Telecommunications
Information
Networking
Architecture
Consortium

Issue Status:
Architecture Board approved

The ConS Reference Point

Version:1.0

Abstract: This document contains the specification of the Connectivity Service (ConS) Reference Point, produced by the ConS review panel as a solution for the TINA-C RFR/S on the ConS Reference Point. 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.

Editors: N. Natarajan and Laurence Demounem

Author(s): ConS Review Panel

Date: February 12, 1997

1. Introduction

This document contains the specification of the **Connectivity Service (ConS) Reference Point** (**ConS-RP**) produced by the ConS review panel as a solution to the TINA-C RFR/S on the ConS Reference Point [1]. This specification is mainly based on the Core Team response but also includes some ideas contained in other ConS RFR responses as well as comments on the Core Team response provided by the ConS review panel. This document includes a detailed requirements specification, an information and a computational specification, with definitions of objects, their interfaces in ODL, and event traces.

It is expected that there will be other releases of the ConS-RP specifications by the end of 1997.

The Connectivity Service can be briefly described as follows. The provider of the connectivity service, hereafter referred to as **Connectivity Provider** (**CP**) enables its clients, hereafter referred to as **Connectivity Users** (**CU**) to setup, modify, and release a **connectivity session** that is made up of one or more **network flow connections** (hereafter simply called **flow connections**). See Glossary for the definitions of the terms flow connection, layer network, connectivity layer network, and so on. A flow connection transports information across a **connectivity layer network** that is made up of one or more **layer networks**. The endpoints of a flow connection are **network flow endpoints** (hereafter simply called **flow endpoints**) and the different endpoints of a flow connection may have different characteristic information associated with them. For example, a flow connection may accept frame relay frames on one endpoint and deliver ATM cells on another endpoint. The connectivity service enables a CU to manage (i.e., setup, release, and modify) a group of flow connections as an aggregated unit. Thus, for example, by releasing a connectivity session, a CU can release all flow connections that are part of the connectivity session.

The connectivity service (like any TINA service) has two parts: an **access part** and a **usage part**. The interactions between a CU and the CP in the access part deal with the following capabilities:

- Authentication of the CU
- Establishment and control of service sessions for usage part interactions

Usage part interactions are structured in terms of service sessions, called **Connectivity service session**s. A connectivity service session is an instantiation of one of the following services:

- **Connectivity Control Service**: Setup, modification, and release of connectivity sessions and flow connections; event reports on operational state changes of a flow connection; management of these event reports.
- **Contract Profile Management Service**: Retrieval and modification of contract profile associated with each CU. The contract profile contains default values for some parameters exchanged in the usage part interactions and security parameters exchanged in access part interactions.

1.1 Main assumptions

The inter-object communication across the ConS reference point occur via a DPE supporting the access transparency that is implicit in the TINA Computational Model [4].

In this version of the specification, security and notification services are explicitly specified. However, it is expected that future ConS-RP specifications will adopt more generic specifications for these services in the form of DPE services, as these specifications become available. The advantage of using generic DPE services is that they can be reused for all the reference points.

It is also assumed that CUs know the Network Flow End Point Pools (See Glossary). How the CUs know is out of the scope of the ConS-RP. (A CU may obtain this information from either the terminal equipment or the CP).

1.2 Inter-relationships with other reference points

The ConS reference point does not require any other TINA reference point to be implemented. However, there is an interrelation with the TCon reference point as described below:

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 and endpoint pools at TCon. For instance, the setup of a network flow connection at ConS will result in the setup of a network flow endpoint at TCon. The Core Team’s response to the RFR/S on TCon Reference Point is described in [11].

2. Business Requirements

This section presents a detailed requirements specification for the ConS Reference Point. These requirements form the basis for the information and computational specifications presented in Sections 3 and 4. However, no assumption on information or computational objects, or interfaces, is made in the requirements specification. The requirements are specified in terms of functions to be supported in the ConS-RP. A function maps to one or more computational operations. This mapping is an aspect of the computational specification. In some mappings, some of the parameters (such as identifiers for connectivity sessions and flow connections) may become redundant. In such cases, these parameters may be omitted in the computational specification.

2.1 Access Requirements[1]

Security context

Access interactions are required for authenticating the CU before usage interactions occur (this requires checking the user's contract profile and includes all the security procedures).

Assumption: A CU is not allowed to pass to another CU the rights concerning an access or connectivity session established with a CP.

The requirements listed in the following sections apply assuming that the CU and the CP have already agreed on the password of the CU and other security parameters. The way the contract is set up initially is out of the scope of TINA (traditional mail, etc.). It is assumed that an authentication procedure will be used prior to usage part interactions. Authentication may also occur periodically as determined by the contract profile parameters.

Usage interface request

ConS-A-1: The CP shall provide a function for the CU to obtain usage interface[2] references in order to perform usage part related operations. An authentication must always be performed before any usage interface reference can be used. The following parameter must be provided:

- Security parameters.

ConS-A-2: Upon receipt of the CU's request, a CP shall return the usage interface references with the following parameters:

- Security parameters
- Reference to usage interfaces.

ConS-A-3: The CP shall send a failure report in response to a usage interface request if it is unable to complete the request; this response shall have a failure code parameter. Failure codes are:

- Security requirements not met.
- Usage interface can not be allocated.

1. A1 to A3 requirements will be harmonized with the retailer access specifications in future versions.
2. This interface either already exists or is to be created with a factory. The CP is in charge of creating it.

Usage interface block request

ConS-A-4: The CU can request the blocking of a usage interface reference. This means that usage interface reference becomes invalid. Authentication must always be performed beforehand. This function shall take the following input parameter:

- Security parameters

ConS-A-5: Upon receipt of the CU’s request, a CP shall return a confirmation report if the request has been processed successfully.

The CP shall report a failure in response to a usage interface block request if the CP is unable to complete the request. In such a case, the CP shall send a failure report to the CU with the following information:

Failure code is either

- Security requirements not met
- Usage interface can not be made invalid[3]

After a successful usage interface block request, the CU will not be able to invoke operations on this usage part interface. If the CU wants to invoke an operation on this interface, the CU will have to request another reference.

Contract termination[4]

ConS-A-6: The CP shall provide a function for terminating the business contract. This will result in the deletion of the contract profile and terminate the business relationship between the CU and the CP. This function shall take the following input parameter:

- Security parameters

ConS-A-7: Upon receipt of a contract session termination request, the CP shall first check the validity of the request. The CP shall treat the request as invalid if at least one of the following conditions holds:

- Invalid security parameters.
- Connectivity sessions not terminated.
- Connectivity service sessions not terminated.

If any of the above conditions holds, the CP shall send a response to the CU indicating the invalidity of the request. If the request is invalid, no action will be taken and the contract will not be terminated.

ConS-A-8: If the request is valid, the CP shall delete the contract profile.

ConS-A-9: If the contract is terminated successfully, the CP shall confirm the termination.

3. For example, because operations are being performed on the interface.
4. The contract is initiated off-line, but on-line contract termination is allowed.

2.2 Usage Requirements

Connection Management

The connection management requirements are categorized into several subsections, each subsection dealing with a specific capability of the connectivity service.

Connectivity Session[5] Setup

ConS-U-1 The CP shall provide a function for setting up a connectivity session. This function shall take the following input parameters:

- *Connectivity session identifier*
- *Information on each flow connection* (see ConS-U-2). This parameter provides information on one or more flow connections that are to be setup as part of a connectivity session setup. It is possible to subsequently create and include more flow connections into the connectivity session.
- *Success criterion*: this parameter specifies the criterion for a successful completion of the session setup operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall setup either all flow connections successfully, or in the event that it is unable to setup a flow connection, it does not setup any of them. If BestEffort is specified, the CP shall attempt to setup as many (at least one) flow connections as possible. This is an optional parameter and is included when two or more flow connections are to be setup as part of the session setup.
- *Initial Administrative State* of the connectivity session; the possible values are Locked and Unlocked.[6]

ConS-U-2: The CU shall provide (and the CP shall accept) the following information for each flow connection that is to be setup during the setup of a connectivity session[7]:

- *Flow connection identifier*.
- *Connection topology* which can be either point-to-point bidirectional, or point-to-multipoint unidirectional[8]. A point-to-point bidirectional flow connection has two flow endpoints, one designated as the *root* endpoint and the other as the *leaf* endpoint, and information can be sent from either endpoint to the other. A point-to-multipoint unidirectional flow connection has two or more endpoints, one of them

5. The connectivity session is represented by the physical connection graph at the information model level.
6. See Section 2.3 for further details on the semantics of the administrative state concept.
7. Setup of several flow connections in one operation is allowed. The interface reference to the terminal(s) for the negotiation of the Flow End Point (FEP) might have to be passed through the ConS-RP. For lack of information on how this reference is passed, in this version, no reference is passed through the ConS-RP.
8. In this version of the specification, only pt-pt bidirectional FCs and pt-mpt unidirectional FCs are supported. A point-to-point unidirectional topology can be realized either by specifying zero bandwidth for one direction in a pt-pt bidirectional connection, or by using a pt-mpt connection with only two flow endpoints. Other multipoint topologies have to be realized using multiple flow connections.

designated as the *root* endpoint and others as the *leaf* endpoints. In a point-to-multipoint unidirectional flow connection, information flows only from the root endpoint to the leaf endpoints.

- *Success criterion*: this optional parameter is included only if the Connection Topology is point-to-multipoint unidirectional. This parameter specifies the criterion for a successful completion of the multipoint flow connection setup; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall either setup connectivity to all leaf endpoints, or in the event that it is unable to setup connectivity to a leaf endpoint, it does not setup the multipoint flow connection. If BestEffort is specified, the CP shall attempt to setup connectivity to as many (at least one) leaf endpoints as possible.
- *Traffic type* which can be one of the following: Constant Bit Rate (CBR), Variable Bit Rate (VBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR). A CBR traffic has only one bandwidth attribute: peak bandwidth; a VBR traffic has two bandwidth attributes: peak bandwidth and average bandwidth; an ABR traffic also has two bandwidth attributes: peak bandwidth and minimum bandwidth; and a UBR traffic has no bandwidth attribute. Further, only the CBR and VBR traffic have delay parameters. These traffic types are distinguished to enable the CP to manage the network resources in an efficient manner and yet guarantee to a CU the bandwidth it specifies for a flow connection.
- *Reliability Class* which can be one of the following: ReleaseOnFailure, and HoldOnFailure. If the class is ReleaseOnFailure, the flow connection is released when it fails. If the class is HoldOnFailure, the flow connection is not released when it fails, but information flow is suspended. Subsequently, when the flow connection becomes operational again, information flow is resumed.[9]
- *Routing Constraint* which is a pair of the form <Constraint Type, Reference Flow Connection>, where Constraint Type is either SameRoute or DifferentRoute, and Reference Flow Connection is an identifier that identifies another flow connection that is a part of the same connectivity session. If the Constraint Type is SameRoute, the CP shall setup the flow connection, if possible, along a path (in terms of switches and links between the switches) that is the same as the route of the referenced flow connection. If the Constraint Type is DifferentRoute, the CP shall setup the flow connection, if possible, along a path (in terms of switches and links between the switches) that is different from the route of the referenced flow connection. Note that the CP makes a best effort to satisfy the routing constraint, but does not guarantee that the constraint will be met. (The CP shall indicate in its response to the CU whether the constraint was met. See ConS-U-3.).
- *Initial Administrative State*: this parameter is optional and can be specified only if the initial administrative state of the connectivity session is unlocked. It is used to specify whether or not the flow

9. It is expected that in either case the CP will attempt to reroute a flow connection when it fails before the CP decides to hold or release the flow connection.

connection should be automatically activated upon setup. If Unlocked is specified, the CP shall set the administrative state of the flow connection to Unlocked, and the flow connection can transport information immediately upon setup. If Locked is specified, the CP shall set the administrative state of the flow connection to Locked, and the flow connection cannot transport information upon setup. At some later time, the CU can explicitly set the administrative state to Unlocked (see ConS-U-14).

For a flow connection, bandwidth and QoS parameters (described below) are specified individually for each endpoint and they are interpreted as follows.[10] For a point-to-point connection, the bandwidth and QoS parameters specified for the root endpoint are associated with the traffic from the root endpoint to the leaf endpoint. Similarly, the bandwidth and QoS parameters specified for the leaf endpoint are associated with the traffic from the leaf endpoint to the root endpoint. For a point-to-multipoint connection, the bandwidth and QoS parameters specified for the root endpoint are taken as the default bandwidth and QoS parameters for the traffic to each leaf endpoint. If, however, bandwidth and QoS parameters are specified for a leaf endpoint, they are associated with the traffic from the root endpoint to the specific leaf endpoint, and they override the default value for the traffic to the specific leaf endpoint. For a point-to-multipoint connection, the bandwidth and QoS parameters specified for a leaf endpoint should be “lesser” than the parameters specified for the root endpoint.

- For each endpoint of the flow connection, the CU shall provide the following information:
    - *Flow Endpoint type* (root or leaf). This identification is required since traffic descriptors are interpreted differently for root endpoints and leaf endpoints.
    - *Flow Endpoint Name*: The flow endpoint name can identify either a **Network Flow Endpoint Pool** or a specific flow end point (including a specific channel name, such as VPI/VCI values in the case of ATM). If the flow endpoint name identifies only a flow endpoint pool, implicit channel selection is used; i.e., the CP makes the channel selection. It is assumed that the flow type name is a part of the endpoint name.
    - *Peak Bandwidth* (specified in units of Mbits per second; applicable only for CBR, VBR, and ABR)[11]
    - *Average Bandwidth* (specified in units of Mbits per second); applicable only if traffic type is VBR or ABR; for ABR, this is interpreted as the minimum bandwidth.)
    - *Maximum Delay* (specified in units of milliseconds; applicable only for CBR and VBR)
    - *Maximum Variation in Delay* (specified in units of milliseconds; applicable only for CBR and VBR)

10. This initial version of ConS-RP requirements addresses only one QoS parameter, i.e., delay.
11. Here and elsewhere in this document, the term “unit” denotes a number that includes fractions.

- *Maximum Error Rate* (specified in bits per second; applicable only for CBR, VBR, and ABR)
- *Initial Administrative State* of the endpoint; the possible values are Locked and Unlocked, and it can be specified only if the initial administrative state of the flow connection is unlocked.

ConS-U-3: If the connectivity session is setup successfully, the CP shall send a response to the CU with the following information on each flow connection that has been setup:

- Flow Connection Identifier
- Root Endpoint Name (this name shall also indicate the channel selection)
- Bound Leaf Endpoint Names (the names of the leaf endpoints that have been bound to the root endpoint; these names shall indicate the channel selection)
- Unbound Leaf Endpoints List; this is a list of pairs <Unbound Leaf Endpoint Name, Failure Code>
- an indication whether the routing constraint specified by the CU was met

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadBound: the endpoint identified by the CU is already part of another flow connection.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

For each flow connection that the CP could not setup, the CP shall report the following information to the CU in its success response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network

- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadBound: the endpoint identified by the CU is already part of another flow connection.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-4: The CP shall report a failure to the CU in response to a connectivity session setup request, if there is a security violation, or the CU specified AllorNone and the CP is unable to setup one of the flow connections specified, or the CU specified BestEffort and the CP is unable to setup even one flow connection.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is due to other reasons, the CP shall report the following information to the CU for each flow connection that the CP could not setup:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadBound: the endpoint identified by the CU is already part of another flow connection.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

Flow Connection Setup[12]

12. The setup of multiple Flow Connections in one operation is supported. A request for the setup of multiple flow connections shall include a Success Criterion parameter.

ConS-U-5: The CP shall provide a function for setting up a flow connection in the context of a connectivity session. This function shall take the following input parameters:

- Connectivity session identifier; this parameter identifies the connectivity session into which the flow connection is to be included.
- Information on the flow connection (as specified in ConS-U-2)

ConS-U-6: If the flow connection is setup successfully, the CP shall send a response to the CU with the following information on the flow connection that has been setup:

- Flow Connection Identifier
- Root Endpoint Name (this name shall also indicate the channel selection)
- Bound Leaf Endpoint Names (the names of the leaf endpoints that have been bound to the root endpoint; these names shall indicate the channel selection)
- Unbound Leaf Endpoints List; this is a list of pairs <Unbound Leaf Endpoint Name, Failure Code>
- an indication whether the routing constraint specified by the CU was met

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadBound: the endpoint identified by the CU is already part of another flow connection.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-7: The CP shall report a failure to the CU in response to a flow connection setup request, if there is a security violation, or the CU specified AllorNone and the CP is unable to setup connectivity to all endpoints specified, or the CU specified BestEffort and the CP is unable to setup connectivity to even one flow endpoint.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is due to other reasons, the CP shall report the following information to the CU in its failure response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadyBound: the endpoint identified by the CU is already part of another flow connection.
- NetworkFailure: one or more network resources that support connectivity to the leaf endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

Network flow endpoints Addition

ConS-U-8: The CP shall provide a function for adding new flow endpoints (leaf endpoints) to an existing point-to-multipoint flow connection. This function shall accept the following input parameters:

- Flow Connection Identifier
- New Flow Endpoints Information (specified below)
- Success Criterion: This parameter specifies the criterion for a successful completion of the flow endpoints addition operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall setup connectivity to either all specified leaf endpoints, or in the event that it is unable to setup connectivity to a leaf endpoint, it does not setup connectivity to any of the specified flow endpoints. If BestEffort is specified, the CP shall attempt to setup connectivity to as many (at least one) leaf endpoints as possible

The CU shall provide the following information on each flow endpoint that is to be added to the flow connection

- *Flow Endpoint Name*
- *Peak Bandwidth* (specified in units of Mbits per second)
- *Average Bandwidth* (specified in units of Mbits per second); applicable only if traffic type is VBR.)
- *Maximum Delay* (specified in units of milliseconds)

- *Maximum Variation in Delay* (specified in units of milliseconds)
- *Maximum Error Rate* (specified in bits per second)
- *Initial Administrative State*

ConS-U-9: If endpoints addition is successful, the CP shall send a response to the CU with the following information on the flow connection that has been modified:

- Bound New Leaf Endpoint Names (the names of the leaf endpoints that have been bound to the root endpoint; these names shall indicate the channel selection)
- Unbound Leaf Endpoints List; this is a list of pairs <Unbound Leaf Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadyBound: the endpoint identified by the CU is already part of another flow connection.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-10: The CP shall report a failure to the CU in response to a flow endpoints addition request, if there is a security violation, or the CU specified AllorNone and the CP is unable to setup connectivity to all endpoints specified, or the CU specified BestEffort and the CP is unable to setup connectivity to even one flow endpoint.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is due to other reasons, the CP shall report the following information to the CU in its failure response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network

- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointAlreadBound: the endpoint identified by the CU is already part of another flow connection.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

Network flow Endpoints Deletion

ConS-U-11: The CP shall provide a function for deleting one or more flow endpoints (leaf endpoints) from an existing point-to-multipoint flow connection. This function shall accept the following input parameters:

- Flow Connection Identifier
- Names of Flow Endpoints to be deleted
- Success Criterion: This parameter specifies the criterion for a successful completion of the flow endpoints deletion operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall either remove connectivity to all specified leaf endpoints, or in the event that it is unable to remove connectivity to a leaf endpoint, it does not remove connectivity to any of the specified flow endpoints. If BestEffort is specified, the CP shall attempt to remove connectivity to as many (at least one) leaf endpoints as possible.

ConS-U-12: If endpoints deletion is successful, the CP shall send a response to the CU with the following information on the flow connection that has been modified:

- Unbound Leaf Endpoint Names (the names of the leaf endpoints that have been removed from the flow connection)
- Bound Leaf Endpoints List; this is a list of pairs <Bound Leaf Endpoint Name, Failure Code>

Failure Code is either

- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not part of the specified flow connection.
- NetworkFailure: one or more network resources that support connectivity to the leaf endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-13: The CP shall report a failure to the CU in response to a flow endpoints deletion request, if there is a security violation, or the CU specified AllorNone and the CP is unable to remove connectivity to all endpoints specified, or the CU specified BestEffort and the CP is unable to remove connectivity to even one flow endpoint, or the CU attempts to delete all flow endpoints of a flow connection.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is because the CU attempts to delete all endpoints of a flow connection, the CP shall send a failure response to the CU indicating this error condition.

If the failure is due to other reasons, the CP shall report the following information to the CU in its failure response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not a part of the specified flow connection.
- NetworkFailure: one or more network resources that support connectivity to the leaf endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

Flow Connection Activation[13]

ConS-U-14: The CP shall provide a function for setting up the administrative state of a flow connection to Unlocked state. Only when the administrative state is Unlocked, the flow connection transports information between its endpoints. This function shall accept the following input parameters:

- Flow Connection Identifier

ConS-U-15: The processing of a flow connection activation request is deemed successful if the CP is able to activate all network resources (i.e., cross connections and link connections) that support the flow connection, and the flow connection is ready for information transport.

ConS-U-16: If flow connection activation is successful, the CP shall send a response to the CU indicating that the flow connection has been activated.

ConS-U-17: The CP shall report a failure to the CU in response to a flow connection activation request if at least one of the following conditions holds:

13. The activation of multiple Flow Connections in one operation is supported. A request for the activation of multiple flow connections shall include a Success Criterion parameter.

- Flow connection identifier is invalid (i.e., the referenced flow connection is nonexistent)
- The flow connection has been activated already (i.e., the administrative state is Unlocked)
- Security access violation: (NotAuthenticated, NotAuthorized)
- The CP is unable to activate all network resources that support the flow connection
- One or more network resources that support the flow connection have failed.
- The CP is unable to access the network resources due to failure of the Kernel Transport Network.

If any of the above conditions holds, the CP shall send a failure response to the CU indicating the failure cause.

ConS-U-18: When a flow connection activation request fails, the administrative state of the flow connection remains unchanged.[14]

Flow Connection Deactivation[15]

ConS-U-19: The CP shall provide a function for setting up the administrative state of a flow connection to Locked state. When the administrative state is set to Locked, transport of information over the flow connection is suspended. This function shall accept the following input parameters:

- Flow Connection Identifier

ConS-U-20: The processing of a flow connection deactivation request is deemed successful if the CP is able to deactivate all network resources (i.e., cross connections and link connections) that support the flow connection, and the transport of information over the flow connection is suspended.

ConS-U-21: If flow connection deactivation is successful, the CP shall send a response to the CU indicating that the flow connection has been deactivated.

ConS-U-22: The CP shall report a failure to the CU in response to a flow connection deactivation request if at least one of the following conditions holds:

- Flow connection identifier is invalid (i.e., the referenced flow connection is nonexistent)
- The flow connection has been deactivated already (i.e., the administrative state is Locked)
- Security access violation: (NotAuthenticated, NotAuthorized)
- The CP is unable to deactivate all network resources that support the flow connection
- One or more network resources that support the flow connection have failed.

14. This implies that the CP shall perform necessary compensating actions to undo actions that it may have performed during the course of activating the flow connection.

15. The deactivation of multiple Flow Connections in one operation is supported. A request for the deactivation of multiple flow connections shall include a Success Criterion parameter.

- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

If any of the above conditions holds, the CP shall send a failure response to the CU indicating the failure cause.

ConS-U-23: When a flow connection deactivation request fails, the administrative state of the flow connection remains unchanged.[16]

Flow Connection Branches Deactivation

ConS-U-24: The CP shall provide a function for selectively disabling traffic from the root endpoint of a point-to-multipoint flow connection to one or more leaf endpoints. Traffic to a specific leaf endpoint is disabled when the administrative state of the leaf endpoint is Locked. This function shall accept the following input parameters:

- Flow Connection Identifier
- Leaf Flow Endpoint Names
- Success Criterion: This parameter specifies the criterion for a successful completion of the flow connection branches deactivation operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall either disable traffic to all specified leaf endpoints, or in the event that it is unable to disable traffic to a leaf endpoint, it does not disable traffic to any of the specified flow endpoints. If BestEffort is specified, the CP shall attempt to disable traffic to as many (at least one) leaf endpoints as possible.

ConS-U-25: If branches deactivation is successful, the CP shall send a response to the CU with the following information on the flow connection that has been modified:

- Deactivated Leaf Endpoint Names (the names of the leaf endpoints to which traffic has been suspended)
- Failed Leaf Endpoints List; this is a list of pairs <Failed Leaf Endpoint Name, Failure Code>

Failure Code is either

- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not part of the specified flow connection.
- FlowEndpointAlreadyDeactivated: the flow endpoint has been deactivated already (i.e., the administrative state is Locked)
- NetworkFailure: one or more network resources that support connectivity to the flow endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

16. This implies that the CP shall perform necessary compensating actions to undo actions that it may have performed during the course of deactivating the flow connection.

ConS-U-26: The CP shall report a failure to the CU in response to a flow branches deactivation request, if there is a security violation, or the CU specified AllorNone and the CP is unable to deactivate all branches specified, or the CU specified BestEffort and the CP is unable to deactivate even one branch.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is due to other reasons, the CP shall report the following information to the CU in its failure response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not a part of the specified flow connection.
- FlowEndpointAlreadyDeactivated: the flow endpoint has been deactivated already (i.e., the administrative state is Locked)
- NetworkFailure: one or more network resources that support connectivity to the flow endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-27: When a flow connection branches deactivation request fails, the administrative state of the specified branches remains unchanged.

Flow Connection Branches Activation

ConS-U-28: The CP shall provide a function for selectively enabling traffic from the root endpoint of a point-to-multipoint flow connection to one or more leaf endpoints. Traffic to a specific leaf endpoint is enabled when the administrative state of the leaf endpoint is Unlocked. This function shall accept the following input parameters:

- Flow Connection Identifier
- Leaf Flow Endpoint Names
- Success Criterion: This parameter specifies the criterion for a successful completion of the flow connection branches activation operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall either enable traffic to all specified leaf endpoints, or in the event that it is unable to enable traffic to a leaf endpoint, it does not enable traffic to any of the specified flow endpoints. If BestEffort is specified, the CP shall attempt to enable traffic to as many (at least one) leaf endpoints as possible.

ConS-U-29: If branches activation is successful, the CP shall send a response to the CU with the following information on the flow connection that has been modified:

- Activated Leaf Endpoint Names (the names of the leaf endpoints to which traffic has been enabled)
- Failed Leaf Endpoints List; this is a list of pairs <Failed Leaf Endpoint Name, Failure Code>

Failure Code is either

- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not part of the specified flow connection.
- FlowEndpointAlreadyActivated: the flow endpoint has been activated already (i.e., the administrative state is Unlocked)
- NetworkFailure: the CP is unable to activate all network resources that support connectivity to the flow endpoint
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-30: The CP shall report a failure to the CU in response to a flow branches activation request, if there is a security violation, or the CU specified AllorNone and the CP is unable to activate all branches specified, or the CU specified BestEffort and the CP is unable to activate even one branch.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is due to other reasons, the CP shall report the following information to the CU in its failure response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not a part of the specified flow connection.
- FlowEndpointAlreadyActivated: the flow endpoint has been activated already (i.e., the administrative state is Unlocked)
- NetworkFailure: the CP is unable to activate all network resources that support connectivity to the flow endpoint
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-31: When a flow connection branches activation request fails, the administrative state of the specified branches remains unchanged.

Flow Connection Modification

ConS-U-32: The CP shall provide a function for modifying the bandwidth and QoS parameters of one or more branches of a flow connection. This function shall accept the following input parameters:

- Flow Connection Identifier
- *Success criterion*: this optional parameter is included only if the Connection Topology is point-to-multipoint unidirectional. This parameter specifies the criterion for a successful completion of the multipoint flow connection modification operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall either modify connectivity to all specified leaf endpoints, or in the event that it is unable to modify connectivity to a leaf endpoint, it does not modify the multipoint flow connection. If BestEffort is specified, the CP shall attempt to modify connectivity to as many (at least one) leaf endpoints as possible.
- For each branch of the flow connection that is to be modified, the CU shall provide the following information:
  - *Flow Endpoint Name*[17]
  - *Peak Bandwidth* (specified in units of Mbits per second)
  - *Average Bandwidth* (specified in units of Mbits per second); applicable only if traffic type is VBR.
  - *Maximum Delay* (specified in units of milliseconds)
  - *Maximum Variation in Delay* (specified in units of milliseconds)
  - *Maximum Error Rate* (specified in bits per second)

ConS-U-33: If flow connection modification is successful, the CP shall send a response to the CU with the following information on the flow connection that has been modified:

- Modified Leaf Endpoint Names (the names of the leaf endpoints that have been modified)
- Unmodified Leaf Endpoints List; this is a list of pairs <Unmodified Leaf Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.

17. Note that the endpoint name is used to identify the branch to be modified.

- FlowEndPointNotAPart: the endpoint identified by the CU is not part of the flow connection.
- NetworkFailure: one or more network resources that support connectivity to the flow endpoint have failed
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-34: The CP shall report a failure to the CU in response to a flow connection modification request, if there is a security violation, or the CU specified AllorNone and the CP is unable to modify all endpoints specified, or the CU specified BestEffort and the CP is unable to modify even one flow endpoint.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is due to other reasons, the CP shall report the following information to the CU in its failure response:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- InsufficientBandwidth: there is no sufficient bandwidth available in the connectivity layer network
- NoPathFound: there is no possible path to the failed endpoint in the connectivity layer network
- QoSCannotBeMet: the required QoS cannot be provided
- InsufficientResources: there is insufficiency of resources, such as channel numbers (e.g., VPI/VCI values).
- NonexistentFlowEndpoint: the endpoint identified by the CU does not exist.
- FlowEndPointNotAPart: the endpoint identified by the CU is not part of the flow connection.
- NetworkFailure: one or more network resources that support connectivity to the flow endpoint have failed
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

Flow Connection Release[18]

ConS-U-35: The CP shall provide a function for releasing a flow connection. When a flow connection is released, all network resources (subnetwork connections and link connections) that support the flow connection are released, and transport of information over the flow connection is stopped. This function shall accept the following input parameters:

18. Release of multiple Flow Connections in one operation is supported.

- Flow Connection Identifier

ConS-U-36: The CP shall treat a flow connection release request as invalid if at least one of the following conditions holds:

- Flow connection identifier is invalid (i.e., the referenced flow connection is nonexistent)
- Security access violation: (NotAuthenticated, NotAuthorized)

If any of the above conditions holds, the CP shall send a failure response to the CU indicating the failure cause.

ConS-U-37: The CP shall process a valid flow connection release request in the following manner. First, it shall send a response to the CU indicating that the flow connection will be released. Then, it shall attempt to release immediately all network resources that support the flow connection. If this is not possible (either due to network element failure or KTN failure), it shall record the release request, and retry to release the network resources at a later time. It shall ensure that the network resources are released eventually.

Connectivity Session Activation

ConS-U-38: The CP shall provide a function for setting up the administrative state of a connectivity session to Unlocked state. When the administrative state of a connectivity session is Unlocked, all flow connections that are part of the session are also activated. This function shall accept the following input parameters:

- Connectivity Session Identifier
- *Success criterion*: this parameter specifies the criterion for a successful completion of the session activation operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall either activate all flow connections on AllorNone basis, or in the event that it is unable to activate a flow connection, it does not activate any of the flow connections. If BestEffort is specified, the CU shall attempt to activate as many (at least one) flow connections as possible on BestEffort basis. This is an optional parameter and is included when two or more flow connections are part of the session.

ConS-U-39: If connectivity session activation is successful, the CP shall send a response to the CU with the following information on each flow connection of the connectivity session:

- Flow Connection Identifier
- Activated Leaf Endpoint Names (the names of the leaf endpoints of the flow connection to which traffic has been enabled)
- Failed Leaf Endpoints List; this is a list of pairs <Failed Leaf Endpoint Name, Failure Code>

Failure Code is either

- NetworkFailure: one or more network resources that support connectivity to the leaf endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-40: The CP shall report a failure to the CU in response to a connectivity session activation request, if there is a security violation, or the connectivity session has been activated already, or the CU specified AllorNone and the CP is unable to activate all flow connections of the connectivity session, or the CU specified BestEffort and the CP is unable to activate even one flow connection.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

- Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is because the connectivity session has been activated already, the CP shall send a failure response to the CU indicating this error condition.

If the failure is due to other reasons, the CP shall report in its failure response to the CU the following information on each flow connection of the connectivity session:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- NetworkFailure: one or more network resources that support connectivity to the leaf endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-41: When a connectivity session activation request fails, the administrative state of the connectivity session remains unchanged.

Connectivity Session Deactivation

ConS-U-42: The CP shall provide a function for setting up the administrative state of a connectivity session to Locked state. When the administrative state of a connectivity session is Locked, all flow connections that are part of the session are deactivated. This function shall accept the following input parameters:

- Connectivity Session Identifier
- *Success criterion*: this parameter specifies the criterion for a successful completion of the session activation operation; the possible values are AllorNone, and BestEffort. If AllorNone is specified, the CP shall deactivate either all flow connections on AllorNone basis, or in the event that it is unable to deactivate a flow connection, it does not deactivate any of the flow connections. If BestEffort is specified, the CU shall attempt to deactivate, on BestEffort basis, as many (at least one) flow connections as possible. This is an optional parameter and is included when two or more flow connections are part of the session.

ConS-U-43: The CP shall report a failure to the CU in response to a connectivity session deactivation request, if there is a security violation, or the connectivity session has been deactivated already, or the CU specified AllorNone and the CP is unable to deactivate all flow connections of the connectivity session, or the CU specified BestEffort and the CP is unable to deactivate even one flow connection.

If the failure is due to security violation, the CP shall send a failure response to the CU supplying the following information:

Security violation code: the possible codes are NotAuthenticated, and NotAuthorized.

If the failure is because the connectivity session has been deactivated already, the CP shall send a failure response to the CU indicating this error condition.

If the failure is due to other reasons, the CP shall report in its failure response to the CU the following information on each flow connection of the connectivity session:

- Flow Connection Identifier
- Failed Endpoints List; this is a list of pairs <Failed Endpoint Name, Failure Code>

Failure Code is either

- NetworkFailure: one or more network resources that support connectivity to the leaf endpoint have failed.
- KTNFailure: the CP is unable to access the network resources due to failure of the Kernel Transport Network.

ConS-U-44: When a connectivity session deactivation request fails, the administrative state of the connectivity session remains unchanged.

Connectivity Session Release

ConS-U-45: The CP shall provide a function for releasing a connectivity session. When a connectivity session is released, all flow connections that are part of the connectivity session are released. This function shall accept the following input parameters:

- Connectivity Session Identifier

ConS-U-46: The CP shall treat a connectivity session release request as invalid if at least one of the following conditions holds:

- Connectivity session identifier is invalid (i.e., the referenced connectivity session is nonexistent)
- Security access violation: (NotAuthenticated, NotAuthorized)

If any of the above conditions holds, the CP shall send a failure response to the CU indicating the failure cause.

ConS-U-47: The CP shall process a valid connectivity session release request in the following manner. First, it shall send a response to the CU indicating that the connectivity session will be released. Then, it shall attempt to release immediately all flow connections that are part of the connectivity session. If some flow connections cannot be released immediately (either due to network element failure or KTN failure), it shall record these conditions, and retry to release such flow connections at a later time. It shall ensure that all flow connections are released eventually.

Fault Management

ConS-U-48: When connectivity is lost between the root endpoint of a flow connection and one or more leaf endpoints (due to network failure), the CP shall send a notification to the CU with the following information:

- Flow Connection Identifier
- List of Flow Leaf Endpoints to which connectivity is lost

The CP sends this notification only when it is unable to reroute the flow connection. If the reliability class of the failed flow connection is ReleaseonFailure, the flow connection is released by the CP when the flow connection fails.

ConS-U-49: When connectivity is restored between the root endpoint of a flow connection and one or more leaf endpoints, the CP shall send a notification to the CU with the following information:[19]

- Flow Connection Identifier
- List of Flow Leaf Endpoints to which connectivity is restored

Contract profile modification

ConS-U-50: The CP shall provide a function for the CU to modify the contract profile parameters associated with the CU.

The contract profile has the following parameters[20]:

- Security parameters
  - Password and key information
  - Security procedure to be used
  - Reauthentication interval and other security parameters
- Connectivity session parameters
  - Default traffic type: CBR, VBR, ABR, or UBR.
  - Default reliability class: releaseonFailure or HoldonFailure.
  - Default initial administrative state for connectivity sessions: Locked or Unlocked.
  - Default initial administrative state for flow connections: Locked or Unlocked.

In a contract profile modification request, a CU may request modification of one or more of the contract profile parameters listed above.

ConS-U-51: Upon receipt of a contract modification request, the CP shall first check the validity of the request. The CP shall treat the request as invalid if at least one of the following conditions holds:

19. This is a conditional requirement and is applicable only for flow connections whose reliability class is HoldOnFailure.

20. It is expected that future versions of ConSRP specifications will support parameters related to all management functional areas.

- Invalid security parameters, unable to perform the requested degree of security.
- Invalid security procedure: procedure not supported.
- Invalid traffic type.
- Invalid reliability class.
- Invalid initial administrative state.

If any of the above conditions holds, the CP shall send a response to the CU indicating the invalidity of the request.

ConS-U-52: If the request is valid, the CP shall update the contract profile with the CU.

ConS-U-53: If the contract profile is updated successfully, the CP shall send a confirmation to the CU.

Miscellaneous Query Capabilities

ConS-U-54: The CP shall provide a function for retrieving information on a specific connectivity session. This function shall take the following input parameters:

- Connectivity Session Identifier

ConS-U-55: Upon receipt of a connectivity session query request, the CP shall first check the validity of the request. The CP shall treat the request as invalid if any of the following conditions holds:

- Connectivity session identifier is invalid (i.e., the referenced connectivity session is nonexistent)
- Security access violation: (NotAuthenticated, NotAuthorized)

If the request is invalid, the CP shall send a response to the CU indicating the invalidity of the request.

If the request is valid, the CP shall send a response to the CU with the following information:

- List of identifiers of flow connections that are part of the specified connectivity session

ConS-U-56: The CP shall provide a function for retrieving information on a specific flow connection. This function shall take the following input parameters:

- Flow Connection Identifier

ConS-U-57: Upon receipt of a flow connection query request, the CP shall first check the validity of the request. The CP shall treat a request as invalid if any of the following conditions holds:

- Flow Connection identifier is invalid (i.e., the referenced flow connection is nonexistent)
- Security access violation: (NotAuthenticated, NotAuthorized)

If the request is invalid, the CP shall send a response to the CU indicating the invalidity of the request.

If the request is valid, the CP shall send a response to the CU with the following information:

- Connection Topology (point-to-point, point-to-multipoint unidirectional)
- Traffic Type
- Reliability Class
- Routing Constraint
- Operational State: the possible values are Failed, Operational, Degraded. The Operational State attribute has the value Degraded when connectivity is lost to some (but not all) of the leaf flow endpoints.
- For each flow endpoint, the following information is included in the response:
  - Flow Endpoint Name
  - Flow Endpoint Type (Root or Leaf)
  - Peak Bandwidth (as specified at setup/modification time)
  - Average Bandwidth (as specified at setup/modification time)
  - Maximum Delay (as specified at setup/modification time)
  - Maximum Variation in Delay (as specified at setup/modification time)
  - Maximum Error Rate (as specified at setup/modification time)
  - Administrative State
  - Operational State

ConS-U-58: The CP shall provide a function by which the CU can find out if two flow endpoints of the connectivity layer network can be potentially bound.[21] This determination is made purely based on the topology of the connectivity layer network including the inter layer network adaptation capabilities that exist in the connectivity layer network.

This function shall take as input two flow endpoint names and return a boolean response: the value “true” indicates that the endpoints can be potentially bound, and the value “false” indicates that the two endpoints cannot be potentially bound.

21. This requirement is useful only if the connectivity layer network consists of more than one layer network.

2.3 State diagram

This section clarifies the notion of administrative states as used in this document. A connectivity session and its component flow connections have separate administrative states, but they are related as follows. If the administrative state of a connectivity session is Unlocked, then the administrative state of a component flow connection may be either Unlocked or Locked; it can be individually changed. But, if the administrative state of a connectivity session is Locked, then it implies that the administrative state of each component flow connection is Locked. A similar relationship holds between the administrative state of a flow connection and the administrative states of its branches. If the administrative state of a flow connection is Unlocked, then the administrative state of a branch of the flow connection may be either Unlocked or Locked; it can be individually changed. But, if the administrative state of a flow connection is Locked, then it implies that the administrative state of each branch of the flow connection is Locked.

Figure 2-1 illustrates these relationships through a state diagram.

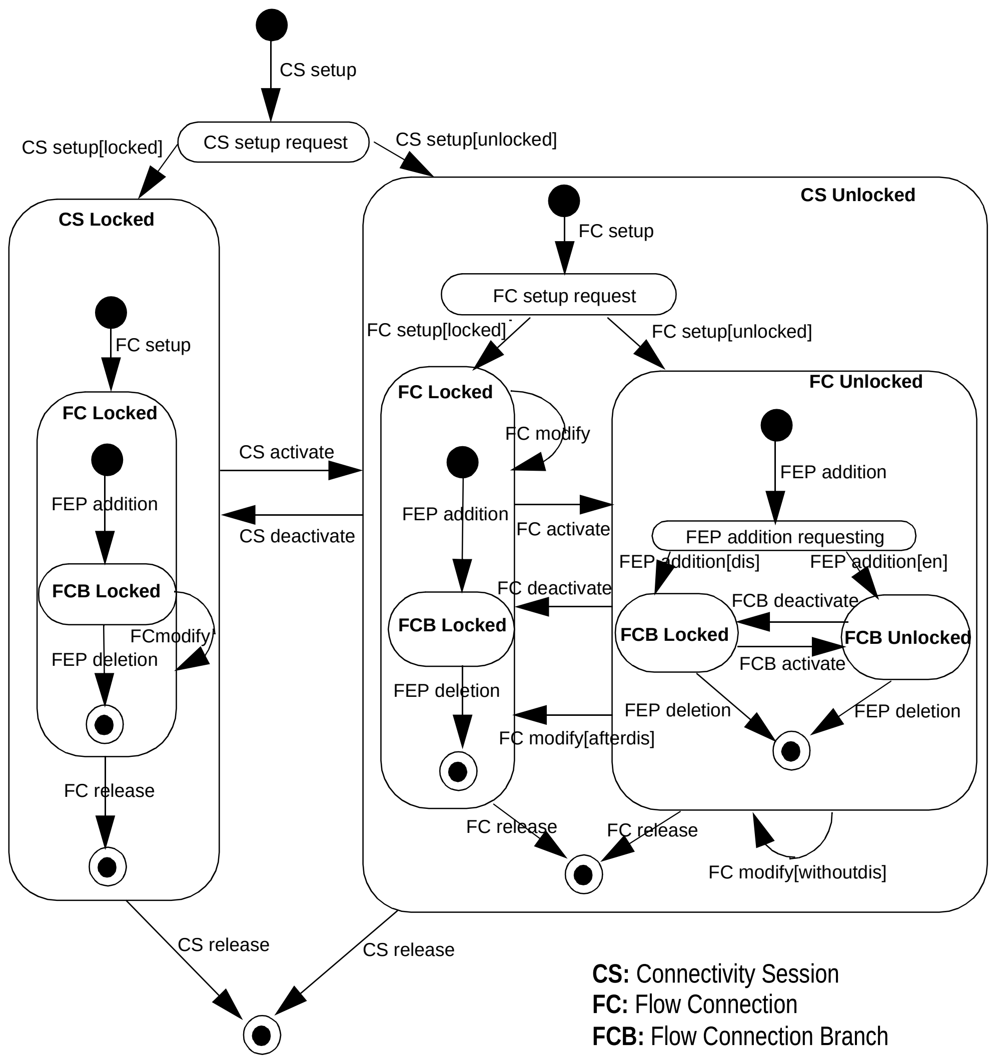


Figure 2-1. State Diagram for Administrative States

3. ConS-RP Information Model

3.1 Access part

Associated with each connectivity user is a contract profile object that contains some context (or profile) information pertaining to the connectivity user. This object is created (outside of the ConS-RP) and exists as long as the business relationship between the CU and the CP exists. The contract profile can be queried and modified via the usage part of the ConS-RP. When the business relationship between a CU and the CP is terminated, the contract profile object associated with the CU is deleted via the ConS-RP.

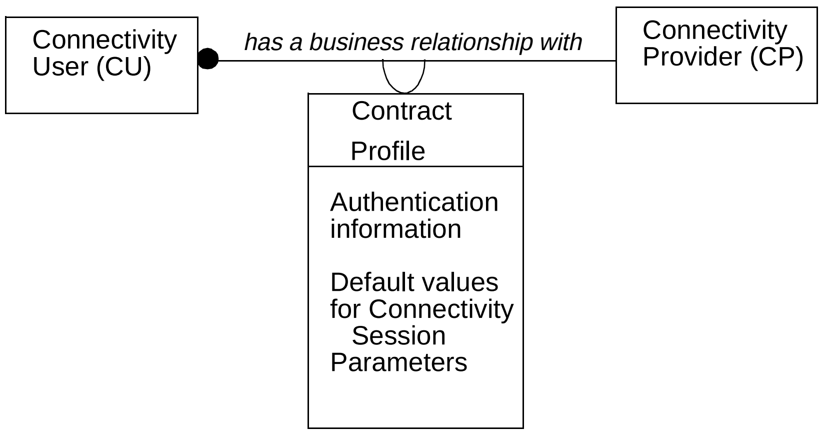

Connectivity User (CU)
has a business relationship with
Connectivity Provider (CP)
Contract Profile
Authentication information
Default values for Connectivity Session Parameters

Figure 3-1. OMT diagram for ConS-RP Access part

- **Connectivity User**
  - An instance of this object type represents an entity that plays the user role in the connectivity reference point
- **Connectivity Provider**
  - An instance of this object type represents the entity that plays the provider role in the connectivity reference point
- **Has a Business Relationship With**
  - An instance of this association represents the business relationship between a CU and a CP
- **Contract profile**
  - An instance of this object type represents information pertaining to the business relationship between a CU and a CP
  - This object exists only when the business relationship between the CU and the CP exists
  - Attributes:

- Authentication information (details not specified in this document). Need the following information: degree of trust, authentication protocol to be used, key information, reauthentication interval, etc.
- Default values for the following parameters: traffic type, reliability class, and initial administrative state for flow connections; initial administrative state for connectivity sessions

3.2 Usage part

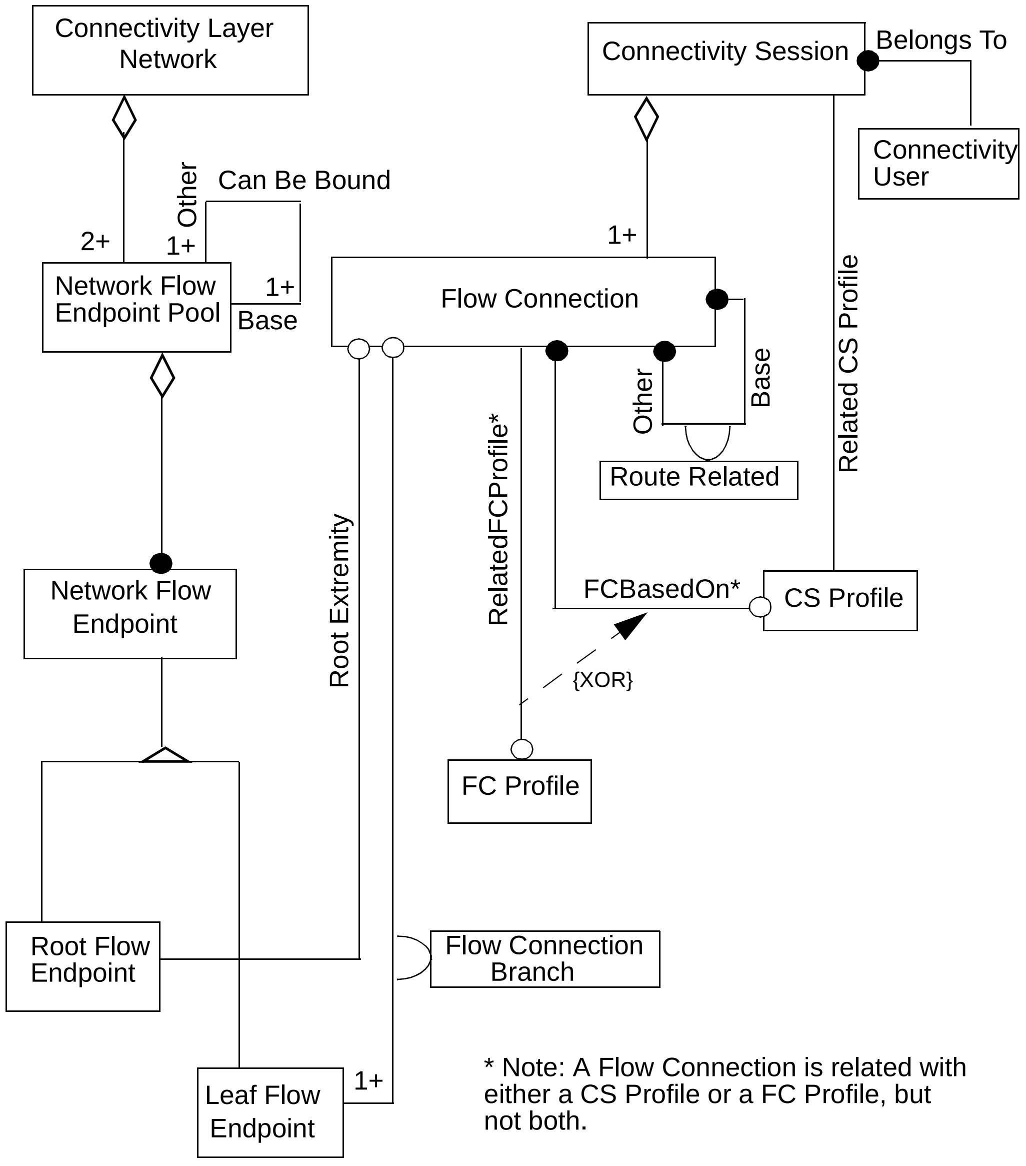


Figure 3-2. OMT diagram for ConS-RP Usage part

- **Connectivity Layer Network**
  - There is only one instance of this object type. It represents the connectivity layer network domain that is under the purview of the CP. In this version of the ConS-RP specification, it is assumed that the entire connectivity layer network is under the purview of a single administration; i.e., there is only one CP domain.
  - Attribute: Connectivity Layer Network Name
- **Network Flow Endpoint Pool**
  - An instance of this object type represents a collection of flow endpoints that are collocated on a CPE.
  - A Network Flow Endpoint Pool has a one to one association with a Network Trail termination Point Pool. Thus, a Network Flow EndPoint Pool does not span layer networks.
  - Attributes
    - NFEPpool Name (this name may include the name of the administrative domain to which the pool belongs; naming conventions are not pre-scribed in this document.)
    - Characteristic information
    - Operational State
    - Administrative State
  - Notifications
    - Creation
    - Deletion

    Example in ATM: NFEPpool = NW address (E164) or
    NW address + (range of) VPIs/VCIs
- **Network Flow Endpoint**
  - This is a non instantiable object type and is defined only for inheritance.
  - This object type represents a flow endpoint. Flow endpoints may be either preconfigured or created only during a flow connection setup.
  - Attributes
    - NFEP Name (Channel Name)
    - Peak Bandwidth
    - Average Bandwidth
    - Maximum Delay
    - Maximum Delay Variation
    - Maximum Error Rate
    - Technology Dependent QoS information

    Example in ATM: NFEP = NFEPpool + VPI (for the setup of a VPC) or
    NFEPpool + VPI/VCI

- **Root Flow Endpoint**
  - A subtype of Network Flow Endpoint
  - An instance of this object type represents the root flow endpoint of a flow connection.
  - The root-leaf distinction is arbitrary for a bidirectional point-to-point flow connection.
- **Leaf Flow Endpoint**
  - A subtype of Network Flow Endpoint
  - An instance of this object type represents a leaf flow endpoint of a flow connection.
- **Can Be Bound**
  - An instance of this association exists for two flow endpoint pools, say A, and B, with A playing the role Base and B playing the role Other, if and only if flow endpoints in A can be potentially bound with flow endpoints in B (based only on the connectivity layer network topology structure).
- **Connectivity Session**
  - An instance of this object type represents a group of flow connections
  - A connectivity session belongs to one and only one Connectivity User
  - Attributes
    - Connectivity Session Name
    - Operational State
    - Administrative State
- **Belongs To**
  - An instance of this association relates a Connectivity Session object with a Connectivity User object if and only if the connectivity session belongs to the connectivity user.
- **Flow Connection**
  - An instance of this object type represents a flow connection
  - Attributes
    - Flow Connection Name
    - Operational State
    - Administrative State
    - Connection Topology
  - Notification
    - State Change
- **Root Extremity**
  - An instance of this association represents the one to one relationship between a flow connection and its root flow endpoint.

- **Flow Connection Branch**
    - An instance of this association represents a point-to-point branch of a flow connection. The instance relates a Flow Connection object and a Leaf Flow End Point object if and only if connectivity has been established between the root endpoint of the flow connection and the leaf endpoint.
    - Link Attributes
        - Operational State
        - Administrative State
- **Route Related**
    - An instance of this association relates two Flow Connection objects, say A and B, with A playing the role Base and B playing the role Other, if and only if the route of A has some relationship with the route of B. The route relationship can be either same route or diverse route.
    - Link Attribute
        - Routing Constraint
- **CS Profile**
    - One such object is associated with each connectivity session. The CS Profile object associated with a connectivity session represents the default values for connectivity profile parameters for flow connections within the connectivity session. The values for these parameters are derived from the contract profile unless overridden with values given explicitly at connectivity session setup time.
    - Attributes:
        - Traffic type
        - Reliability class
        - Initial administrative state for flow connections
        - Flow connection notification destination
- **Related CS Profile**
    - An instance of this association relates a Connectivity Session object and a CS Profile object. This association is established when the connectivity session is setup.
- **FC Based On**
    - An instance of this association relates a Flow Connection object and a CS Profile object if and only if *all* connectivity profile parameters of the flow connection are in accordance with the default values contained in the CS Profile object.
- **FC Profile**
    - A FC Profile object represents connectivity profile parameters associated with a specific flow connection. This object exists if and only if one or more connectivity profile parameters of the flow connection have values that are different from the default values contained in the CS Profile object associated with the connectivity session.

- Attributes:
    - Traffic type
    - Reliability class
    - Initial administrative state for flow connection branches
    - Flow connection notification destination

- **Related FC profile**
    - An instance of this association relates a Flow Connection object and a FC Profile object if and only if the flow connection has one or more connectivity profile parameter values that are specific to the flow connection.

4. ConS-RP Computational Model

4.1 The object model

Figure 4-1 shows the computational model for ConS-RP. This ConS-RP specification prescribes the interfaces shown in Figure 4-1, and the behavior of the connectivity user and connectivity provider with respect to these interfaces. The individual objects are shown for ease of understanding, but are not prescriptive. An implementer may define a different set of objects. As long as the same set of interfaces is offered across the ConS-RP, and the same behavior is exhibited by the overall set of interfaces, compliance with this ConS-RP specification is fulfilled.

The interfaces in ConS-RP are categorized into two parts, i.e. access part and usage part. The individual (descriptive) objects and their (prescriptive) interfaces are discussed in sections 4.2 (access) and 4.3 (usage). Section 4.4 gives a precise specification of the objects and their interfaces in Object Definition Language (ODL). Section 4.5 describes event trace diagrams for interactions at ConS-RP.

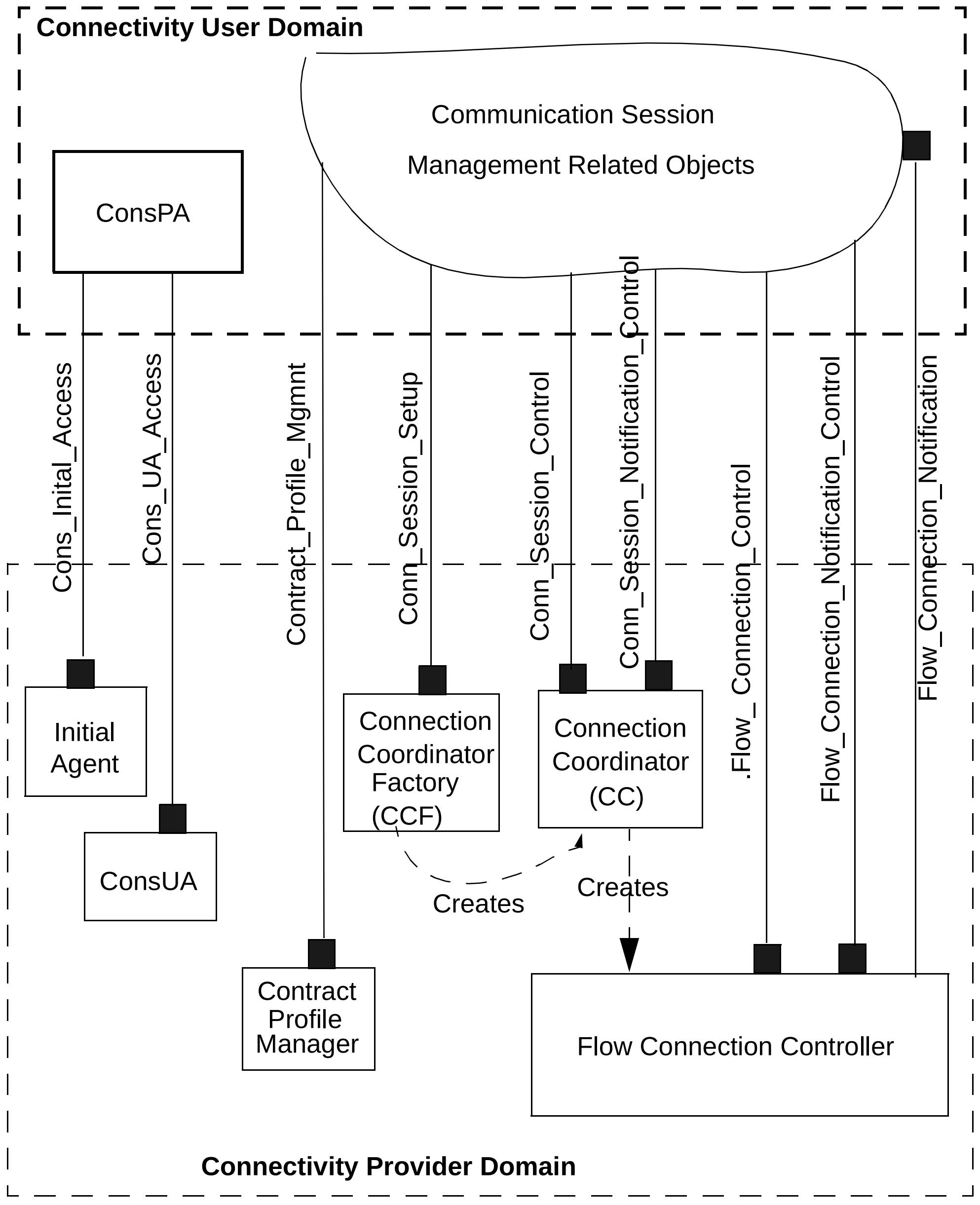

Connectivity User Domain
Communication Session
Management Related Objects
ConsPA
Cons_Inital_Access
Cons_UA_Access
Contract_Profile_Mgmnt
Conn_Session_Setup
Conn_Session_Control
Conn_Session_Notification_Control
.Flow_ Connection_Control
Flow_Connection_Notification_Control
Flow_Connection_Notification
Initial Agent
ConsUA
Connection Coordinator Factory (CCF)
Connection Coordinator (CC)
Creates
Creates
Contract Profile Manager
Flow Connection Controller
Connectivity Provider Domain

Figure 4-1. Cons-RP Computational Architecture

4.2 Objects for Access Part

4.2.1 Initial Agent Object

This object provides the function of authenticating a CU. It offers an interface called Cons_Initial_Access that provides the authentication function as described below.

4.2.1.1 Cons_Initial_Access Interface

This interface provides the function of authenticating a CU. Reference to this interface is supplied to the CU 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. A CU uses this interface as a bootstrap interface that can be used to obtain the reference to the access session interface (Cons_UA_Access) provided by a ConsUserAgent object.

Since interfaces for authentication services provided by TINA DPE have not yet been specified, this submission adapts the principal authentication interface specified in CORBA Security Services Specification [15].

This interface provides the following operations:

- *Authenticate*
  - This operation is used for the first round of authentication. It is also used for reauthentication.
  - Input: import from corba specs.
  - Output: import from corba specs.
- *Continue_Authenticate*
  - This operation is used for subsequent rounds of authentication
  - Input: import from corba specs.
  - Output: import from corba specs.
- *Get_Cons_UA_Access_Interface*
  - This function is used for obtaining a reference to the Cons_UA_Access interface offered by a ConsUserAgent object
  - Input: Some security information
  - Output: Cons_UA_Access interface reference

4.2.2 ConsUserAgent Object

This object represents a connectivity user (CU) that has setup a business relationship with the connectivity provider (CP). This object is created when the business relationship is setup and exists as long as the business relationship exists. It provides an interface called Cons_UA_Access that provides operations for establishing and controlling service sessions in this reference point, hereafter referred to as **Connectivity Service Sessions**.

A connectivity service session is a service session between a CU and the CP in which the CU uses a set of related management/control capabilities of the CP. A connectivity service session is an instantiation of one of the following services:

- **Contract Profile Management Service**: This service provides the following capabilities to the CU:
  - Retrieval of contract profile information associated with the CU
  - Modification of contract profile information associated with the CU
- **Connectivity Control Service**: In a connectivity service session of this type, a CU can either setup new connectivity sessions and control them or control one or more existing connectivity sessions. This service provides the following capabilities to the CU:
  - Setup of new connectivity sessions
  - Setup of new flow connections within a connectivity session
  - Connectivity session activation, deactivation, and release
  - Flow connection branches addition, deletion, activation, and deactivation
  - Flow connection activation, deactivation, and release
  - Request the CP to send notifications when the operational state changes of flow connections (individually specified) occur
  - Receive notifications from the CP when such changes occur

4.2.2.1 Cons_UA_Access Interface

This interface provides operations for the establishment, control, and deletion of connectivity service sessions. It also provides for terminating the business relationship between the CU and the CP.

This interface provides the following operations:

- *Establish_Service_Session*
  - This operation is used for instantiating a connectivity service session. The input argument identifies the connectivity service that is to be instantiated. If the connectivity service session is setup successfully, the operation returns as results the service session identifier and a reference to an (usage part) interface associated with the connectivity service session. The interface reference returned varies with the service requested as follows:
    - *Contract Profile Management Service*: a Contract_Profile_Mgmnt interface reference is returned as result.
    - *Connectivity Control Service*: a Conn_Session_Setup interface reference is returned as result.
  - Input:
    1. Security information
    2. Connectivity Service Type (Contract profile management or Connectivity Control)
  - Output:
    1. Success or Failure indication
    2. Connectivity Service Session Id
    3. A Contract_Profile_Mgmnt or Conn_Session_Control interface reference
- *Delete_Service_Session*
  - This operation is used for deleting a connectivity service session.
  - Input:
    1. Security information
    2. Connectivity Service Session Id
  - Output:
    1. Success or Failure indication
- *List_All_Service_Sessions*
  - This operation is used for obtaining either the session identifier of connectivity service sessions of a specific type or the type code and session identifier of all connectivity service sessions initiated by the CU.
  - Input:
    1. Security information
    2. Service type code (Contract Profile Management, Connectivity Control, All)

- Output:
    1. Success or Failure indication
    2. A list of pairs of the form <A,B> where A is the service type code and B is the Service Session Id of a connectivity service session initiated by the CU.

- *Get_Service_Session_interface*
  - This operation is used for obtaining the reference to either the Contract_Profile_Mgmnt interface or the Conn_Session_Setup interface that is associated with a specific connectivity service session.
  - Input:
    1. Security information
    2. Connectivity Service Session Id
  - Output:
    1. Success or Failure indication
    2. A Contract_Profile_Mgmnt or Conn_Session_Setup interface reference (depending on the type of the specified connectivity service session).

- *Terminate_Contract*
  - This operation is used for terminating the business relationship between the CU and the CP. This operation is permitted only if there is no connectivity session or connectivity service session associated with the CU.
  - Input:
    1. Security information
  - Output:
    1. Success or Failure indication

4.3 Objects for Usage Part

4.3.1 Contract Profile Manager Object

This object manages the contract profile information associated with a CU. It offers a single interface called Contract_Profile_Mgmnt as described below.

4.3.1.1 Contract_Profile_Mgmnt Interface

This interface provides operations for retrieval and modification of information stored in the Contract Profile Information object associated with a CU.

This interface provides the following operations:

- *Get_Authentication_Info*
  - Input: Security information
  - Output:
    1. Success or Failure indication
    2. Authentication information
- *Update_Authentication_Info*
  - Input:
    1. Security information
    2. Authentication information
  - Output: Success or Failure indication
- *Get_Contract_Profile_Parameters*
  - Input: Security information
  - Output:
    1. Success or Failure indication
    2. Default values of parameters in the following format:
       - Default connectivity session initial administrative state
       - Default flow connection initial administrative state
       - Default traffic type
       - Default reliability class
       - Default flow connection notification destination interface
- *Update_Contract_Profile_Parameters*
  - Input:
    1. Security information
    2. Default values for parameters in the format specified above
  - Output: Success or Failure indication

- Note: Updating the contract profile does not affect the parameter values contained in existing CS Profile objects.

4.3.2 Connection Coordinator Factory Object

This object serves as the factory object for connectivity sessions. It has one interface, called, Conn_Session_Setup, and this interface provides operations for the following capabilities: connectivity session setup, listing all connectivity sessions belonging to the CU, and obtaining references to interfaces for controlling a specific connectivity session.

4.3.2.1 Conn_Session_Setup Interface

- *Setup_Connectivity_Session*
  - This operation is used for setting up a connectivity session. If the invocation is successful, this operation creates a Connection Coordinator object for controlling the connectivity session. When a connectivity session setup is requested, a CU may request setup of one or more flow connections.
  - Input:
    1. Security information
    2. Connectivity Session Name
    3. Connectivity Session Parameters List (optional; required only if one or more parameters of the connectivity session being setup override the default values in the contract profile)
    4. Flow connection setup information (optional; required only if setup of one or more flow connections is requested as part of the connectivity session setup)
  - Output:
    1. Success or Failure indication
    2. Conn_Session_Control interface reference
    3. Conn_Session_Notification_Control interface reference
    4. A list of tuples of the form <A, B, C, D, E> where A is the name of a flow connection that has been setup, B and C are references to the Flow_Connection_Control and the Flow_Connection_Notification_Control interfaces associated with the flow connection identified by A, D is the list of flow endpoints that are bound to the flow connection identified by A, and E is the list of flow endpoints that could not be bound to the flow connection identified by A.
- *List_All_Connectivity_Sessions*
  - This operation is used for obtaining the names of all connectivity sessions belonging to the CU.
  - Input:
    1. Security information
  - Output:
    1. Success or Failure indication

2. A list of names of the connectivity sessions belonging to the CU

- *Get_Conn_Session_Control_Interfaces*
  - This operation is used for obtaining references to the Conn_Session_Control and Conn_Session_Notification_Control interfaces associated with a specific connectivity session.
  - Input:
    1. Security information
    2. Connectivity Session Name
  - Output:
    1. Success or Failure indication
    2. Conn_Session_Control interface reference
    3. Conn_Session_Notification_Control interface reference

4.3.3 Connection Coordinator Object

One instance of this object type exists for each connectivity session that has been setup. This object is created by the Connection Coordinator Factory object. 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. A Connection Coordinator object offers two interfaces, Conn_Session_Control and Conn_Session_Notification_Control. The Conn-Session_Control interface provides all operations related to the control of a connectivity session except those related to flow connection notification control. The latter group of operations is provided by the Conn_Session_Notification_Control interface.

4.3.3.1 Conn_Session_Control Interface

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 update the default flow connection notification destination in the CS Profile. Further, it provides an operation that can be used to query if two flow endpoints can be potentially bound.

This interface provides the following operations:

- *Setup_Flow_Connections*
  - This operation is used for setting up one or more flow connections within the connectivity session associated with the Connection Coordinator object. If the invocation is successful, this operation creates a Flow_Connection_Controller object for controlling each flow connection that is setup. When a flow connection setup is requested, a CU must request set up of al least one flow connection branch.
  - Input:

1. Security information
2. Flow connections setup information

- Output:
  1. Success or Failure indication
  2. A list of tuples of the form <A, B, C, D, E> where A is the name of a flow connection that has been setup, B and C are references to the Flow_Connection_Control and the Flow_Connection_Notification_Control interfaces associated with the flow connection identified by A, D is the list of flow endpoints that are bound to the flow connection identified by A, and E is the list of flow endpoints that could not be bound to the flow connection identified by A.

• *Activate_Flow_Connections*
  - This operation is used for activating one or more (possibly all) flow connections of the connectivity session associated with the Connection Coordinator object.
  - Input:
    1. Security information
    2. Success Criterion
    3. A boolean parameter which, if set to true specifies that the entire connectivity session is to be activated, and if set to false specifies only some specified flow connections are to be activated.
    4. Names of flow connections to be activated. This parameter is required only if the boolean parameter is set to false.
  - Output:
    1. Success or Failure indication
    2. List of triples of the form <FC, FEP, Status> where FC is the name of a flow connection, FEP is the name of an endpoint (root or leaf) of the flow connection referenced by FC, and Status is either “Activated” or “UnableToActivate”.

• *Deactivate_Flow_Connections*
  - This operation is used for deactivating one or more (possibly all) flow connections of the connectivity session associated with the Connection Coordinator object.
  - Input:
    1. Security information
    2. Success Criterion
    3. A boolean parameter which, if set to true specifies that the entire connectivity session is to be deactivated, and if set to false specifies only some specified flow connections are to be deactivated.
    4. Names of flow connections to be deactivated. This parameter is required only if the boolean parameter is set to false.
  - Output:

1. Success or Failure indication
2. List of triples of the form <FC, FEP, Status> where FC is the name of a flow connection, FEP is the name of an endpoint (root or leaf) of the flow connection referenced by FC, and Status is either “Deactivated” or “UnableToDeactivate”.

• *Release_Flow_Connections*

- This operation is used for releasing one or more (possibly all) flow connections of the connectivity session associated with the Connection Coordinator object. If release of all flow connections is requested, the connectivity session is also released.
- Input:
  1. Security information
  2. A boolean parameter which, if set to true specifies that the entire connectivity session is to be released, and if set to false specifies only some specified flow connections are to be released.
  3. Names of flow connections to be released. This parameter is required only if the boolean parameter is set to false.
- Output: Success or Failure indication

• *Get_Connectivity_Session_Info*

- This operation is used for retrieving the connectivity session information.
- Input:
  1. Security information
- Output:
  1. Success or Failure indication
  2. Connectivity session name
  3. Operational State of the connectivity session
  4. Administrative state of the connectivity session
  5. CS Profile Information
  6. List of names of flow connections that are components of the connectivity session associated with the Connection Coordinator object.

• *Get_Flow_Connection_Control_Interfaces*

- This operation is used for obtaining the references to the Flow_Connection_Control and Flow_Connection_Notification_Control interfaces associated with one or more (possibly all) flow connections of the connectivity session associated with the connection coordinator object.
- Input:
  1. Security information
  2. A boolean parameter which, if set to true specifies that control interfaces for all flow connections are to be obtained; if set to false specifies that

control interfaces to only some specified flow connections are to be obtained.

3. Names of flow connections whose control interfaces are to be obtained. This parameter is required only if the boolean parameter is set to false.

- Output:
  1. Success or Failure indication
  2. A list of triples of the form <A, B, C> where A is the name of a component flow connection, and B and C are references to the Flow_Connection_Control and Flow_Connection_Notification_Control interfaces associated with the flow connection identified by A.

• *Can_Be_Bound* [22]

- This operation is used for determining if two given flow endpoint pools satisfy the Can Be Bound relationship.
- Input:
  1. Security information
  2. Flow endpoint pool 1 name
  3. Flow endpoint pool 2 name
- Output:
  1. Success or Failure indication
  2. Boolean result

4.3.3.2 Conn_Session_Notification_Control Interface

This interface provides operations for controlling notifications emitted by the CP regarding flow connections of the connectivity session associated with the Connection Coordinator object. This interface provides operations for enabling and disabling notifications for one or more (possibly all) flow connections. Further, it provides an operation for updating the Flow Connection Notification Destination parameter contained in the CS profile object of the connectivity session associated with the Connection Coordinator object.

It provides the following operations:

• *Enable_Conn_Session_Notification*

- This operation is used for instructing the CP to emit notifications regarding one or more flow connections of the connectivity session.
- Input:
  1. Security information

22. It is recognized that this operation does not logically belong to this interface. It may be moved to another interface in the next version of ConS-RP specification.

2. A boolean parameter which, if set to true specifies that notifications for all flow connections are to be enabled, and if set to false specifies that notifications for only some specified flow connections are to be enabled.
3. Names of flow connections for which notifications are to be enabled. This parameter is required only if the boolean parameter is set to false.

- Output: Success or Failure indication

• *Disable_Conn_Session_Notification*

- This operation is used for instructing the CP to suspend the emission of notifications regarding one or more flow connections of the connectivity session.
- Input:

1. Security information
2. A boolean parameter which, if set to true specifies that notifications for all flow connections are to be suspended, and if set to false specifies that notifications for only some specified flow connections are to be suspended.
3. Names of flow connections for which notifications are to be suspended. This parameter is required only if the boolean parameter is set to false.

- Output: Success or Failure indication

• *Update_CS_Flow_Connection_Notification_Destination*

- This operation is used to update the value of the Flow Connection Notification Destination parameter contained in the CS Profile object. This operation affects only the CS profile of the connectivity session associated with the Connection Coordinator object.
- Input:

1. Security information
2. Reference to the Flow_Connection_Notification interface offered by CU for receipt of the notifications

- Output: Success or Failure indication

4.3.4 Flow Connection Controller Object

One instance of this object type exists for each flow connection that has been setup. This object is created by the Connection Coordinator object associated with the connectivity session of which the flow connection is a part. A Flow Connection Controller object provides operations for controlling the associated flow connection including the flow connection release operation. When a flow connection is released, the associated Flow Connection Controller object is deleted.

4.3.4.1 Flow_Connection_Control Interface

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

This interface provides the following operations:

- *.Add_Flow_Connection_Branches*
  - This operation is used for adding one or more branches to the flow connection associated with the Flow_Connection_Controller object.
  - Input:
    1. Security information
    2. Success Criterion
    3. Flow connection branches setup information
  - Output:
    1. Success or Failure indication
    2. List of names of leaf flow endpoints that have been successfully attached to the flow connection.
    3. List of pairs of the form < FEP, Failure Code> where FEP is the name of a leaf endpoint that could not be attached to the flow connection, and Failure Code identifies the reason for the failure.
- *Delete_Flow_Connection_Branches*
  - This operation is used for removing one or more branches (possibly all branches) from the flow connection associated with the Flow_Connection_Controller object. If deletion of all branches is requested, the flow connection is released.
  - Input:
    1. Security information
    2. Success Criterion
    3. A boolean parameter which, if set to true specifies that the entire flow connection is to be deleted, and if set to false specifies only some specified branches are to be deleted.
    4. Names of leaf flow endpoints to be deleted. This parameter is required only if the boolean parameter is set to false.
  - Output:
    1. Success or Failure indication
    2. List of names of leaf flow endpoints that have been successfully deleted from the flow connection.
    3. List of pairs of the form < FEP, Failure Code> where FEP is the name of a leaf endpoint that could not be deleted from the flow connection, and Failure Code identifies the reason for the failure.

- *Activate_Flow_Connection_Branches*
  - This operation is used for activating one or more branches (possibly all branches) of the flow connection associated with the Flow_Connection_Controller object.
  - Input:
    1. Security information
    2. Success Criterion
    3. A boolean parameter which, if set to true specifies that the entire flow connection is to be activated, and if set to false specifies only some specified branches are to be activated.
    4. Names of leaf flow endpoints to be activated. This parameter is required only if the boolean parameter is set to false.
  - Output:
    1. Success or Failure indication
    2. List of names of leaf flow endpoints (identifying the branches) that have been successfully activated.
    3. List of pairs of the form < FEP, Failure Code> where FEP is the name of a leaf endpoint (branch) that could not be activated, and Failure Code identifies the reason for the failure.
- *Deactivate_Flow_Connection_Branches*
  - This operation is used for deactivating one or more branches (possibly all branches) of the flow connection associated with the Flow_Connection_Controller object.
  - Input:
    1. Security information
    2. Success Criterion
    3. A boolean parameter which, if set to true specifies that the entire flow connection is to be deactivated, and if set to false specifies only some specified branches are to be deactivated.
    4. Names of leaf flow endpoints to be deactivated. This parameter is required only if the boolean parameter is set to false.
  - Output:
    1. Success or Failure indication
    2. List of names of leaf flow endpoints (identifying the branches) that have been successfully deactivated.
    3. List of pairs of the form < FEP, Failure Code> where FEP is the name of a leaf endpoint (branch) that could not be deactivated, and Failure Code identifies the reason for the failure.
- *Modify_Flow_Connection_Branches*

- This operation is used for modifying one or more branches to the flow connection associated with the Flow_Connection_Controller object.
- Input:
  1. Security information
  2. Success Criterion
  3. Flow connection branches modification information
- Output:
  1. Success or Failure indication
  2. List of names of flow endpoints (identifying the branches) that have been successfully modified.
  3. List of pairs of the form < FEP, Failure Code> where FEP is the name of a flow endpoint that could not be modified, and Failure Code identifies the reason for the failure.

• *Get_Flow_Connection_Info*
  - This operation is used for retrieving the flow connection information.
  - Input:
    1. Security information
  - Output:
    1. Success or Failure indication
    2. Connection topology
    3. Traffic type
    4. Routing constraint
    5. Flow Connection Notification Destination
    6. Reliability Class
    7. List of tuples, one tuple for each flow endpoint, giving the following information on the flow endpoint:
       - Flow endpoint name
       - Flow endpoint type (root or leaf)
       - Peak bandwidth
       - Average bandwidth
       - Maximum delay
       - Maximum delay variation
       - Administrative state
       - Operational state

4.3.4.2 Flow_Connection_Notification_Control Interface

This interface provides operations for controlling the emission of notifications by the CP regarding the flow connection associated with the Flow_Connection_Controller object.

It provides the following operations:

- *Enable_Flow_Connection_Notification*
  - This operation is used for instructing the CP to emit notifications regarding the flow connection.
  - Input: Security information
  - Output: Success or Failure indication
- *Disable_Flow_Connection_Notification*
  - This operation is used for instructing the CP to suspend emission of notifications regarding the flow connection.
  - Input: Security information
  - Output: Success or Failure indication
- *Set_Flow_Connection_Notification_Destination*
  - This operation is used for communicate to the CP the reference to the Flow_Connection_Notification interface offered by the CU for receipt of the notifications regarding the flow connection.
  - Input:
    1. Security information
    2. Reference to the Flow_Connection_Notification interface offered by CU for receipt of the notifications
  - Output: Success or Failure indication

4.3.4.3 Flow_Connection_Notification Interface

This interface is used by the Flow_Connection_Controller object as a client. It uses this interface to send the notifications regarding the associated flow connection.

It provides the following operation:

- *Flow_Connection_Status_Changed*
  - This operation is used to notify the CU when the operational state of one or more branches of the flow connection changes.
  - Input:
    1. Flow connection name
    2. A list of tuples of the form <FEP, FEP Type, Status, Additional_Info> where FEP is the name of a flow endpoint, FEP Type is either “Root” or “Leaf”, Status is either “Operational” or “Not Operational”, and Additional_ Info is a text string that conveys to the CU some additional information (not specified in this document) concerning the status change.

4.4 Specification of computational objects and interfaces

4.4.1 Common Data Types

All commonly used data types for the IDL interfaces associated with the ConS reference point are defined in this section. For the convenience of referencing the common types, the interface “ConsCommonDefs” is created to contain these commonly used types.

```
/* IDL of interface i_ConsCommonDefs */

#ifndef i_ConsCommonDefs_IDL
#define i_ConsCommonDefs_IDL


/*
 * Description:   This file defines some commonly used data types and
 * ~~~~~~~~~~~    structures.
 *
 */

interface i_ConsCommonDefs
{


    /*
     * Name types
     *
     */

    typedef sequence <string> t_Name;

    struct t_TagValue
        {
        string tag;
        string value;
        };

    typedef sequence <t_TagValue> t_FepPoolName;

    typedef sequence <t_FepPoolName> t_FepPoolList;

    typedef sequence <t_TagValue> t_FepHandle;

    struct t_FepName
        {
        t_FepPoolName pool;
        t_FepHandle fep;
        };

    typedef t_Name t_FlowConnName;

    typedef t_Name t_ConnSessionName;


    /*
     * Data types for state attributes
     *
     */

    enum t_OperationalState
        {Failed,       /* Resource toatlly inoperable */
```

```
         Operational,  /* Resource fully operable */
         Degraded      /* Resource partially operable */
        };

    enum t_AdministrativeState
        {
        Locked, /* Resource usage is prohibited */
        Unlocked /* Resource usage is permitted */
        };


    /*
     * Layer network related definitions
     *
     */

    typedef string t_CharacteristicInfo;

    typedef t_CharacteristicInfo t_LNtype;

    /*
     * Flow endpoint related definitions
     *
     */

    enum t_FepType {Root, Leaf};

    enum t_FepRefType {PartialRef, FullRef};

    union t_FepRef switch(t_FepRefType)
        {
        case PartialRef:t_FepPoolName pref;
        case FullRef:t_FepName fref;
        };

    typedef sequence <t_FepRef> t_FepRefList;


    /*
     * Traffic descriptor definitions
     *
     */

    enum t_TrafficType {CBR, VBR, ABR, UBR};

    struct t_TrafficDescriptor
        {
        unsigned long peakBW;/* Mbits/sec */
        unsigned long averageBW; /* Mbits/sec */
        };

    struct t_QoSDescriptor
        {
        unsigned long maxErrorRate; /* bits/(10**6) sec */
        unsigned long maxDelay; /* msec */
        unsigned long maxDelayVariation;/* msec */
        };


    /*
     * Flow endpoint information
     *
     */
```

```
struct t_FepInfo
    {
    t_FepRef ref;
    t_FepType fType;
    t_TrafficDescriptor tDesc;
    t_QoSDescriptor qDesc;
    t_AdministrativeState adminState;
    t_OperationalState opState;
    };


/*
 * Flow Connection related information
 *
 */

enum t_ConnTopology {PointToPoint, PointToMultipoint};

enum t_ReliabilityClass {ReleaseOnFailure, HoldOnFailure};

enum t_SuccessCriterion {AllOrNone, BestEffort};

enum t_RoutingOption {SamreRoute, DifferentRoute};

struct t_RoutingConstraint
    {
    t_FlowConnName refConn;
    t_RoutingOption option;
    };


/*
 * Connnectivity Service Session related definitions
 *
 */

enum t_ConnectivityService {ContractProfileMgmnt, ConnControl};

typedef unsigned long t_ServiceSessionId;


/*
 * Operation request-response related definitions
 *
 */

enum t_FailureCode
    {InsufficientBandwidth, InsufficientResources, QoSCannotBeMet,
    NoPathFound, NetworkFailure, KtnFailure};

enum t_ParametersTag { AdminState, TraffType, RelClass };

union t_ParameterValue switch(t_ParametersTag)
    {
    case AdminState:t_AdministrativeState state;
    case TraffType:t_TrafficType traffType;
    case RelClass:t_ReliabilityClass rclass;
    };

typedef sequence <t_ParameterValue> t_ParametersList;
typedef sequence <t_FlowConnName> t_FlowConnNameSeq;

typedef sequence <t_ConnSessionName> t_ConnSessionNameSeq;

/*
 * Security handle definition
 *
 */

typedef long t_Credentials;

typedef t_Credentials t_SecHandle;


/*
 * Commonly used exceptions (in alphabetical order)
 *
 */

exception ConnectivitySessionsExist { };

exception ConnSessionActiveAlready { };

exception ConnSessionDeactiveAlready { };

exception FlowConnBranchesActiveAlready {
    t_FepRefList list;
};

exception FlowConnBranchesDeactiveAlready {
    t_FepRefList list;
};

exception FlowEndPointsAlreadyBound {
    t_FepRefList list;
};

exception InvalidAuthenticationInfo { };

exception InvalidConnSessionName { };

exception InvalidConnSessionInfo {
    string info;
};

exception InvalidDefaultValues { };

exception InvalidFlowConnBranchesInfo {
    t_FepRefList list; string info;
};

exception InvalidFlowConnInfo {
    t_FlowConnName connName; string info;
};

exception InvalidFlowConnName {
    t_FlowConnName connName;
};

exception InvalidServiceSessionId { };

exception NotAuthenticated { };
```

```
    exception NotAuthorized { };

    exception NonExistentFlowEndPoints {
        t_FepRefList list;
    };

    exception NonExistentPools {
        t_FepPoolList list;
    };

    exception NotificationDestinationNotSet { };

    exception ServiceSessionsExist { };

    exception InvalidServiceName { };


};

#endif /* i_ConsCommonDefs_IDL */
```

4.4.2 Flow_Connection_Notification Interface

```
/* IDL of interface i_FlowConnNotification */

#ifndef i_FlowConnNotification_IDL
#define i_FlowConnNotification_IDL

#include “i_ConsCommonDefs.idl”


/*
 * This interface is used by a FlowConnectionController object
 * as a client for sending notifications to a CU regarding changes in the
 * operational state of the associated flow connection
 *
 */

interface i_FlowConnNotification
{

    struct t_FepStatusInfo
        {
        t_FepRef ref;
        t_FepType type;
        t_OperationalState state;
        string     additionalInfo;
        };

    typedef sequence <t_FepStatusInfo> t_ConnNotifInfo;

    void flow_connection_status_changed
        (in t_FlowConnName connName, in t_ConnNotifInfo info);


};

#endif /* i_FlowConnNotification_IDL */
```

4.4.3 Flow_Conn_Notification_Control Interface Specification

```
/* IDL of interface i_FlowConnNotificationControl */

#ifndef i_FlowConnNotificationControl_IDL
#define i_FlowConnNotificationControl_IDL

#include “i_FlowConnNotification.idl”


/*
 * This interface is used by a CU to control the emission of notifications
 * by a FlowConnectionController object regarding changes in the
 * operational state of the associated flow connection
 * This operation is used for instructing the CP to emit notifications
 * regarding the flow connection. The invocation results in an
 * exception if the CU has not set the destination notification interface
 *
 */

interface i_FlowConnNotificationControl
{

    void enable_flow_connection_notification
        (in t_SecHandle handle)
        raises(NotAuthenticated,NotAuthorized,NotificationDestinationNotSet);

    void disable_flow_connection_notification
        (in t_SecHandle handle)
        raises(NotAuthenticated, NotAuthorized);

    void set_flow_connection_notification_destination
        (in t_SecHandle handle, in i_FlowConnNotification destination)
        raises(NotAuthenticated, NotAuthorized);


};

#endif /* i_FlowConnNotificationControl_IDL */
```

4.4.4 Flow_Conn_Control Interface Specification

```
/* IDL of interface i_FlowConnControl */

#ifndef i_FlowConnControl_IDL
#define i_FlowConnControl_IDL

#include “i_FlowConnNotificationControl.idl”


/*
 * This interface provides operations for manipulating the flow connection
 * associated with the Flow Connection Controller that offers this
 * interface.
 *
 */
```

```
interface i_FlowConnControl
{

    typedef sequence <t_FepRef> t_SuccFepList;

    struct t_FailedFep
        {
        t_FepRef ref;
        t_FailureCode code;
        };

    typedef sequence <t_FailedFep> t_FailedFepList;

    struct t_FepDesc
        {
        t_FepRef ref;
        t_FepType fType;
        t_TrafficDescriptor tDesc;
        t_QoSDescriptor qDesc;
        t_ParametersList list; /* Used for specifying only the
               admin state */
        };

    typedef sequence <t_FepDesc> t_FepDescSeq;

    struct t_FlowConnInfo
        {
        t_FlowConnName connName;
        t_ConnTopology topology;
        t_AdministrativeState adminState;
        t_OperationalState opState;
        t_TrafficType traffType;
        t_ReliabilityClass relClass;
        t_RoutingConstraint routeConstraint;
        i_FlowConnNotification notifIf;
        sequence <t_FepInfo> fepInfoList;
        };


    /*
     * This operation is used for adding one or more branches to the
     * flow connection
     *
     */

    void add_flow_connection_branches
        (in t_SecHandle handle,
         in t_SuccessCriterion criterion,
         in t_FepDescSeq descList,
         out t_SuccFepList boundList,
         out t_FailedFepList unboundList)
        raises(NotAuthenticated, NotAuthorized, NonExistentFlowEndPoints,
               FlowEndPointsAlreadyBound, InvalidFlowConnBranchesInfo);

    /*
     * This operation is used for deleting one or more branches of the
     * flow connection
     *
     */

    void delete_flow_connection_branches
        (in t_SecHandle handle,
         in t_SuccessCriterion criterion,
         in boolean allFlag, // set to true if all branched are to be deleted
         in t_FepRefList list,
         out t_SuccFepList list1,
         out t_FailedFepList list2)
        raises(NotAuthenticated, NotAuthorized, NonExistentFlowEndPoints);

    /*
     * This operation is used for activating one or more branches of the
     * flow connection
     *
     */

    void activate_flow_connection_branches
        (in t_SecHandle handle,
         in t_SuccessCriterion criterion,
         in boolean allFlag, // set to true if all branched are to be activated
         in t_FepRefList list,
         out t_SuccFepList activatedList,
         out t_FailedFepList activationFailedList)
        raises(NotAuthenticated, NotAuthorized, NonExistentFlowEndPoints,
                FlowConnBranchesActiveAlready);

    /*
     * This operation is used for deactivating one or more branches of the
     * flow connection
     */

    void deactivate_flow_connection_branches
        (in t_SecHandle handle,
         in t_SuccessCriterion criterion,
         in boolean allFlag, // set to true if all branched are to be deactivated
         in t_FepRefList list,
         out t_SuccFepList deactivatedList,
         out t_FailedFepList deactivationFailedList)
        raises(NotAuthenticated, NotAuthorized, NonExistentFlowEndPoints,
                FlowConnBranchesDeactiveAlready);

    /*
     * This operation is used for modifying one or more branches of the
     * flow connection
     */

    void modify_flow_connection_branches
        (in t_SecHandle handle,
         in t_SuccessCriterion criterion,
         in t_FepDescSeq descList,
         out t_SuccFepList modifiedList,
         out t_FailedFepList unmodifiedList)
        raises(NotAuthenticated, NotAuthorized, NonExistentFlowEndPoints,
                InvalidFlowConnBranchesInfo);

    /*
     * This operation is used for retrieving information
     * on the flow connection
     */

    void get_flow_conn_info
        (in t_SecHandle handle,
         out t_FlowConnInfo connInfo)
        raises(NotAuthenticated, NotAuthorized);

};
#endif /* i_FlowConnControl_IDL */
```

4.4.5 Conn_Session_Control Interface Specification

```
/* IDL of interface i_ConnSessionControl */

#ifndef i_ConnSessionControl_IDL
#define i_ConnSessionControl_IDL

#include “i_FlowConnControl.idl”


/*
 * This interface provides operations for manipulating a connectivity
 * session. One such interface is associated with each connectivity
 * session
 *
 */


interface i_ConnSessionControl
{

    struct t_FlowConnDesc
        {
        t_FlowConnName connName;
        t_ConnTopology topology;
        t_ParametersList list;
        t_RoutingConstraint routeConstraint;
        sequence <t_FepDesc> t_fepDescList;
        };


    typedef sequence <t_FlowConnDesc> t_FlowConnDescSeq;

    struct FlowConnResponse
        {
        t_FlowConnName connName;
        i_FlowConnControl fCtrlIf;
        i_FlowConnNotification_Control fNotifCtrlIf;
        t_FepRef rootFep;
        t_SuccFepList boundLlist;
        t_FailedFepList unboundList;
        };

    typedef sequence <t_FlowConnResponse> t_FlowConnResponseSeq;

    enum t_ActivationStatus {Activated, UnableToActivate, Deactivated,
                UnableToDeactivate};

    struct t_ActivationResponse {
        t_FlowConnName connName;
        t_FepRef ref;
        t_ActivationStatus status;
        };

    typedef sequence <t_ActivationResponse> t_ActivationResponseSeq;


    struct t_FlowConnInterfaces {
        t_FlowConnName connName;
        i_FlowConnControl ctrlIf;
        i_FlowConnNotificationControl notifCtrlIf;
        };
```

```
typedef sequence <t_FlowConnInterfaces> t_FlowConnInterfacesSeq;


struct t_ConnSessionProfileInfo
    {
    t_AdministrativeState initialConnSessionAdminState;
    t_AdministrativeState initialFlowConnectionAdminState;
    t_TrafficType traffType;
    t_ReliabilityClass relClass;
    i_FlowConnNotification notifIf;
    };


struct t_ConnSessionInfo
    {
    t_ConnSessionName sessionName;
    t_AdministrativeState adminState;
    t_OperationalState opState;
    t_ConnSessionProfileInfo info;
    sequence <t_FlowConnName> flowConnList;
    };


/*
 * This operation is used for creating a flow connection
 *
 */

void setup_flow_connections
    (in t_SecHandle handle,
     in t_ParametersList list,
     in t_SuccessCriterion criterion,
     in t_FlowConnDescSeq flowConnDescList,
     out t_FlowConnResponseSeq resp)
    raises(NotAuthenticated, NotAuthorized, InvalidFlowConnInfo);

/*
 * This operation is used for activating the connectivity session
 *
 */

void activate_flow_connections
    (in t_SecHandle handle,
     in t_SuccessCriterion criterion,
     in boolean allFlag, //set to true if all flow connections are to be
                        // activated
     in t_FlowConnNameSeq flowConnList,
     out t_ActivationResponseSeq resp)
    raises(NotAuthenticated, NotAuthorized, InvalidFlowConnName,
           ConnSessionActiveAlready);

/*
 * This operation is used for deactivating the connectivity session
 *
 */

void deactivate_flow_connections
    (in t_SecHandle handle,
     in t_SuccessCriterion criterion,
     in boolean allFlag, //set to true if all flow connections are to be
                        // deactivated
     in t_FlowConnNameSeq flowConnList,
         out t_ActivationResponseSeq resp)
        raises(NotAuthenticated, NotAuthorized, InvalidFlowConnName,
            ConnSessionDeactiveAlready);

    /*
     * This operation is used for releasing the connectivity session
     *
     */

    void release_flow_connections
        (in t_SecHandle handle,
         in boolean allFlag, // set to true if all flow connections are to be released
         in t_FlowConnNameSeq flowConnList)
        raises(NotAuthenticated, NotAuthorized, InvalidFlowConnName);


    /*
     * This operation is used for retrieving information
     * on the connectivity sessio
     *
     */

    void get_conn_session_info
        (in t_SecHandle handle,
         out t_ConnSessionInfo info)
        raises(NotAuthenticated, NotAuthorized);

    /*
     * This operation is used for retrieving the references to the
     * i_FlowConnControl and i_FlowConnNotificationControl interfaces
     * associacted with the flow connections of the conncectivity session
     *
     */

    void get_flow_conn_control_interfaces
        (in t_SecHandle handle,
         in boolean allFlag, // set to true if control interfaces for
                             // all flow connections are to be retrieved
         in t_FlowConnNameSeq flowConnList,
         out t_FlowConnInterfacesSeq connIfList )
        raises(NotAuthenticated, NotAuthorized, InvalidFlowConnName);

    /*
     * Operation for determining if two given floe endpoint pools
     * satisfy the Can Be Bound relationship
     *
     */

    boolean can_be_bound
        (in t_SecHandle handle,
         in t_FepPoolName pool1,
         in t_FepPoolName pool2)
        raises(NotAuthenticated, NotAuthorized, NonExistentPools);

};
#endif /* i_ConnSessionControl_IDL */
```

4.4.6 Conn_Session_Notification_Control Interface Specification

```
/* IDL of interface i_ConnSessionNotificationControl */

#ifndef i_ConnSessionNotificationControl_IDL
#define i_ConnSessionNotificationControl_IDL

#include “i_FlowConnNotification.idl”


/*
 * One such interface is associated with each connectivity session.
  * This interface is used by a CU to control the emission of notifications
 * by FlowConnectionController objects regarding changes in the
 * operational state of the flow connections that are part of the
 * connectivity session.
 *
 */

interface i_ConnSessionNotificationControl
{

    void enable_Conn_Session_notification
        (in t_SecHandle handle,
         in boolean allFlag, // set to true if notifications for all flow connections
                             // are to be enabled
         in t_FlowConnNameSeq flowConnList)
        raises(NotAuthenticated,NotAuthorized,NotificationDestinationNotSet);

    void disable_Conn_Session_notification
        (in t_SecHandle handle,
         in boolean allFlag, // set to true if notifications for all flow connections
                             // are to be disabled
         in t_FlowConnNameSeq flowConnList)
        raises(NotAuthenticated, NotAuthorized);

    /*
     * This operation is used for updating the default destination notification
     * interface in the CS Profile object
     *
     */

    void update_CS_flow_connection_notification_destination
        (in t_SecHandle handle,
         in i_FlowConnNotification destination)
        raises(NotAuthenticated, NotAuthorized);


};

#endif /* i_ConnSessionNotificationControl_IDL */
```

4.4.7 Conn_Session_Setup Interface Specification

```
/* IDL of interface i_ConnSessionSetup */

#ifndef i_ConnSessionSetup_IDL
#define i_ConnSessionSetup_IDL

#include “i_ConnSessionControl.idl”
#include “i_ConnSessionNotificationControl.idl”


/*
 * This interface provides operations for a connectivity session setup,
 * listing all connectivity sessions belonging to a CU, and obtaining references
  * to interfaces for controlling a specific connectivity session.
*/

interface i_ConnSessionSetup
{
    /*
     * This operation is used for setting up a connectivity session. When a
     * Setup of one or more flow connections may be requested as part of
     * a connectivity session setup
    */

    void setup_connectivity_session
        (in t_SecHandle handle,
         in t_ConnSessionName consName,
         in t_ParametersList list,
         in i_FlowConnNotification notifIf,
         in t_FlowConnDescSeq flowConnDescList,
         out i_ConnSessionControl ctrlIf,
         out i_ConnSessionNotificationControl notifCtrlIf,
         out t_FlowConnResponseSeq resp)
        raises(NotAuthenticated, NotAuthorized, InvalidFlowConnInfo);

    /*
     * This operation is used for obtaining the names of all connectivity sessions
     * belonging to the CU
    */

    void List_all_connectivity_sessions
        (in t_SecHandle  handle,
         out t_ConnSessionNameSeq connSessionList)
        raises(NotAuthenticated,NotAuthorized);


    /*
     * This operation is used for obtaining references to the i_ConnSessionControl
     * and i_ConnSessionNotificationControl interfaces associated with
     * a specific connectivity session
    */

    void get_conn_session_control_interface
        (in t_SecHandle handle,
         in t_ConnSessionName name,
         out i_ConnSessionControl ctrlIf,
         out i_ConnSessionNotificationControl notifCtrlIf )
        raises(NotAuthenticated,NotAuthorized,InvalidConnSessionName);

};
#endif /* i_ConnSessionSetup_IDL */
```

4.4.8 Contract_Profile_Mgmnt Interface Specification

```
/* IDL of interface i_ContractProfileMgmnt */

#ifndef i_ContractProfileMgmnt_IDL
#define i_ContractProfileMgmnt_IDL

#include “i_ConnSessionControl.idl”


/*
 * One such interface is associated with each contract profile
 * information object. Each contract profile information object is
 * associated with some CU. This interface provides operations for the
 * retrieval and modification of contract profile information associated
 * with a CU.
 *
 */

interface i_ContractProfileMgmnt
{
    /*
     * Contract Profile Information related definitions
     *
     */

    typedef sequence <string> t_AuthenticationInfo;

    struct t_ContractProfileInfo
        {
        t_AuthenticationInfo authInfo;
        t_ConnSessionProfileInfo profileInfo;
        };


    void get_authentication_info
        (in t_SecHandle handle,
         out t_AuthenticationInfo authInfo   )
        raises(NotAuthenticated, NotAuthorized);

    void update_authentication_info
        (in t_SecHandle handle,
         in t_AuthenticationInfo authInfo   )
        raises(NotAuthenticated, NotAuthorized, InvalidAuthenticationInfo);

    void get_contract_profile_parameters
        (in t_SecHandle handle,
         out t_ConnSessionProfileInfo profileInfo   )
        raises(NotAuthenticated, NotAuthorized);

    void update_contract_profile_parameters
        (in t_SecHandle handle,
         in t_ConnSessionProfileInfo profileInfo   )
        raises(NotAuthenticated, NotAuthorized, InvalidDefaultValues);


};

#endif /* i_ContractProfileMgmnt_IDL */
```

4.4.9 Cons_UA_Access Interface Specification

```
/* IDL of interface i_ConsUAAccess */

#ifndef i_ConsUAAccess_IDL
#define i_ConsUAAccess_IDL

#include “i_ContractProfileMgmnt.idl”
#include “i_ConnSessionSetup.idl”


/*
 * One such interface is associated with each connectivity user (CU).
 * This interface provides operations for the establishment, control, and
 * deletion of connectivity service sessions. It also provides an
 * operation for terminating the business relationship between the CU
 * and the CP.
 *
 */

interface i_ConsUAAccess
{

    union t_SessionInterf switch(t_ConnectivityService)
        {
         case ContractProfileMgmnt:i_ContractProfileMgmnt profileMgmntIterf;
         case ConnControl:i_ConnSessionSetup setupIf;
        };

    enum t_ServiceTypeCode {ContractProfileMgmnt, ConnControl, All};

    struct t_ServiceSessionInfo
        {
        t_ConnectivityService service;
        t_ServiceSessionId sessionId;
        };

    typedef sequence <t_ServiceSessionInfo> t_ServiceSessionInfoSeq;



    void establish_service_session
        (in t_SecHandle handle,
         in t_ConnectivityService service,
         out t_ServiceSessionId sessionId,
         out t_SessionInterf interf   )
        raises(NotAuthenticated, NotAuthorized, InvalidServiceName);

    void delete_service_session
        (in t_SecHandle handle,
         in t_ServiceSessionId sessionId   )
        raises(NotAuthenticated,NotAuthorized,InvalidServiceSessionId);

    void List_all_service_sessions
        (in t_SecHandle handle,
         in t_ServiceTypeCode typeCode,
         out t_ServiceSessionInfoSeq serviceSessionList   )
        raises(NotAuthenticated, NotAuthorized);

    void get_service_session_interface
        (in t_SecHandle handle,
         in t_ServiceSessionId sessionId,
         out t_SessionInterf interf   )
        raises(NotAuthenticated,NotAuthorized,InvalidServiceSessionId);

    void terminate_contract
        (in t_SecHandle handle   )
        raises(NotAuthenticated, NotAuthorized, ConnectivitySessionsExist,
                ServiceSessionsExist);


};

#endif /* i_ConsUAAccess_IDL */
```

4.4.10 Cons_Initial_Access Interface Specification

```
/* IDL of interface i_ConsInitialAccess */

#ifndef i_ConsInitialAccess_IDL
#define i_ConsInitialAccess_IDL

#include “i_ConsUAAccess.idl”


/*
 * This interface is derived from PrincipalAuthenticator
 * object interface of CORBA Security Specs.[15, pp.106-107]
 *
 * This interface provides the function of authenticating
 * a connectivity user.
 *
 */

interface i_ConsInitialAccess
{

    typedef unsigned long AuthenticationStatus;

    typedef unsigned long AuthenticationMethod;

    typedef unsigned long Opaque;

    typedef unsigned long AttributeList;

    typedef unsigned long Credentials;

    AuthenticationStatus authenticate
        (in AuthenticationMethod method,
         in string security_name,
         in Opaque auth_data,
         in AttributeList privileges,
         out Credentials creds,
         out Opaque continuation_data,
         out Opaque auth_specific_data  );

    AuthenticationStatus continue_authentication
        (in Opaque response_data,
         inout Credentials creds,
         out Opaque continuation_dAta,
         out Opaque auth_specific_data  );

    /*
```

```
     * The following operation is not part of CORBA Security.
     *
     */

    void get_Cons_UA_Access_interface
        (in Credentials creds,
         out i_ConsUAAccess cons_UA_Access_interface  );


};

#endif /* i_ConsInitialAccess_IDL */
```

4.4.11 FlowConnectionController Object Specification

```
// ODL of object template FlowConnectionController


object FlowConnectionController
{

    behavior
        behaviorText

        “   One such object exists for each flow connection
            that has been setup. This object is created by a
            ConnectionCoordinator object. This object offers
            an interface, Flow_Conn_Control, that provides
            operations for manipulating the associated flow
            connection. The capabilities offered by this
            object are parts of the Connectivity Control
            Service.

            This object is deleted when the associated flow
            connection is released using an operation defined
            in the i_FlowConnControl interface. If the
            operational state of the associated flow connection
            changes, this object uses an interface,
            i_FlowConnNotification, offered by a CU, and
            reports the operational state change to the CU.
            This object also offers another interface,
            i_FlowConnNotificationControl, that can be used by
            CUs to control the emission of operational state
            change notifications.”;


    supports
        i_FlowConnControl, i_FlowConnNotificationControl;

    initial
        i_FlowConnNotificationControl;

};
```

4.4.12 ConnectionCoordinator Object Specification

```
// ODL of object template ConnectionCoordinator

object ConnectionCoordinator
{

    behavior
        behaviorText

        “   One such object exists for each connectivity
            session that has been setup. This object is
            created by the Connection CoordinatorFactory
            object. This object offers two interfaces:
            an interface, i_ConnSessionControl, that provides
            operations for manipulating the connectivity session;
            an interface, i_ConnSessionNotificationControl, that
            provides flow connection notification control at
            connectivity session level.

            This object serves as the factory object for
            flow connections that are components of the
            associated connectivity session.
            The capabilities offered by this object are
            parts of the Connectivity Control Service.
            This object is deleted when the associated
            connectivity session is released using an
            operation defined in the i_ConnSessionControl
            interface.”;


    supports
        i_ConnSessionControl, i_ConnSessionNotificationControl;

    initial
        i_ConnSessionControl;

};
```

4.4.13 ConnectionCoordinatorFactory Object Specification

```
// ODL of object template ConnectionCoordinatorFactory

object ConnectionCoordinatorFactory
{

    behavior
        behaviorText

        “   This object is the factory object for connectivity sessions.
            It offers an interface, i_ConnSessionSetup, that provides an
            operation for connectivity session setup,an operation for
            listing all connectivity sessions belonging to a CU, and
            an operation for obtaining references to i_ConnSessionControl
            and i_ConnSessionNotificationControl interfaces for
            controlling a specific connectivity session. The capabilities
            offered by this object are parts of the Connectivity Control
            Service.
        ”;
    supports
        i_ConnSessionSetup;

    initial
        i_ConnSessionSetup;

};
```

4.4.14 ContractProfileManager Object Specification

```
// ODL of object template ContractProfileManager

object ContractProfileManager
{

    behavior
        behaviorText

        “   This object manages the contractProfile information
            associated with a Connectivity User. It offers an
            interface i_ContractProfileMgmnt that provides
            operations for the retrieval and modification of
            contract profile information.”;

    supports
        i_ContractProfileMgmnt;

    initial
        i_ContractProfileMgmnt;

};
```

4.4.15 ConsUserAgent Object Specification

```
// ODL of object template ConsUserAgent

object ConsUserAgent
{

    behavior
        behaviorText

        “   This object represents a connectivity user (CU)
            that has setup a business relationship with the
            connectivity provider (CP). This object is created
            when the business relationship is setup and exists
            as long as the business relationship exists. It
            provides an interface called Cons_UA_Access that
            provides operations for establishing and controlling
            service sessions in this reference point, i.e.,
            Connectivity service sessions.”;


    supports
        i_ConsUAAccess;

    initial
        i_ConsUAAccess;

};
```

4.4.16 Initial Agent Object Specification

```
// ODL of object template InitialAgent

object InitialAgent
{

    behavior
        behaviorText

        “   This object is the initial access point to a
            Connectivity Provider’s domain. It provides
            the function of authenticating a Connectivity
            User. It offers an interface i_ConsInitialAccess
            that provides the authentication function.”;


    supports
        i_ConsInitialAccess;

    initial
        i_ConsInitialAccess;

};
```

4.5 The event traces

We illustrate the usage of ConS-RP objects and interfaces specified in this document through a few of the representative event sequences in the following service scenarios.

- ConS access session
- Contract profile management service
- Connectivity control service
    - Connectivity session setup
    - Connectivity session modification
    - Connectivity session deletion
    - Connectivity session monitoring

The reader should be aware that the following event sequences illustrate only the typical usage and its associated operations, not the exhaustive list of usages and operations. For example, the contract profile management service allows several operations such as *Get_Authentication_Info*, *Update_Authentication_Info*, etc. In the following event trace sequence, however, only a few of these operations are used, just enough to illustrate how the contract profile management service can be used.

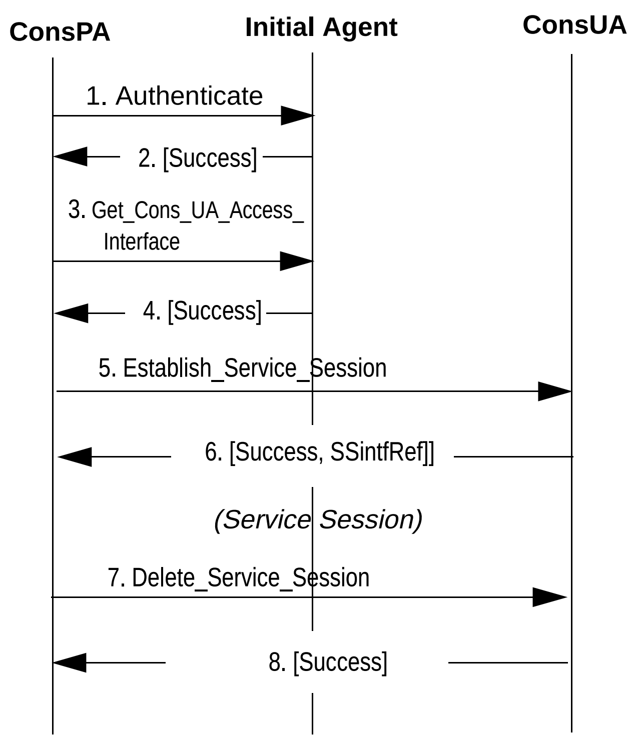
4.5.1 ConS Access Session

Preconditions

ConsPA has a reference to the Cons_Initial_Access interface of Initial Agent.

Postconditions

A secure association between ConsPA and ConsUA is established. Upon completion of *Get_Cons_UA_Access_Interface* (Step 4), ConsPA is ready to establish a service session. All the following service sessions can be made secure, by using the secure access session and a service session key.



SSintfRef: Connectivity Service Session Interface Reference

Figure 4-2. ConS Access Session

4.5.2 Contract Profile Management Service

Description

Upon completion of *Establish_Service_Session* (Step 2), an interface reference to the Contract_Profile_Mgmnt interface is passed back to the ConsPA, and the ConsPA (or some object in the CU domain) invokes contract profile management operations.

Preconditions

Access session between ConsPA and ConsUA is established.

Postconditions

The service session is deleted.

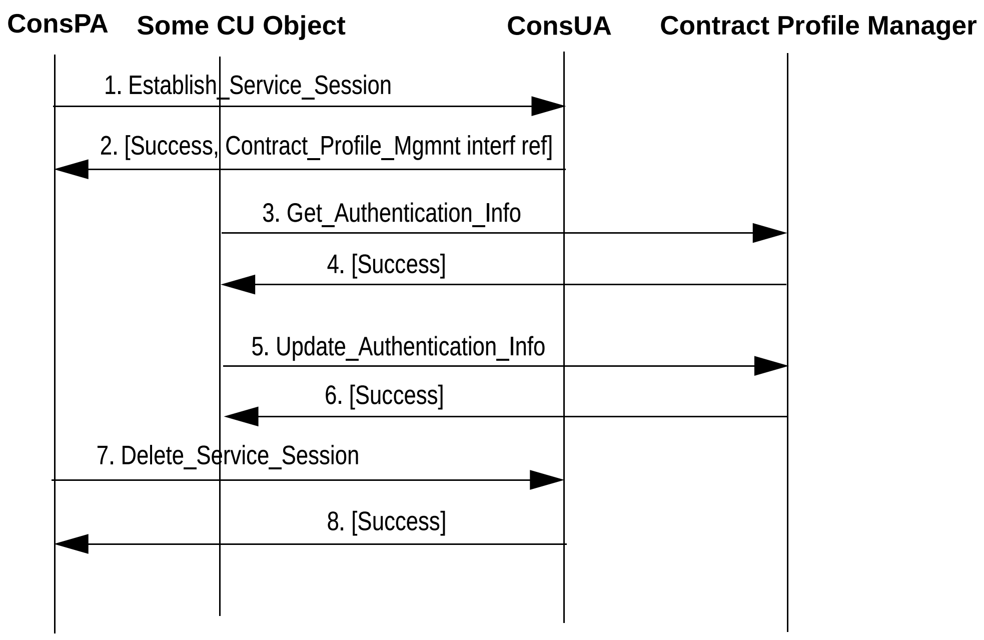

ConsPA
Some CU Object
ConsUA
Contract Profile Manager
1. Establish_Service_Session
2. [Success, Contract_Profile_Mgmnt interf ref]
3. Get_Authentication_Info
4. [Success]
5. Update_Authentication_Info
6. [Success]
7. Delete_Service_Session
8. [Success]

Figure 4-3. Contract Profile Management Service

4.5.3 Connectivity Control Service

4.5.3.1 Connectivity Session Setup

Description

CSM in the connectivity user domain requests the connectivity provider for creation of a connectivity session.[23]

Preconditions

An access session between ConsPA and ConsUA is established. A connectivity control service is established by ConsPA, from which the CSM has obtained a Conn_Session_Setup interface reference.

Postconditions

A connectivity session with one or more flow connections is created. The connectivity control service session continues.

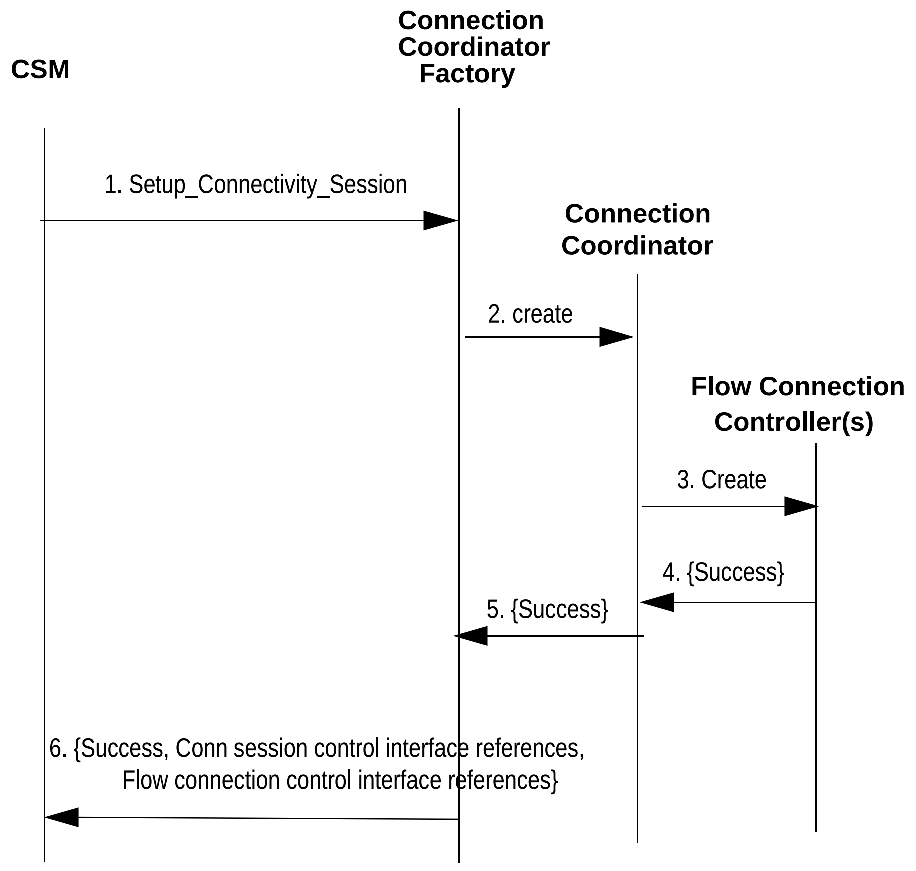

CSM
Connection Coordinator Factory
1. Setup_Connectivity_Session
Connection Coordinator
2. create
Flow Connection Controller(s)
3. Create
4. {Success}
5. {Success}
6. {Success, Conn session control interface references, Flow connection control interface references}

Figure 4-4. Connectivity Session Setup

23. Objects in the CU domain (such as CSM) are used in the event scenarios only for illustration. The usage of these objects is not prescriptive in this specification.

4.5.3.2 Connectivity Session Modification

Description

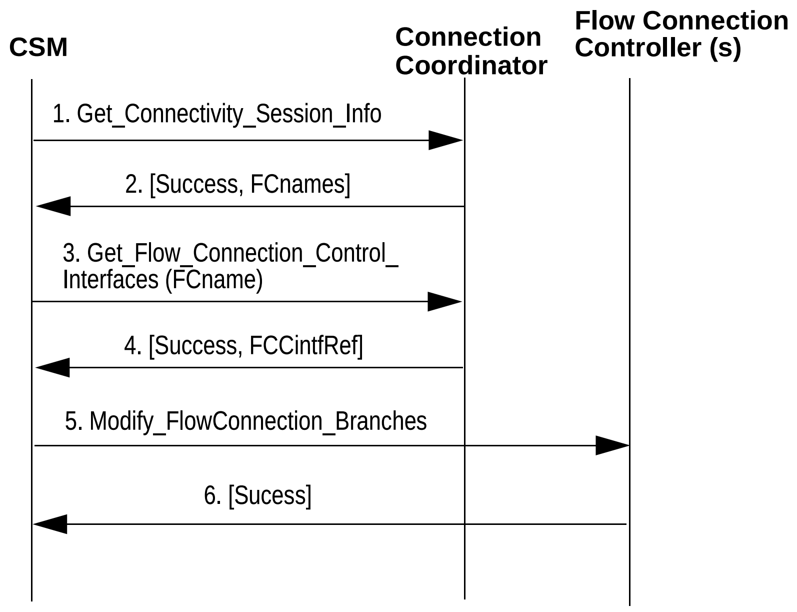
CSM in the connectivity user domain modifies a branch of a flow connection in the connectivity session.

Preconditions

An access session between ConsPA and ConsUA is established. A connectivity control service session for the modification of the connectivity session is established.

Postconditions

A flow connection in the connectivity session is modified. The connectivity control service session continues.


CSM
Connection Coordinator
Flow Connection Controller (s)
1. Get_Connectivity_Session_Info
2. [Success, FCnames]
3. Get_Flow_Connection_Control_ Interfaces (FCname)
4. [Success, FCCintfRef]
5. Modify_FlowConnection_Branches
6. [Sucess]

FCname: flow connection name
FCCintfRef: Flow_Connection_Control interface reference

Figure 4-5. Connectivity Session Modification

Note: Steps 1 to 4 in the above event trace are required only if the CSM does not already have the flow connection control interface reference.

4.5.3.3 Connectivity Session Deletion

Description

CSM in the connectivity user domain deletes a connectivity session.

Preconditions

An access session between ConsPA and ConsUA is established. A connectivity control service session is established for controlling the specific connectivity session.

Postconditions

The connectivity session is deleted.

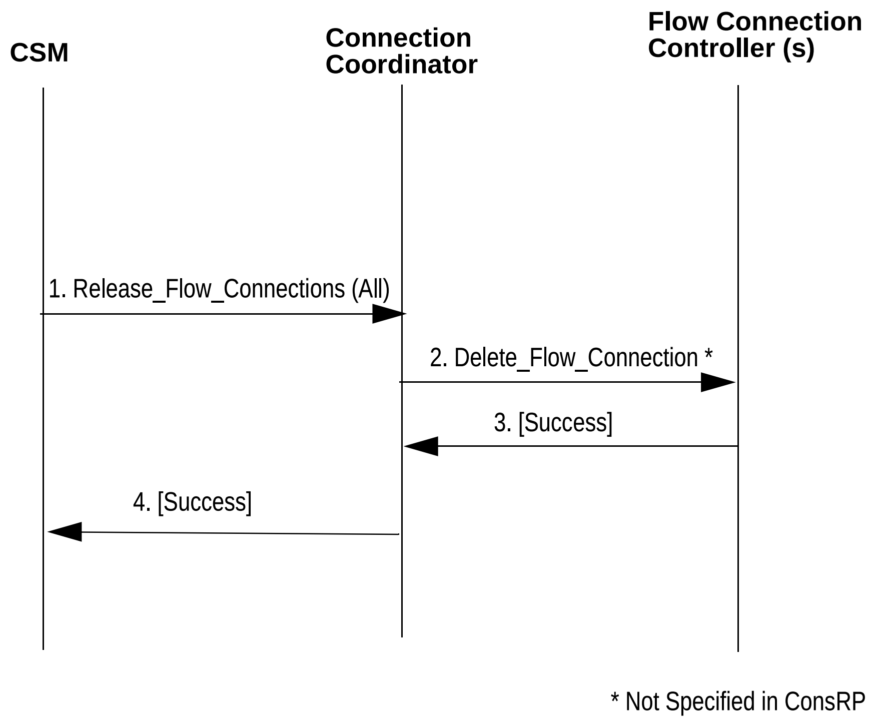

CSM
Connection Coordinator
Flow Connection Controller (s)
1. Release_Flow_Connections (All)
2. Delete_Flow_Connection *
3. [Success]
4. [Success]
* Not Specified in ConsRP

Figure 4-6. Connectivity Session Deletion

4.5.3.4 Connectivity Session Monitoring

Description

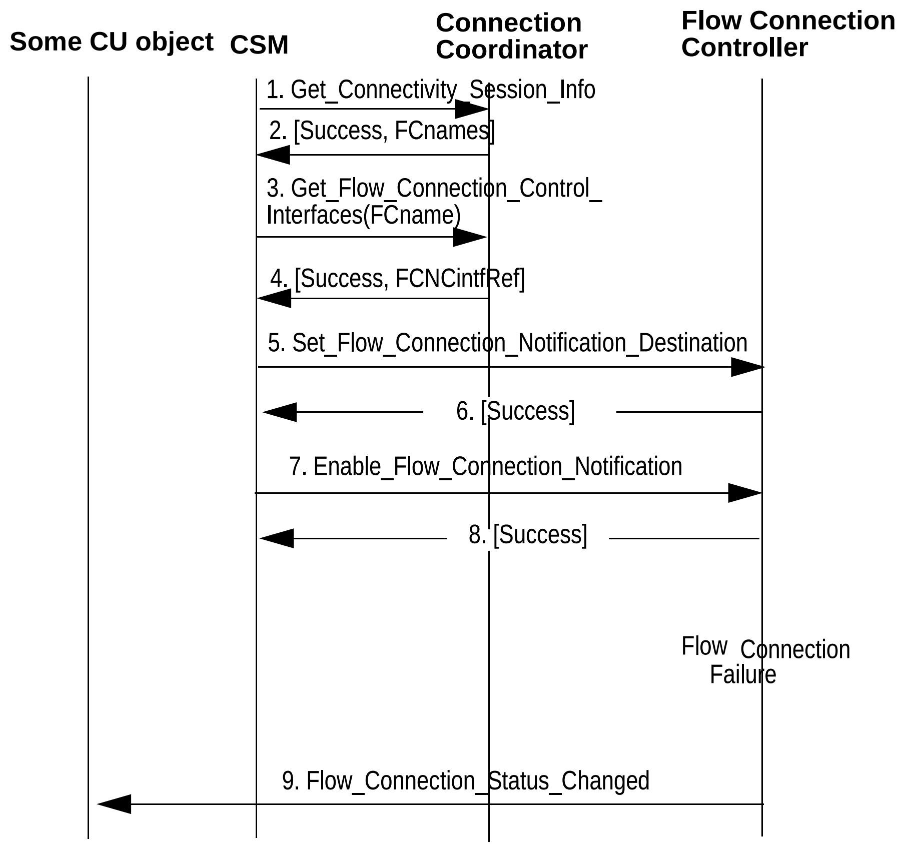
Flow connection manager reports a flow connection failure.

Preconditions

An access session between ConsPA and ConsUA is established. A connectivity control service session for controlling the specific connectivity session is established.

Postconditions

A flow connection failure notification is received by an object in the CU domain.


Some CU object
CSM
Connection Coordinator
Flow Connection Controller
1. Get_Connectivity_Session_Info
2. [Success, FCnames]
3. Get_Flow_Connection_Control_ Interfaces(FCname)
4. [Success, FCNCintfRef]
5. Set_Flow_Connection_Notification_Destination
6. [Success]
7. Enable_Flow_Connection_Notification
8. [Success]
Flow Connection Failure
9. Flow_Connection_Status_Changed

FCname: flow connection name
FCNCintfRef: Flow_Connection_Notification_Control interface reference

Figure 4-7. Connectivity Session Monitoring

Note: Steps 1 to 8 in the above event trace are required only if the flow connection notification destination interface is different from the default contained in the connectivity session profile. Even in such a case, steps 1 to 4 are required only if the CSM does not already have the flow connection notification control interface reference.

5. Suggestions for Further Work

The specification presented in this document is the first baseline specification of the ConS Reference Point and has addressed only a subset of the capabilities that can be associated with this reference point. Some of the topics that can be addressed in future versions of ConS-RP specifications are listed below:

- In terms of management capabilities, this initial specification has focused only on connection management. Future ConS-RP specifications need to address all management functional areas using concepts specified in TINA Network Resource Architecture [13]. Among the management areas, accounting and security management can be addressed first.
- Future ConS-RP specifications can address the issue of using generic DPE specifications for notification and security services, instead of the specifc services presented in this specification.
- Access part of future ConS-RP specifications should be aligned with the access part of other reference point specifications.

6. References

6.1 TINA Documents

[1] *Request for Refinements and Solutions: The ConS Reference Point*, Version 2.0, August 9, 1996.

[2] *TINA Reference Points*, Version 3.1, Document No. EN_RCJ.030_3.1_96, June 4, 1996.

[3] *Information Modelling Concepts*, Document No. TB_EAC.001_1.2_94, April 3, 1995.

[4] *Computational Modelling Concepts*, Version 3.2, Document No. TP_HC.012_3.2_96, May 17, 1996.

[5] *TINA Object Definition Language Manual*, Version 2.3, Document No. TR_NM.002_2.2_96, July 22, 1996.

[6] *Network Resource Information Model Specification*, Document No. TB_LR.010_2.1_95, August 25, 1995.

[7] *Service Architecture*, Version 4.0, Document No. TB_RM.001_4.0_96, October 28, 1996.

[8] *Network Resource Information Model Specification*, TINA Baseline Draft, May 15, 1996.

[9] *Connection Management Architecture*, Document No. TB_JJB.005_1.5_94, March 1995.

[10] *Connection Management Specifications*, Document No. TP_NAD.001_1.2_95, March 1995.

[11] *Response to a Request for Refinements and Solutions: The TCon Reference Point*, Version 1.0, November 8, 1996.

[12] *TINA Glossary*, Version 1.0, 1996.

[13] *Network Resource Architecture*, Version 3.0, Document No. NRA_v3.0_97_02_10, February 10, 1997.

6.2 Other documents

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

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

7. Acronyms

ABR	Available Bit Rate
ATM	Asynchronous Transfer Mode
CBR	Constant Bit Rate
ConS	Connectivity Service
ConSPA	Connectivity Service Provider Agent
ConSRP	Connectivity Service Reference Point
ConSUA	Connectivity Service User Agent
CP	Connectivity Provider
CPE	Customer Premises Equipment
CS	Connectivity Session
CSM	Communication Session Manager
CU	Connectivity User
DPE	Distributed Processing Environment
FC	Flow Connection
KTN	Kernel Transport Network
NFEP	Network Flow EndPoint
NFEPpool	Network Flow EndPoint pool
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
TCon	Terminal Connectivity Service
UBR	Unspecified Bit Rate
VBR	Variable Bit Rate

8. Glossary

In addition to the terms defined in TINA-C Glossary [12], this document uses several new terms. These terms are defined below in alphabetical order:

- **Connectivity Control Service:** A TINA service that provides capabilities for setup, modification, and release of **Connectivity Session**s. Using this service, **Network Flow Connection**s that are part of a connectivity session can be controlled either individually or in aggregation.
- **Connectivity Layer Network**: A transport network that is made up of one or more layer networks. The characteristic information accepted by a connectivity layer network can be different from the characteristic information delivered by the connectivity layer network. Within a connectivity layer network, a layer network may be directly connected with one or more other layer networks. Since different layer networks transport different characteristic information, the layer networks interconnection involves adaptation of characteristic information; i.e., information transported by one layer network is adapted and converted into information transported by another layer network.
- **Connectivity Provider**: A TINA application (composed of one or more TINA computational objects) that provides the Connectivity Service.
- **Connectivity Service**: A TINA service that consists of two component services: the **Contract Profile Management Service** and the **Connectivity Control Service**
- **Connectivity Service Reference Point (ConsRP)**: A TINA reference point that represents the interactions between a Connectivity Provider and a Connectivity User.
- **Connectivity Service Session**: A service session between a **Connectivity User** (CU) and a **Connectivity Provider**(CP) in which the CU uses a set of related management and control capabilities of the CP. A connectivity service session is an instantiation of either the **Contract Profile Management Service** or the **Connectivity Control Service**.
- **Connectivity Session**: A set of Network Flow Connections that have been grouped together for some purpose, e.g., connections that are part of the same communication session, connections that have related lifetimes, and so on.
- **Connectivity User**: A TINA application (composed of one or more TINA computational objects) that uses the Connectivity Service.
- **Contract Profile Management Service**: A TINA service that provides capabilities for retrieval and modification of the contract profile information associated with a Connectivity User.
- **Flow Connection**: See **Network Flow Connection**.
- **Flow Connection Branch:** See **Network Flow Connection Branch**.
- **Flow Endpoint**: See **Network Flow Endpoint**.
- **Flow EndPoint Pool**: See **Network Flow Endpoint Pool**.
- **Layer Network**: A transport network made up of components of a specific transmission and/or switching technology that transports information of a specific format, coding and rate. The information type transported by a layer

network is called its characteristic information. The termination points at which a layer network accepts or delivers its characteristic information are called trail termination points.

- **Network Flow Connection**: A network resource that transports information across a connectivity layer network between two or more Network Flow Endpoints. The characteristic information associated with the different flow endpoints of a flow connection may be different. A Network Flow Connection is made up of one or more trails.
- **Network Flow Connection Branch:** The connectivity between the root Network Flow Endpoint of a point-to-multipoint Network Flow Connection and a leaf Network Flow Endpoint of the flow connection.
- **Network Flow Endpoint**: An abstraction that represents in a technology independent manner a termination of a Network Flow Connection. Has a one-to-one association with a trail termination point.
- **Network Flow Endpoint Pool**: A topological component of a Connectivity Layer Network. Represents the potential for terminations of Network Flow Connections in a CPE. A network flow endpoint pool is either a collection of link termination points that are collocated on the same CPE or a portion of a link termination point. The link termination points that make up a network flow endpoint pool belong to the same layer network.
- **Root Flow Endpoint**: For a point-to-multipoint unidirectional Network Flow Connection, the source Network Flow Endpoint where the traffic originates is called the Root Flow Endpoint. For a point-to-point Network Flow Connection, one of the two Network Flow Endpoints associated with the connection is arbitrarily designated as the Root Flow Endpoint.
- **Leaf Flow Endpoint**: For a point-to-multipoint unidirectional Network Flow Connection, a sink Network Flow Endpoint where the traffic terminates is called a Leaf Flow Endpoint. For a point-to-point Network Flow Connection, one of the two Network Flow Endpoints associated with the connection is arbitrarily designated as the Leaf Flow Endpoint.
- **Trail**: A network resource that transports information across a Layer Network between two or more Trail Termination Points and ensures the integrity of the information transfer.
- **Trail Termination Point**: A termination of a trail where the characteristic information associated with the trail is accepted and/or delivered.