provider.

TCON-U-6 The connectivity consumer shall report a failure to the connectivity provider in response to a NFEP setup request, if the connectivity consumer is unable to setup the NFEP specified. One of the following failure codes may be returned: NotAuthenticated, NotAuthorised, InsufficientResources, InsufficientBandwidth, QoSCannotBeMet, NonExistentEndPoint, EndpointAlreadyInUse, ResourceError.

NFEP activation

TCON-U-7 The connectivity consumer shall provide a function for changing the administrative state of a NFEP to Unlocked state. Only when the administrative state is Unlocked, the NFEP transports information.

TCON-U-8 Upon receipt of a NFEP activation request, the connectivity consumer shall first check the validity of the request. If the request is invalid, the connectivity consumer shall send a response to the connectivity provider indicating the invalidity of the request.

TCON-U-9 The processing of a NFEP activation request is deemed successful if the connectivity consumer is able to activate all terminal resources that support the NFEP, and the NFEP is ready for information transport.

TCON-U-10 If NFEP activation is successful, the connectivity consumer shall send a response to the connectivity provider indicating that the NFEP has been activated.

TCON-U-11 The connectivity consumer shall report a failure to the connectivity provider in response to a NFEP activation request if the connectivity consumer is unable to activate all terminal resources for which it has the responsibility. In such a case, the connectivity consumer shall send a failure response to the connectivity provider. When an activation request fails, the NFEP remains in Locked state. One of the following failure codes may be returned: NotAuthenticated, NotAuthorised, NonExistentEndPoint, EndpointAlreadyActive, EquipmentError.

NFEP deactivation

TCON-U-12 The connectivity consumer shall provide a function for changing the administrative state of a NFEP to Locked state. When the administrative state is set to Locked, transport of information over the NFEP is suspended.

TCON-U-13 Upon receipt of a NFEP deactivation request, the connectivity consumer shall first check the validity of the request. It the request is invalid, the connectivity consumer shall send a response to the connectivity provider indicating the invalidity of the request.

TCON-U-14 The processing of a NFEP request is deemed successful if the connectivity consumer is able to deactivate all terminal resources that support the trail termination, and the transport of information over the NFEP is suspended.

TCON-U-15 If NFEP point deactivation is successful, the connectivity consumer shall send a response to the connectivity provider indicating that the NFEP has been deactivated.

TCON-U-16 The connectivity consumer shall report a failure to the connectivity provider in response to a NFEP deactivation request if the connectivity consumer is unable to deactivate all network resources that support the NFEP. In such a case, the connectivity consumer shall send a failure response to the connectivity provider. When a deactivation request fails, the NFEP remains in Unlocked state. One of the following failure codes may be returned: NotAuthenticated, NotAuthorised, NonExistentEndPoint, EndpointNotActive, ResourceError.

NFEP modification

TCON-U-17 The connectivity consumer shall provide a function for modifying the QoS parameters of the NFEP.

TCON-U-18 Upon receipt of a NFEP modification request, the connectivity consumer shall first check the validity of the request. If the request is invalid, the connectivity consumer shall send a response to the connectivity provider indicating the invalidity of the request. Note that if a request is invalid, the connectivity consumer shall not modify the NFEP specified.

TCON-U-19 If a NFEP modification request is valid, the connectivity consumer shall modify the NFEP specified in the request in accordance with the parameters specified in the request.

TCON-U-20 If a NFEP is modified successfully, the connectivity consumer sends a response to the connectivity provider indicating that the modification was successful.

TCON-U-21 The connectivity consumer shall report a failure to the connectivity provider in response to a NFEP modification request, if the connectivity consumer is unable to modify the specified NFEP. One of the following failure codes may be returned: Not authenticated, Not authorised, InsufficientResources, InsufficientBandwidth, QoSCannotBeMet, NonExistentEndPoint, ResourceError.

NFEP release

TCON-U-22 The connectivity consumer shall provide a function for releasing a NFEP. When a NFEP is released, all terminal resources that support the NFEP are released, and transport of information over the NFEP is stopped. More precisely, the supporting NWTTP is deleted. If applicable (i.e. in case the link is not pre-provisioned) the NWCTP is deleted as well. If dynamic creation of NFEPs upon set-up is used (i.e. if the NFEP was not pre-provisioned), then the NFEP is deleted as well.

TCON-U-23 Upon receipt of a NFEP release request, the connectivity consumer shall first check the validity of the request. If the request is invalid, the connectivity consumer shall send a response to the connectivity provider indicating the invalidity of the request.

TCON-U-24 The connectivity consumer shall process a valid NFEP release request in the following manner. First, it shall send a response to the connectivity provider indicating that the NFEP will be released. Then, it shall attempt to release immediately all terminal resources that support the NFEP. If this is not possible (e.g. due to terminal failure), it shall record the release request, and retry to release the terminal resources at a later time. It shall ensure that the terminal resources are released eventually.

Status change reports

TCON-U-25 The connectivity consumer shall provider a function for notifying changes in the status of NFEP's to the connectivity provider.

2.2.3.2 Configuration Management requirements

Information about NFEPpools

TCON-U-26 When a network flow endpoint pool is created or deleted or modified by the connectivity consumer, the connectivity consumer shall send a notification to the connectivity provider, with the following information:
- NFEPpool name
- Characteristic information

TCON-U-27 The connectivity consumer shall provide a function allowing the connectivity provider to query the status of NFEPpools.

2.3 Non functional requirements

None identified.

3. Business Role Life-Cycle

This version of TCon does not define the business role life-cycle.

4. The information model

4.1 Access part

Figure 4-1 shows the information view on the TCon access part.

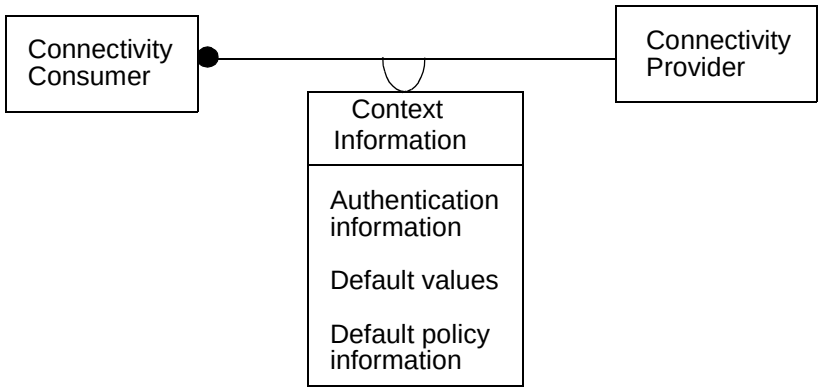

Figure 4-1. OMT diagram for TCon access part

The figure shows the following objects:

Connectivity Consumer: An instance of this object type represents an entity that plays the connectivity consumer role in the TCon reference point. Its attributes are:

- Connectivity consumer name

Connectivity Provider: An instance of this object type represents the entity that plays the connectivity provider role in the TCon reference point. Its attributes are:

- Connectivity provider name

Context Information: An instance of this object type represents information pertaining to the business relationship between a connectivity consumer and a connectivity provider. This object exists only when the business the relationship between the connectivity consumer and the connectivity provider exists. Its attributes are:

- Authentication information (details to be determined). The following information is needed: degree of trust, authentication protocol to be used, key information, re-authentication interval, etc.
- Default values for the following parameters: traffic type, reliability class, initial administrative state
- Policy information: this specifies if connectivity provider needs configuration change reports and NFEP failure reports.

4.2 Usage part

Figure 4-2 shows the OMT diagram for the information model relevant to the TCon usage part.

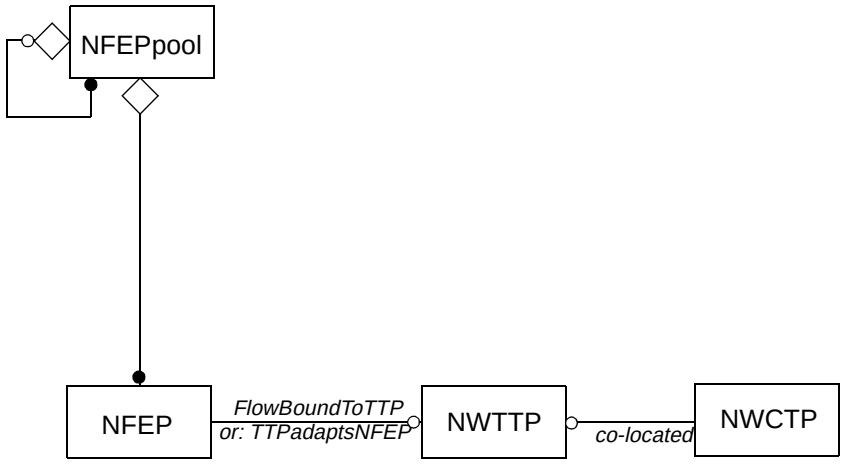

Figure 4-2. Information model for TCon usage part

The following objects are present in this model:

Network Flow End Point (NFEP): This is a technology independent representation of a particular end point of a network flow connection, in a connectivity consumer domain. It has the following attributes:

- NFEP Name
- QoS description
- Lifetime (i.e. pre-provisioned or not)

A state diagram for NFEPs is given in Section 4.2.1.

Network Flow End Point pool (NFEPpool): This is a collection of zero or more NFEP's. The NFEPpool may consist of an aggregation of zero or more NFEPpools. It has the following attributes:

- NFEPpool name
- Characteristic information
- Operational state
- Administrative state

Furthermore, it has the following notifications:

- Creation
- Deletion

Network Trail Termination Point (NWTTP): Termination point for trails, see NRIM [3]. Its attributes are:

- NWTTP name
- QoS description

Network Connection Termination Point (NWCTP): Termination point for subnetwork connections, transport connections and link connections, see NRIM [3]. In the context of TCon it is the termination point for the link connection to the connectivity consumer domain. Its attributes are:

- NWCTP name
- Operational state
- QoS description
- Lifetime (i.e. pre-provisioned or not)

Furthermore, the following relationships are shown in Figure 4-2:

FlowBoundToTTP: An instance of this relationship represents a direct association of a NFEP to a NWTTP.

TTPadaptsNFEP: An instance of this relationship implies that an adapter is used between NFEP and NWTTP.

Co-located: An instance of this relationship represents a co-location of a NWTTP to a NWCTP.

4.2.1 State diagram for NFEP

Figure 4-3 shows the state diagram for a NFEP (OMT notation). This state diagram is helpful in understanding the effects of the operations that are described in chapter 5.

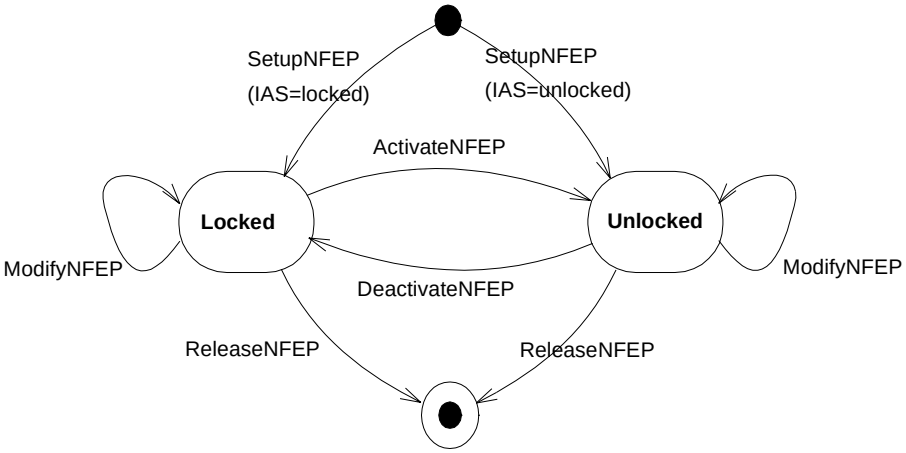

Figure 4-3. State diagram for a NFEP

Once created, a NFEP can be in two states:

- **Locked**: In this state the NFEP is fully configured, but is not ready to send and/or receive information.

- **Unlocked**: In this state the NFEP is fully configured, and is sending and/or receiving information.

Creation, deletion, and transitions between the states are achieved by operations on the computational object that encapsulates the NFEP information object, and is described in chapter 5.

5. The computational model

Figure 5-1 shows the computational model for TCon. TCon prescribes the interfaces shown in Figure 5-1, and the behavior of the connectivity consumer and connectivity provider with respect to those interfaces. The individual objects are shown for ease of understanding, but are not prescriptive. An implementor may define a distinct set of objects. As long as the same set of interfaces is offered across the TCon reference point, and the same behavior is exhibited by the overall set of interfaces, compliance with the TCon reference point is fulfilled.

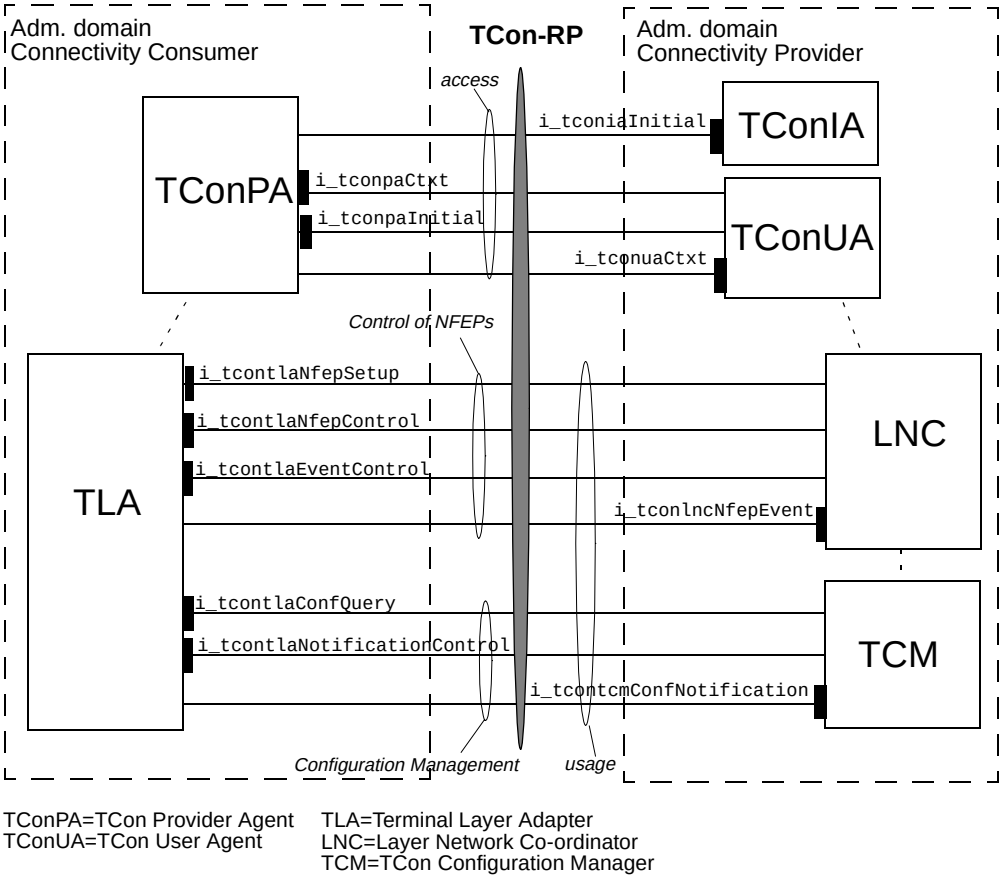

Figure 5-1. Computational model for TCon

The interfaces on TCon are categorized into two parts, i.e. access and usage. The usage part itself is subdivided into “Control of NFEPs”, and “Configuration Management”. The individual (descriptive) objects and their (prescriptive) interfaces are discussed in sections 5.2 (access) and 5.3 (usage). Section 5.4 gives a specification of the objects and their interfaces in Object Definition Language (ODL), while section 5.5 specifies event trace diagrams for interactions at TCon.

5.1 Objects for access

5.1.1 TCon Provider Agent (TConPA)

The TConPA represents a connectivity provider in a connectivity consumer's domain. It performs the access part related functions in the connectivity consumer's domain. An instance of a TConPA is created every time access interactions are needed, and lasts at least as long as these interactions.

The TConPA provides the following interfaces:

5.1.1.1 `i_tconpaInitial` interface: Start access

This interface is used by the TConUA to make the TConPA initiate access interactions. It provides the following operation:

`invite`: This operation signals to the TConPA that is should start the establishment of a secured binding with the TConUA. It contains the following parameters:

- Input: Context information (to be defined).
- Output: Accept or refuse.

This interface may be registered with the UA, or with a naming/location service, or it may be found through the DPE. This aspect is not specified in this reference point specification.

5.1.1.2 `i_tconpaCtxt` interface: Context management

This interface is used by the UA for retrieval of information from the connectivity consumer domain. It provides the following operation:

`get_reference`: This operation is used by the TConUA to get references to TCon usage interfaces in the connectivity consumer domain. Its parameters are:

- Input: Requested interface reference type.
- Output: Interface reference of requested type.

5.1.2 TCon Initial Agent (TConIA)

This object is the initial access point to a connectivity provider's domain. It provides the following interface:

5.1.2.1 `i_tconiaInitial` interface: Request access

A connectivity consumer uses this interface to request the start of access interactions to the connectivity provider, and to obtain an interface reference to its User Agent.

Reference to this interface is supplied to the connectivity consumer through either electronic means (e.g., registered in a trader) or by some off-line means. This aspect is not specified in this reference point specification.

This interface provides the following operations:

request_access: This operation is used by the TConPA to request the start of access interactions:

- Input: User id.
- Output: security information.

request_uaref: This operation is used by the TConPA to reveal its own **i_tconpaCtxt** interface reference, and to get a reference to the **i_tconuaCtxt** interface of the User Agent:

- Input: Interface reference of **i_tconpaCtxt**.
- Output: Interface reference of **i_tconuaCtxt**.

5.1.3 TCon User Agent (TConUA)

This object represents a connectivity consumer in a connectivity provider's domain. There is one instance per connectivity consumer of a TConUA in a connectivity provider's domain. This object is created when the business relationship between connectivity consumer and connectivity provider is setup, and exists as long as the business relationship exists. It provides the following interface:

5.1.3.1 i_tconuaCtxt Interface: Context_Management

This interface provides functions for retrieval and modification of information stored in the Context Information object associated with a connectivity consumer.

This interface provides the following operations:

get_authentication_info: This operation is used for obtaining authentication information. Its parameters are:

- Input: Security information (details to be determined)
- Output:
  1. Success or Failure indication
  2. Authentication information (details to be determined)

update_authentication_info: This operation is used for updating authentication information. Its parameters are:

- Input:
  1. Security information (details to be determined)
  2. Authentication information (details to be determined)
- Output: Success or Failure indication

get_default_values: This operation is used for obtaining default values for NFEP management. Its parameters are:

- Input: Security information (details to be determined)
- Output:
  1. Success or Failure indication
  2. Default values information (details to be determined)

update_default_values: This operation is used for updating the default values for NFEP management. Its parameters are:

- Input:
  1. Security information (details to be determined)
  2. Default values information (details to be determined)
- Output: Success or Failure indication

get_policy_information: This operation is used for obtaining policy information. It parameters are:

- Input: Security information (details to be determined)
- Output:
  1. Success or Failure indication
  2. Policy information (details to be determined)

update_policy_information: This operation is used to update policy information. Its parameters are:

- Input:
  1. Security information (details to be determined)
  2. Policy information (details to be determined)
- Output: Success or Failure indication

get_reference: This operation is used to get references to TCon usage interfaces in the connectivity provider domain. Its parameters are:

- Input: Requested interface type;
- Output: Interface reference for requested type.

5.2 Objects for Usage

The specification of the TLA, LNC and TCM and their interfaces, as provided by this document, is technology independent. In some cases the given specification may directly be usable for implementation, but specialization for different technologies is allowed for other cases.

5.2.1 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.

5.2.1.1 `i_tcontlaNfepSetup` interface: Setup of NFEPs

This interface covers requirements TCON-U-2 to TCON-U-6. It provides the following operation:

`setup_nfep`: This operation is used to set-up the NFEP in the connectivity consumer domain (requirement TCON-U-2). The connectivity consumer will execute the operation in accordance with requirements TCON-U-2 to TCON-U-6 (e.g. report success or failure in the response to the operation). 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. The parameter list is as follows:

- Input:
  - Correlation Identifier
  - NFEPreference[3]
  - NFEPType
  - ProposedNFEPInformation[4]
  - QoSDescription
  - ChoiceCapability[5]
  - InitialAdministrativeState
  - ApplicationInformation
- Output:
  - Result (success, failure)
  - NFEP name
  - ResolvedNFEPInformation[6]
  - TCon-TLA-NFEPControlReference
  - Error cause, if applicable

5.2.1.2 `i_tcontlaNfepControl` interface: Manipulation of NFEPs

This interface covers the requirements TCON-U-7 to TCON-U-25. It provides the following operations:

`activate_nfep`: This operation is used to activate a NFEP in the connectivity consumer domain (requirement TCON-U-7). The connectivity consumer will execute the operation in accordance with requirements TCON-U-7 to TCON-U-11 (e.g. report success or failure in the response to the operation). 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. The parameter list is as follows:

- Input
  - NFEP name
- Output
  - Result

3. This parameter contains a reference to an NFEP in case the terminal offers pre-provisioned NFEPs that can be used, or a reference to an NFEPpool if an NFEP has to be created.
4. This parameter is specified in detail in Section 5.4 (ODL specifications). It includes a list of NWCTPs (e.g. channel numbers) among which the connectivity consumer may or must choose, depending on the value of the parameter “ChoiceCapability”.
5. This parameter indicates whether the connectivity consumer must choose NWCTPs from the List_NFEPInformation parameter, or may choose itself.
6. This parameter specifies the name of the NWTTP that the connectivity consumer has created, and the chosen NWCTP.

- Error cause, if applicable

`deactivate_nfep`: This operation is used to de-activate an active NFEP in the connectivity consumer domain (requirement TCON-U-12). The connectivity consumer will execute the operation in accordance with requirements TCON-U-12 to TCON-U-16 (e.g. report success or failure in the response to the operation). 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. The parameters of this operation are as follows:

- Input
  - NFEP name
- Output
  - Result
  - Error cause, if applicable

`modify_nfep`: This operation is used to modify the characteristic of the NFEP (e.g. bandwidth, QoS, bearer service type, etc.) (requirement TCON-U-17). The connectivity consumer will execute the operation in accordance with requirements TCON-U-17 to TCON-U-21 (e.g. report success of failure in the response to the operation). This operation may be invoked in the *locked* state or the *unlocked* state. It will not result in a state change.Its parameter list is as follows:

- Input
  - NFEP name
  - NewQoSDescription
- Output
  - Result
  - Error cause, if applicable

`release_nfep`: This operation is used to release a NFEP in the connectivity consumer domain (requirement TCON-U-22). The connectivity consumer will execute the operation in accordance with requirements TCON-U-22 to TCON-U-24 (e.g. report success or failure in the response to the operation). 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[7], the supporting NWCTP is deleted as well. Similar, the NFEP is deleted if it is not pre-provisioned[8]. The parameters of this operation are:

- Input
  - NFEP
- Output
  - Result
  - Error cause, if applicable

7. The information model in Section 4.2 includes an attribute indicating whether the NWCTP is pre-provisioned.
8. The information model in Section 4.2 includes an attribute indicating whether the NFEP is pre-provisioned.

5.2.1.3 `i_tcontlaEventControl` interface

This interface allows a LNC 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_event_notification`: This operations is used for instructing the connectivity consumer to emit notifications regarding the NFEP. Its parameters are:

- Output: Success or failure indication

`disable_event_notification`: This operations is used for instructing the connectivity consumer to suspend emission of notifications regarding the NFEP. Its parameters are:

- Output: Success or failure indication

`set_notification_destination`: This operation is used for instructing the TLA about the `i_tconlncNfepEvent` interface where notifications should be sent. Its parameters are:

- Input: Reference to the `i_tconlncNfepEvent` interface offered by the connectivity provider for receipt of notifications.
- Output: Success or failure indication

5.2.1.4 `i_tcontlaConfQuery` interface

This interface allows a TCM to query the TLA about available NFEPpools and their status. It provides the following operation:

`get_nfeppools`: This operations is used for retrieving the names of all flow endpoint pools that the connectivity consumer domain supports. Its parameters are:

- Output: Network flow endpoint pool name, characteristic information

5.2.1.5 `i_tcontlaNotificationControl` interface

This interface allows a TCM to control the emission of notifications by the TLA regarding available NFEPpools. It provides the following operations:

`enable_notification`: This operations is used for instructing the connectivity consumer to emit notifications regarding NFEPpools. Its parameters are:

- Output: Success or failure indication

`disable_notification`: This operations is used for instructing the connectivity consumer to suspend emission of notifications regarding NFEPpools. Its parameters are:

- Output: Success or failure indication

`set_notification_destination`: This operation is used for instructing the TLA about the `i_tcontcmConfNotification` interface where notifications should be sent. Its parameters are:

- Input: Reference to the `i_tcontcmConfNotification` interface offered by the connectivity provider for receipt of notifications regarding configuration changes in NFEPpools.

- Output: Success or failure indication

5.2.2 Layer Network Co-ordinator (LNC)

The LNC manages network flow connections within a layer network domain. It interacts with the TLA to manipulate the endpoints of these network flow connections, i.e. the NFEPs. There is one LNC in the connectivity provider's domain for every layer network for every administrative domain. The LNC controls the manipulation of NFEPs through the TLA's `i_tcontlaNfepControl` interface, while itself provides the following interface on TCon[9]:

5.2.2.1 `i_tconlncNfepEvent` interface: Notification of NFEP events

Through this interface the TLA can notify the LNC 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. Its parameters are:

- Input: NFEP name, operational state (failed, operational, degraded)

5.2.3 TCon Configuration Manager (TCM)

This object, in the connectivity provider's domain, maintains a view on the available connectivity configuration in the connectivity consumer's domain. To this end it may actively query the `i_tcontlaConfQuery` interface on the TLA in the connectivity consumer's domain, or it may be notified of configuration changes by the TLA through the `i_tcontcmConfNotification` interface, which is described in the following subsection:

5.2.3.1 `i_tcontcmConfNotification` interface

Through this interface the TLA can notify the TCM of changes in its connectivity configuration. The following operations are provided:

`nfeppool_created`: This operation is used for notifying that a new network flow endpoint pool has been created. Its parameters are:

- Input: Network flow endpoint pool name, characteristic information.

`nfeppool_deleted`: This operation is used for notifying that a network flow endpointpool has been deleted. A network flow endpoint pool may only be deleted if there are no network flow endpoints in that pool set-up or in the process of being set up.

- Input: Network flow endpoint pool name.

9. The LNC provides other interfaces as well, but these are outside the scope of TCon.

5.3 Specification of computational objects

This section provides the formal ODL-specification of the objects, interfaces and operation as described in the previous section.

5.3.1 Common definitions

```
// TconCommonDefinitions.odl.h

typedef string t_InterfaceReference;

// Name types

typedef sequence <string> Name;
typedef Name t_NfeppoolName;
typedef sequence <t_NfeppoolName> t_NfeppoolList;
typedef unsigned long t_NfepHandle;
struct t_NfepName {
           t_NfeppoolName pool;
           t_NfepHandle nfep };

// NFEP Information

enum t_NfepType { root, leaf, bidirectional };

enum t_NfepRefType { PartialRef, FullRef };

union t_NfepRef switch (t_NfepRefType)
    {
    case PartialRef: t_NfeppoolName pref;
    case FullRef: t_NfepName fref;
    };

struct t_NfeppoolInformation {
           //to be determined
           };

typedef any t_Nwttp; // To be imported?

typedef any t_Nwctp; // To be imported?

struct t_NfepInformation {
           t_Nwttp NWTTP,
           t_Nwctp NWCTP
           };

typedef sequence <t_NfepInformation> t_List_NfepInformation;

// QoS descriptors

// typedef unsigned long t_PkBandwidth;
// typedef unsigned long t_AvBandwidth;
//      These two definitions are required for the following example:

struct t_BandwidthDescription {
           // Description depends on technology.
           // For instance for ATM:
           // t_PkBandwidth PkBandwidth;
           // t_AvBandwidth AvBandwidth
           }
```

```
// typedef unsigned long t_MaxDelay;
// typedef unsigned long t_MaxVarDelay;
//      These two definitions are required for the following example:

struct t_DelayDescription {
           // Description depends on technology.
           // For instance for ATM:
           // t_MaxDelay MaxDelay;
           // t_MaxVarDelay MaxVarDelay
           }

// typedef unsigned long t_MaxErrRate;
//      These two definitions are required for the following example:

struct t_ErrorRateDescription {
           // Description depends on technology.
           // For instance, for ATM:
           // t_MaxErrorRate MaxErrorRate
           }

// enum t_TrafficType { CBR, VBR, ABR, UBR };
//      Another example.

enum t_ReliabilityClass { ReleaseOnFailure, HoldOnFailure };

struct t_QosDescription {
           t_BandwidthDescription BandwidthDescription;
           t_DelayDescription DelayDescription;
           t_ReliabilityClass ReliabilityClass;
           // Others, for instance:
           // t_TrafficType TrafficType
           };

// Types for state attributes

enum t_AdministrativeState { Unlocked, Locked };

enum t_OperationalState { Failed, Operational, Degraded };

// Various types

enum t_ChoiceCapability; { FromPool, FromList }
           // Indicates whether the terminal may choose NWCTP itself

typedef string t_CorrelationId;

typedef any t_ApplicationInformation;

enum t_Result { Success, Failure };

typedef Credentials SecHandle; //Credentials is defined in CORBA Security
                                  // specs.
```

5.3.2 Access Part Definitions

```
// TconAccessPartDefinitions.odl.h

typedef string t_UserId;

typedef any t_SecurityInfo;

typedef any t_ContextInfo;
```

```
enum t_AccessErrorCode { NotAuthenticated, NotAuthorised };

exception AccessError { t_AccessErrorCode AccessErrorCode };
```

5.3.3 TConPA

```
// TconPA.odl.h

#include “TconCommonDefinitions.odl.h”
#include “TconAccessPartDefinitions.odl.h”

object TConPA {
    behaviour
        “This object support access functions in the connectivity
         consumer’s domain.”
    requires
        i_tconiaInitial;
        i_tconuaCtxt;
        // It will also require access to the TLA, but this is
        // beyond the scope of the TCon reference point.
    initial
            // Initialization is beyond the scope of TCon
    supports
        i_tconpaInitial;  // UA -> PA
        i_tconpaCtxt;           // UA -> PA
            // It will also support interfaces for access by other
            // objects within the connectivity consumer’s domain, but
            // this is outside the scope of the TCon RP.
    };

interface i_tconpaInitial {  // UA -> PA
    behaviour
        “Through this interface the UA may initiate
         access interactions.”;
    usage
        “”;

    void invite(
        in t_ContextInfo ContextInfo,
        out t_Result Result );
    };

interface i_tconpaCtxt {  // UA -> PA
    behaviour
        “Through this interface the UA may retrieve information
         from the PA.”;
    usage
        “”;

    void get_reference(
        in t_ReferenceType ReferenceType,
        out t_InterfaceReference Reference)
        raises (AccessError);
    };
```

5.3.4 TConIA

```
// TconIA.odl.h

#include “TconCommonDefinitions.odl.h”
```

```
#include “TconAccessPartDefinitions.odl.h”

object TConIA {
    behaviour
        “This object is the initial access point to a connectivity
         provider’s domain.”
    requires

    initial
            // Initialization is beyond the scope of TCon
    supports
        i_tconiaInitial;  // PA -> IA
    };

interface i_tconiaInitial {  // PA -> IA
    behaviour
        “Through this a PA can request access to the connectivity
         provider’s domain.”;
    usage
        “”;

    void request_access (
        in t_UserId UserId,
        out t_SecurityInfo SecurityInfo)
        raises (AccessError);

    void request_uaref (
        in t_InterfaceReference PA-Ctxt-Reference)
        out t_InterfaceReference UA-Ctxt-Reference)
        raises (AccessError);
```

5.3.5 TConUA

```
// TconUA.odl.h

#include “TconCommonDefinitions.odl.h”
#include “TconAccessPartDefinitions.odl.h”

object TConUA {
    behaviour
        “This object represents the connectivity consumer in the
         provider’s domain.”
    requires
        i_tconpaInitial;
        i_tconpaCtxt;
            // It will also require access to objects within the
            // connectivity provider’s domain (e.g. LNC), but this
            // is beyond the scope of TCon.
    initial
            // Initialization is beyond the scope of TCon
    supports
        i_tconuaCtxt;       // PA -> UA
                // It will also support interfaces for access by other
            // objects within the connectivity provider’s domain, but
            // this is outside the scope of the TCon RP.
    };

interface i_tconuaCtxt {  // PA -> UA
    behaviour
        “This interface provides functions for retrieval and
         modification of information stored in the Context Information
```

```
         object associated with a CU”
    usage
        “”;

// The following operations are to be refined

    ContextMgmntStatus get_authentication_info (
        out AttributeTypeList     attributes
        );
    ContextMgmntStatus update_authentication_info (
        inout AttributeTypeListattributes
        );
    ContextMgmntStatus get_default_values (
        out AttributeTypeList     attributes
        );
    ContextMgmntStatus update_default_values (
        inout AttributeTypeList attributes
        );
    ContextMgmntStatus get_policy_information (
        in AttributeTypeList      attributes
        );
    ContextMgmntStatus update_policy_information (
        inout AttributeTypeListattributes
        );

    void get_reference (
        in t_ReferenceType ReferenceType,
        out t_InterfaceReference Reference)
        raises (AccessError);
    };
```

5.3.6 Usage part definitions

```
// TconUsagePartDefinitions.odl.h

enum t_SetupErrorCode {
            NotAuthenticated,
            NotAuthorised,
            InsufficientResources,
            InsufficientBandwidth,
            QoSCannotBeMet,
            NonexistentEndPoint,
            EndpointAlreadyInUse,
            ResourceError
        };

exception SetupError { t_SetupErrorCode SetupErrorCode };

enum t_ActivationErrorCode {
            NotAuthenticated,
            NotAuthorised,
            NonexistentEndPoint,
            EndPointNotAvailable,
            EndpointAlreadyUnlocked,
            ResourceError
        };

exception ActivationError
                { t_ActivationErrorCode ActivationErrorCode };

enum t_DeactivationErrorCode {
            NotAuthenticated,
```

```
            NotAuthorised,
            NonexistentEndPoint,
            EndpointNotUnlocked,
            ResourceError
        };

exception DeactivationError
            { t_DeactivationErrorCode DeactivationErrorCode };

enum t_ModificationErrorCode {
            NotAuthenticated,
            NotAuthorised,
            InsufficientResources,
            InsufficientBandwidth,
            QoSCannotBeMet,
            NonexistentEndPoint,
            ResourceError
        };

exception ModificationError
            { t_ModificationErrorCode ModificationErrorCode };

enum t_ReleaseErrorCode {
            NotAuthenticated,
            NotAuthorised,
            NonexistentEndPoint,
        };

exception ReleaseError { t_ReleaseErrorCode ReleaseErrorCode };

enum t_GetNfeppoolsErrorCode {
            NotAuthenticated,
            NotAuthorised,
            ResourceError
        };

exception GetNfeppoolsError {
            t_GetNFEPpoolsErrorCode GetNFEPpoolsErrorCode };

enum t_StatusChangeErrorCode {
            NotAuthenticated,
            NotAuthorised,
            NonexistentEndPoint,
        };

exception StatusChangeError
            { t_StatusChangeErrorCode StatusChangeErrorCode };

enum NfeppoolConfErrorCode {
            NotAuthenticated,
            NotAuthorised,
            NFEPpoolUnknown
        };

exception NfeppoolConfigurationError
            { t_NfeppoolConfErrorCode NfeppoolConfErrorCode };
```

5.3.7 TLA

```
// TLA.odl

#include “TconCommonDefinitions.odl.h”
```

```
#include “TconUsagePartDefinitions.odl.h”

object TLA {
    behaviour
        “This object supports the control of NFEPs. It allows their
         setup, reservation, activation, deactivation, modification
         and release.”
    requires
            i_tconlncNfepEvent;
            i_tcontcmConfNotification;
            // The TLA will also require access to specific interfaces
            // of the TCSM and the TCC, but this is beyond the scope of
            // the TCon specification, and should be specified
            // elsewhere.
    initial
            // Initialization is beyond the scope of TCon
    supports
        i_tcontlaNfepSetup;
        i_tcontlaNfepControl;
        i_tcontlaEventControl;
        i_tcontlaConfQuery;
        i_tcontlaNotificationControl;
            // It will also support interfaces for access by TCSM or
            // TCC, but this outside the scope of the TCon RP, and
            // should be specified elsewhere
    };

interface i_TcontlaNfepSetup {  // LNC -> TLA
    behaviour
        “Through this interface NFEPs can be set-up.”;
    usage
        “”;

    void setup_nfep (
        in t_CorrelationId CorrelationId,
        in t_QosDescription QoSDescription,
        in t_NfepRef NfepRef,
        in t_NfepType NFEPType,
        in t_List_NfepInformation ProposedNFEPInformation,
        in t_ChoiceCapability ChoiceCapability,
        in t_AdministrativeState InitialAdministrativeState,
        in t_ApplicationInformation ApplicationInformation,
        out t_Result Result,
        out t_NfepName NFEPName,
        out t_NfepInformation ResolvedNFEPInformation,
        out t_InterfaceReference TCon-TLA-NFEPControlReference)
        raises (SetupError);
    };

interface i_tcontlaNfepControl {  // LNC -> TLA
    behaviour
        “Through this interface NFEPs can be manipulated,
         i.e., activated, deactivated, modified and released.”;
    usage
        “”;

    void activate_nfep (
        in t_NfepName NFEPName,
        out t_Result Result)
        raises (ActivationError);

    void deactivate_nfep (
        in t_NfepName NFEPName,
```

```
        out t_Result Result)
        raises (DeactivationError);

    void modify_nfep (
        in t_NfepName NFEPName,
        in t_QoSDescription QoSDescription,
        out t_Result Result)
        raises (ModificationError);

    void release_nfep (
        in t_NfepName NFEPName,
        out t_Result Result)
        raises (ReleaseError);
    };

interface i_tcontlaEventControl {  // CMC -> TLA
    behaviour
        “This interface allows an LNC to control the emission of event
        notifications regarding the operational state of NFEPs by the TLA.”
    usage
        “”;

    void enable_event_notification (
        out t_Result Result);

    void disable_event_notification (
        out t_Result Result);

    void set_notification_destination
        in t_InterfaceReference i_tconlncNfepEvent(
        out t_Result Result);

    };

interface i_tcontlaConfQuery {  // CMC -> TLA
    behaviour
        “This interface supports querying of configuration information
         in the connectivity consumer’s domain.”
    usage
        “”;

    void get_nfeppools (
        out t_NfeppoolList NFEPpoolList)
        raises (GetNFEPpoolsError);
    };

interface i_tcontlaNotificationControl {  // CMC -> TLA
    behaviour
        “This interface allows a TCM to control the emission of notifications by the
TLA regarding available NFEPpools.”
    usage
        “”;

    void enable_notification(
        out t_Result Result);

    void disable_notification(
        out t_Result Result);

    void set_notification_destination(
        in t_InterfaceReference i_tcontcmConfNotification(
        out t_Result Result);
```

```
    };
```

5.3.8 LNC

```
// LNC.odl
#include “TconCommonDefinitions.odl.h”
#include “TconUsagePartDefinitions.odl.h”

object LNC {
    behaviour
        “This object co-ordinates the establishment of connections
         within a layer network domain. Through TCon it makes use of
         the TLA to establish the NFEP of a network flow connection.
         It also offers an interface to the TLA (FFS)”
    requires
            i_tcontlaNfepSetup;
            i_tcontlaNfepControl;
            // The LNC will also require access to for instance
            // interfaces of CPs and other LNCs,, but this is outside
            // the scope of the TCon and should be specified elsewhere.
    supports
        i_tconlncNfepEvent // TLA -> LNC
        // The LNC will also support other interfaces, but these are
        // outside the scope of TCon.
    };

interface i_tconlncNfepEvent { // TLA -> LNC
    behaviour
        “This interface is used by the TLA to notify the LCN of changes
         in the operational state of a NFEP”;
    usage
        “”;

    void status_change (  // TLA -> LNC
        in t_NfepName NFEPName,
        in t_OperationalState NFEPStatus)
        raises (StatusChangeError);
    };
```

5.3.9 Terminal Configuration Manager

```
// TCM.odl

#include “TconCommonDefinitions.odl.h”
#include “TconUsagePartDefinitions.odl.h”

object TCM {
    behaviour
        “This object allows the connectivity provider to maintain
         a view on the available connectivity in the connectivity
         consumer’s domain.”
    requires
        i_tcontlaConfQuery
            // It will also require other interfaces, but these are
            // outside the scope of TCon
    supports
        i_tcontcmConfNotification
            // It will also support other interfaces, but these are
            // outside the scope of TCon
    };
```

```
interface i_tcontcmConfNotification {  // TLA -> TCM
    behaviour
        “This interface allows to TLA to notify the connectivity
         provider about the connectivity consumer’s connectivity
         configuration (i.e. creation, deletion and status change of
        NFEPpools.”
    usage
        “”;

    void nfeppool_created (
        in t_NfeppoolName NFEPpoolName,
        in t_NfeppoolInformation NFEPpoolInformation)
        raises (AccessError);

    void nfeppool_deleted (
        in t_NfeppoolName NFEPpoolName)
        raises (NFEPpoolConfigurationError) );
    };
```

5.4 The event traces

Conventions: The operation name is the same as in the ODL specification. Operation returns are numbered and labelled with empty square brackets.

5.4.1 Login

This section specifies the event traces for login. Login allows a connectivity consumer to initiate the set-up of a secured binding with the connectivity provider. The procedure is shown in Figure 5-2. The consecutive steps are described below.

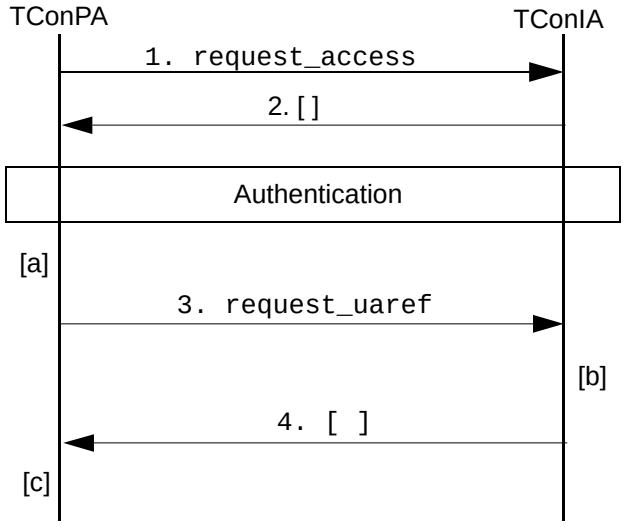

Figure 5-2. Login of a connectivity consumer

Preconditions

None.

Postconditions

Upon successful completion of login sequence, TConPA is ready to start secure usage interactions with TConUA.

1. **`request_access`**: First, initial access is requested by TConPA.
2. TConIA answers the request providing the necessary security information to perform authentication.

   Authentication takes place. It is not detailed here since it is not intended to prescribe any authentication mechanism for TCon RP.

a. Upon successful execution of the authentication, the TConPA will issue the `request_uaref` operation to the TConIA, revealing its own `i_tconpaCtxt` interface reference, and requesting an interface reference to the `i_tconuaCtxt` interface.

2. **`request_uaref`**: The following parameter is passed:
   - `i_tconpaCtxt` reference

b. The TConIA checks if a secured binding exists with the TConPA. If so, it will return the reference of the `i_tconuaCtxt` interface. Otherwise, it will return an indication “NotAuthenticated” to the TConPA.

3. The following parameter is passed in the return:
   - `i_tconuaCtxt` reference

c. If success is returned, the TConPA is now ready to perform procedures related to context management. Furthermore, usage related interactions may now be performed on TCon.

5.4.2 Invitation

This section specifies the event traces for invitation. Invitation allows the connectivity provider to initiate the establishment of a secured binding with the connectivity consumer. The procedure is shown in Figure 5-3. The consecutive steps are described below.

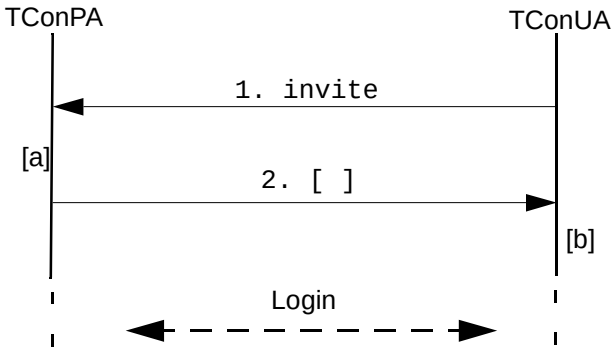

Figure 5-3. Invitation of a connectivity consumer

Preconditions

None.

Postconditions

Upon successful completion of invitation sequence, TConPA and TConUA are ready to start secure usage interactions.

1. **`invite`**: With this operation the TConUA signals to the TConPA that the connectivity provider wants to initiate the establishment of a secured binding with the connectivity consumer. Typically, the connectivity provider will do this if connectivity is about to be established, but a secured binding does not exist yet.

a. The TConPA analyses the request, and will decide whether it accepts the request. If it does not it will return “Refused”. The procedure will then stop after step 2. If, however, the TConPA accepts the request, it will return the value “Accept”.

2. This return of the **`invite`** operation indicates accept or refuse.

b. If 2 indicates “Refused”, the procedure is finished as far as TCon is concerned. The connectivity provider may take further actions (e.g. towards ConS), but these are not visible to TCon. If the returned value indicates “Accept”, the TConUA will start the Login procedure with TConIA as described in the previous section.

5.4.3 Retrieval of connectivity consumer domain interface references

This section specifies the event traces for retrieval of connectivity consumer domain interface references by the connectivity provider. The procedure is shown in Figure 5-4. The consecutive steps are described below.

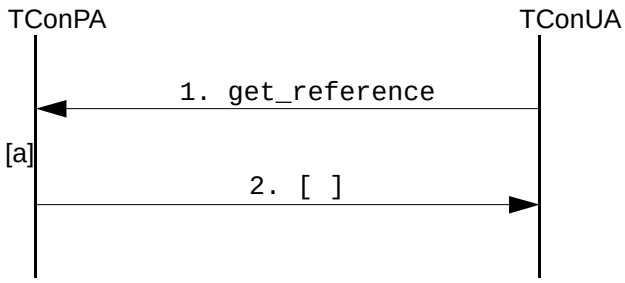

Figure 5-4. Retrieval of CC domain interface references

Preconditions

TConPA and TConUA have already started a secure access session.

Postconditions

TConUA obtains the reference to a usage interface in the consumer domain. A usage interface may be of types such as `i_tcontlaNfepSetup` or `i_tcontlaConfQuery`.

1. **`get_reference`**: With this operation the TConUA requests the TConPA to provide interface references of usage interfaces. The following parameters are passed:
   - `ReferenceType`: This parameter specifies the usage interface type that is needed. This version of TCon supports the following two values for the elements in this list: `i_tcontlaNfepSetup` and `i_tcontlaConfQuery`.

a. The TConPA will check if a secured binding with the connectivity provider exists. If not it will return “NotAuthenticated”. Otherwise it will check whether the requested ReferenceType is allowed. If not it will return “NotAuthorised”. Otherwise the TConUA will return an interface reference of the requested type.

2. This is the return of the **`get_reference`** operation. The following parameters are passed in the return:
   - `Reference`: This parameter contains the requested reference.
   - An exception code, if applicable.

5.4.4 Retrieval of connectivity provider domain interface references

This section specifies the event traces for retrieval of connectivity provider domain interface references by the connectivity consumer. The procedure is shown in Figure 5-5. The consecutive steps are described below.

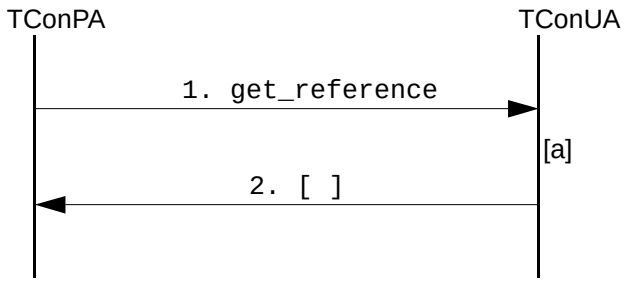

Figure 5-5. Retrieval of CU domain interface references

Preconditions

TConPA and TConUA have already started a secure access session.

Postconditions

TConPA obtains the reference to a usage interface in the provider domain. A usage interface may be of types such as `i_tconlncNfepEvent` or `i_tconcmcConfNotification`.

1. **`get_reference`**: With this operation the TConPA requests the TConUA to provide interface references of usage interfaces. The following parameters are passed:
   - `ReferenceType`: This parameter specifies the usage interface type that is needed. This version of TCon supports the following two values for the elements in this list: `i_tconlncNfepEvent` and `i_tconcmcConfNotification`.

a. The TConUA will check whether the connectivity consumer has previously been authenticated. If not it will return “NotAuthenticated” as an error. otherwise, the TConUA will continue and check whether the requested ReferenceType is allowed. If not it will return “NotAuthorised”. Otherwise the TConUA will return an interface reference of the requested type.

2. This is the return of the **`get_reference`** operation. The following parameters are passed in the return:
   - `Reference`: This parameter contains the requested reference.
   - An exception code, if applicable.

5.4.5 NFEP set-up

This section specifies the event traces for NFEP set-up. The procedure is shown in Figure 5-6. The consecutive steps are described below.

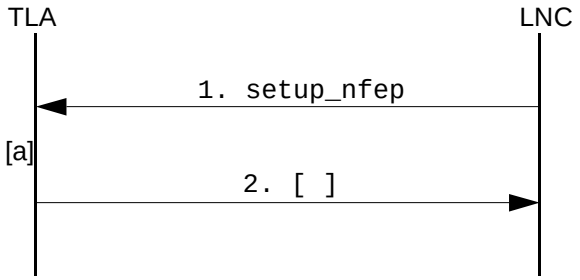

Figure 5-6. NFEP set-up

Preconditions

TConPA and TConUA have already started a secure access session. Using a session key, TLA and LNC are within a secure service session.

Postconditions

Upon successful completion of an NFEP set-up sequence, an NFEP is set up in the consumer domain.

1. **`setup_nfep`**: With this operation the LNC requests the TLA to set-up a NFEP. The following parameters are passed in the operation:
   - `CorrelationId`: The use of this parameter is described in requirement TCON-U-1.
   - `QoSDescription`: This parameter defines the bandwidth and QoS, as described in requirement -.
   - `NFEPRef`: This parameter defines either the NFEPpool from which the NFEP should be chosen, or the name of the NFEP to be used, in case pre-provisioned NFEPs are available.
   - `NFEPType`: This parameter defines the type of the NFEP (root, leaf, bi-directional).
   - `ProposedNFEPInformation`: This parameter contains a list of NWCTPs out of which an NWCTP (to which the NFEP is to be bound) may be chosen, or must be chosen, depending on the parameter `ChoiceCapability`.
   - `ChoiceCapability`: This parameter indicates whether the connectivity consumer must choose NWCTPs from the `ProposedNFEPInformation`, or may choose itself. The LNC can thus force a NWCTP to be used, by having only one NWCTP in the `ProposedNFEPInformation`.
   - `InitialAdministrativeState`: Indicates whether the NFEP should automatically be activated upon set-up, i.e. whether the initial administrative state of the NFEP upon setup should be Unlocked or Locked.

- `ApplicationInformation`: This parameter allows application information to be transferred over the TCon reference point.

a. The TLA will act upon the receipt of the operation in accordance with requirements TCON-U-3 to TCON-U-4, and then issue a response as described in the following step.

2. This is the return of the **`setup_nfep`** operation. It contains the following parameters:

   - `Result`: This parameter describes the result of the **`SetupNFEP`** operation, i.e. success or failure.
   - `NFEPName`: This is the NFEP that the TLA has chosen, and set-up.
   - `ResolvedNFEPInformation`: This parameter contains the chosen NWTTP and NWCTP.
   - `i_tcontlaNfepControlReference`: This is the reference of the interface that the connectivity provider should use for further control operations on the created NFEP.
   - `SetupError`: This exception is raised in case of an error. Possible failure causes are defined in requirement TCON-U-6.

5.4.6 NFEP activation

This section specifies the event traces for NFEP activation. The procedure is shown in Figure 5-7. The consecutive steps are described below.

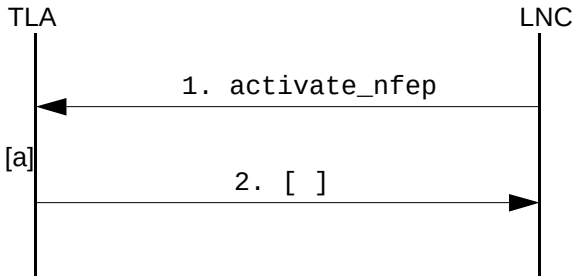

Figure 5-7. NFEP activation

Preconditions

TConPA and TConUA have already started a secure access session. Using a session key, TLA and LNC are within a secure service session.

Postconditions

Upon successful completion of an NFEP activation sequence, an NFEP is activated in the consumer domain.

1. **`activate_nfep`**: With this operation the LNC requests the TLA to activate a NFEP, that previously has been reserved. The following parameters are passed in the operation:
   - `NFEPName`: This is the NFEP that is requested to be activated.

a. The TLA will act upon the receipt of the operation in accordance with requirements TCON-U-8 to TCON-U-9, and then issue a response as described in the following step.

2. This is the return of the **`ActivateNFEP`** operation. It contains the following parameters:
   - `Result`: This parameter describes the result of the **`ActivateNFEP`** operation, i.e. success or failure.
   - `ActivationError`: This exception is raised in case of an error. Possible failure causes are as defined in requirement TCON-U-11.

5.4.7 NFEP deactivation

This section specifies the event traces for NFEP deactivation. The procedure is shown in Figure 5-8. The consecutive steps are described below.

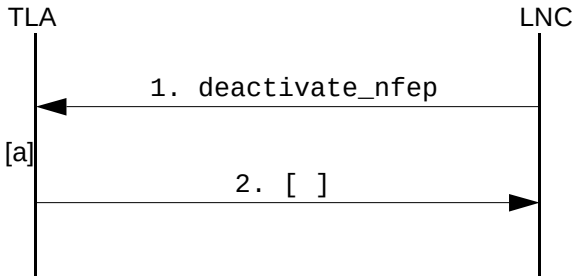

Figure 5-8. NFEP deactivation

Preconditions

TConPA and TConUA have already started a secure access session. Using a session key, TLA and LNC are within a secure service session.

Postconditions

Upon successful completion of an NFEP deactivation sequence, an NFEP is deactivated in the consumer domain.

1. **`deactivate_nfep`**: With this operation the LNC requests the TLA to deactivate a NFEP. The following parameters are passed in the operation:
   - `NFEPName`: This is the NFEP that is requested to be deactivated.

a. The TLA will act upon the receipt of the operation in accordance with requirements TCON-U-13 to TCON-U-14, and then issue a response as described in the following step.

2. This is the return of the **`deactivate_nfep`** operation. It contains the following parameters:
   - `Result`: This parameter describes the result of the **`DeactivateNFEP`** operation, i.e. success or failure.
   - `DeactivationError`: This exception is raised in case of an error. Possible failure causes are as defined in requirement TCON-U-16.

5.4.8 NFEP modification

This section specifies the event traces for NFEP modification. The procedure is shown in Figure 5-9. The consecutive steps are described below.

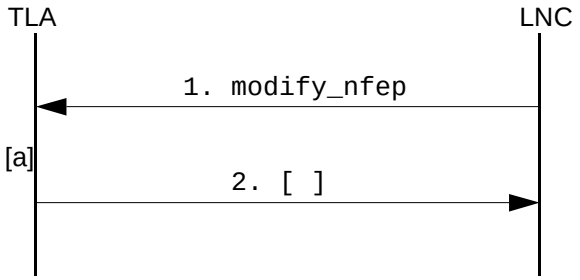

Figure 5-9. NFEP modification

Preconditions

TConPA and TConUA have already started a secure access session. Using a session key, TLA and LNC are within a secure service session.

Postconditions

Upon successful completion of an NFEP modification sequence, an NFEP in the consumer domain is modified in its properties such as the bandwidth, QoS parameters.

1. **`modify_nfep`**: With this operation the LNC requests the TLA to modify the bandwidth and QoS parameters of a NFEP. The following parameters are passed in the operation:
   - `NFEPName`: This is the NFEP that is requested to be modified.
   - `QoSDescription`: This parameter defines the bandwidth and QoS, as described in requirement -.

a. The TLA will act upon the receipt of the operation in accordance with requirements TCON-U-18 to TCON-U-19, and then issue a response as described in the following step.

2. This is the return of the **`modify_nfep`** operation. It contains the following parameters:
   - `Result`: This parameter describes the result of the **`ModifyNFEP`** operation, i.e. success or failure.
   - `ModifyError`: This exception is raised in case of an error. Possible failure causes are as defined in requirement TCON-U-21.

5.4.9 NFEP release

This section specifies the event traces for NFEP modification. The procedure is shown in Figure 5-10. The consecutive steps are described below.

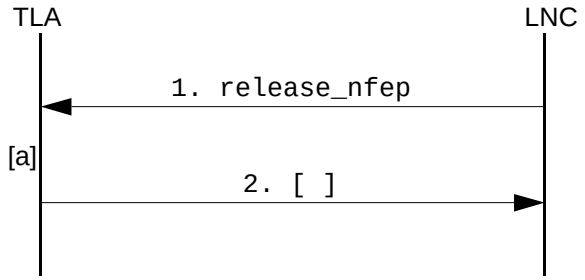

Figure 5-10. NFEP release

Preconditions

TConPA and TConUA have already started a secure access session. Using a session key, TLA and LNC are within a secure service session.

Postconditions

Upon successful completion of an NFEP release sequence, an NFEP in the consumer domain is released.

1. **`release_nfep`**: With this operation the LNC requests the TLA to release a NFEP. The following parameters are passed in the operation:
   - `NFEPName`: This is the NFEP that is requested to be released.

a. The TLA will act upon the receipt of the operation in accordance with requirements TCON-U-23 to TCON-U-24, and then issue a response as described in the following step.

2. This is the return of the **`release_nfep`** operation. The following exception may be raised:
   - `ReleaseError`: This exception is raised in case of an error. See the ODL specification for possible failure causes.

5.4.10 Query NFEPpools (Get NFEPpools)

This section specifies the event traces for querying the TLA about the NFEPpools it supports. The procedure is shown in Figure 5-8. The consecutive steps are described below.

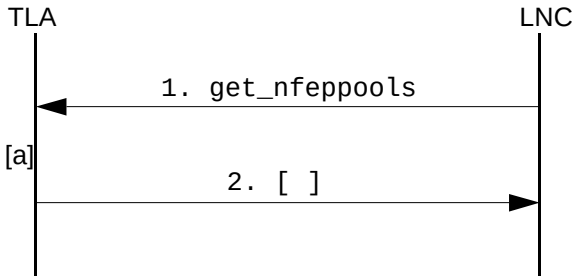

Figure 5-11. NFEP deactivation

Preconditions

TConPA and TConUA have already started a secure access session. Using a session key, TLA and LNC are within a secure service session.

Postconditions

Upon successful completion of an Query NFEPpools sequence, a list of available NFEPpools in the consumer domain is made known to the provider (LNC).

1. **`get_nfeppools`**: With this operation the LNC requests the TLA to reveal the NFEPpools the connectivity consumer domain supports.

a. The TLA will act upon the receipt of the operation in accordance with requirements TCON-U-13 to TCON-U-14, and then issue a response as described in the following step.

2. This is the return of the **`get_nfeppools`** operation. It contains the following parameters:

   - `NFEPpoolList`: This parameter contains the list of NFEPpools supported by the connectivity consumer.
   - `Result`: This parameter describes the result of the **`GetNFEPpools`** operation, i.e. success or failure.
   - `GetNFEPpoolsError`: This exception is raised in case of an error.

5.5 Reuse of interfaces

A common specification of the access interfaces may be shared between Ret, ConS and TCon (possibly with some specialization as required).

6. Engineering model

Not provided in this version of TCon.

7. Miscellaneous

-

8. References

8.1 TINA Baselines

[1] TINA reference points, Document N0 EN_TCJ.030_3.1_96, version 3.1, June 4

[2] TINA-C Core Team, *The ConS Reference Point*, Version 1.0, November 7, 1996

[3] N. Natarajan, H. Flinck, *Network Resource Information Model Specification*, May 15, 1996 Draft

[4] TINA Information modelling Concepts, Document N0 TB_EAC.001_3.0_94, Dec.94

[5] TINA Object Definition Language Manual, Document N0 TR_NM.002_2.2_96, Version 2.3, July 96

[6] TINA Glossary, Version 1.0, 1996.

8.2 Other documents

[7] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, *Object-Oriented Modeling and Design*, Prentice Hall: Englewood Cliffs, N.J.:, 1991.

[8] CORBA Security, OMG Document Number 95-12-1, Dec. 1995.

9. Acronyms

ABR	Available Bit Rate
ATM	Asynchronous Transfer Mode
ConS	Connectivity Service reference point
CTP	see NWCTP
DPE	Distributed Processing Environment
LNC	Layer Network Co-ordinator
LTP	Link Termination Point
NFEP	Network Flow EndPoint
NFEPpool	Network Flow EndPoint pool
NWCTP	NetWork Connection Termination Point
NWTTP	NetWork Trail Termination Point
ODL	Object Definition Language
OMT	Object Modelling Technique
QoS	Quality of Service
Ret	Retailer reference point
RFR/S	Request for Refinements and/or Solutions
RP	Reference Point
SFEP	Stream Flow EndPoint
SNW	SubNetWork
TCM	TCon Configuration Manager
TCon	Terminal Connectivity reference point
TConPA	TCon Provider Agent
TConUA	TCon User Agent
TLA	Terminal Layer Adapter
TTP	see NWTTP
UBR	Unspecified Bit Rate
VBR	Variable Bit Rate

10.Glossary

Refer to the TINA-C Glossary [6].