Deliverable-5.4

Consolidated Network Management System

Deliverable Editor: Victor Alvarez, BISDN

Publication date:	30-June-2016
Deliverable Nature:	Report/Software
Dissemination level (Confidentiality):	PU (Public)
Project acronym:	PRISTINE
Project full title:	PRogrammability In RINA for European Supremacy of virTualIsed NEtworks
Website:	www.ict-pristine.eu
Keywords:	multi-layer management, configuration management, RINA, performance management, RIB model, Manager, Management Agent
Synopsis:	This document describes the Network Management System for the second iteration of PRISTINE DMS.

The research leading to these results has received funding from the European Community's Seventh Framework Programme for research, technological development and demonstration under Grant Agreement No. 619305.

List of Contributors

Deliverable Editor: Victor Alvarez, BISDN
i2CAT: Eduard Grasa
TSSG: Micheal Crotty, Miguel Ponce de Leon, Jason Barron
CN: Roberto Riggio, Kewin Rausch
ATOS: Javier Garcia, Juan Vallejo, Miguel Angel Puente
TRT: Hamid Asgari

Disclaimer

Executive Summary

This document describes the results and major contributions to the Network Management System for RINA networks that the PRISTINE project has been researching and developing. RIB design, NMS System Proof of Concept implementation and validation, configuration management, performance management and security management have been the major areas of work.

RIB design : The work performed in D5.4 enables the specification and validation of RIB objects using a formal language. RIB object model compilers and code generation tools enable developers to seamlessly incorporate the RIB model into their RINA management applications.

- **RIB: Language**. Previous WP5 deliverables (D5.1 and D5.2) provided an informal, descriptive specification of a language to describe the objects used to model the state of the systems in a RINA network (represented in a logical schema called the RIB, Resource Information Base). D5.4 has completed this work by producing a specification of a formal language to model the RIB objects: GDRO (Guidelines for the Definition of RIB Objects). Section 2 introduces the language constructs and Annex 1 provides the full specification. With GDRO in place, it is possible to define a RIB object model whose correctness can be formally verified.
- **RIB: Verification tools**. PRISTINE has provided a reference compiler to assist in the production of GDRO models. This allows any produced GDRO models to be validated against the language constraints enforced by GDRO. Java code generation tools are also part of the toolchain, freeing developers involved in the implementation of Managers or Management Agents managing RINA networks from coding classes to represent the RIB objects. Last but not least, the RIB toolchain also supports automatic documentation generation from the GDRO source files.
- **RIB: Object model specification**. GDRO and its associated tool-chain has been leveraged by WP5 partners to complete the formal specification of the RIB objects required to manage a RINA network, one of the key deliverables of PRISTINE WP5. The RIB object model is now consistent and fully validated - as compared to previous deliverables -

and has been incorporated into the Manager code-base. This is a crucial step in enabling real-world management of RINA networks.

Status of implementation and validation : Implementation and validation activities have been ongoing through the period covered by this deliverable.

- **Status of the implementation**. Since D5.3 we have continued to improve the implementation of the Manager and the Management Agent with new features and stability improvements. The current software release provides enough functionality to enable centralized configuration management of RINA networks via the PRISTINE Manager. The implementation will continue to be improved as part of WP6, since the Manager and Management Agent will be used in WP6 experiments and demonstrators (not only to showcase and measure Management function per se, but as a way to facilitate the configuration of experiments).
- **Validation**. A public demo showcasing the RINA Network Management System developed by PRISTINE was held in the TNC 2016 conference. The demo validated all the steps required for configuring a RINA network of 12 nodes: i) systems bootstrapping, Management Agents enrolling to the NMS DAF; ii) Manager discovering and keeping track of all managed systems; iii) Manager instantiating and configuring DIFs in various systems, which involves: creation of IPC Processes, registration to N-1 DIFs, configuring them with the DIF policies and discovering neighbour IPC Processes. In addition to its validation purpose, the demo was useful in providing a real example of the RINA benefits when it comes to Network Management.

Configuration Management : Supporting reusable templates, and a management case study has been undertaken.

- **DIF templates**. One of the unique features of the RINA architecture is that every layer has the same two protocols with different concrete syntaxes and different policies. Therefore, it is possible to design a common DIF template that captures specification of any DIF, as well as its configuration. This extremely simplifies network configuration management compared to current approaches, which require different configurations for different layers depending on the protocols that are

used. D5.4 reports on the initial attempt to design a DIF specification template as well as its associated tool-chain.

- **Large DC Network configuration management case study: current tech vs. RINA**. This section performs a comparative analysis in the complexity of managing an IP-based and a RINA-based large-scale multi-tenant data centre networks. Configuration management is the main target of the analysis although some hints on performance and security management are also provided. The analysis shows that the commonality built into the RINA architecture and the single type of recursive layer with a uniform API greatly reduces the complexity of the models the Network Management System (NMS) uses to understand the state of the managed network.

Performance Management : Adding inference on events, providing a centralised resource reservation, and supporting VNF configuration.

- **Event Inference**. Performance management concerns itself with the identification of sub-optimal configurations and behaviour within the RINA network. The goal is to optimise the system based on a set of high-level strategy goals. This allows performance tuning, where the allocated resources are optimised to the aggregate needs, increasing available resources to over-loaded DIF's and reducing resources to under-utilised ones. This section develops an approach to let the NMS autonomously tune the performance of a DIF as a reaction to notifications from Management Agents indicating some issues. Once received, the Notifications can be correlated to a DIF and analysed using some advanced machine learning techniques. Automated policy responses can be included in the declarative configuration specification, allowing the NMS to activate them on certain trigger events. Here, strategies are employed to inspect, prioritise and ultimately decide if corrective actions are desirable.
- **Centralised resource reservation strategy for RINA**. This section presents a different approach to the multipath routing algorithms described in previous deliverables, taking a step forward the strategy of the forwarding decisions. By leveraging the central Manager in the RINA network and the RIB information objects, it has been possible to develop a centralised reservation strategy to determine the best end-to-end path in a multipath environment, taking into account the QoS requirements of the flows and the bandwidth capacity of the DIFs.

This decision strategy is complemented with a reroute algorithm that allows the manager to move and optimise the distribution of the flows between the available paths once they have been allocated. Experimental results presented at the end show the benefits of using this strategy over previously developed multipath algorithms in RINA to minimize the amount of rejected flow requests.

- **Virtual Network Function Forwarding Graph (VNF-FG) Configuration**. This section describes of to configure a VNF-FW using the NFV Over RINA (NORI) tool-kit. The forwarding graph describe the logical sequence of VNF(s) a certain portion of the traffic must traverse as well as their order. The section shows how using RINA it is possible to specify custom packet processing operations by combining different VNFs. As opposed to legacy TCP/IP (even the ones based on recent SDN solution such as OpenFlow), RINA allows to chain different VNFs and to specify precise QoS policies on such virtual link, providing NFV administrators with an unprecedented level of control of their network services.

Security Management : This section outlines the main goals of managing the security of RINA networks, and discusses in more detail a specific use case: security management in networks supporting MLS (Multi-Level Security).

Table of Contents

List of Figures

List of acronyms

3PP	Third Party Provider
AMQP	Advanced Message Queuing Protocol
CACE	Common Application Connection Establishment
CDAP	Common Distributed Application Protocol
CLI	Command Line Interface
DAF	Distributed Application Facility
DAP	Distributed Application Process
DIF	Distributed-IPC-Facility
DMS	Distributed Management System
DSL	Domain Specific Language
DTCP	Data Transfer and Control Protocol
EFCP	Error and Flow Control Protocol
ES	Event System
FCAPS	Fault, Configuration, Accounting, Performance and Security
IPC	Inter-Process Communication
IPCP	Inter-Process Communication Process
IPCM	Inter-Process Communication Manager
NM	Network Management
NMS	Network Management Service
OODA	Observe, Orient, Decide, and Act
OSS	Operation Support System
PDU	Protocol Data Unit
RIB	Resource Information Base
RINA	Recursive Inter-Network Architecture
SDU	Service Data Unit
SCTP	Stream Control Transport Protocol
VLAN	Virtual LAN
VM	Virtual Machine

1. Introduction

Today computer networks are designed as multiple co-operating layers that perform different functions implemented by a diverse set of protocols. Allowing distributed applications to achieve an optimal performance through the network requires the different layers to collaborate, so that application quality requirements can be enforced down to the physical wires. Each layer and protocol needs to be properly configured, and this configuration must be continuously updated due to the dynamic nature of the network operational environment (node and link failures, degradation of physical layer conditions) or changes in the volume and characteristics of the traffic offered to the network.

Network Management Systems (NMSs) are in charge of keeping the network operating in optimal conditions, monitoring the behaviour of the different layers that make up the network, reasoning about their current and desired states, and taking any corrective actions to update the network configuration if needed (due to misalignments in performance or reliability and security issues). Trough years the role of NMSs has switched from performing low-level control decisions to taking part in higher levels of decision, since more and more autonomic control functions such as routing, resource allocation, flow recovery or congestion control have been embedded into the network layers. Thus, we can characterize the goal of modern NMSs to be that of “monitor and repair”, as illustrated in Figure 1.

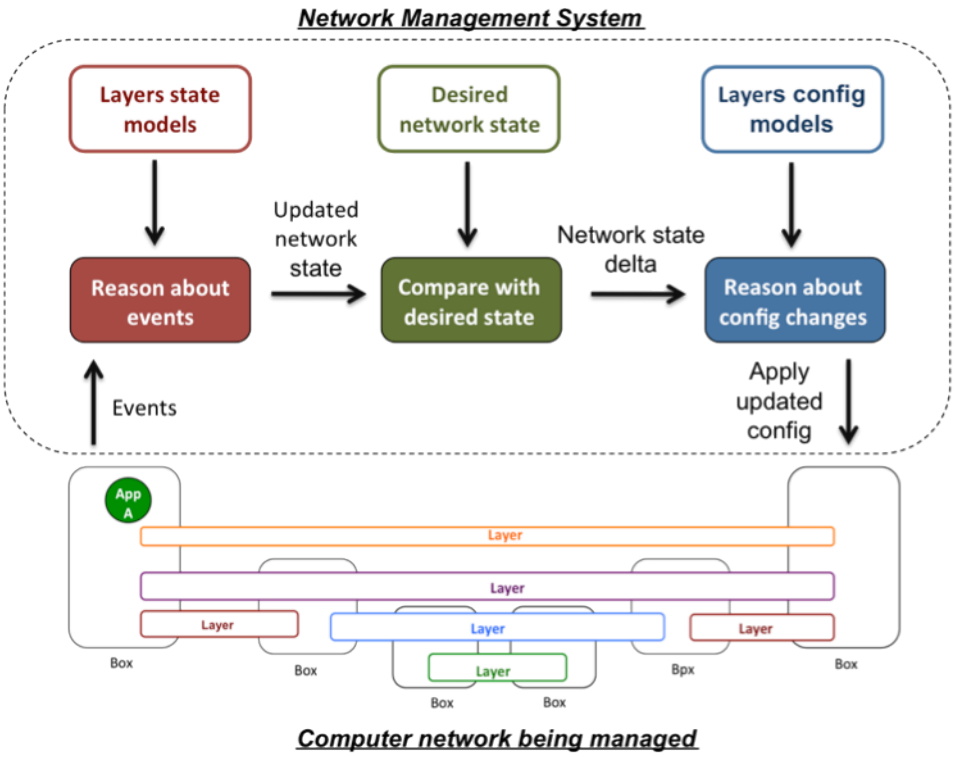

Figure 1. Model of the Network Management System as a closed control loop

Figure 1 conceptualizes the main functions of a NMS, forming a closed control loop with the computer network being managed [cnsm15]. The NMS receives events (notifications, reports) from the network, which contain relevant information about changes in the state of one or more layers. The first step of the NMS is to reason about these events, performing data analytics in order to understand what is the updated network state. To do so, the NMS needs models about the state of the different layers that compose the network. Once it has understood the updated state, the NMS has to compare it with a reference desired state – provided by the network managers – and obtain the "state delta" between the desired network state and the current network state. Finally, this state delta is used as the input to a module that needs to reason about the changes in the network configuration that need to be applied in order to get as close as possible to the desired network state (the "repair" function of the NMS). This stage requires the NMS to have models of the network layers configuration. After applying the configuration changes, the NMS will get more events that will allow it to reason about the effectiveness of the new configuration (closing the loop of the control system).

The current Internet architecture is based on the paradigm of functional layering: each layer performs a different function, usually implemented by

multiple protocols. For example: the transport layer performs end-to-end error and flow control via protocols such as TCP, UDP or SCTP (Stream Control Transport Protocol). The network layer performs relaying over a number of data link layers of different technology. IP is the universal protocol at this layer, but a number of other protocols perform the dynamic control functions of the network layer, such as routing or resource reservation. Data link layers take care of the transmission of data over physical links of different characteristics; therefore different protocols are used depending on the link characteristics. In addition to the layers present in the theoretical architecture (L1 to L4 with applications on top), practical network deployments include also other type of layers, such as "layers 2.5" like MPLS – Multi Protocol Label Switching, separated customer and provider data link layers in Provider Backbone Bridging (PBB) or "virtual layers" on top of transport in virtual network overlays to allow for multi-tenancy in data-centres (NVGRE – Network Virtualization using Generic Routing Encapsulation or VXLAN – Virtual eXtensible Local Area Network [icton13]).

We can see that managing networks made up of functional layers becomes complex very fast, since each layer has a different state, configuration and service model that the NMS must understand. What is more, interaction between layers varies depending on the type of layer and protocol; therefore predicting what will be the effect of a configuration change across multiple layer becomes very hard. Last but not least, there are multiple network management protocols (the protocols used to get notifications from the network and update its configuration) such as SNMP (Simple Network Management Protocol), CMIP (Common Management Information Protocol), WBEM (Web-Based Enterprise Management) or NETCONF and multiple languages to model the functionalities of the different layers, such as SMI (Structure of Management Information), SID (Information Framework), CIM (Common Information Model) or YANG. The combination of NETCONF as a management protocol and YANG as a modelling language [commag10] is lately gaining traction in the ISP industry, but it still does not provide a complete coverage of all the different network layers found in a typical provider network today. SDN, Software Defined Networking, has also been proposed to simplify network management, by reducing the number of autonomous functions present in the hardware and logically centralizing them at certain network points.

However it does not solve the inherent problem of functional layering: each layer is different.

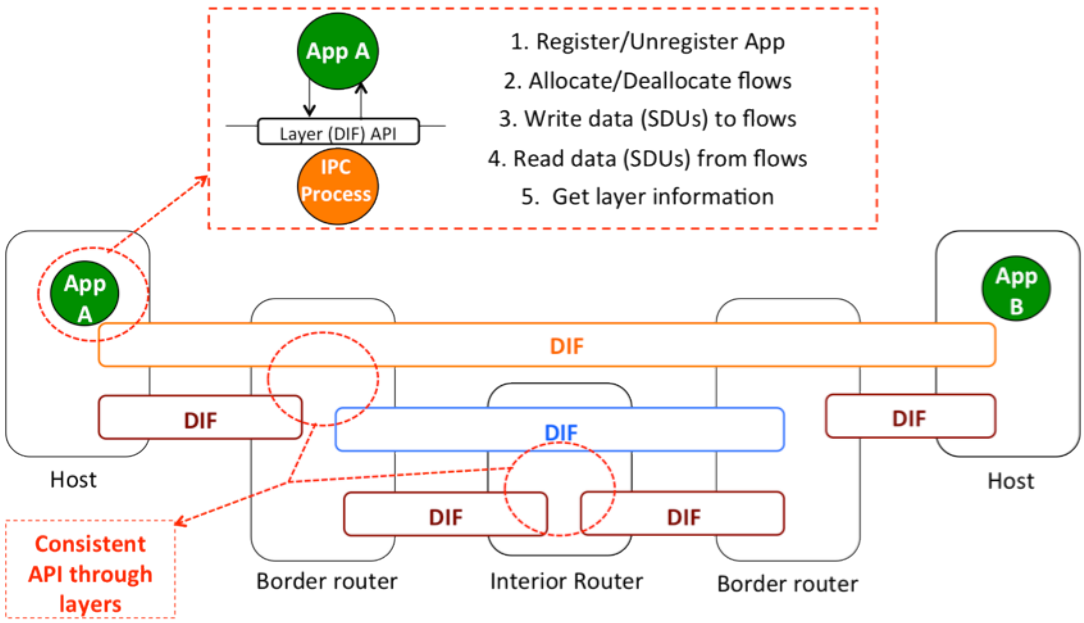

Figure 2. Structure of RINA

As shown in Figure 2, in RINA all layers provide the same service (Inter Process Communication or IPC between two or more application processes), and perform the same functions. All the functions of a layer can be separated in to the fixed parts (mechanism) and the variable parts (policies), therefore the behaviour of a layer can also be tailored to its operational environment. Managing RINA networks is radically more simple than managing the networks of today, because: i) there is a single type of layer, therefore the NMS only needs to understand one state and configuration model; ii) since all layers are the same and provide the same API, if is much easier to reason about the interaction of multiple layers and last but not least iii) there is a single management protocol that is sufficient to manage all layers of the network. The simplicity gains are huge: not only network management will become cheaper, but automating the troubleshooting and reconfiguration of multi-layer networks will become feasible, allowing the network operator to predict the effect of updating the configuration of multiple layers at once.

2. RIB Design

The following section outlines the proposed language used for Resource Information Base (RIB) for the PRISTINE project. Within RINA, the RIB is constructed with two object hierarchies:

Object *inheritance* hierarchy
: The various utility RIB object (RO) definitions required to represent the common information and data manipulated during the operation of a DMS. This information is essentially static in the sense it is defined by the managed object language (GDRO) and can only be extended by adding additional managed object definitions at runtime.

Object *containment* hierarchy
: This hierarchy captures the runtime information and instance related configuration and state of the DMS. This information contains all the necessary detail required to represent both DIFs and DAFs, their current state, and inter-relations.

Previous versions of the RIB have been captured in a semi-formal way, where the inheritance and containment relationships have been presented in UML models, and the semantics represented as textual information or comments (in another file). Explicitly, defining a formal representation allows:

1. Combination of varying syntax definitions into a common textual format,
2. A compiler to verify the definitions,
3. Provides the possibility to allow cross-checks between RIB Objects, and validations of the types,
4. And finally, allows the possibility for automating these checks.

The next section describes the language used to capture the RIB objects in a formal way.

2.1. Guidelines for the Definition of RIB Objects

A full formal description of GDRO is given in the annex. This section aims to be a gentler introduction to the GDRO langauge. Guidelines for

the Definition of RIB Objects (GDRO) shares some of its motivation from Guidelines for the Definition of Management information Objects (GDMO) as specified by the ISO. However a number of simplifications have been applied, to make the language easier to manage. Some key features to highlight are:

1. Removal of multiple inheritance for classes.
2. Avoidance of ATTRIBUTE-GROUPS and PACKAGES
3. Better support for specifying behaviour in terms of policies and strategies.
4. A simplification of notification specifications

The following table gives a brief overview of the chief constructs that make up the GDRO language, and how they may be combined in specifying RIB Objects.

Table 1. GDRO feature summary table

Construct	Behaviour	Inheritance	Attribute	Operation	Containment	Notification
class	Yes	Yes	Yes	Yes	Yes	Yes
attribute	Yes	No	-	No	No	No
operation	Yes	No	Yes	-	No	No
containment	Yes [a]	No	No	No	-	No
notification	Yes	No	Yes	No	No	-

[a] Only create and delete strategies are allowed.

2.1.1. GDRO: Rib object example

Perhaps the best way to introduce GDRO is to show an example. The following is a example of a GDRO class definition, in this case the class for "ManagmentAgent" RIB object.

```
ro class ManagementAgent ❶
  behavior
      "This class represents a Management Agent Instance." ❷
  extends /Classes/ApplicationProcess  ❸

  // optional attribute definitions
  // optional operation definitions.
```

```
  // optional notification definitions
  // optional containment specification

  registered-as ERoot Classes(1) 13  ❹
;
```

❶ Declaration of the ManagementAgent RIB class
❷ A description of the purpose of the class
❸ A link to the class it inherits from.
❹ Unique registration identifier

A RIB object class definition is terminated by the end of a file, i.e. no closing keyword is required. Every class is required to have a unique identifier. This class is registered in the "Classes(1)" name-space, and a unique id of "13" The attribute, operation, notification and containment sections are described in greater detail below.

Remember a GDRO class can only inherit from a single parent class. Multiple inheritance is not currently supported.

Attribute definition

A RIB object can technically have **zero or more** defined attributes. An attribute must have a type, a name and a description of the purpose of the attribute. Attributes can be defined as shown in the following example:

```
attributes ❶
  /Attributes.ManagementAgentId       agentId ❷
      "uniquely identifies the Management Agent within the Processing
 System" ❸
  /Attributes.MasterAgent                masterAgent
      "True if the Management Agent is the master of this processing
 system"
; ❹
```

❶ Attributes definition must begin with the "attributes" keyword
❷ Every attribute has a type and an attribute name.
❸ A description of the purpose of the attribute. An attribute definition is terminated by a new line.
❹ A ";" terminating the list of attributes.

The names of attributes must be unique within the class. The type must be defined in the above case "ManagmentAgentId" is defined in the "Attributes" folder with the type name "ManagementAgentId.gdro".

2.1.2. Attribute type definition

Every attribute is defined by a type. An example of a type definition is given below:

```
ro type definition T_DIFInfo ❶
  "The information of a DIF, including its configuration" ❷

  T_String                type        "The type of DIF"
  /Types.T_APNamingInfo   name        "The name of the DIF"  ❸
  /Types.T_DIFConfig      difConfig   "The DIF configuration"

  registered-as ERoot Types(5) 8  ❹
; ❺
```

❶ The "ro type definition" keywords followed by the name of the type
❷ A description of the purpose of the type
❸ A contained type attribute
❹ Unique registration identifier.
❺ Type definitions are terminated by a ";"

Types can contain other named types as components of there specification (see 3 above). The syntax here follows the convention set for attributes, with a type, name and a description. In the above example, the `policySet` type member is defined in the "Types" folder in the "T_PolicyConfig" file. Every type must have unique registration id, in this case, defined in the "Types(5)" name-spaces, with a unique id within that name-space of "8". All type definitions are terminated by a ";".

Policy definition

Policies share some common traits with classes, in that they can contain attributes, inherit from other policies, but not other classes. All policies must inherit from `RINAPolicy`. A sample definition of a policy is given below:

```
ro policy RIBUpdatePolicy  ❶
```

```
        behavior "Defines how often a set of objects in the RIB need to be
updated,
          performing what remote operations and on which IPC Processes"
        extends /Policies/RIBDaemon     ❷
        ❸
        registered-as ERoot Policies(4) RIBDaemon(8) 1  ❹
;  ❺
```

❶ The definition of a type "T_NSMConfig"
❷ The Policy is an extension of the RIBDaemon policy. RIBDaemon policy in turn inherits from RINAPolicy
❸ In this case no additional attributes are defined for the policy
❹ Unique registration identifier.
❺ Type definitions are terminated by a ";"

The `RIBUpdatePolicy` inherits a single attribute from `RINAPolicy` called `policyConfig` with a type of POLICY_CONFIG. POLICY_CONFIG is a set of (name,value) pairs that contain any necessary policy parameters that are needed by the policy.

The following is an example of a RIB object class making use of a defined policy. Policies are defined at the same level as an attribute, and by definition policies can be set or unset based on the policy type.

```
ro class RIBDaemon
    behavior
        "This class represents a RIB Daemon Application Entity."
    extends /Classes/ApplicationEntity

    attributes
        /Attributes.RIBVersionList      supportedRIBVersions "The list of
RIB versions supported by this App/IPCP"
    ;

    policies  ❶
        /Policies/RIBDaemon/RIBUpdatePolicy,
        /Policies/RIBDaemon/RIBReplicationPolicy,
        /Policies/RIBDaemon/RIBSubscriptionPolicy,
        /Policies/RIBDaemon/RIBLoggingPolicy  ❷
    ; ❸
```

❶ The "policies" keyword

❷ A reference to the "RIBLoggingPolicy"
❸ Policies are terminated by a ";"

In the above example, four policies are referenced, which can apply to a "RIBDaemon".

As policies are not explicitly named, there is an implicit understanding that only one instance of each policy type can be present in a RIB object instance at a time.

Operation definition

A RIB object can technically have **one or more** defined CDAP operations. An operation has a CDAP operation type , a description of the purpose and side effects of invoking the operation. Each RIB object is required to support a "read" operation. An example can be found below:

```
operations ❶
    read "read Management Agent naming information" ❷
        out        T_String        managementAgentInfo ❸
          "The Management Agent naming information: DAP name/instance, "
          "list of synonyms and ipc process id; plus the flag indicating "
          "if it is the master" ❹
    cancel-read "cancel ongoing read operation" ❺
; ❻
```

❶ Operations definition must begin with the "operations" keyword
❷ A CDAP operation and a description of the purpose of the attribute.
❸ The keyword "out", followed by a return type and its name.
❹ A description of what is returned.
❺ An operation definition is terminated by a new line beginning with the next operation.
❻ A ";" terminating the list of operations.

Each individual CDAP operation definition is terminated by a new line. The operation name must correspond to a CDAP operation name, ie. one of [create, delete, read, cancel-read, write, start or stop]. A return type is optional, however if it is not specified then a generic CDAP operation response is generated, giving a result code (0 = success). This is imposed by the use of CDAP. In the above example, there is a explicit return object of type *String*.

Containment definition

A RIB object can technically have **zero or more** defined contained objects. Each containment has a contained RIB object class, a synonym and optional defined strategies for creation an deletion. An example, can be found below:

```
contains ❶
    /Classes/DAFManagement as "dafmanagement" ❷
        create-strategy "This object is automatically created on "
          "ManagementAgent creation." ❸
        delete-strategy "This object is automatically deleted on "
          "ManagementAgent destruction."
    /Classes/IPCManagement as "ipcmanagement"
        create-strategy "This object is automatically created on "
          "ManagementAgent creation."
        delete-strategy "This object is automatically deleted on "
          "ManagementAgent destruction."
    /Classes/RIBDaemon as "ribdaemon"
        create-strategy "This object is automatically created on "
          "ManagementAgent creation."
        delete-strategy "This object is automatically deleted on "
          "ManagementAgent destruction."
; ❹
```

❶ Containments definition must begin with the "contains" keyword
❷ The contained object type and an explicit name
❸ A description of the purpose of the attribute. An attribute definition is terminated by a new line.
❹ A ";" terminating the list of containments.

This containment is slightly unusual in that there is a single object of each type. In this form the "as" keyword is used to give an explicit "name" to be registered. At (2), this is "dafManagement". Containments are unique in that they can have strategies associated. One example, is where the "container" is responsible for the creation (and deletion) of the contained object.

This form should only be used when there is a one-to-one relationship between the container RIB class and the contained RIB class. A second, more general example is given below:

```
contains
   /Classes/DirectoryForwardingTableEntry with-attribute key    ❶
     create-strategy "This object is created when a new entry "
      "is added to the Directory Forwarding Table"
     delete-strategy "This object is removed when an entry is "
      "removed from the Directory Forwarding Table"
 ;
```

❶ Containment where the registration attribute is specified

Here the "with-attribute" keyword is given to associate an identifier to be registered. For example, the value of the attribute "key" in the object to be contained is used to register the object in the container. By convention, this registration is done via an Attribute Value Assertion for example, "key=4222". This form should be used when multiple objects of the same RIB object class need to contained in the container class, i.e. a one-to-many relationship.

2.1.3. Notification definition

Notifications are treated as first class objects, so there definition mirrors that of RIB object classes. A key difference is that notifications are not allowed to have dynamic behaviour specified. Notifications are not allowed to contain operations (with a defined behaviour), or containment specifications (which specify creation and deletion behaviour). An example is given below:

```
ro notification CreateIPCProcess    ❶
  behavior
      "Triggered when an IPC Process is created"  ❷

  attributes  ❸
    /Types.T_APNamingInfo namingInfo
      "The naming information of the IPC Process"  ❹
    T_Int     ipcpId
      "The id of the IPC Process within the computing system"
  ;

  registered-as ERoot Notifications(3) 1  ❺
;
```

❶ Declaration of the ManagementAgent RIB class
❷ A description of the purpose of the class
❸ A list of attributes contained in the notification.
❹ An attribute with a complex type
❺ A unique registration

In the above example, a notification is defined by the "notification" keyword, followed by the name of the notification. Notifications are allowed to specify behaviour. By convention this behaviour description should describe the behaviour that may trigger the notification to be generated. A notification is made up from a set of attributes, each with a given type. In the example above (4), an attribute called "namingInfo" contains a type "T_APNamingInfo" defined in the "Types" folder. As with classes, all notifications are required to be uniquely registered, here in the "Notifications(3)" name-space with a unique id of "1".

2.2. RIB verification tools

PRISTINE has provided a reference compiler to assist in the production of GDRO models. This allows any produced GDRO models to be validated against the language constraints enforced by GDRO.

The GDRO compiler has three ways of interacting with it.

1. From the command line (for verification of models)
2. From the command line (for code generation)
3. Or from the maven plugin (for regeneration of the RIB model code)

2.2.1. GDRO compiler (GDRO validation)

The following command invokes the GDRO compiler on the RIB model contained in the `rib-model/pristine` folder. This model is build with the specification `pristine-specification` containing meta-data e.g. version, update date, etc.

```
./bin/sh/gdro-comp-tool.sh -m ./rib-model/pristine -s ./rib-model/
pristine-specification
```

The command responds with errors if there are syntax errors in the GDRO model. For illustration purposes, here is a sample of an error:

```
gdro-comp-tool: error
 => Neighbor2.gdro:16:5 - extraneous input 'underDIFs' expecting {'ERoot',
 VAL_STRING, ':', '.', '>', ';', '/'} ❶
gdro-comp-tool: found 1 errors
gdro-comp-tool: problem parsing GDRO files: found 1 errors, cannot
 continue
```

❶ Error showing the line number, file and a error message.

In this example, the GDRO compiler was expecting a Type definition, not the type name "underDIFs". Additional information on the checks performed can be done via the `--show-gcc` command line option.

Additional information on the model, for example, the number of classes defined can be produced through the use of the `--show-statistics` command line argument.

2.2.2. GDRO compiler (for java code generation)

The GDRO compiler can also be used to generate Java code for the RIB. In this form a "target" is specified on the command line as follows:

```
./bin/sh/gdro-comp-tool.sh -m ./rib-model/pristine -s ./rib-model/
pristine-specification -t java -o ./generated
```

The generated classes extend the `RO_Base` class, and implement getters and setters to access the RIB. The Java files are placed in the `./generated` sub-folder of the current directory.

2.2.3. GDRO compiler (with Maven integration)

The final method is to use the GDRO compiler as part of automated checks or for code generation within an existing build. For this use-case a Maven plug-in is provided that can be seamlessly integrated into automated or semi-automated builds.

2.3. RIB object model

All the model objects for a RINA network described in D5.2 have been formally specified using GDRO and validated with the tool-chain designed for this purpose. During the exercise inconsistencies in the model as well

as missing items were identified and fixed. The end result is a formally verified RIB object model for RINA networks described in GDRO. Annex Appendix B contains the model documentation, auto-generated by the tool chain from the source GDRO definitions.

3. State of the implementation and validation

3.1. State of the implementation

The software architecture and high-level design of the Manager and the Management Agent has not changed from the one reported in Deliverable D5.3 [D53]. From the work planned in D5.3, the items explained below have been completed at the time of writing this deliverable - in addition to the usual bug fixing and stability improvements.

3.1.1. Management Agent / RINA implementation

- **Delegation of MA RIB objects managed by the IPC Process**. When the Manager invokes a CDAP operation over an object that belongs to the sub-tree of one of the IPC Processes in the system, the Management Agent forwards the request to the relevant IPCP. Once it gets the one or more responses associated to the request, it aggregates them and sends back to the Manager with the proper object names. This functionality was partially implemented in D5.3, and had not been tested in conjunction with the Manager. Now it is fully implemented and verified working.
- **Exporting of kernel information via sysfs**. RINA components in the kernel now export key information to user space via the `sysfs` file system - information such as send/received/dropped bytes via N-1 flows, length for RMT queues, bytes sent/received via EFCP, number of retransmissions, etc. -. This information is captured by the IPC Process Daemon by inspecting the sysfs files with information relevant to each IPCP, and written/updated into the IPCP RIB. The Management Agent can export this information when the Manager operates over the IPCP RIB objects that contain this information. Notifications between the Management Agent and the Manager are not implemented yet.
- **Better integration of Management Agent with IPCM**. The Management Agent synchronizes with the IPC Manager on IPC Process creation and destruction events, so that the MA RIB is always consistent with the running configuration. Commands to query the RIB of the Management Agent from the IPCM console have been added to facilitate local administration and debugging.

3.1.2. Manager

- **Full specification of RIB schema and RIB object model using GDRO**. Improved the tool-chain (model checker, compiler, code generator, automatic documentation in various formats) to auto-generate Java code and documentation from GDRO RIB object models, as specified in section 2 of this report. Specified all GDRO objects required for the RINA RIB model, enhancing the specification provided as part of D5.2 and fixing its inconsistencies.
- **Improved strategies for IPCP creation and deletion**. The Manager can now instantiate an IPCP Process with a particular DIF configuration, register it to multiple N-1 DIFs and instruct the IPCP to discover multiple neighbours; all using a single CDAP command. Remote deletion of IPC Processes has also been implemented.
- **Full implementation of NMS-DAF enrollment policy**. After establishing and application connection to the Manager, the Manager reads the Management Agent state and stores it in its RIB.
- **Support for new Manager shell commands**. Support for new commands to list the systems being managed and their state.

3.2. Validation

In order to validate and showcase the current functionalities of the PRISTINE Network Management System, a public demo took place in the TNC conference 2016 [tnc16demo]. The demo consisted in the configuration of a small multi-tenant capable Data Centre Network based on RINA via the NMS developed in PRISTINE.

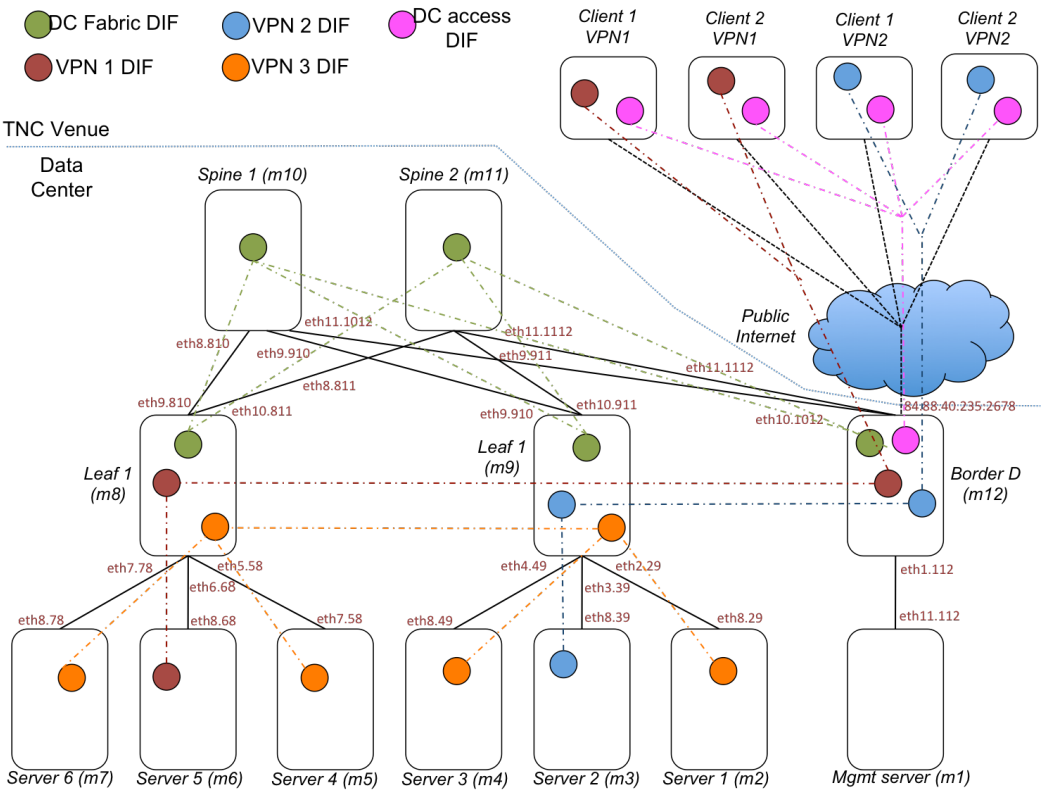

Figure 3. TNC 2016 demo, systems and DIFs (top view)

Figure 3 shows all the systems and DIFs involved in the demo, except for the Management DIF. The DC network consisted in 12 systems: a management server, 5 servers to host user applications, 2 Top-of-rack routers (the leafs), 2 spine routers and the DC border router. The DCN network was organized in a *fabric* DIF that connected all the DC routers and supported several *VPN* DIFs dedicated to different tenants. Some of the *VPN* DIFs spanned to the demo venue over the Internet, allowing systems in the demo venue to join their DIF of choice to get access to the applications hosted in the DC servers (for the demo it was different server instances of the Quake 3 multi-player game). Figure 4 illustrates the vertical organization of the different DIFs in the DC.

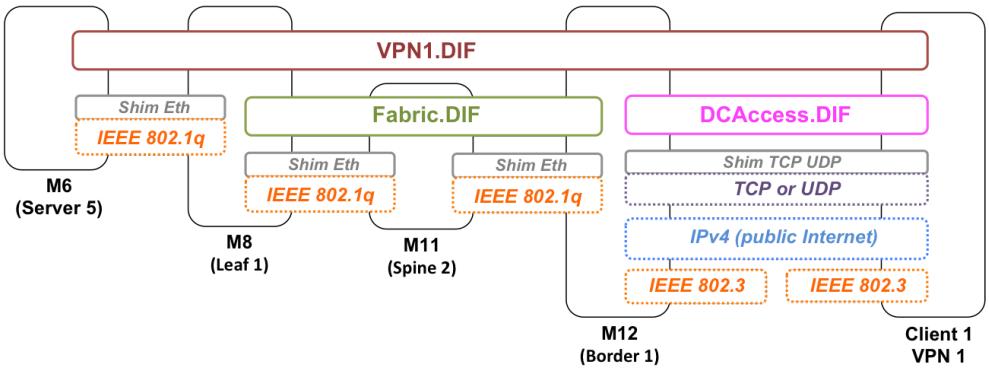

Figure 4. TNC 2016 demo, systems and DIFs (side view)

PRISTINE's NMS, hosted in server 1 (labelled *m1*), was in charge of configuring all the DIFs in all the systems of the DC (*m2* to *m12*). A full tutorial to recreate the demo scenario is available in [tnc16demotut], but this section will summarize the main steps carried out during the configuration of the systems in the DC.

All the DC systems (*m1* to *m12*) had an initial configuration to bootstrap RINA, create all the shim DIFs, create an IPC Process belonging to the "Management DIF" and join this DIF, which is dedicated to support the NMS-DAF. Note that this configuration is not strictly required since the NMS-DAF can use any DIF that has enough scope to let the Manager and the Management Agents communicate; but it is a usual practice to enforce a high degree of isolation between user and DC management traffic. The following code snippet shows the configuration of system *m3*, the configuration of other systems is analogous to this one (and can be obtained in [tnc16demotut]).

```
{
  "configFileVersion" : "1.4.1",
  "localConfiguration" : {
    "installationPath" : "/usr/local/irati/bin",
    "libraryPath" : "/usr/local/irati/lib",
    "logPath" : "/usr/local/irati/var/log",
    "consolePort" : 32766,
    "pluginsPaths" : ["/usr/local/irati/lib/rinad/ipcp", "/lib/
modules/4.1.10/extra"]
  },
  "ipcProcessesToCreate" : [ {
    "type" : "shim-eth-vlan",
    "apName" : "m3.13",
    "apInstance" : "1",
    "difName" : "13"
   }, {
    "type" : "shim-eth-vlan",
    "apName" : "m3.39",
    "apInstance" : "1",
    "difName" : "39"
   }, {
    "type" : "normal-ipc",
    "apName" : "m3.nms",
    "apInstance" : "1",
    "difName" : "NMS.DIF",
    "difsToRegisterAt" : ["13"]
```

```
   } ],
  "difConfigurations" : [ {
    "name" : "13",
    "template" : "shim-eth-vlan.dif"
  }, {
    "name" : "39",
    "template" : "shim-eth-vlan8.dif"
  }, {
    "name" : "NMS.DIF",
    "template" : "nms.dif"
  } ],
  "addons" : {
     "mad" : {
        "managerAppName" : "m3.mad-1--",
        "NMSDIFs" : [{ "DIF": "NMS.DIF" } ],
        "managerConnections" : [ {
           "managerAppName" : "rina.apps.manager-1--",
           "DIF" : "NMS.DIF"
        } ]
     }
  }
}
```

All systems start with the Management Agent trying to allocate a flow to the Manager (whose application name is specified in the configuration file) via the management DIF (*NMS.DIF*). Once the flow is allocated, the MA establishes an application connection with the Manager and enrols to the Management DAF. The DC admin can inspect the systems that have joined the Management DAF via the Manager shell by typing the following command:

```
listMas dialect:dif
dms @ ws-server on ws://localhost:8887: dms: Response from <DMS_MANAGER>

List of connected MAs

3 {m4.mad,1} Connected
4 {m8.mad,1} Connected
1 {m10.mad,1} Connected
2 {m6.mad,1} Connected
11 {m3.mad,1} Connected
10 {m7.mad,1} Connected
9 {m9.mad,1} Connected
7 {m2.mad,1} Connected
```

```
8 {m12.mad,1} Connected
5 {m11.mad,1} Connected
6 {m5.mad,1} Connected
```

Then the Manager is ready to configure the DC. To do so, it needs to instruct the MAs of the relevant systems to instantiate IPC Processes (IPCPs), assign them to relevant DIFs, register them to N-1 DIFs and help the IPCPs discover their neighbours. All this is achievable via the Manager console, which can also run scripts where the commands have already been written into a file. This allows network managers to write the configuration required to create a DIF once, and run it multiple times. The configuration of each DIF is also centrally maintained in the Manager via DIF templates; which specify the syntax of the protocols (EFCP, CDAP), RIB object model and policy configurations for each DIF. The following snippets show the scripts that were used to create the *fabric*, *vpn1*, *vpn2* and *vpn3* DIFs respectively.

Manager script to create the fabric DIF

```
createIPCP dialect:dif, appName:m10.fabric, appInst:1,
 difName:fabric.DIF, difTemplateName:fabric, maName:m10.mad, ipcpAddr:10,
 difN1NameList:810;910;1012
createIPCP dialect:dif, appName:m11.fabric, appInst:1,
 difName:fabric.DIF, difTemplateName:fabric, maName:m11.mad, ipcpAddr:11,
 difN1NameList:811;911;1112
wait 2000
createIPCP dialect:dif, appName:m8.fabric, appInst:1, difName:fabric.DIF,
 difTemplateName:fabric, maName:m8.mad, ipcpAddr:8, difN1NameList:810;811,
 neighbors:m10.fabric/810/fabric.DIF;m11.fabric/811/fabric.DIF
wait 2000
createIPCP dialect:dif, appName:m9.fabric, appInst:1, difName:fabric.DIF,
 difTemplateName:fabric, maName:m9.mad, ipcpAddr:9, difN1NameList:910;911,
 neighbors:m10.fabric/910/fabric.DIF;m11.fabric/911/fabric.DIF
wait 2000
createIPCP dialect:dif, appName:m12.fabric, appInst:1,
 difName:fabric.DIF, difTemplateName:fabric, maName:m12.mad,
 ipcpAddr:12, difN1NameList:1012;1112, neighbors:m10.fabric/1012/
fabric.DIF;m11.fabric/1112/fabric.DIF
```

Manager script to create the VPN1 DIF

```
createIPCP dialect:dif, appName:m6.vpn1, appInst:1, difName:vpn1.DIF,
 difTemplateName:vpn1, maName:m6.mad, ipcpAddr:6, difN1NameList:68
```

```
wait 1000
createIPCP dialect:dif, appName:m8.vpn1, appInst:1,
 difName:vpn1.DIF, difTemplateName:vpn1, maName:m8.mad, ipcpAddr:8,
 difN1NameList:68;fabric.DIF, neighbors:m6.vpn1/68/vpn1.DIF
wait 2000
createIPCP dialect:dif, appName:m12.vpn1, appInst:1,
 difName:vpn1.DIF, difTemplateName:vpn1, maName:m12.mad, ipcpAddr:12,
 difN1NameList:fabric.DIF;overlay.DIF, neighbors:m8.vpn1/fabric.DIF/
vpn1.DIF
```

Manager script to create the VPN2 DIF

```
createIPCP dialect:dif, appName:m3.vpn2, appInst:1, difName:vpn2.DIF,
 difTemplateName:vpn2, maName:m3.mad, ipcpAddr:3, difN1NameList:39
wait 1000
createIPCP dialect:dif, appName:m9.vpn2, appInst:1,
 difName:vpn2.DIF, difTemplateName:vpn2, maName:m9.mad, ipcpAddr:9,
 difN1NameList:39;fabric.DIF, neighbors:m3.vpn2/39/vpn2.DIF
wait 2000
createIPCP dialect:dif, appName:m12.vpn2, appInst:1,
 difName:vpn2.DIF, difTemplateName:vpn2, maName:m12.mad, ipcpAddr:12,
 difN1NameList:fabric.DIF;overlay.DIF, neighbors:m9.vpn2/fabric.DIF/
vpn2.DIF
```

Manager script to create the VPN3 DIF

```
createIPCP dialect:dif, appName:m2.vpn3, appInst:1, difName:vpn3.DIF,
 difTemplateName:vpn3, maName:m2.mad, ipcpAddr:2, difN1NameList:29
createIPCP dialect:dif, appName:m4.vpn3, appInst:1, difName:vpn3.DIF,
 difTemplateName:vpn3, maName:m4.mad, ipcpAddr:4, difN1NameList:49
createIPCP dialect:dif, appName:m5.vpn3, appInst:1, difName:vpn3.DIF,
 difTemplateName:vpn3, maName:m5.mad, ipcpAddr:5, difN1NameList:58
createIPCP dialect:dif, appName:m7.vpn3, appInst:1, difName:vpn3.DIF,
 difTemplateName:vpn3, maName:m7.mad, ipcpAddr:7, difN1NameList:78
wait 1000
createIPCP dialect:dif, appName:m8.vpn3, appInst:1,
 difName:vpn3.DIF, difTemplateName:vpn3, maName:m8.mad, ipcpAddr:8,
 difN1NameList:58;78;fabric.DIF, neighbors:m5.vpn3/58/vpn3.DIF;m7.vpn3/78/
vpn3.DIF
wait 2000
createIPCP dialect:dif, appName:m9.vpn3, appInst:1,
 difName:vpn3.DIF, difTemplateName:vpn3, maName:m9.mad, ipcpAddr:9,
 difN1NameList:29;49;fabric.DIF, neighbors:m2.vpn3/29/vpn3.DIF;m4.vpn3/49/
vpn3.DIF;m8.vpn3/fabric.DIF/vpn3.DIF
```

As it can be seen it doesn't matter which layer the Manager is configuring, it always performs the same kind of actions using the same abstractions (creation of IPCPs, assigning them to DIFs via templates, registering to N-1 DIFs, discover neighbours). As stated in previous sections, a common model for configuration and state management greatly simplifies the task of the Manager, allowing it to perform more sophisticated changes at a lower cost.

4. Configuration Management

4.1. Introduction

Figure 5 illustrates a typical configuration management scenario for RINA networks. Each system has to configure one or more IPC Processes belonging to different DIFs. The system configuration is "owned" by the Management Agent (MA), who is responsible for the life-cycle management of all the IPC Processes in the system (there can be more than one MA in the same system, taking responsibility for subsets of IPC Processes, but they must form a hierarchy with a single master MA at its root). Management Agents collaborate with other peer Management Agents in the same Management Domain, or can be coordinated by specialized processes performing a more centralized manager role. PRISTINE is focused with configurations in which a logically centralized Manager manages all the Management Agents in the same Management Domain. The Manager and Management Agents in the same domain form the Network Management DAF (NMS-DAF). Manager and MAs in the same domain interact via CDAP operations targeting the objects in the RIBs of the Management Agents.

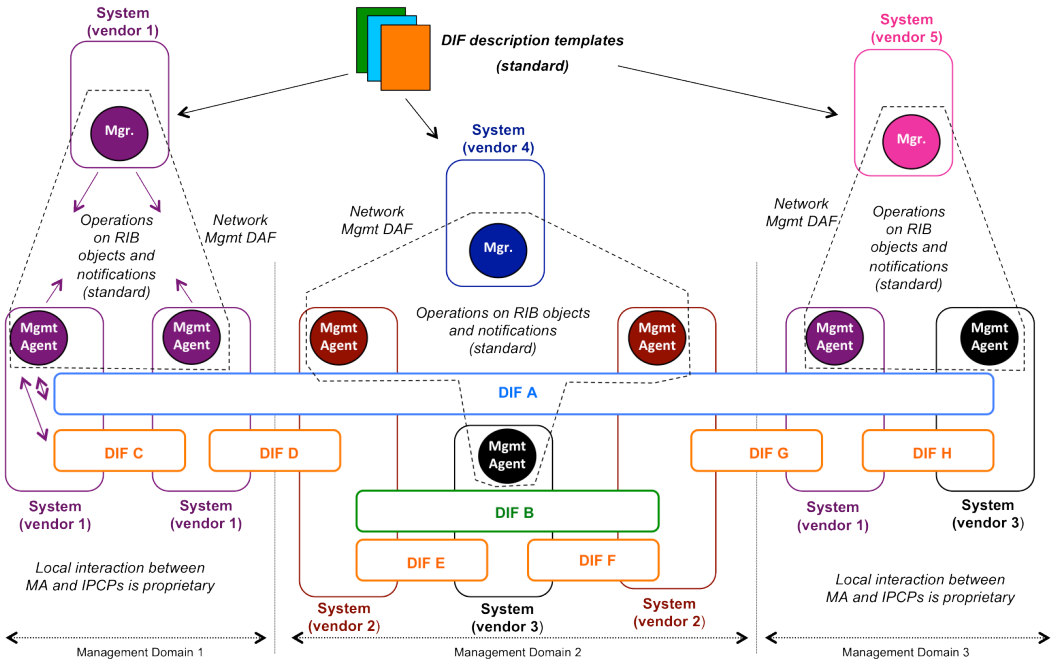

Figure 5. DIF configuration management scenario

One of the key aspects for a practical configuration management solution is to enable a rich market of multiple vendors of RINA systems, while keeping

the solutions of the various vendors interoperable. In order to fulfil this interoperability requirement at the Management System level i) Managers of different vendors must be able to understand the same DIF specification templates and ii) Managers of multiple vendors must be able to successfully interact with Management Agents of multiple vendors.

Since in RINA all DIFs have the same mechanisms configured with different policies, it is possible to create an abstract model for a DIF specification that applies to all types of DIFs. This DIF specification model captures the services that the DIF provides (in terms of QoS cubes and its associated ranges of parameters), the services that the DIF requires from N-1 DIFs to operate (again in terms of ranges of QoS parameters), the DIF static parameters (such as *Maximum SDU Size* or *Maximum PDU Lifetime*), its data transfer syntax (length of the fields in the EFCP header), its data transfer policies (for EFCP, the RMT and SDU Protection), its supported RIB versions and its layer management policies (authentication, enrolment, flow allocation, name-space management, routing, resource allocation or security management).

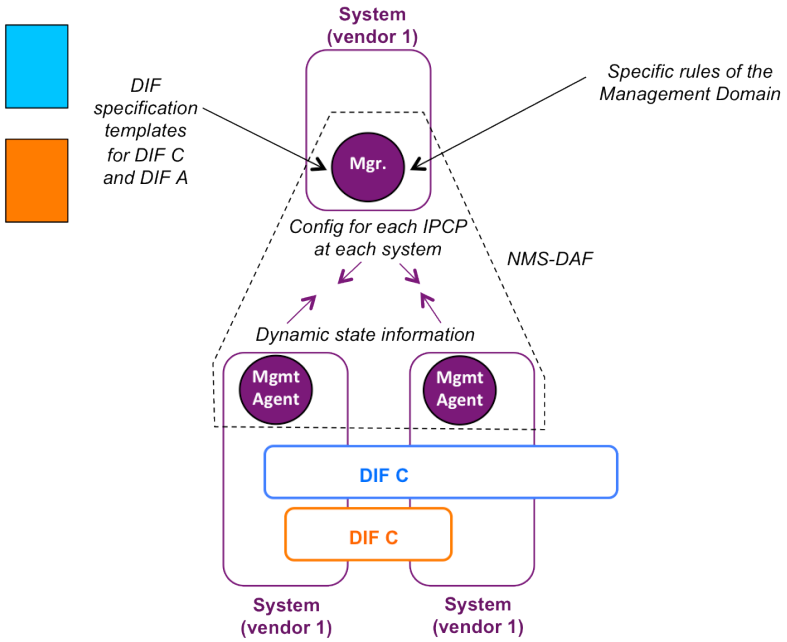

Figure 6. Generation of IPCP configurations from DIF specifications

We expect several forms of proforma from these DIF specification templates. The simple case is a DIF Specification that specifies everything. Others might specify some elements but only put bounds on some policies, allowing flexibility in behaviour within limits, or detailed to be filled in for specific networks. Some might require a minimum set of policies be

supported, etc. Some DIF Specifications may be intended as frameworks to simplify configuring more specialized DIFs. Independently of its degree of completeness, DIF specification templates should be standardised since they constitute a vendor-independent way of modelling the configuration (partial or full) of a DIF. Using the DIF specification templates as input together with other information (such as the role of a specific system in the DIF - border router vs. interior router, etc. -, the IPC Process internal state and specific rules of the Management Domain) Manager processes in the NMS-DAF can generate complete configurations for the different IPC Processes running in the different systems that belong to a certain DIF, as shown in Figure 6.

As it will be seen later in this section, one of the key advantages of RINA for Network configuration management is that the same specification template can be re-used across all layers, whereas in the current Internet one would require a different template for each different layer/technology or even "service" provided by the network (IPv4/IPv6 templates, layer 2 templates for several layer 2 technologies, BGP/MPLS VPN templates, LTE-specific templates, etc). The reminder of this chapter is structured as follows: section 4.2 describes an initial specification for a DIF template and a example of its usage specifying a couple of DIFs; section 4.3 performs a comparative study of the complexity of managing configuration in a RINA and an IP-based data-centre networks.

4.2. DIF Template specification

A common template for specifying DIFs is a key element for enabling interoperable Network Management products in the RINA ecosystem. Figure 7 illustrates the multiple roles that the different stakeholders play in the life-cycle of DIF specifications. The DIF specification template is one of the core RINA standards, defined by the Standards Development Organisation (SDO) in charge of maintaining the core RINA specifications. Tool producers can develop a set of tools to facilitate working with the template such as user interfaces to capture input towards a specific DIF specification or automatic DIF specification validators.

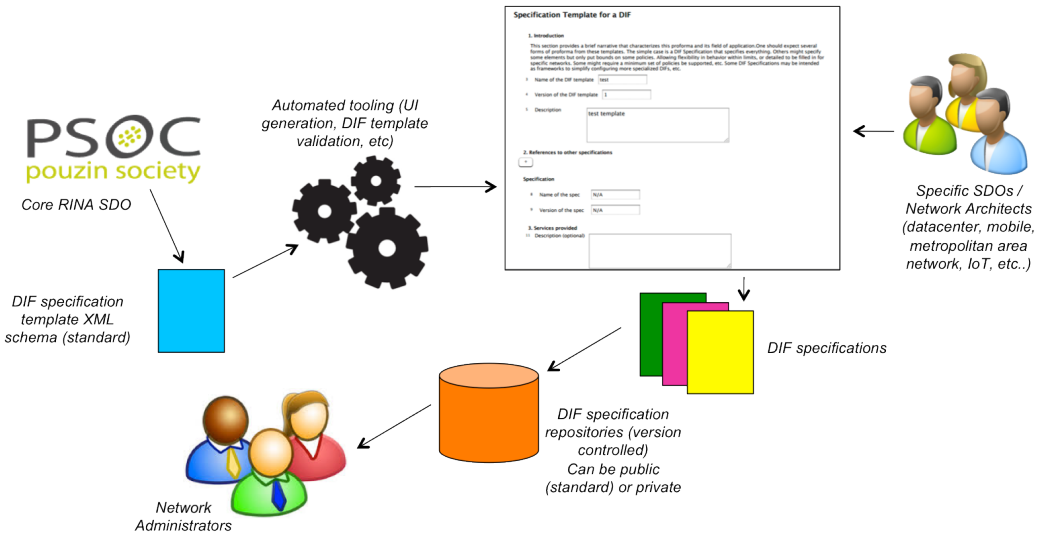

Figure 7. Example DIF specification procedure

Specific SDOs can use the template and associated tools to produce DIF specifications for DIFs belonging to different operating environments such as data-centres, fixed or cellular mobile access, metropolitan area networks, backbone networks, inter-networks supporting different types of applications, etc. Network architects can also use the template to specify private DIFs that need not to be standard. DIF specifications produced by the aforementioned stakeholders would be stored in repositories, so that they could be version controlled and easily searchable. These repositories could be public or private, depending on the nature of the DIF specifications (public standard vs. private DIFs). Finally, architects or administrators of individual networks would search for the DIF specifications better tailored to their network goals, download them and either use them right away (in the case of complete DIF specifications with no policy options) or tailor them to their operational environment (in case the DIF specification offered a range of policy choices for some of the DIF components). As explained in section 3.1, the complete DIF specification is an input to the Network Manager.

PRISTINE has worked on a first experimental proof of this concept (PoC) and chosen a set of tools and languages for it (mainly XML and XML schema, as explained below). This choice of tools and languages may or may not be the ideal one for different real-life deployment scenarios of market segments, since there are a lot of factors other than technical merit influencing product development. Therefore PRISTINE is not necessarily recommending the tools and languages chosen for this PoC as the only way

to realize the concept, rather just as a valid option out of others that may be equivalent (such as YANG or JSON).

The key choice for the PoC is the language in which to describe the DIF specifications, which needs to support a - ideally built-in - mechanism to validate individual DIF specifications against the standard DIF specification template. The choice of the language should facilitate the use of automated tooling to perform model checking and automated User Interface (UI) generation, and should not preclude human inspection and analysis of DIF specification files. XML, the eXtensible Markup Language, provides a good compromise between these three requirements: XML schema allows to specify a model/template to validate individual DIF descriptions, it is a structured text representation format and there are a plethora of available open source tools to auto-generate simple UIs (such as HTML forms) and to validate XML documents against an XML schema.

The full XML schema for the DIF specification template used in this PoC is provided in Annex C, but the following paragraphs describe the main components of its structure. Note that this are all the configurable elements of the DIF. As it will later be seen in the examples section, not all the DIFs need to have all the elements, nor all the policies of a particular element (specially true for simple DIFs like point-to-point wired DIFs).

4.2.1. DIF specification template overview

Introduction

This section provides a brief narrative that characterizes this proforma and its field of application. This template only specifies those parameters and policies that must be known when the DIF is created. Note that in the template below Delimiting Modules, Data Transfer and Data Transfer Control policies may be different for different flows. Similarly, SDU Protection Modules may be different for different (N-1)-ports. This determination is under the control of the `AllocationAE`.

One should expect several forms of proforma from these templates. The simple case is a DIF Specification that specifies everything. Others might specify some elements but only put bounds on some policies. Allowing flexibility in behaviour within limits, or detailed to be filled in for specific networks. Some might require a minimum set of policies be supported,

etc. Some DIF Specifications may be intended as frameworks to simplify configuring more specialized DIFs, etc.

References

This section should list the versions of the specifications this assumes and the other more detailed specifications that are cited below.

Services provided

- **QoS-cubes supported**. This section gives a precise definition of the QoS-cubes supported by this proforma and their QoS-ids.
- **System-specific APIs**. This section specifies one or more definitions of the API for specific system environments that conform to the Service Definition. This should also specify the API flow control policy for this QoS-cube. Primarily an indication of when the user of the flow will be blocked. If there are no APIs in the implementation, this section is N/A.

Services required

- **Ranges of QoS required from the N-1 DIF**. This section lists the ranges of QoS that this DIF requires. Note that while these ranges of QoS form a QoS-cube, they need not be precisely the same as the QoS-cubes provided by the (N-1)-DIF. They would not necessarily be assigned a QoS-cube-id.

Data transfer and data transfer control

- **DIF Parameters**. Static DIF parameters such as maximum PDU size, maximum SDU size, maximum PDU Lifetime, maximum time an ACK is delayed before sending, etc.
- **EFCP Concrete syntax**. The length of the values of fields in the EFCP PCI (addresses, connection-endpoint-ids, qos-id, length, sequence number).
- **Delimiting**. Specify the name and version of the delimiting policy, as well as any parameters required for its correct configuration.
- **QoS-cubes policies**. For each QoS-cube defined in the "service provided" section, define all the EFCP policies associated to it (DTP policies, DTCP Flow control policies if needed and DTCP retransmission control policies if needed). For each policy defined,

provide its name, version and any parameters required for its correct configuration.

- **Relaying and Multiplexing**. Provide the names, versions and required parameters for each of the RMT policies in this DIF: scheduling, maximum queue, queue monitoring and PDU forwarding.
- **SDU Protection policies**. For each SDU protection policy supported by the DIF provide the name, version and configuration parameters.

Layer management

- **CACEP**. CACEP is the common mechanism for creating application connections. Hence, its concrete syntax must be either the default or known a priori, i.e. defined in the specification, because there is no means to negotiate it. Another aspect of the CACEP configuration are the different authentication policies supported in the DIF, with the names, versions and configurable parameters.
- **CDAP**. Specify the concrete syntax to be used in this DIF. This is negotiable by CACEP.
- **RIB definition**. Specify the name, version(s) and configurable parameters of the policy defining the RIB object model for this DIF.
- **RIB Daemon**. Specify the names, versions and configurable parameters of the RIB Daemon policies: update policy, replication policy, subscription policy, logging policy and RIB access control policy.
- **Enrolment**. Specify the names, versions and configurable parameters of the enrolment policies supported in this DIF.
- **Name-space Management**. Specify the names, versions and configurable parameters of the RIB Daemon policies: address assignment/validation policies, directory forwarding policy, directory forwarding table generator policy.
- **Flow Allocator**. Specify the names, versions and configurable parameters of the Flow Allocator policies: Allocate notify policy, allocate retry policy, new flow request policy, sequence roll-over policy, flow monitoring policy, new flow access control policy.
- **Resource Allocator**. Specify the names, versions and configurable parameters of the Resource Allocator policies: PDU Forwarding Table Generator policy, QoS Management policy, per QoS-cube congestion management policy.

- **Routing**. Specify the name, version and configurable parameters of the routing policy for this DIF.
- **Security management**. Specify the names, versions and configurable parameters of the Security Management policies: Credential Management policy, auditing policy.

4.2.2. DIF configuration for large-scale data centers (DCs)

Recently two well-known web companies published some details about their large-scale data centres (Facebook [facebookdc], Google [googledc]). These DCs follow the current trend of building a large scalable DC fabric using a multi-stage Clos topology, in order to achieve high bi-sectional bandwidth between servers and scale up to a large numbers of them, while providing multiple paths for redundancy. These DC fabrics can be Layer 2, Layer 3 or a combination of both as described in [dcfabric]. Network virtualization technologies can be deployed on top of the DC fabric in order to create isolated "slices" dedicated to individual customers (usually referred to as "tenants").

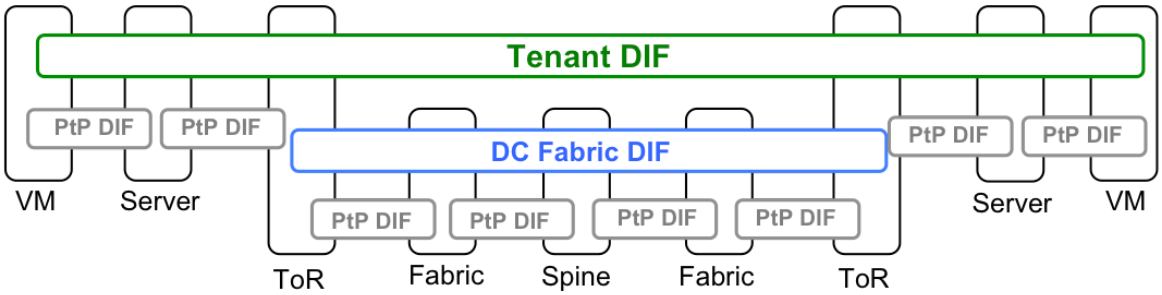

Figure 8. Multi-stage close fabric DC design with RINA (single DC)

The following Figures illustrate the design of a Facebook-style DC as described in [facebookdc] using RINA. Figure 8 depicts the different types of systems in a single DC (VMs, servers, Top-of-Rack switches, fabric switches and spine switches), the different DIFs and their scope. The RINA-enabled DCN network is partitioned into three main types of DIFs: i) several point-to-point DIFs, for the physical links between systems; ii) a single DC-Fabric layer, acting as a large distributed switch; and iii) multiple tenant DIFs, isolated and customized as per the requirements of the different tenants. Figure 9 shows a more complex configuration involving two DCs. Each DC has its own DC fabric DIF, but both DCs share a top-level DIF that connects together all the servers in both DCs under the same pool (enabling for seamless VM mobility across DCs, for example).

This configuration requires a new type of system - the Edge switch - which act as the DC border routers.

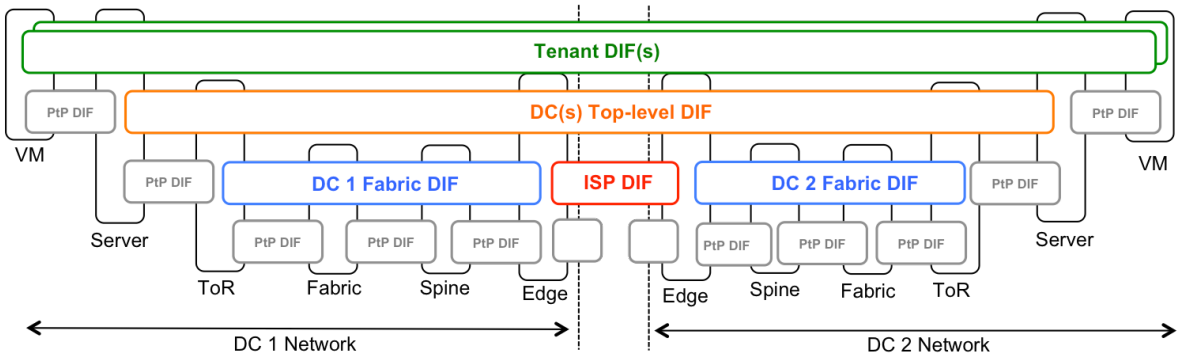

Figure 9. Multi-stage close fabric DC design with RINA (multiple DCs)

Configuration template for Point to Point DIFs

Annex C provides an example of a DIF specification for a point-to-point DIF over a wired media (as it could be short reach fiber optics or cat-6 twisted pair), similar to the characteristics provided by the 10 Gigabit Ethernet (10GbE) standard. While the DIF specification is provided for illustrative reasons, it is expected that DIFs on top of physical media would be fully implemented in hardware and require minimal to no configuration (the router/switch would identify the NIC cards as belonging to that particular type of DIF, similar to how current NICs are classified as 10 GbE, 1 GBE, etc.).

This DIF could be used for point to point links between ToR, fabric and spine switches. Like 10GbE the DIF provides a maximum capacity of 10 Gbps, and a maximum SDU size of 9000 bytes. However it brings additional capabilities inherited from exploiting the common DIF structure:

- **It can support multiple QoS classes** (cubes). In the example two classes are defined: low loss - low latency and best effort. The service definitions for both classes provide explicit bounds on the maximum delay and SDU loss probability allowed.
- **It can support multiple flows**. Unlike traditional Ethernet, which just supports different protocols (a single instance of them) on the same Ethernet link via the Ether-type field, this DIF supports up to 256 parallel flows (the limitations is due to the choice of 1 byte CEP-ids, if more flows where needed a 2-byte CEP-id field could have been defined, but 256 seems large enough for the DIF environment).

- Since it is a point to point DIF, **there is no need for addresses** (there is just one possible source for packets arriving at any of both destinations, and vice versa).

CRC	QOS	DST CEPID	SRC CEPID	PDU Type	FLAGS	PAYLD
4-bytes	1-bytes	1-bytes	1-bytes	1-byte	1-byte	n-bytes

Figure 10. PCI of the PDUs in the Point to Point Wired DIF

The resulting header structure for this DIF is represented in Figure 10.

Configuration template for DC Fabric DIFs

Annex C provides a example specification for the DCN Fabric DIF, which is explained in the following paragraphs. The DIF, whose goal is to support different tenant DIFs, provides 4 QoS cubes: i) loss and delay sensitive; ii) delay sensitive; iii) loss sensitive and iv) best effort.

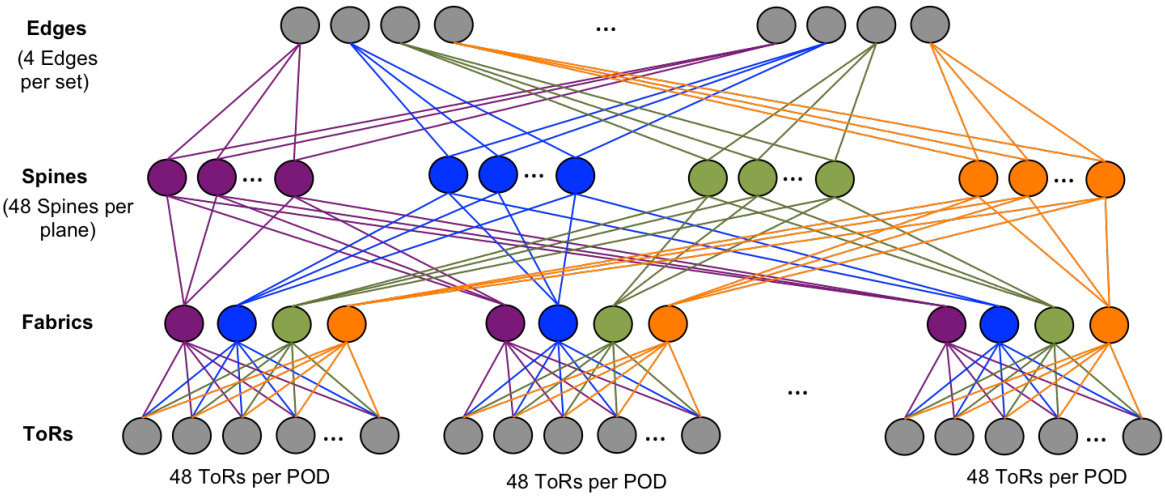

Figure 11. Connectivity graph of the IPCPs in the DC Fabric DIF

Data transfer

The EFCP syntax of the DCN fabric DIF is the following one:

- **Address length**: 3 bytes, which is enough for encoding topological addresses of more than 1M nodes, following the scheme explained in D3.3 chapter 3 [D33].
- **CEP-id length**: 2 bytes, which is enough to support 65.000 concurrent EFCP connections between the same pair of nodes.

- **QoS-id length**: 1 byte, enough to encode the 4 QoS classes provided by the DIF.
- **Sequence number length**: 4 bytes.

EFCP flow control policies use a sliding window approach, with the window size adjusting according to the DIF's congestion avoidance policies, with the RMT providing ECN marks if its queues go beyond a certain threshold. The RMT uses an ECMP-style hash-based forwarding policy to load-balance the different flows provided by the DIF over the multiple N-1 flows provided by diverse N-1 point to point DIFs (using the highly redundant connectivity in the DC environment). Scheduling policies are based on the QTAMux system as described in D3.3 chapter 2 [D33], allowing the DIF to treat the different flows according to the four different levels of service listed before.

Layer management

In addition to default policies and Google Protocol Buffers encoding for CDAP, the most important specific layer management policies used in the DCN fabric DIF are described in the following bullet points:

- *Centralized address management*. IPCPs in the DCN fabric DIF obtain their addresses from a few specialized IPCPs that maintain a fully replicated database with the DIF's address assignment (mapping IPCP process names to addresses). This configuration makes sense in a DC environment since all systems are co-located and under the management of the same organization.
- *Centralized application directory*. A few selected systems in the DCN fabric DIF implement a fully replicated directory that keeps track of applications registered in the DIF (mapping their application names to the addresses of the IPCPs where they are registered). All other IPCPs in this DIF forward flow allocation requests to one of those IPCPS.
- *Flow request policy*. Flows are mapped to one of the four QoS cubes available in the DIF depending on their delay/loss requirements.
- *Links state only errors routing policy*. Routing is link-state but only N-1 flow failures are disseminated, since default forwarding rules can be generated from the topological address encoding as explained in D3.3 chapter 3 [D33].

4.3. Analysis of the structural complexity of the configuration of large-scale Data Centre Networks

This section compares the structural complexity of an IP-based and a RINA-based multi-tenant data centre (DC) network, and how this differences in complexity impact network configuration management.

4.3.1. Multi-tenant DataCentre Network

Modern web-scale Data-centre Networks (DCNs) usually are designed with a multi-stage Clos topology to provide high cross-bisectional bandwidth and high levels of redundancy using a large number of moderate cost networking devices. The Figure below provides an illustration of the physical layout of the modular Facebook DCN [facebookdc], which features a modular design to allow DCs to scale as their computing, storage and networking requirements evolve. The basic unit of modularity is the POD, which is a bunch of computing racks interconnected by a single-stage Clos network. Each rack has `Z` servers, connected to a ToR (Top of the Rack switches). Each POD has `M` racks (and therefore `M` ToRs), which are interconnected via 4 Fabric switches. PODs are connected to each other via a number of Spine switches organized in 4 different planes: each Spine plane has `Y` Spine switches. Traffic that doesn't leave the DC (East-West traffic) is not processed by any additional network devices; but for North-South traffic (in/out the DC), the DC has a number of Edge switches (organized into groups, 4 switches per group) that are also connected to the switch planes. The whole design can be characterized by the parameters shown in Table 2.

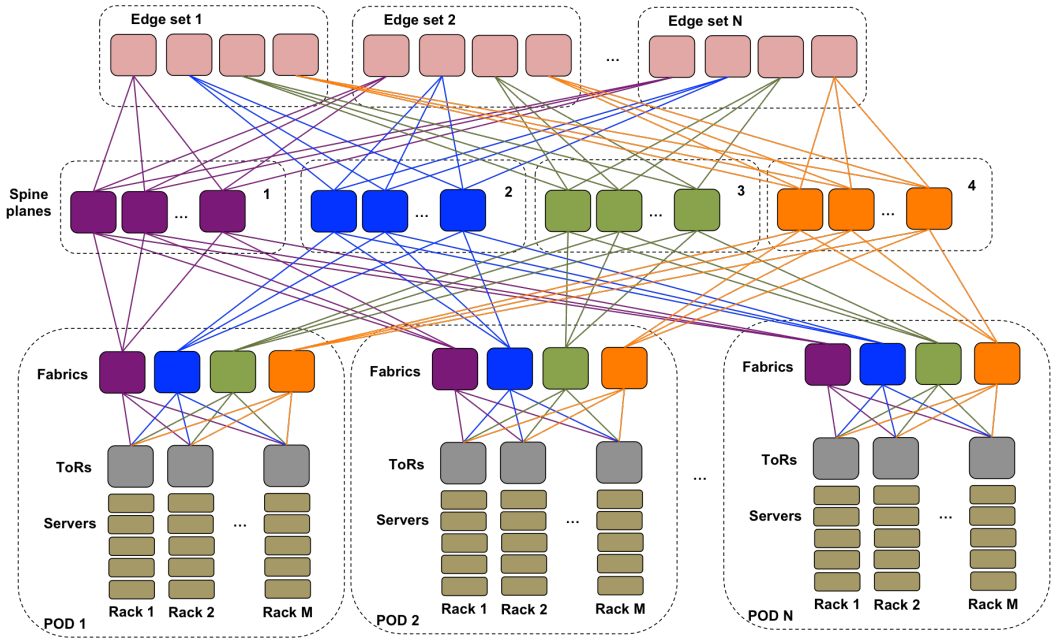

Figure 12. Physical layout of the modular Facebook DataCentre Network (DCN)

Table 2. Parameters defining the Facebook DCN and its max value

Parameter name	Max. value (for a full DC)
# Servers per rack (Z)	48
# Racks per POD (M)	48
# Spine planes (X)	4
# Spines per plane (Y)	48
# Edge sets (U)	4
# PODs	95

The following subsections analyse how a DC such as the one described above can provide its customers a service consisting in a set of Virtual Machines interconnected by isolated networks, independent of those from other tenants. DCs that are capable of providing this service are usually referred to as supporting “multi-tenancy” (where each tenant is allocated a subset of the DC computing, storage and networking resources).

IP-based DCN

There are multiple ways to design the network of a DC based on a multi-stage Clos Fabric, such as the one described by Facebook. Layer 2 (L2), Layer 3 (L3) or a combination of both approaches could be used in different parts of the DCN fabric, depending on the types of applications to be deployed in the DC and other requirements. For example, L2 technologies such as Q-in-Q or MAC-in-MAC could be used between ToR and Fabric

switches, with L3 technologies being used between Fabrics and Spines. In this section we focus on an "all L3 DCN fabric" approach for two reasons: i) to minimize the number of protocols and technologies in the DCN – since we will perform a "best-case" comparison with RINA; and ii) since this is also the approach followed by Facebook. In order to implement the multi-tenant overlays on top of the DCN fabric we will use Ethernet VPN (EVPN) [evpn] technologies - a VXLAN data plane with a BGP control plane for MAC and host IP address learning-, since EVPN enables scalable and flexible L2 and L3 overlay networks over an L3 DCN fabric in a protocol-efficient way. Finally, we assume that the DC Network Management system can manage all the devices and protocols in the DCN via NETCONF and YANG [netconfyang].

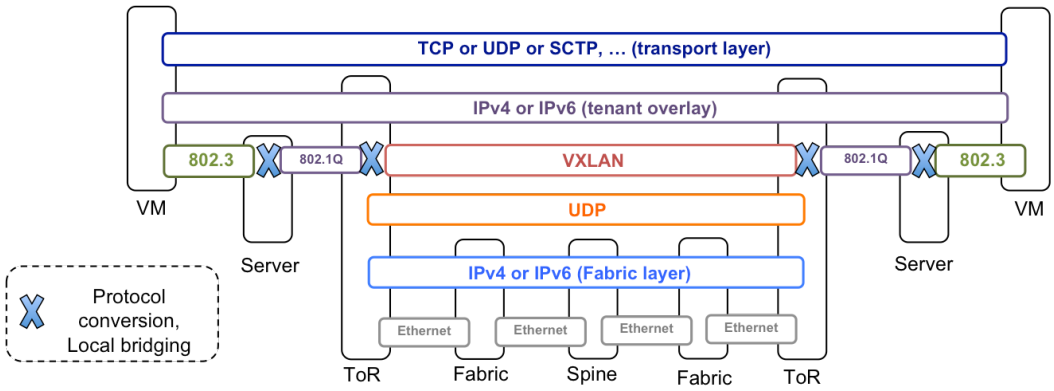

Figure 13. Data plane for VM to VM communication

Figure 13 shows the data plane of the IP-based solution, for VM to VM communication (VMs can communicate to external machines in the public Internet or a customer's internal network via a Gateway, but this document only focuses on the analysis of inter-VM communication within the DC). VMs have one or more virtual Ethernet interfaces, which are connected by internal procedures to one of the Server's software Ethernet bridges. There the server typically tags the traffic generated by different VMs with a different VLAN id, and sent to an upstream Top of Rack switch (ToR). The ToR encapsulates the tagged Ethernet frame into a VXLAN frame (VLAN to VXLAN mappings have been populated by the eVPN control plane), and sends the resulting packet to the destination IP address in the VXLAN frame through the DC fabric. Equal Cost Multi-Path (ECMP) is heavily used in the data plane to leverage the large number of paths between each pair of nodes (either for load balancing or resiliency purposes). The fabric control plane – based on eBGP – keeps the ECMP groups up to date, adding or removing members as Ethernet links go up and down.

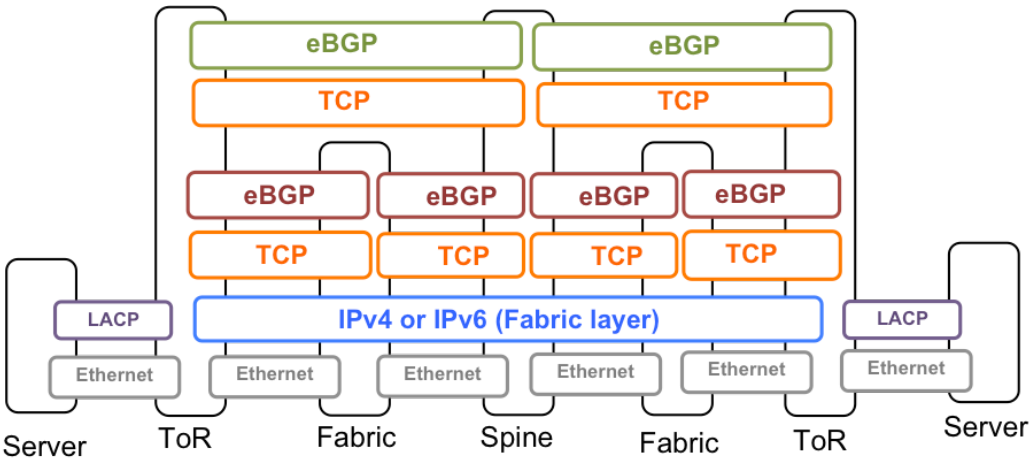

Figure 14. Control plane for VM to VM communication

The control plane of the IP-based DC-solution is shown in Figure 14. As opposed to the data-plane structure, in which the overlay layers are multiplexed over the fabric layers, the control-plane features an essentially flat structure and is overlaid on top of the fabric IP layer (this is a key difference with the RINA structure). eBGP is deployed as the only routing protocol of the data centre fabric, following the design in [lapukhov], which divides the different DCN fabric switches into private Autonomous Systems (AS). All spine switches are grouped together as a single AS, all the fabric switches of the same group are another AS and finally each ToR with all the servers connected to it form another AS. Expressing the number of ASs as a function of the number of PODs in the DC (N) we obtain the following expression: N*(racks per POD + 1) + 1. Therefore the maximum number of ASs in a full DC – taking the numbers of Table 2 – is of 4656. As explained in [lapukhov], this number is larger than the number of private AS numbers (considering 2-byte AS numbers), therefore special BGP configurations to allow for re-using private AS numbers have to be employed. BGP is also the key protocol in the eVPN overlay control plane, in which ToRs exchange eVPN routes amongst them (using BGP multi-protocol extensions and a dedicated address family). In order to avoid setting up a full mesh of BGP sessions between ToRs, some spine switches (or alternatively dedicated servers) can be configured as BGP route reflectors. Finally, in order to increase the availability of the servers, ToRs can be designed as two separate chassis that are connected to each server with redundant connections. In this configuration, the Link Aggregation Control Protocol (LACP) is required to provide the servers with transparent multi-homing over the separate physical Ethernet links to each ToR chassis.

RINA-based DCN

Figure 15 shows a conceptual diagram of the layers in the RINA-based Data Centre Network solution. As opposed to the IP-based design, in RINA both the data transfer (data plane) and layer management (control plane) functions of a layer are part of the same layer (DIF in the RINA terminology). The RINA-design that is equivalent to the IP-based designed features three different types of layers: i) a DC-Fabric DIF, which brings together all the ToR, Fabric and Spine switches as a large distributed switch; ii) multiple tenant DIFs, allowing tenants to connect dedicated computing resources (VMs) via performance-isolated and secure networking and iii) multiple Point-to-Point (PtP) DIFs, providing connectivity over individual physical links.

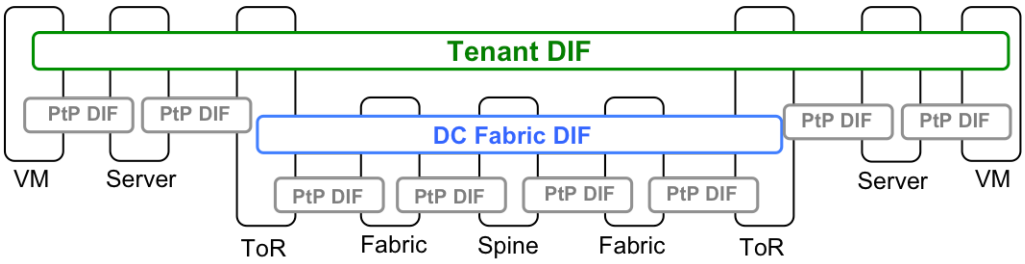

Figure 15. Layers in the RINA-based DCN

The Tenant DIFs is expanded to show a simplified view of the protocol processing performed by each IPC Process in that layer (the protocol structure of the other layers is identical, only specific policies change). EFCP is the single data transfer/data transfer control protocol, used to provide end-to-end flows to applications using the DIF (where the ends are defined by the scope of the layer). IPCPs that are at the endpoints of the flow encapsulate and decapsulate SDUs, optionally provide flow and/or retransmission control and forward the resulting PDUs to the next hop towards the destination IPCP. Intermediate IPCPs relay the PDUs belonging to different flows according to a forwarding policy maintained by the layer management functions (routing, flow allocation, resource allocation, security management, name-space management).

In parallel, all layer management functions exchange information (encoded as objects) with its neighbours via the CDAP protocol. CDAP allows layer management functions to perform 6 operations (create, delete, read, write, start, stop) targeting objects of its neighbour IPCPs. Therefore what changes from layer management function to layer management

function are the objects and operations carried by CDAP, not the protocol. This is a big simplification in terms of network management and operation compared to IP-based designs, even more taking into account that this model is consistently followed by all layers.

In terms of specific policies for each layer, we will briefly describe the ones used for routing and forwarding in the DC Fabric DIF; policies for each tenant DIF could be different and customized to the tenant DIF goals. Taking into account that the DC has a highly regular connectivity graph, the use of topological addressing would minimize the need for exchanging routing information and the amount of entries in the forwarding tables. Each IPCP would know how to forward PDUs to all destination address by just inspecting the destination address and comparing it to the addresses of all direct neighbour IPCPs. Only failed links would need to be disseminated via routing updates; upon learning about a failed link the routing function would compute an exception to the default forwarding rules and add an entry to the IPCP forwarding table. Since it is the case that usually multiple paths to the destination IPCP will exist, the forwarding policy would include a ECMP-style logic in order to load-balance PDUs of different flows over the different paths.

4.3.2. Configuration of the DC Fabric

This section compares the complexity in configuring the fabrics of the RINA and IP-based solutions. Configuring the fabric involves configuring the ToR, Fabric, Spine and Edge devices. For simplicity we will assume that the DC Fabric provides no support for traffic differentiation and therefore a simple FIFO scheduling policy would suffice.

IP-based solution

We will assume that when a device bootstraps all Ethernet interfaces are enabled by default. All devices in the IP-based DC fabric require a similar configuration, consisting in getting IP addresses for all the Ethernet interfaces (MAC addresses are already assigned to the physical interface by the interface vendor), configuring BGP sessions and ECMP groups. In order to configure BGP, at least the following information is needed: AS number, router-id (IP address), configuration of routing policies, maximum number of entries per ECMP group and for each session to

each directly connected neighbour an IP address and its AS number. The following table summarizes the main entities managed in each layer.

Table 3. Main managed entities for Point-to-Point Ethernet Links

Interfaces	Ethernet interfaces, need unique MAC address (one per interface)
Data transfer protocol syntax	IEEE 802.3 (Ethernet)
Management protocol	NETCONF
Management models	yang-common-types [yangcommon], yang-interfaces [yangif]

Table 4. Main managed entities for DC-fabric IPv4 layer

Interfaces	IPv4 interfaces, need IP address (one per interface), unique in the layer.
Data transfer protocol syntax	IPv4 syntax, TCP syntax (TCP is used by the control plane)
Forwarding entity	router, one per device in the layer, has FIB entries (forwarding table)
Forwarding strategy	Longest prefix matching, ECMP
Scheduling strategy	FIFO (needs maximum-queue size) – assuming no traffic differentiation in the fabric
Routing protocol	BGP with different routing policies. Needs AS numbers, router-id (IP address), neighbours' IP addresses and AS numbers. Maintains RIB.
Management protocol	NETCONF
Management models	yang-common-types [yangcommon], yang-interfaces [yangif], yang-ip [yangip], yang-routing [yangroute], yang-bgp [yangbgp]

RINA-based solution

Similar to the IP-based case, we assume that the IPC Processes (IPCPs) belonging to the point-to-point DIFs are set-up when the device (ToR, Fabric, Spine or Edge) bootstraps. Then the Management System would create a single IPCP per device belonging to the DC-Fabric DIF, configure them with an address and the policies described in Table 6. After that the Management System would instruct each IPC Process to enrol with all directly connected neighbours (so that neighbour IPCPs are able to exchange layer management information). Note that the Manager could just configure a few IPCPs and let all the other IPCPs in the DC layer automatically obtain their configuration by enrolling to those already-configured IPCPs.

Table 5. Main managed entities for Point-to-Point DIFs

Interfaces	Physical wire driver
Data transfer protocol syntax	EFCP (length of fields in PCI optimized for physical media). No addresses.
Management protocol	CDAP
Management models	dif-common-mom Appendix B

Table 6. Main managed entities for the DC-fabric DIF

Interfaces	Port-ids to N-1 flows, just need port-id (locally –device- unique identifier)
Data transfer protocol syntax	EFCP (length of fields in the PCI optimized for the layer). Need address (one per device in the layer), unique in the layer
Forwarding entity	Relaying and Multiplexing Task (RMT), one per device in the layer, has forwarding table entries.
Forwarding strategy	Longest prefix matching, ECMP
Scheduling strategy	FIFO (needs max-queue size) – assuming no traffic differentiation in the fabric
Routing protocol	CDAP with link-state routing policy and topological addressing. Maintains RIB.
Directory protocol	CDAP with centralized directory policy. Maintains Directory Forwarding Table.
Management protocol	CDAP
Management models	dif-common-mom Appendix B

EFCP (data transfer) and CDAP (layer management, network management) are the only protocols required in every DIF. The fields in the EFCP PCI (source/destination addresses, QoS-id, source/destination CEP-ids, sequence number, length) are the same across layers, with only its length changing from layer to layer. All layer management functions (such as routing or the directory) are CDAP policies that manipulate different objects via the CDAP protocol actions (create, delete, read, write, start, stop).

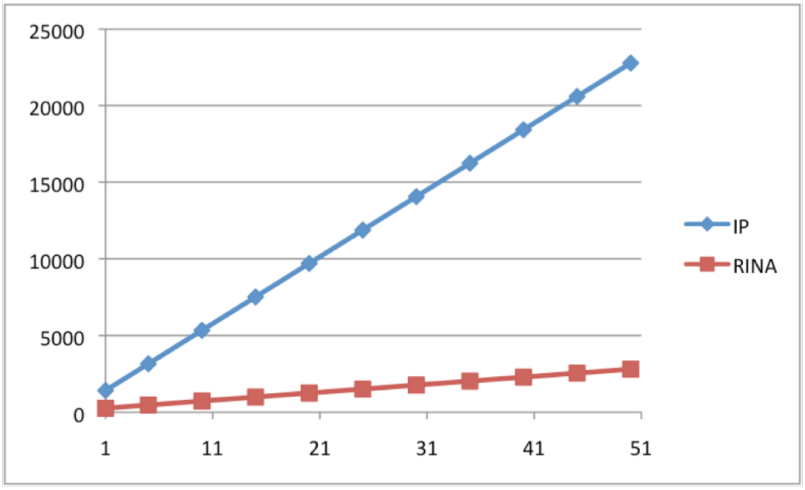

Figure 16. # of addresses in the DCN fabric as a function of the # of PODs in the DC

The complete naming and addressing architecture of RINA [day16] also contributes to reducing the management overhead by minimizing the number of addresses that needs to be configured in each layer. The Figure above shows the number of addresses needed in the DCN Fabric as a function of the number of PODs. In the IP-based solution each interface needs two addresses (MAC and IPv4), therefore the number of addresses is a function of the number of links plus the number of devices (since BGP needs a router-id per BGP-speaking device). In the RINA-based solution Point-to-Point DIFs don't need addresses (data can only have a possible source and a possible destination) and the DC-Fabric DIF just needs one address per device in the DIF (i.e. per IPCP). The simple formulas below express the number of addresses required for both solutions:

- **IP, # of addresses**: 4*Links in DCN fabric + Nodes in the DCN Fabric (IP) = 436N + 976
- **RINA, # of addresses**: Nodes in the DCN Fabric = 52N + 208

4.3.3. Configuration of tenant overlays

Tenant overlays have to bring together a number of VMs under a performance and security-isolated connectivity domain, crating the illusion that each tenant has its own dedicated "slice" of data centre resources. Configuring a tenant overlay requires updating the state of ToR and server devices, as well as instantiating VMs hosted in the servers. The size of a tenant overlay may go from small to moderate, but in any case is expected to be at least one or two orders of magnitude smaller than the size

of the DC Fabric layer (considering a large-scale data centre with a capacity of 100k servers as shown in Table 2).

IP-based

In order to create a tenant overlay three different types of systems have to be configured: Virtual Machines, servers and ToRs. The configuration of Virtual Machines is simple: based on assigning IP addresses to the VM's virtual Ethernet interfaces. Servers must segment traffic coming from different VMs per tenant and multiplex it and send it to one of the two ToRs is connected two via an Ethernet link. The initial configuration of the server requires the creation of a Link Aggregation Group (LAG) with the two Ethernet interfaces that are connected to the uplink ToRs. Then, every time VMs belonging to a different tenant overlay are instantiated a virtual Ethernet bridge is created. The VM's virtual Ethernet interfaces are attached to this bridge. The logical Ethernet interface representing the LAG group is also "partitioned" into multiple VLANs, one per tenant DIF. Each VLAN interface is attached to the virtual bridge belonging to the tenant network.

When creating a tenant overlay a number of VMs will be instantiated on a number of servers, which will extend the layer 2 connectivity of the VMs to a set of ToRs as explained in the previous paragraph. In order to complete the tenant overlay, these set of ToRs must be connected together under the same private, L2 domain. L2 Ethernet VPNs over the DC fabric is the proposed way to implement this functionality for the IP-based solution discussed in this paper. In order to do so, a full mesh of VXLAN tunnels is created between ToRs belonging to the same tenant overlay. VXLAN tunnels transport Ethernet traffic over UDP and the IP layer of the DCN fabric.

For each VXLAN tunnel the Management System has to instantiate two Virtual Tunnel Endpoints (VTEPs), one at each end of the tunnel. Each VTEP needs to be bound to a local IP address and UDP port number, and associated to the IP address and UDP port number of the remote endpoint. Finally an Ethernet VRF (E-VRF) instance is created in each ToR, in order to connect together VTEPs and 802.1q interfaces belonging to the same tenant overlay. The E-VRF instance is like an Ethernet bridge, with the exception that it doesn't use data-plane learning techniques to populate its MAC forwarding table: the table is populated via routes learned by BGP.

Therefore, a full mesh of BGP instances has to be configured between all ToRs participating in the provisioning of E-VPN services - or alternatively a number of spine switches can be set-up as route reflectors to increase the scalability of the control plane,

Table 7. Main managed entities at the IP-based tenant overlay layer

Interfaces	Ethernet interfaces: need MAC address (one per interface). 802.1q interfaces: need VLAN-id. VTEP interfaces: need VXLAN-id, local IP address and UDP port, remote IP address and UDP port. IPv4 interfaces: need IP address (one per interface), unique in tenant overlay
Data transfer protocol syntax	IEEE 802.3 (Ethernet), IEEE 802.1q, IPv4, UDP, VXLAN, TCP
Forwarding entity	Router: one per VM. Ethernet bridge: one per server per tenant overlay. E-VRF: one per ToR per tenant overlay
Forwarding strategy	Exact address matching
Scheduling strategy	FIFO (needs max-queue size) – assuming no traffic differentiation in the fabric
Routing protocol	BGP with multi-protocol extensions. Needs route distinguishing and VPN targets
Directory protocol	DNS (resolve domain names of apps executing in the tenant DIF to IP @s)
Redundancy protocol	Link Aggregation Control Protocol – needs local Ethernet interface addresses
Management protocol	NETCONF
Management models	yang-common-types [yangcommon], yang-interfaces [yangif], yang-ip [yangip], yang-routing [yangroute], yang-bgp [yangbgp], yang-bridging [yangbridge], yang-vxlan [yangvxlan], yang-evpn [yangevpn], yang-lacp [yanglacp]

RINA-based

Unlike the IP-based solution for tenant overlays described in the previous section, the configuration of VMs, servers and ToRs when a new tenant overlay DIF is instantiated is quite similar [vrijders16]. In all the systems belonging to the tenant DIF the Management system has to instantiate a single IPCP belonging to this DIF, and configure the data transfer and layer management policies as in the DC-fabric DIF case. Supporting N-1 flows between IPCPs will be established over Point to Point DIFs (between servers and ToRs) or over the DC-Fabric DIF (between ToRs), but this is transparent to the IPCPs in the tenant overlay DIF: all the DIFs provide

the same service API to its users, regardless of its internal policies or implementation.

Table 8. Main managed entities for the DC-fabric DIF

Interfaces	Port-ids to N-1 flows, just need port-id (locally –device- unique identifier)
Data transfer protocol syntax	EFCP (length of fields in the PCI optimized for the layer). Need address (one per device in the layer), unique in the layer
Forwarding entity	Relaying and Multiplexing Task (RMT), one per device in the layer, has forwarding table entries.
Forwarding strategy	Longest prefix matching, ECMP (load-balancing/redundancy at server level)
Scheduling strategy	FIFO (needs max-queue size) – assuming no traffic differentiation in the fabric
Routing protocol	CDAP with link-state routing policy. Maintains RIB.
Directory protocol	CDAP with distributed directory policy. Maintains Directory Forwarding Table.
Management protocol	CDAP
Management models	dif-common-mom Appendix B

Table 8 summarizes the main information that the Manager needs to configure in each IPCP at the different devices (VMs, servers, ToRs). As with the DC-Fabric DIF, EFCP and CDAP are the only protocols used, while RMT is the only type of forwarding entity. The only differences with the DC-fabric DIF are in the policies used for routing and the distributed directory. Since tenant overlay DIFs are significantly smaller than the DC-fabric one, a normal link-state routing policy with flat identifiers would be enough. Since the structure of the graph of tenant DIFs will be also quite regular, topological addresses could also be used to allow routing to scale better. The small/medium size of these types of DIFs suggests that a directory policy following a distributed approach (similar to the dissemination of routing advertisements in link-state strategies) would be adequate.

The instantiation of tenant DIFs in the RINA-based use case is simpler than in the IP-based use case. In RINA only one type of forwarding entity exists (the RMT), compared to the three different forwarding entities used in the IP case (IP routers, Ethernet bridges and Ethernet VRFs). While in RINA there is a single data transfer protocol – EFCP – the IP-based design uses Ethernet with VLANs (IEEE 802.1Q), IPv4, VXLAN, UDP and TCP. The IP-based solution is also more complex in terms of interface

types (physical Ethernet, logical Ethernet, Virtual Tunnel Endpoint, IP) compared to the RINA case (just N-1 port-ids). Even if its design has been chosen to minimize the number of control plane protocols, the IP-based solution still features more control plane protocols (BGP, DNS, LACP) than the RINA case, in which only CDAP with routing and directory policies is used. This situation is reflected in the management models of the IP-based solution, which grow in complexity as new requirements are introduced in the design. This fact is due to the lack of commonality and invariants in the IP protocol suite, in which the approach of dealing with different operational environments or new requirements is usually to design new protocols from scratch. In contrast, RINA networks leverage a common structure that captures the mechanisms that are invariant with respect to the requirements of each networking use case, and has built-in hooks for deploying optimized policies into its single data transfer (EFCP) and layer management (CDAP) protocols.

4.3.4. Conclusions

This section has described some of the advantages of managing a RINA network consisting in multiple layers compared to managing an equivalent IP-based network in the context of a large-scale multi-tenant data centre network. The commonality offered by the RINA layers together with a consistent QoS and security models in all layers from the application to the wire allows RINA to minimize the number of management models required to model all layers and protocols, therefore simplifying a lot the design of Network Managers. Since Managers can reason about simpler models, their behaviour will be able to become more sophisticated and reliable, increasing the degree of network automation.

These benefits are due to RINA's effort in separating mechanism from policy; that is, extracting invariants through all layers of the communication stack. Gains in simplicity compared to all-IP networks will be higher for larger networks with more diversity, like service provider networks. These networks typically feature different segments (access, aggregation, core, interconnection), featuring different underlying data plane and control plane protocols. In contrast, RINA maintains its generic layer with two protocols, only policies change from layer to layer.

5. Performance management

5.1. Manager Inference

Performance management concerns itself with the identification of sub-optimal configurations and behaviour within the RINA network. Specifically, to investigate appropriate complex event processing techniques, develop and evaluate them within the scenarios identified by PRISTINE. The work here builds on the work done in Configuration management and the general information modelling work done in the RIB. Specifically, the work presented aims to show how for a single scenario :

- The problem can be identified from the Notification reports sent from the Management Agent (MA)
- The Notifications can be correlated to a DIF and analysed using some advanced machine learning techniques.
- Analyse application impact of the correlated notification reports, for example, analysis on the actual usage of network resources as opposed to the declared expected usage.
- Investigate ways automated policy responses can be included in the declarative configuration specification. Here, strategies are employed to inspect, prioritise and ultimately decide if corrective actions are desirable.
- Optimise the system based on a set of high-level strategy goals. This allows performance tuning, where the allocated resources are optimised to the aggregate needs, increasing available resources to over-loaded DIF's and reducing resources to under-utilised ones.

Self-adaptation principles can be applied to network management tasks to optimise the Quality of Service (QoS) of network services deployed over a recursive network architecture under limited network resources availability. This section outlines a novel self-adaptive network management approach capable of learning the levels of importance associated with traffic flows. The aim is to prioritise the most requested traffic flows and perform graceful degradation by shedding least requested flows thereby freeing up much needed network resources. Network service performance optimization is achieved through high-level decisions that come from the analysis of low-level network data. The self-

adaptive framework correlates low-level network QoS parameters that are analysed using a supervised Machine Learning (ML) algorithm to yield performance predictions, responding to them autonomously by modifying underlying networking configurations if required. This allows the network management system to adapt to and actively learn from changes in network service performance characteristics or user demands that can occur and may vary over the lifetime of the network service provision.

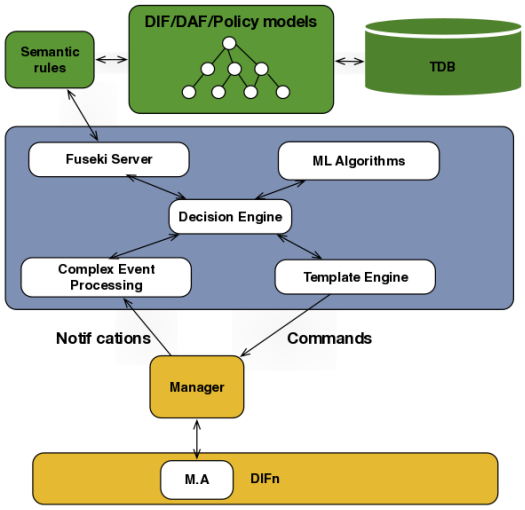

Figure 17. Semantic Network Management Framework

The functions of the self-adaptive framework are outlined in Figure 17 and include event monitoring/correlation for identifying and relating pertinent network events, machine learning for making network service "importance" predictions and model-driven development techniques for establishing (on demand) a QoS cube in a DIF layer. Management policies provide the high-level goals that actively guide the decision-making process of the self-adaptive framework to ensure adherence to high-level management objectives.

The self-adaptive framework correlates generated events with their associated DIF and calculates the effective network management impact caused by the event and specifically application impact. Semantic web rules are used to update and query the knowledge model to accurately reflect the current runtime state of the system. For example, analysis on the actual usage of network resources as opposed to the declared

expected usage or if there are historical basis to these alarms (e.g. greater than 20% under utilisation, greater than 10% admission failures, etc.). This allows network service performance tuning, where the allocated network resources are optimised to the aggregate needs, increasing resources to over-loaded network services and reducing resources to under-utilised ones (i.e. load shedding). This involves adjusting the DIF configurations and in particular the QoS cube and Resource allocation policy parameters to a more optimised form.

The recursive nature of RINA facilitates a much better aggregation of events and correlation of alarms, with a consistent protocol for data transfer differentiated by means of policies at each layer. The availability of a single management protocol and consistent object model for the Management Information Base (MIB) known in RINA as a Resource Information Base (RIB) for the logical representation of information held by the IPC Process (IPCP) for the operation of the DIF, supports a simpler and more powerful application of the ontologies and machine learning models.

Machine learning from statistical learning theory is applied to balance desired changes in network configuration with protecting existing configurations. The function analyses changes in network service demand from specific users or user groups and harnesses a supervised machine learning algorithm to facilitate domain adaptation by developing a system of network service demand prediction and provisioning which allows the underlying network to resize and resource itself to serve predicted network service demand according to various parameters such as bandwidth, undetected bit error rate, delay and jitter. This is achieved while optimising network service performance, maximising use of available underlying network resources and minimising overall network costs for network service providers. ML is performed over identified subsets of the knowledge model to make predictions for more optimal network service provisioning.

The function performs automated re-configuration of DIFs. The framework considers QoS cube configurations deployed for provisioning currently active network services, and those requested for newly deployed applications. Should there be a mismatch and the application or is deemed important enough, a QoS cube (tailored to the requirements of the new application) can be created in the DIF and the resource allocation policy adjusted to the relative "importances" of the QoS cubes. Thus, it learns

when an application's network requirements are not being met and where appropriate, takes remedial actions.

An automated network service provisioning use case is outlined in Figure 18 and is based on a content distribution network (CDN). Specifically, the use case shows how learned flow "importance" can be used to shed less important flows. In the use case, a user that started watching a football game on their mobile phone in Standard Definition (SD) quality. However, half-way through the game the user decides to watch the remainder of the game on their TV in High Definition (HD) quality. Unfortunately, the underlying network flow currently only supports a SD quality QoS rate. This means the user's device (TV) initiates a flow request to the "App cache HD" application process on the Delivery Server (IPCP3) for a higher rate QoS cube (HD). The Resource Allocator discovers there are insufficient network resources available to support HD quality video in the DIF which results in the *allocateNotifyPolicy* firing with a flow allocation failed result. The *allocateNotifyPolicy* is an example of a RINA policy where the behaviour of the network can be "programmed" with a specific set of actions. In this case, generating a fault event for the Management Agent which has been configured to report these flow allocation failures to the DMS.

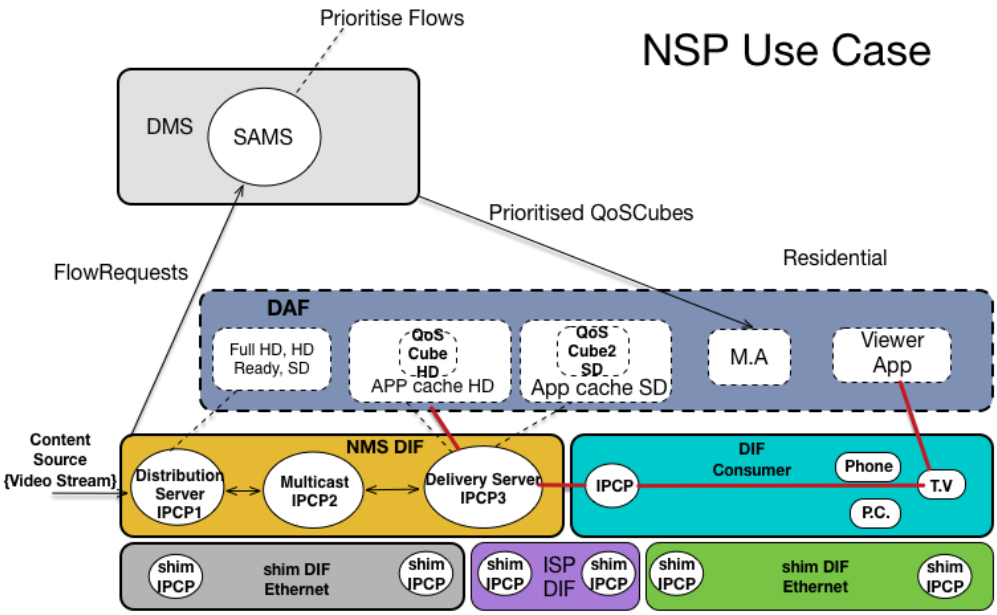

Figure 18. Use Case Scenario

RINA conceives a DIF Management System (DMS) as a centralized tool to perform management tasks over the systems of the network capable of

making complex configuration changes affecting multiple layers at once and optimizing the performance of a set of layers working together. The DMS follows a manager-agent (MA) model for its network management tasks using two protocols, the Common Application Connection Establishment Phase (CACEP) allowing application processes to establish an application connection and the Common Distributed Application Protocol (CDAP) enabling distributed applications to communicate at an object level rather than using serialisation to assist the DMS runtime operations.

Within the DMS there are a number of management strategies in operation. Later, we will define a DMS strategy to automatically add QoS cubes to DIFs, so we need a "check" in the DMS, in the form of a strategy for automatically removing QoS cubes from a DIF if they are unused. The *resourceAllocationPolicy* parameters are then adjusted to give higher "importance" to other QoS cubes. A second set of management strategies (for HD) are triggered and examine the failed flow request. These determine that the flow (and associated QoS cube) is of higher "importance" than some of the existing flows, thus CDAP actions are generated. The following list details the steps that are performed:

- The DMS sends a CDAP create operation (via the Management Agent) to create a QoS cube in the consumer DIF.
- The DMS sends sends a CDAP write to adjust the relative "importance" of the QoS cube within the *resourceAllocationPolicy* parameters. Subsequent flow allocations will discard flows from the least "important" QoS cube.
- The *allocateRetryPolicy* in the Consumer DIF is configured to retry the flow allocation four times (at 500, 1000, 2000 and 3000 ms)
- The *allocateRetryPolicy* fires and the DIF allocator retries the flow allocation. This time the flow allocation succeeds.
- Flow allocator then notifies the application process (in this case the App cache) to accept the flow allocation request.
- Assuming a positive response, the flow allocator reserves required resources, and responds to the TV application.
- A new flow (HD) is instantiated as the response is acted upon.
- Lower "importance" flows are discarded and closed.

In the above scenario, event correlation is trivial as there is a single event triggering the action. In a more realistic environment, QoS degradation

is a possibility leading to multiple QoS Violation events being generated (from each node concerned). Additionally, an application instance may have multiple flows, with different QoS cubes active concurrently, more advanced correlation algorithms are needed to isolate the affected application instance, as multiple instances of the viewing application could be in use on the same network segment.

Our prototype implementation includes creation of ontology models used to represent both the RINA network within the CDN and the network services running over the RINA network. Fault reporting (to the DMS) is not fully implemented yet, so failure reports are simulated. However, the policy mechanism hooks are part of the RINA SDK, which also includes the Management Agent. The RINA SDK are implemented in C++, and is available from the GitHub repository [rinaimpl2015]. Most of the other components in the prototype are implemented in Java.

The network (DIF/DAF) and policy models were created in the web ontology language (OWL) [motik2009owl] using Protege [Gennari02theevolution], a tool for creating and editing ontologies where a network model and a policy model have been defined for both DIF/DAF and management policy models. SPARUL [seaborne2008sparql], a semantic web language was used to query and update the domain and policy knowledge bases. In particular, semantic web queries were executed over the properties of concepts specified in the knowledge model to act as filter returning only a subset of the individuals from the complete network and policy models. Fuseki [foundation2015] was used to load the required domain and policy ontologies, issue semantic queries over the loaded ontological knowledge bases and store the results in a data structure. A Jena API [mcbride2002jena] is used for manipulating the ontology models.

RabbitMQ [rabbitmq2015] is an open-source messaging broker based on the Advanced Message Queuing Protocol (AMQP)[amqp2015] is used to aggregate and correlate the large number of network events, so that only the most pertinent network events are used to update the knowledge models and consequently used as input to the machine learning algorithm. Weka [hall2009weka] is a machine learning software tool written in Java that implements many machine learning algorithms such as naive Bayes, Bayesian networks and decision tree learners, etc. for performing data analysis and predictive modelling tasks. Weka supports a number of essential data mining functions such as data pre-processing, clustering,

classification and regression. In our approach we opted to use Weka's naive Bayes algorithm, a well known supervised learning algorithm whose classification approach is based on probabilistic knowledge. Naive Bayes is trained to classify flow requests to an associated importance as shown in the following Table.

Table 9. Importance Training information

Importance	1	2	3	4
Delay	100ms	250ms	600ms	2s
Jitter	20ms	45ms	55ms	65ms
Loss	10^{-5}	0.5%	1.5%	4%
Application	VoIP	HD-video	SD-video	Best effort

Thus each failed flow request receives an importance rating which assists in the decision making process. The premise being that within this consumer DIF, higher importance flow requests (and associated QoS cube) should be accommodated even at the expense of existing lower importance ones. A more complex classifier could also take into account the device initiating the flow and if suitable RINA authentication policies are applied, the person using the device.

Based on the results of this orientation step, a decision can be made to allow the new flow (and support the associated QoS class) within the current DIF according to high-level management policies that dictate the overall network service behaviour. Assuming the flow allocation is to be allowed, a simple test is made in the form of a SPARUL query for the existence of an appropriate "importance" QoS cube in the DIF. The result triggers the current DIF configuration to be either maintained as is or modified to support a new QoS class. We used the String Template engine to create pre-defined DIF templates with place-holders for the various QoS Cubes for DIFs. If a DIF configuration change is required, the DIF QoS place-holders are populated with the appropriate QoS parameters for that network service according to the goals of the management policy and deployed onto the devices.

5.2. Centralized resource reservation strategy for RINA

Existing multipath routing techniques most commonly rely on per hop decisions. This approach is taken due to its simplicity of implementation, since to reach the destination, each node performs forwarding decisions

only based on local information, i.e. the link congestion of the possible next hops. The work in [Kvalbein2009multi] shows one implementation of this approach.

However, in complex networks with many different possible paths to reach the destination, only suboptimal results can be obtained with previous approaches, as per hop forwarding decisions are not aware of the overall network status. Depending on the characteristics of the traffic and the network, it may be desirable to trade off some node configuration simplicity to calculate the best possible paths for each type of traffic, in order to achieve the optimal forwarding solution at each hop. For that, end to end path calculations are used, requiring either having the nodes to be aware of the status of the network using different synchronization mechanisms [Elwalid2001mate][Kandula2005walking] or introducing an external centralised element to act as a decision manager, as shown in [Fares2010hedera].

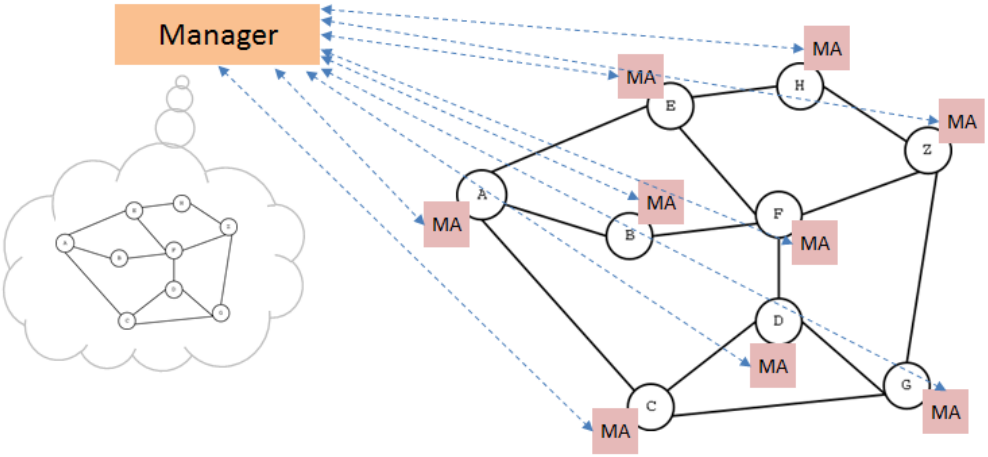

Figure 19. Centralised resource reservation

The characteristics of the RINA framework provide a solution to achieve optimal multipath forwarding decisions by exchanging information between IPCPs and introducing a central manager in the network. All IPCPs in a DIF share the information by means of the Resource Information Base (RIB) which acts as a distributed database among them. Furthermore, the central manager integrated in the RINA architecture has the overall view of the whole network, allowing it to calculate the best solution for the multipath forwarding decision on each node. This section describes a centralised multipath strategy that exploits the aforementioned elements in combination with the multipath routing policies developed in WP3 to

determine optimal load balancing forwarding decisions on each IPCP of a DIF.

5.2.1. Description of the strategy

The flow allocation strategy is described in Figure 2 – the NMS DIF is not shown for clarity reasons. The key idea is that during the allocation of the flow (step 1, 2) the Management Agent (MA) forwards the flow allocation request to the Manager of the RINA network (step 3). The Manager will then process the notification, use the information available in the RIB about the congestion in the DIF to compute the best route and associate the new flow to a specific port of the N-1 DIF on every IPCP along the path. After that it issues CDAP messages to the Management Agents of the systems through which the flow will be routed, in order to modify the configuration of the relevant IPCP's forwarding table (step 4). Last but not least, the Manager forwards the Flow Allocation request to the MA in the system where the target application is running (step 5, 6). There the IPCP processes the flow allocation request the usual way (step 7). If the application accepts the flow, the Flow Allocation Response is sent back to the source IPCP (step 8); otherwise it is proxied to the Manager so that it frees all the resources that had been allocated to the flow.

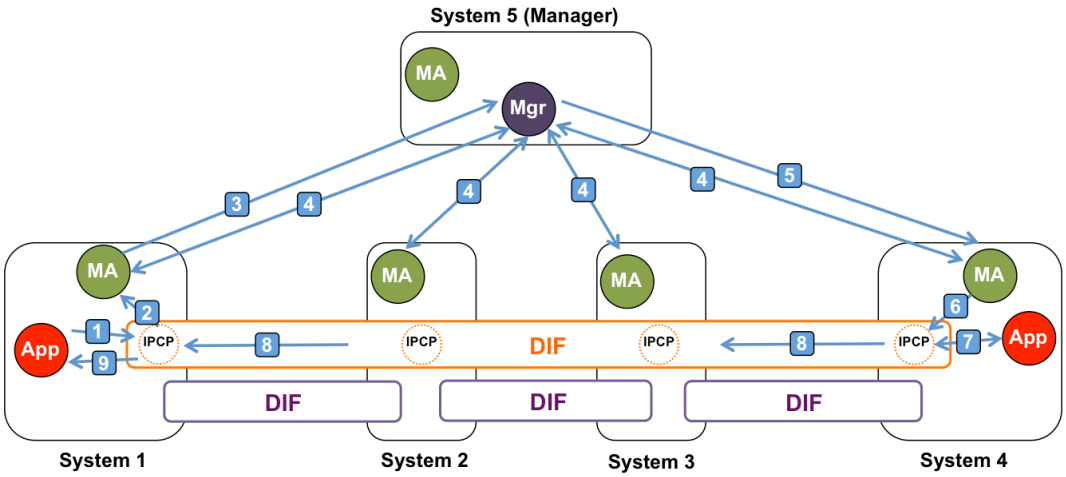

Figure 20. Description of the flow allocation strategy

The following paragraphs explain the procedure with more detail. This strategy has been designed for data centres fabric DIFs allocating flows between IPCPs of higher-level DIFs. In this context it is expected that the vast majority of flows would be accepted by the target application (an IPCP in a DIF of a higher rank), therefore the NMS would rarely need to undo the allocation of resources. If this condition was not met, the actual reservation

of resources (creation of new PDU Forwarding Table entries in the IPCPs on the path of the flow) would have to be deferred until the flow had been accepted by the target application.

1. Flow allocation request

When an application requests a new flow, the Flow Allocator follows the usual flow allocation procedure: select the EFCP policies for the flow, instantiate and configure EFCP. It then creates a new flow object in the RIB with state set to “allocation in progress”. However, after that it forwards the CDAP Create request on the Flow object to the MA in its system. After that it waits for the response.

Once the Manager receives the request, it tries to find a route with enough resources for the flow as explained in the next section. If it doesn’t succeed, it sends a CDAP Create Response Message to the MA, indicating that the flow has been rejected due to lack of resources. Then the MA forwards it to the IPCP, who deletes the EFCP instance and notifies the application that requested the flow.

2. Flow allocation request

At this point the Manager will try to find a path in the DIF capable of supporting the bandwidth requirements of the flow, and in case that there is more than one option, select the most appropriate one. To do that the manager will need to know the state of all the transmission queues of all the possible ports in the DIF, so it will use the information in its RIB, which is already getting via notifications from each Management Agent in the network. Specifically, the Manager checks the following RIB objects of each IPCP:

- Neighbors
- PDUForwardingTable
- RMTQueuePair

The selection of the path is performed according to the following algorithm represented as pseudo-code:

```
ALGORITHM: PathSelector
# input
NodeDataBase # Data Base with all the necessary information from RIB
```

```
newFlow      # New flow to be routed
fwdAlgorithm # Specific forwarding algorithm for allocating new
             # flows presented in D33

# output
PathInfo     # Information of the selected path

PathSelector(NodeDatabase, newFlow, fwdAlgorithm)
BEGIN
  if newFlow.dst reachable from newFlow.src
    PosiblePaths[]
    PortCongestion[]
    QoSCongestion[]

    recursivePathFinder(newFlow.src, newFlow.dst,
        newFlow.qos, newFlow.flowid, NodeDataBase, PosiblePaths)
    if PosiblePaths.size>0
        for i less than pathLength
          for each path in posiblePaths
            if not shared link
              if PortCongestion[path] < PosiblePaths[path].hops[i]
                    .PortCongestion
                PortCongestion[path]=PosiblePaths[path].hops[i].
                     PortCongestion
               QoSCongestion[path]=PosiblePaths[path].hops[i].
                    QoSCongestion

              end if
            end if
          end for
        end for

        PathInfo = PathSelector(PosiblePaths, PortCongestion,
            QoSCongestion, algorithm)
    else
        PathInfo = reroute(newFlow.src, newFlow.dst, newFlow.qos,
        newFlow.flowid, NodeDataBase)
    end if
  else
    PathInfo = Empty
  end if
  if PathInfo != Empty
      updateBW(PathInfo)
  end if
  return PathInfo
```

```
END
```

The `recursivePathFinder` function will return the set of the minimum length paths that have enough bandwidth to allocate the new flow. After that, the best path will be selected based on a configurable decision algorithm. The one used by default uses the maximum bandwidth congestion of every port in each path as a metric to avoid possible bottlenecks in the network, as the good results in [Kandula2005walking] show using a similar strategy.

The `PathSelector` function can also take into consideration both the accumulated congestion of the ports and the specific congestion associated to the QoS-id of the new flow. This can become useful if the algorithm is dealing with different QoS priorities, thus being able to balance not only the overall load but the load associated to each QoS as well. The objective of this strategy is to facilitate the burden of discarding low priority (e.g. best-effort) traffic in case of congestion.

In case no path is able to allocate the flow, the algorithm will try to reroute existing flows by changing the N-1 DIF ports associated to them to make room for the allocation of the new flow. This reroute process consists in moving smaller flows to another path as shown in the image.

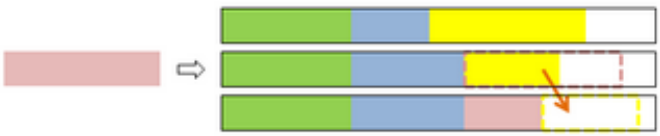

Figure 21. Reroute technique

The reroute technique will try to act following the next principles:

- Move the minimum amount of flows.
- Move the minimum amount of bandwidth.
- Move flows with the lowest priority first.

According to those fundamentals the algorithm will try to reroute first the biggest flows in terms of required bandwidth and find a new path changing the minimum number of hops with respect to the original, in order to reduce the number of changes and reroute actions needed. Finally, once the new path has been correctly located, the Manager will update this

information internally to make sure the correct QoS assurance of future flows.

3. Path configuration

The last part of the strategy is the path configuration in the IPCPs. In order to support that, a new object has been defined and included in the RIB: the `PreForwardingTableEntry`. This new object binds the EFCP connection identifier to the N-1 port to which the flow will be forwarded at each IPCP in the path of the flow. The connection identifier is the combination of source and destination cep-ids as well as the qos-id. When a new flow needs to be allocated the Manager, after calculating the path, introduces a new entry in the `PDUForwardingTable` using the CDAP "CREATE" message with the port of the N-1 DIF through which the flow is going to be forwarded. If all the configuration changes are successfully applied, the Manager moves to the next phase. Otherwise the changes are rolled back and the source MA notified about the failure allocating the flow.

The forwarding policy in the IPCPs is simple: when a PDU arrives, a match for the EFCP connection id is looked up first. If the Manager has already configured the path for the associated flow, the PDU is sent through the selected port. In case the entry is not found, a match for the destination address / qos-id will be looked up and the PDU will be routed based on the configured hop-by-hop forwarding policy.

`PreForwardingTableEntry` objects are extensions of the `PDUForwardingTableEntry` objects.

```
/**
 * RO Class PreForwardingTableEntry - maps flow id to the N-1 port-id
 where the PDU will be forwarded
 *
 */
ro class PreForwardingTableEntry
  behavior
    "Entry in the pre-forwarding table. Maps flow id (composed by "
    "source and destination cepids) to the N-1 port-ids where the "
    "PDU will be forwarded"

  attributes
    /Attributes.TableKey key     "Unique key of this entry in the table"
    /Attributes.FlowId   flowId  "Id of the flow the PDU belongs to"
```

```
    /Attributes.Port     portId  "N-1 port-id where the PDU will be
forwarded"
 ;

 operations
     create "invoked to add a static entry to the pre-forwarding table"
         in /Types.T_PreForwardingTableEntry prefTableEntry
            "The data of the pre-forwarding table entry"
     delete "invoked to remove a static entry from the table"
     read "read information from the table"
         out /Types.T_PreForwardingTableEntry prefTableEntry
            "The data of the pre-forwarding table entry"
     cancel-read "cancel ongoing read operation"
 ;

 registered-as ERoot Classes(1) 57
;
```

4. Flow allocation: notifying the target system

After having successfully configured the changes in the network, the Manager forwards the Flow Allocation request (CDAP Create message on a Flow object) to the MA in the target system (where the destination application is running). There the MA forwards it to the IPCP of the relevant DIF, who continues with the default Flow Allocation procedure: creation of EFCP instances and notification of the incoming flow request to the application. If the application accepts the flow, the Flow Allocator sends the CDAP Create response message to the source IPC Process, who considers the Flow allocated and notifies the application.

If the application rejects the flow, the CREATE flow response CDAP message is sent to the Manager via the MA, who removes all the PDU Forwarding Table Entries associated to the flow from all the IPCPs in its path and forwards the CREATE flow response CDAP message to the source MA. The source MA forwards the request to the relevant IPC Process, who removes the EFCP instance and notifies the source application.

5.2.2. Experimentation results

To evaluate the performance of the centralised resource reservation strategy presented in this section, the same experiments carried out in the deliverable D3.3 have been repeated using the new strategy.

The network configuration is show in the image:

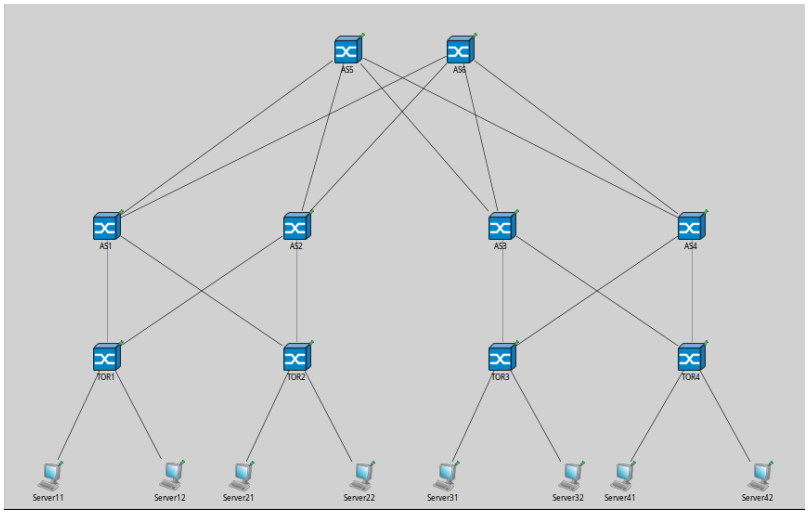

Figure 22. Network configuration for the experiment

In the experiment Server11 and Server12 are sending traffic to Server41 and Server42 with the following specifications: the first QoS class (QoS1) defines a bandwidth of 40% of the total DIF capacity, the second one (QoS2) requires 10% of the capacity and the last one (QoS3) is associated to 1% of the maximum bandwidth. The number of flows for each QoS class was the following: 1 for QoS1, 4 for QoS2 and 20 for QoS3, giving a total of 100% of the DIF bandwidth. The flow allocation requests are randomly initiated to simulate the variance of the traffic in the network.

The results obtained with the multipath routing using a hop-by-hop strategy for the forwarding decisions were the following:

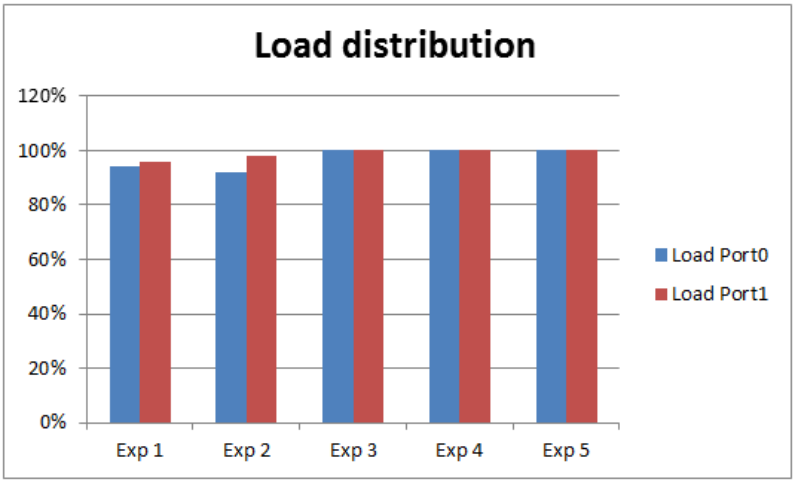

Figure 23. Load distribution in TOR1 using a hop-by-hop forwarding strategy

Theoretically the load in each port should be 100% but in the first two experiments some of the flows were rejected because there was not enough

bandwidth in any of the available ports. The results obtained with the centralised resource reservation strategy are shown next:

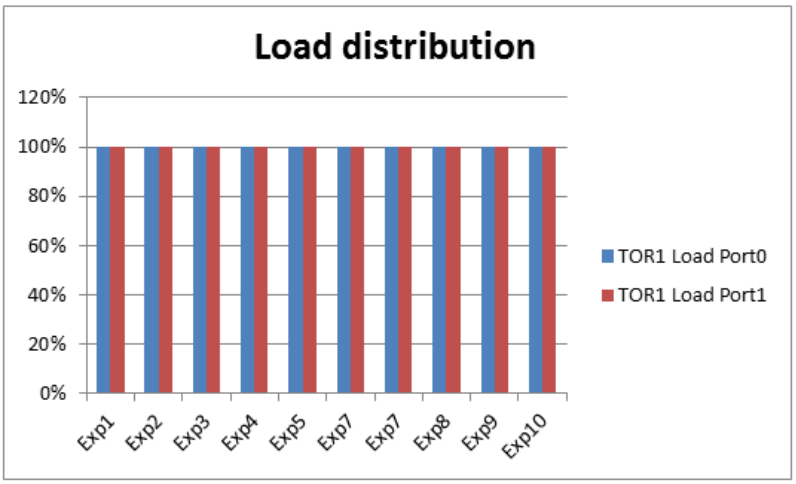

Figure 24. Load distribution in TOR1 using the centralised resource reservation strategy

In this case, the strategy was able to correctly distribute all the flows in every experiment. This was possible thanks to the re-routing actions that were taken once insufficient bandwidth in a port for a new flow was detected. Thus, the expected results of having 100% utilisation were achieved. It is interesting to also check the load distribution in the next hop of the paths.

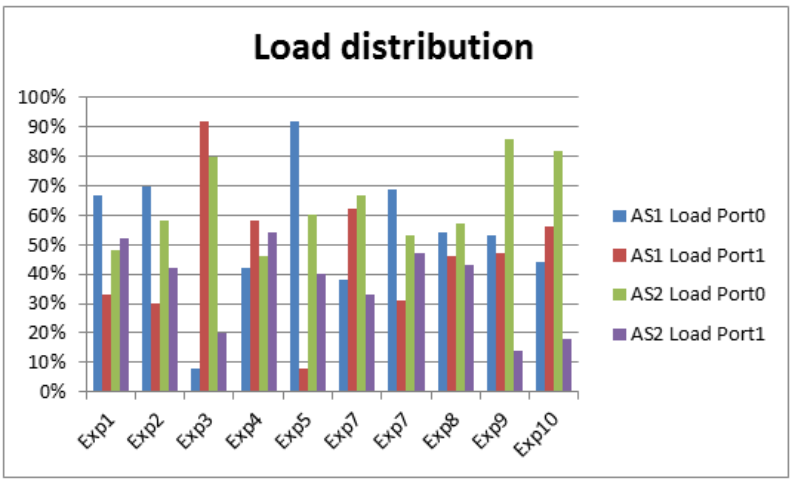

Figure 25. Load distribution in AS1 and AS2 switches using the centralised resource reservation strategy

Contrary to the results obtained in the hop-by-hop load balancing multipath decisions, where the load was kept around the 50% on each output port, here a wider variation between port loads is found. To better explain that behaviour, it is required to analyse as well the number of reroute actions to allocate all the flows that were taken in each execution.

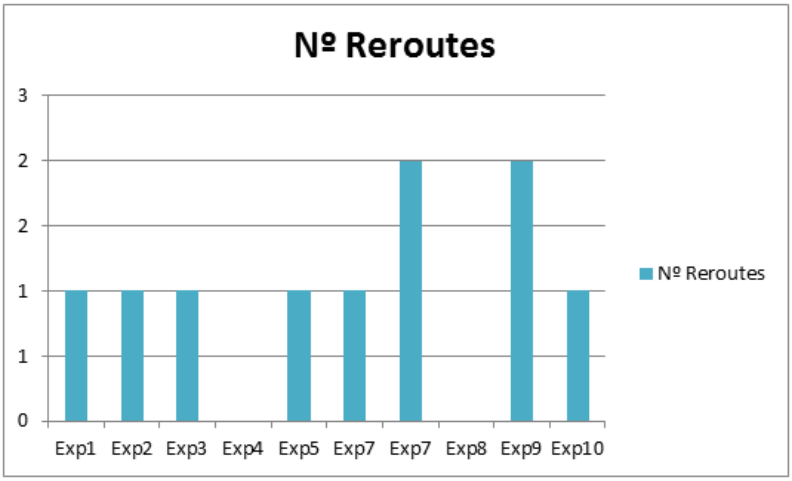

Figure 26. Number of reroute actions taken by the Manager to allocate all flows

From this figure, it is clear than the two executions without reroutes have a load distribution closer to the expected 50%-50% balanced one. The reason behind this is that the reroute policy was designed to apply the big flows algorithm explained in D3.3, where the Manager tried to make room for new flows prioritising the movement of the biggest ones. This specific policy was chosen in order to minimise the number of reroute actions and therefore the time required to complete the flow allocation.

Another experiment was performed sending traffic from Server11, Server12, Server21 and Server22 to Server31, Server32, Server41, and Server42. In this case, the load was set up to get 100% bandwidth utilization in every port and the focus was put in the number of reroute actions needed to allocate all the flows. The numbers obtained for ten iterations are shown in the figure:

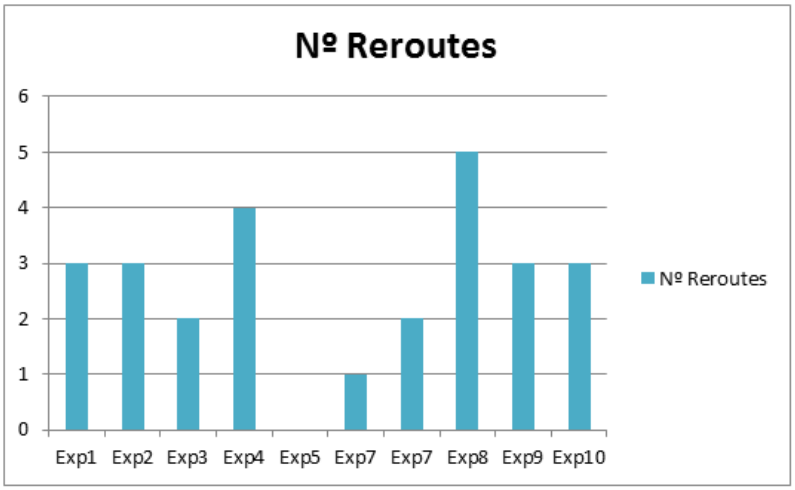

Figure 27. Number of reroute actions for a 100% bandwidth utilisation

In one case, no reroutes were needed whereas the maximum number of reroutes performed was 5. On average, 3 (rounded up from 2.6) reroutes were necessary to allocate flows to fill the maximum available bandwidth. From this experiment, it is clear that the capability of re-routing traffic is very important to achieve allocation ratios of 100% bandwidth utilisation.

Finally the centralised reservation strategy for multipath has been tested in a situation of low load in the DIF to determine how the Manager distributes the flows without the having to trigger reroute actions. In this case the servers are generating traffic to achieve a port load of 50% of the bandwidth in the TOR1.

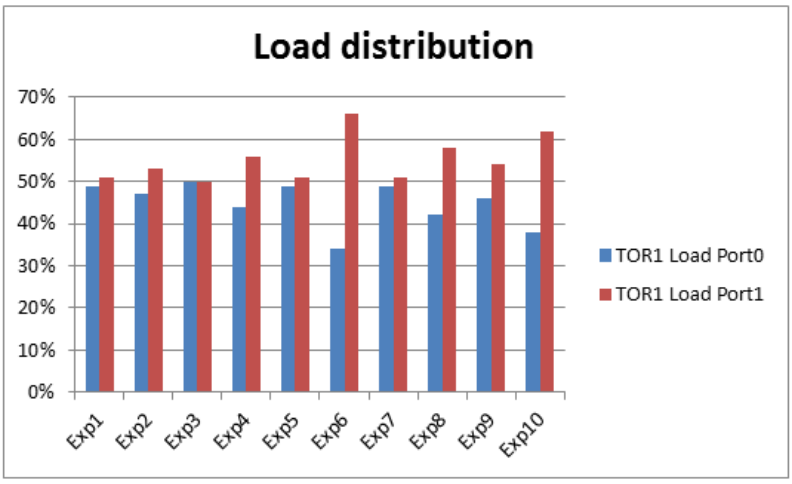

Figure 28. Load distribution in TOR1 for 50% bandwidth capacity

As it can be seen, this multipath strategy does not prioritise the equal load distribution at each hop, leading to a different balance of the traffic between the two N-1 DIF ports than in the case of previous hop-by-hop multipath decision algorithms.

Additionally, the load distribution of next hop nodes AS1 and AS2 was also registered, as the traffic arriving to those nodes is less uniform due to the forwarding decisions taken in TOR1 and TOR2 nodes.

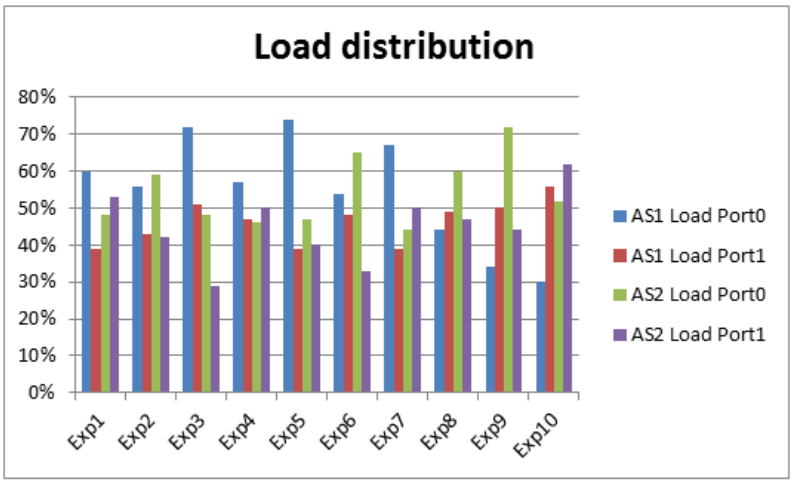

Figure 29. Load distribution in AS1 and AS2 nodes for 50% bandwidth capacity

Similar to the previous figure, the load distribution presents more variance, frequently loading one of the two N-1 DIF ports more than the other. In this strategy however, the Manager looks for a correct load balance over the whole end to end path, taking into account the most congested hop in each path as the key metric.

5.2.3. Conclusions

The centralised resource reservation presented in this section has proven to be able to define forwarding decisions for each IPCP in a DIF that guarantee the correct allocation of new flows, even when the network is working at 100% capacity. For that, re-routing actions are sometimes required, moving existing flows from one N-1 DIF port to another to free space for new flows. By prioritising the re-routing of flows with more bandwidth requirements, the strategy attempts to reach a solution with the least possible changes in the routes.

Although centralised resource reservation strategies can already be done in the IP world through, for example, the use of MPLS protocol, they require more complex network engineering to be implemented than with the use of RINA. First, RINA architecture has built-in support of a central control manager to supervise the network that communicates with IPCPs using Management Agents. Second, the RIB acts as a distributed database between IPCPs of a DIF and the Manager that contains all the objects required to perform any configuration change in the policies with a simple communication protocol (CDAP). Finally, as already mentioned for previous developed multipath policies, the integrated definition of QoS

cubes to classify the flows is essential to achieve a good optimisation of the traffic distribution across multiple end-to-end paths.

5.3. NFV Chain Configuration

Network operators are currently transitioning from hardware-based middle-box models, where network functions, such as fire-walling, load balancing, and caching, are implemented as vertically integrated solutions, to Network Function Virtualization (NFV) where the same operations are performed by software instances running on general purpose virtualized networking and computing infrastructures.

A Virtual Network Function (VNF) consist usually in one or more virtual machines that run over general purpose hardware. Such virtual machines are then connected in such a way that the network traffic is redirected from one to another in an ordered way.

The main benefit of this architecture is the possibility to implement network functions, that previously required specialized hardware, using general purpose devices. Virtualization also allows to scale up/down in terms of performances and throughput, allowing to reach flexibility levels which was difficult to reach because of the set-up cost.

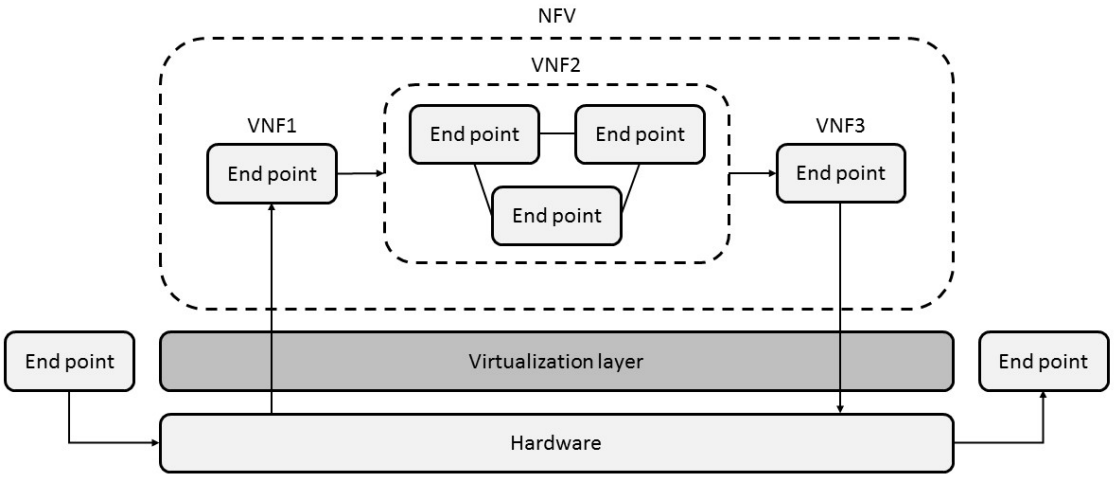

Figure 30. Classic example of NFV

The figure above depicts a generic network service deployed on top of an NFV Infrastructure.

5.3.1. Compatibility with existing NFV software

Several NFV solutions, and in particular several VNF implementations, are already available on the market. As a result it is desirable to maintain compatibility with such technologies even when a disruptive and clean slate technology such as RINA is used in the networking fabric. This allow to reuse previously built solutions with minimal updates and without the necessity to replace all the hardware. This can be done thanks to the layering nature of RINA itself. In RINA every layer is independent from the other, and just share Quality of Service information when it has to relay over another one. Existing NFV software relay on specific network protocol (mainly IP) in order to work in a transparent way and without too much loss of performance.

In this section we want to describe a way to deploy and configure a RINA network capable of transporting IP over RINA. This will allow us to transport legacy NFV Network Service over a RINA network without requiring any change to the actual VNF. On the other hand changes to the NFV Management and Orchestration layer may be required.

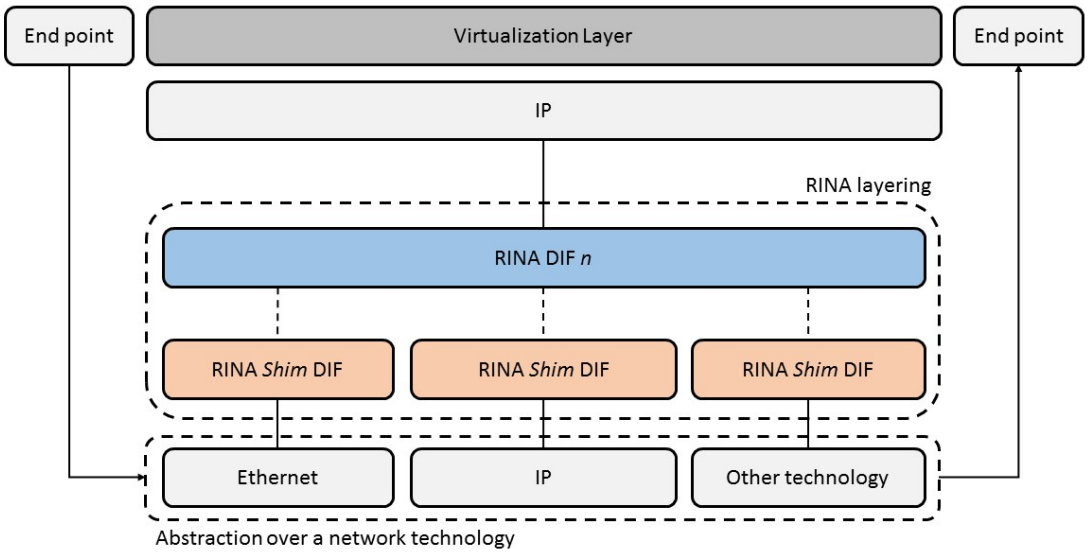

Figure 31. RINA and legacy NFV working together for compatibility.

Is important to underline the fact that the solution presented in this section is not limited only to IP over RINA scenarios, on the contrary it can be applied in principle to any other networking technology, e.g. WiFi or LTE. This also applies to lower layers which allow to carry RINA over an another link technology. In fact it is always possible to introduce a new shim which enable the RINA network to “travel” over another network (Ethernet, IP, wifi, etc…), and such change does not affect the entire stack already set up, but only the bottom layer. The entire VNF virtualization layer, being on the topmost position, is not affected by such changes and can continue to operate without modifications.

5.3.2. Configuration of a single NFV chain

We will now describe how an NFV Forwarding Graph can be deployed over a RINA-enabled network. The Application Entity, creates a virtual interface using TUN/TAP technology which the VNF virtual machine can use to receive and send traffic. The interface must be configured with a suitable IP address. Notice however that it is not necessary to set-up any routing strategy, since RINA will implement such functionalities at a granularity that would not be available in a standard IP network.

Is also possible to introduce different traffic shaping and congestion control strategies directly in the RINA layers, and these will directly affect the encapsulated IP packets that are travelling over it. This allows to use protocols like UDP and TCP together with other congestion control strategies which helps you to further tune the performance of your network.

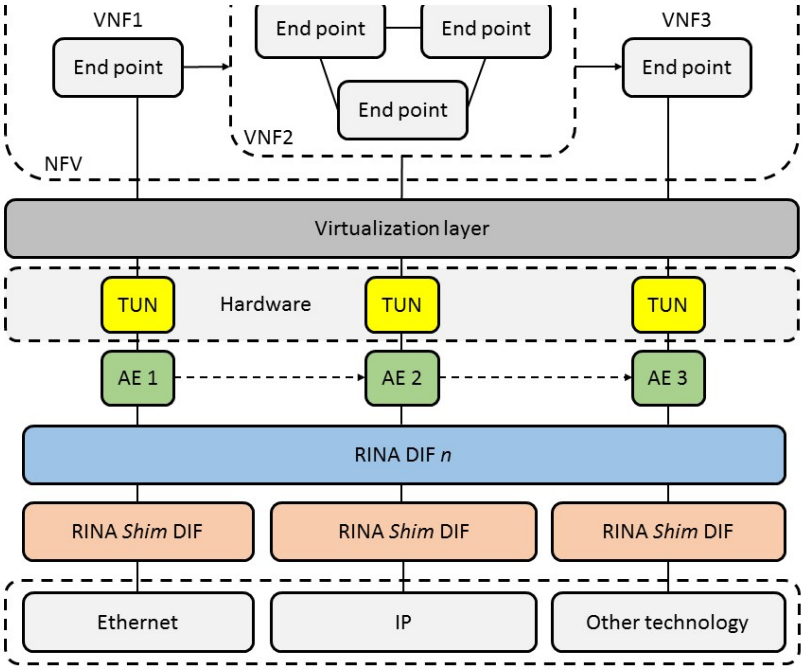

Figure 32. Configuration of a single NFV chain with RINA.

As can be seen in the picture, RINA inter-networking layers will offer connections functionalities to the various virtual interfaces created on the NFV nodes. Since the interfaces are not connected with each other, RINA is in charge of forwarding the messages in the network in order to reach the desired endpoint (and changing routing/forwarding allows use to change this strategy, potentially at runtime). The NFV Forwarding Graph is now inherited by the Application Entities which, following their own logic, must know who is the next element in the forwarding chain.

5.3.3. Configuration of multiple NFV chains

Now that the necessary tools to use legacy NFV software over a RINA substrate have been introduced, we can have a look at how such tools can be combined in a data-centre environment to build a full NFV Management and Orchestration solution.

Obviously such data-centre must be RINA-compatible, and this mean that every link which connects its nodes (servers, Top of Racks, Aggregating Switches and Border Routers) will be bootstrapped with a Shim DIF over a technology (Ethernet, for example). This will allow nodes to be ready to support RINA-like communications.

Still having point-to-point only communication is not enough, so an additional DIF is configured during bootstrap: the DC DIF. This layer

spans over the entire data-centre, and every IPC Process enrolled to it is considered a member of this DIF (for the moment we will not consider inter-data-centre communications). Such additional DIF, which is supported by the Shims, allows to reach every single node of the DC, giving us the access to the pool of resources necessary to support NFV operations.

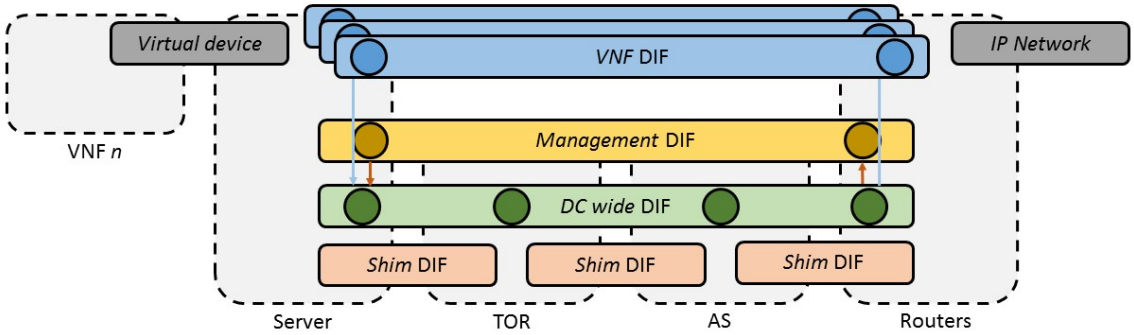

Figure 33. Configuration of a multiple NFV chains with RINA.

Since a dynamic configuration is more appropriate for this case, another layer is needed, which stands just over the DC one and interconnects the Management Agents (MA) with the central Manager, creating a separate network used only by the DC administrators. The MA will then be the ones who, reacting to Manager commands, will create/dispose the VNF in the DC. It's also their job to set-up the initial configuration of the Application Entities necessary to create the Forwarding Graph for that specific VNF.

This method allows to logically isolate every NFV from each other, to avoid any kind of interference which can happens by a bad configuration of the AE which register on that DIF. Any AE which is present on an NFV layer will have only the possibility to communicate with other AE which are member of that layer. This also does not limit any kind of resource sharing in case this will become necessary: one Server node (and so the VMs on it) can always be shared between two NFV by just introducing a new IPC Process which then enrol to a particular NFV chain.

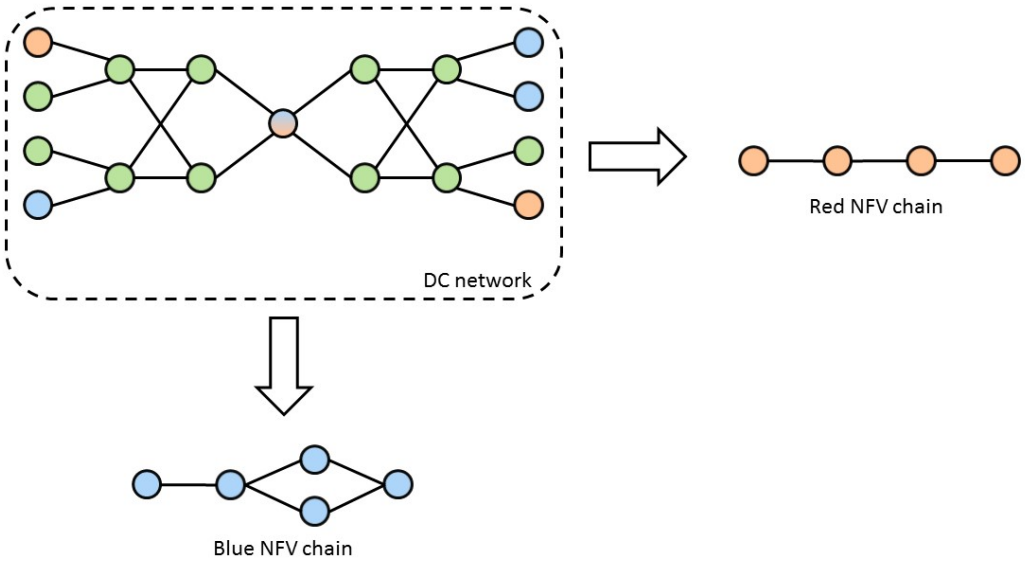

Figure 34. Different NFV chains are isolated from each other within the same DC.

Once the chain has been established, only the entry/exit point are left, and they are in charge of introduce the traffic into the DC and to dispatch the result of it to the requester. If a configuration where the stream of data arrives from legacy application is taken into account, then a sort of gateway which translates from IP to RINA and back is necessary on the Border Router, or on specialized machine just after it. This gateway will be the entry point (traffic which comes from the outside) and exit point (traffic which comes from the inside) of the NFV chain.

6. Security Management

The DMS performs three key roles in managing the security of a RINA network: configuration management, performance management and security monitoring. Configuring security in a RINA network may place a large overhead on the human network operator and increases the likelihood of mistakes in the network configuration, which can introduce security vulnerabilities and impact on the network performance. For example, in order to authenticate successfully, each IPCP in a DIF must be configured with authentication credentials that can be trusted and verified by all of the other IPCPs in the DIF. This requires credentials to be generated and distributed to each IPCP and these credentials should be unique for each IPCP. If an IPCP is given the incorrect credentials, then the IPCP may be able to enrol in a DIF that it is not authorised to, which breaches the confidentiality of the DIF. Alternatively, the IPCP may not be able to enrol in a DIF which it is authorised to join, which can affect the availability of the network for application processes and IPCPs in higher layer DIFs. Automating the configuration of the network is therefore important for reducing the burden on the human Network Manager and minimising the risk of misconfiguration.

There is always a trade off between security and performance, as security mechanisms, such as SDU Protection and Authentication, introduce additional processing and delay in forwarding PDUs, which can lead to congestion in the network. The impact of security on network performance is particularly adverse if the security mechanisms have been misconfigured. For example, if IPCP has been configured with the wrong authentication credentials or its credentials have expired, it will be unable to authenticate successfully and so will not be able to enrol in the DIF. The DMS therefore performs an important role in monitoring the performance of the network to ensure that the impact of security mechanisms on the performance is within acceptable limits and to take actions, e.g. reconfiguring the security mechanisms if it is not.

Networks are subject to attacks from malicious parties. Since the key aim of a RINA network is to provide an inter-process communication service to applications in a more efficient way, the main attacks will focus on disrupting this communication service. While security mechanisms, e.g. SDU Protection, can be deployed in RINA, these are not sufficient to stop all

attacks. It is therefore important to perform security monitoring to detect attacks on the network. Although similar to performance monitoring, security monitoring focuses on specific threats that have been identified through a security risk assessment process. It aims to identify processes that are not behaving as expected and the take actions against them. The IPCP's security management component and MA at the local-level or DMS Manager at the domain-level can maintain realistic courses of action that pro-actively or reactively address attacks to the RINA network. In response to the monitoring indications (e.g. detection of divergence from conditions of normal operations using different security metrics) that an attack is under way the DMS Manager, or MA on its behalf, can take actions.

Deliverable D4.3 extensively described the RINA' security components and their interactions with DMS manger and MAs. In the sections below we focus on Multi-Level Security.

6.1. Multi-Level Security

Multi-level Security (MLS) is an example where security management performed by the DMS is critical. Multi-Level Security refers to access control mechanisms for protecting data or "objects" that can be classified at various sensitivity levels, from processes or "subjects" who may be cleared at various levels of trust . A strict definition of MLS includes a formal model of classification levels for data and clearance levels for users, together with rules to prevent inappropriate access by users to data that is at a higher classification level than their clearance. Such a model is appropriate in many high assurance applications, and is often mandated in government and military contexts by policy.

MLS solutions for RINA have been proposed in D4.2 and D4.3 that use two components to create MLS-enabled networks: Communications Security and a Boundary Protection Component (BPC).

6.1.1. Security Management for Communications Security

Communications Security strictly protects the end-to-end transfer of data, enabling sensitive data to be sent over untrusted network by cryptographically protecting the confidentiality and integrity of data. This ensures that data cannot be inappropriately read from the communication channel (e.g. via eavesdropping or accidental leakage), and that data at

different classification levels is not inappropriately mixed. It also includes authentication of the end points to ensure that they are suitable for accepting the data being communicated, based on its classification level. For a MLS-enabled RINA network, communications security is achieved using SDU Protection policies to encrypt and/or apply integrity protection to SDUs. Several options for the placement of the SDU Protection policies are possible depending on the requirements of the network. SDU Protection policies can be applied in the end devices, i.e. the systems that are sending and receiving the data, or at boundaries between trusted and untrusted networks. Configuration management for Communications Security is complex. It requires coordination between Authentication and SDU Protection across the network. It may use pre-placed symmetric keys, in which case all IPCPs need to be configured with the correct symmetric keys. Alternatively, it may use a key agreement protocol to establish keys, which requires the IPCPs to be configured with the correct security parameters and public key material. Communications security also requires the ability to reconfigure IPCPs on the fly, e.g. if a key is compromised. The DMS is therefore critical in simplifying the configuration management.

6.1.2. Security Management for BPC

To make an MLS system more practical generally necessary to allow for at least some capability to send data from a high system (a trusted network) to a low system (untrusted network), e.g. to allow higher cleared users to send emails to lower cleared users. This capability needs to be carefully controlled to prevent accidental or deliberate release of sensitive information by users and to protect the high system from receiving malicious code from the low system. The BPC is used to control such a flow of data, to ensure that data transferred from the high system is actually at a suitable classification level for the low system. It may also control data imported to sensitive network, e.g. check for mal-ware.

A means of integrating the functionality of a BPC into a RINA-based network is specified in D4.3. The aim of the BPC is to control application data sent between two networks at different security levels to ensure it is appropriate for the recipient. The data sent from an AP is routed via the BPC node according to the routing policy. The BPC then inspects the data packets; makes a decision whether the packet can be forwarded to the

recipient; and enforces the decision. If the BPC determines that the data is not appropriate for the destination, it takes appropriate action according to its policy. There are several options how the BPC handles SDUs that are unsuitable for the destination, depending on the threat model and types of data being inspected.

Intercepting the SDUs and performing the inspection will add a delay to the forwarding of packets. It can also increase the packet loss, since the BPC's policy may mandate that SDUs are blocked from being forwarded if is they are not at an appropriate classification for the recipient. The BPC will therefore impact the performance of the network and so performance monitoring is important to ensure that the network performance is within acceptable limits. If the limits are exceeded, the DMS can reconfigure the BPC's policies, for example to redact SDUs to remove sensitive data, rather than blocking them. In addition, monitoring the BPC's operation may detect attacks, e.g. if the BPC is blocking many SDUs from a particular AP, it may suggest than a malicious party is attempting to exfiltrate sensitive data. The DMS therefore plays an important role in monitoring the impact of the BPC on the network's performance, as well as monitoring for malicious behaviour.

The configuration of the RINA network is critical to the BPC to ensure that IPCPs can only enrol in DIFs that are appropriate for their clearance level and that all data flows between different security levels are routed via the BPC. It requires policies for authentication and routing to be coordinated across the RINA network. Misconfiguration of these policies could enable SDUs to bypass the BPC and so avoid inspection. The DMS therefore plays an important role in ensuring that the network is configured correctly and monitoring to detect configuration issues.

Deliverable D4.3 extensively described the options for the use of MLS solution across managed networks, the use of SDU protection by MLS and how the MLS policies are configured via MAs at application or at DIF level where these policies are to be kept as RIB objects.

6.1.3. RIB Example: Key configuration

A fundamental aspect of MLS is the management of keys. For various reasons outlined in [D43], the Key Manager is treated as a separate component, so that it can securely hold keys (or tokens) with the minimum

of dependencies on other components, to minimise the attack vector. However, to allow the DMS to configure an individual IPCP or a DIF, a reference to the Key must be available in the RIB. This key reference is stored as part of the Security Management section of the management DAF. A `KeyReference` is defined as follows:

```
ro class KeyReference
  behaviour "This class represents a key reference to an authentication/"
    "encryption key.  It contains everything but the actual key itself."
  attributes
    /Attributes.T_String    name     "Name of the key reference"
    /Attributes.T_String    type     "A key type, diffe-hellman etc."
    /Attributes.T_DateTime  expires  "A timestamp after which the key is"
                                     " no longer valid"
  ;
 registered-as ERoot Classes 56;
;
```

A `KeyReference` contains the meta-data for a given key. The name is guaranteed unique within a given management DAF. The type field contains the type of key being referred to, and the expires field to indicate for how long the referenced key is valid for. However, using a `KeyReference` directly is somewhat limiting for certain scenarios. For example, when a given Key needs to be replaced, an alternate may also be needed, or to facilitate key rotation. This implies that the configured authentication and encryption policies do not use a `KeyReference` directly, but refer to a container, that can hold one or more keys. A key container is defined as follows:

```
ro class KeyContainer {
  behaviour "This class represents a key container, holding one or "
    "more keys. (eg. to allow rotation of keys)"
  attributes
    /Attributes.T_String  keyContainerID  "The identity of the container"
  ;

  contains
    /Classes/KeyReference as "refs"
    create-strategy "These objects are manually created"
    delete-strategy "These objects are automatically deleted when a "
                    "KeyContainer is deleted"
  ;
```

```
  registered-as ERoot Classes 57
;
```

Configured IPCP's or DIF's can refer to a `KeyContainer` within their configuration. Individual `KeyReference` can be added or removed as necessary to facilitate the higher goals. For example, should a IPCP become compromised (and potentially the key it is using) an alternative `KeyReference` can be added to the container, to facilitate authentication with a new (uncompromised) key. It is convenient to think of the RIB as capturing references to keys in use, and the `KeyManager` is managing securely the secrets associated with these keys.

7. Conclusions and future work

PRISTINE has designed and built the first prototype of a Network Management System to manage RINA networks. In doing so it has started to explore the advantages of managing the configuration, performance and security of such kind of networks compared to managing the networks of today. The commonality provided by RINA (all the layers have the same structure and protocols, with different policies) has allowed PRISTINE researchers to design a single RIB object schema to model the state of the different layers in the network.

This common RIB model reduces a lot the number of concepts/objects the Manager has to understand in order to manage the network. We have performed an initial worst-case (for RINA) comparison on the complexity of the configuration models for managing a large-scale Data Centre network, showing an already important simplification in configuring the network with respect to the "TCP/IP protocol suite" case. RINA benefits will still be larger in more heterogenous networks such as service provider networks.

Performance management of RINA networks uses the consistent QoS model followed by all RINA layers. A common layer API which allows layer users to express the requirements for their flows, coupled with the QoS cube model which maps the flow requirements to specific policies for different elements of the DIF (data transfer, scheduling, resource allocation, congestion management) facilitates reasoning about the network performance. We have applied this model to design a centralized resource allocation strategy, in which the Manager processes all the flow allocation requests in the network and configures the forwarding of the PDUs of each flow according to the network state exported by the Management Agents. We have also outlined a novel self-adaptive network management approach capable of learning the levels of importance associated with traffic flows in order to prioritise the most requested ones and perform graceful degradation by shedding the least requested flows.

The NMS prototype (Manager and Management Agents) has been demonstrated in an international conference and is currently being used by PRISTINE researchers to configure the DIFs used in their experiments with the IRATI implementation and the PRISTINE SDK. In that regards, work to improve the NMS prototype will continue within WP6, fixing the

bugs reported by experimenters and adding the most important missing features to make the prototype more usable. Last but not least, the ICT ARCFIRE project [arcfire] will continue developing and experimenting with the NMS prototype, focusing on converged network service providers as its main use case.

References

[amqp2015] Advanced Message Queuing Protocol, OASIS, Available at: https://www.amqp.org/, Accessed: 2015-10-23, 2015.

[arcfire] ICT ARCFIRE project website, Available at: online[1]

[commag10] J. Schonwalder, M. Bjorklund, P. Shafer. *Network configuration management using NETCONF and YANG*. IEEE Communications Magazine, September 2010.

[cnsm15] S. van der Meer, J. Keeney, L. Fallon. *Dynamically adaptive policies for dynamically adaptive telecommunication networks*. Proceedings of the 11th Conference on Network and Service Management (CNSM 2015).

[D33] Pristine consortium. *Deliverable-3.3: Final specification and consolidated implementation of scalable techniques to enhance performance and resource utilization in networks*. Accessed: 2016-06-10.

[D43] Pristine consortium. *Deliverable-4.3: Final specification and consolidated implementation of security and reliability enablers*. Accessed: 2016-06-10.

[D53] Pristine consortium. *Deliverable-5.3: Proof of concept DIF management system*. Accessed: 2016-06-10. Available online[2].

[day16] J. Day, 2016. *How naming, addressing and routing should work. PSOC Tutorial*. Accessed 2016-04-14. Available at: online[3]

[dcfabric] Brocade White Paper, *Data Center Fabric Architectures*. Online[4]

[evpn] A. Sajassi, J. Drake, N. Bitar, A. Isaac, J. Uttaro, N. Henderickx, 2015. *A Network Virtualisation Overlay Solution using EVPN*. IETF, L2VPN Working group; draft RFC draft-ietf-bess-evpn-overlay-02.

[1] http://ict-arcfire.eu
[2] http://ict-pristine.eu/wp-content/uploads/2013/12/pristine_d53-proof-of-concept-dms.pdf
[3] http://pouzinsociety.org/education/rina/mobility_multi_homing_multicast
[4] https://www.brocade.com/content/dam/common/documents/content-types/whitepaper/brocade-data-center-fabric-architectures-wp.pdf

[facebookdc] Alexey Andreyev, *Introducing data center fabric, the next-generation Facebook data center network* [Online]. Available at: Online[5]

[foundation2015] Apache, *Fuseki: serving {RDF} data over {HTTP}}* , Available at Online[6], Accessed: 2015-11-30.

[googledc] A. Singh, J. Ong, et al. *Jupiter Rising: A Decade of Clos Topologies andCentralized Control in Google's Datacenter Network*. In SIGCOMM, London, United Kingdom, August 2015.

[Gennari02theevolution] John H. Gennari and Mark A. Musen and Ray W. Fergerson and William E. Grosso and Monica Crubzy and Henrik Eriksson and Natalya F. Noy and Samson W. Tu, *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development* , International Journal of Human-Computer Studies}, vol. 58, pp. 89—123, 2002.

[hall2009weka] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, *The WEKA data mining software: an update*, ACM SIGKDD explorations newsletter, vol.11, issue 1, pp.10—18, 2009, ACM

[icton13] S. Azodolmolky, P. Wieder, R. Yahyapour. *SDN-based cloud-computing networking*. ICTON 2013.

[lapukhov] P. Lapukhov, A. Premji, J. Mitchell, 2014. *Use of BGP for routing in large-scale data centres*. IETF Network Working Group, draft-lapukhov-bgp-routing-large-dc-07

[mcbride2002jena] B. McBride, *Jena: A semantic web toolkit* , Internet Computing IEEE, Vol. 6, Issue. 6, pp. 55—59, 2002.

[motik2009owl] Motik, B. and Patel-Schneider, P.F. and Parsia, B. and Bock, C. and Fokoue, A. and Haase, P. and Hoekstra, R. and Horrocks, I. and Ruttenberg, A. and Sattler, U. and others, *OWL 2 web ontology language: Structural specification and functional-style syntax*, W3C Recommendation, 2009.

[5] https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/

[6] https://jena.apache.org/documentation/serving_data/

[netconfyang] J. Schonwalder, M. Bjorklund, P. Shafer. *Network configuration management using NETCONF and YANG*. IEEE Communications Magazine, September 2010.

[rabbitmq2015] RabbitMQ, Pivatol, Available: online[7], Accessed: 2015-10-23, 2015.

[rinaimpl2015] RINA implementation, IRATI Stack, Available: online[8], Accessed: 2015-11-10, 2015.

[seaborne2008sparql] Seaborne, Andy and Manjunath, Geetha and Bizer, Chris and Breslin, John and Das, Souripriya and Davis, Ian and Harris, Steve and Idehen, Kingsley and Corby, Olivier and Kjernsmo, Kjetil and others, *SPARQL/Update: A language for updating RDF graphs* , W3C Member Submission, Vol. 15, 2008.

[tnc16demo] E. Grasa, M. Crotty, B. Gaston, S. van der Meer, D. Staessens, S. Vrijders. *Configuration of a multi-tenant DC network based on RINA*. Life demo at TNC 2016[9], Prague, June 2016.

[tnc16demotut] TNC 2016 demo tutorial, Available online[10].

[vrijders16] S. Vrijders, V. Maffione, D. Staessens, F. Salvestrini, M. Biancani, E. Grasa, D. Colle, M. Pickavet, J. Day, L. Chitkushev, 2016. *Reducing complexity of Virtual Machine Networking*. IEEE Communications Magazine, April 2016 issue.

[yangcommon] J. Schoenwalder, 2013. *Common YANG data types*. IETF, RFC 6991.

[yangif] M. Bjorklund, 2014. *A YANG data model for Interface management*. IETF RFC 7223.

[yangip] M. Bjorklund, 2014. *A YANG data model for IP management*. IETF RFC 7277.

[yangroute] L. Lhotka, A. Lindem, 2016. *A YANG data model for routing management*. IETF NETMOD Working Group, draft-ietf-netmod-routing-cfg-21.

[7] http://www.rabbitmq.com/
[8] https://github.com/IRATI/stack
[9] https://tnc16.geant.org/core/event/53
[10] https://github.com/IRATI/stack/wiki/Tutorial-4:-Multi-tenant-DCN-TNC-2016

[yangbgp] A. Shaik, R. Shakir, K. Patel, S. Hares, K. D'Souza, D. Bansal, A. Clemm, A. Zhdankin, M. Jethanandani, X. Liu, 2015. *BGP model for service provider networks*. IETF interdomain routing group, draft-shaikh-idr-bgp-model-02.

[yangvxlan] H. Fangwei, R. Chen, M. Milligan, Q. Zu, 2015. *YANG data model for VXLAN protocol*. IETF NVO3 working group, draft-chen-nvo3-vxlan-yang-02.txt.

[yangbridge] M. Holness, 2015. *IEEE 802.1Q YANG module specifications*. IEEE 802.1 working group.

[yangevpn] P. Brissete, H. Shah, Z. Li, A. Liu, K. Tiruveedhula, T. Singh, I. Hussain, J. Rabadan, 2015. *YANG data model for EVPN*. BESS working group, draft-brissete-bess-evn-yang-01

[yanglacp] OpenDaylight project, 2015. *YANG module for LACP*. Accessed 15 April 2016. Available: Online[11]

[Kvalbein2009multi] Kvalbein, Amund, Constantine Dovrolis, and Chidambaram Muthu. *Multipath load-adaptive routing: Putting the emphasis on robustness and simplicity*. Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on. IEEE, 2009.

[Elwalid2001mate] A Elwalid, C. Jin, S. Low, I. Widjaja. *MATE: MPLS Adaptive Traffic Engineering*. IEEE Infocom, 2001.

[Kandula2005walking] S. Kandula, D. Katabi, B. Davie, A. Charny. *Walking the Tightrope: Responsive Yet Stable Traffic Engineering*. SIGCOMM'05, Aug. 2005.

[Fares2010hedera] Al-Fares, Mohammad, et al. *Hedera: Dynamic Flow Scheduling for Data Center Networks*. NSDI. Vol. 10. 2010.

[11] https://git.opendaylight.org/gerrit/gitweb?p=lacp.git

A. Appendix: RIB language grammar

A.1. Specification

The specification provides all required information about a GDRO specification.

- a specification date
- a specification version
- a set of declared paths for specification elements
- well defined standard classes

The following conditions apply:

gdroSpecification-001
a compiler must accept only one GDRO specification and should throw an error if more than one specification is given

gdroSpecification-002
the description of the specification cannot be blank

The rule gdroSpecification is defined as follows:

```
gdroSpecification:
    VAL_STRING+ specDate specVersion specPathDeclaration
    specStandardClasses
;
```

A.1.1. Specification Date

Each GDRO specification must define a data stating when it was finished.

The following conditions apply:

specDate-001
the date cannot be blank

specDate-002
the date should be of format 'dd-MMM-yyyy', e.g. 17-Mar-2016

The rule specDate is defined as follows:

```
specDate:
    AT_DATE VAL_STRING
;
```

A.1.2. Specification Version

Each GDRO specification must define a version.

The following conditions apply:

specVersion-001
: the version cannot be blank

specVersion-002
: the version should be of format 'Version.Major.Minor', e.g. '1.2.3' (see http://semver.org for details on semantic versioning)

The rule specVersion is defined as follows:

```
specVersion:
    AT_VERSION VAL_STRING
;
```

A.1.3. Path Declarations

A specification must declare what paths in the element tree are available. Each path is declared for a particular element, i.e. RO classes, attributes, notifications, policies, or type definitions. For each element, one or more paths can be declared. Each path then becomes the root path for definitions of the associated element.

For instance, if we declare two paths for RO classes, say /RINA/Classes and /Pristine/Classes, then we have two roots for RO class definitions, one for RINA and one for Pristine.

Since a declared path is the root for a particular element, no other element definitions are allowed in this path (or any subpath). For instance, for our two RO class paths, we can only define RO classes in them, not attributes or policies.

This also means that the final node in any path declaration can only be associated with one element. For instance, given our two RO class paths, we cannot declare another path for RO attributes of /RINA/Classes/Attributes. This would mean that we will have mixed elements in any sub path of / Attributes (classes and attributes). So paths are not allowed to cross each other, except for nodes that are not associated with a GDRO element.

This is very similar to a directory structure for which, two types of directories exist:

- a normal directory that can contain any other directory
- a directory associated with a particular element, containing only those elements and directories for those elements

This rule allows to declare the paths for the GDRO elements

- RO Classes,
- RO Attributes,
- RO Notifications,
- RO Policies, and
- RO Type Definitions.

The following conditions apply:

specPathDeclaration-001
at least one path must be defined for each of the elements (this is realized by the grammar rules)

specPathDeclaration-002
a path must not be declared more than once and paths are not allowed to cross each other

specPathDeclaration-003
a path cannot be the sub-path of another path

The rule specPathDeclaration is defined as follows:

```
specPathDeclaration:
    AT_PATH_REGISTRATIONS pathClasses pathAttributes pathNotifications
```

```
    pathPolicies pathTypeDefinitions
;
```

Path for RO Classes

RO classes can be defined in any of the paths defined using this rule. RO class registrations can extend any RO class path further.

The following conditions apply:

pathClasses-001
: RO classes can only be defined in a path defined, if RO classes are defined outside a given path an error needs to be thrown

The rule pathClasses is defined as follows:

```
pathClasses:
    CLASSES eid_registration_list
;
```

Path for RO Attributes

RO attributes can be defined in any of the paths defined using this rule. RO attribute registrations can extend any RO attribute path further.

The following conditions apply:

pathAttributes-001
: RO attributes can only be defined in a path defined, if RO attributes are defined outside a given path an error needs to be thrown

The rule pathAttributes is defined as follows:

```
pathAttributes:
    ATTRIBUTES eid_registration_list
;
```

Path for RO Notifications

RO notifications can be defined in any of the paths defined using this rule. RO notification registrations can extend any RO notification path further.

The following conditions apply:

pathNotifications-001
RO notifications can only be defined in a path defined, if RO notifications are defined outside a given path an error needs to be thrown

The rule pathNotifications is defined as follows:

```
pathNotifications:
    NOTIFICATIONS eid_registration_list
;
```

Path for RO Policies

RO policies can be defined in any of the paths defined using this rule. RO policies registrations can extend any RO policy path further.

The following conditions apply:

pathPolicies-001
RO policies can only be defined in a path defined, if RO policies are defined outside a given path an error needs to be thrown

The rule pathPolicies is defined as follows:

```
pathPolicies:
    POLICIES eid_registration_list
;
```

Path for RO Type Definitions

RO type definitions can be defined in any of the paths defined using this rule. RO type definitions registrations can extend any RO type definition path further.

The following conditions apply:

pathTypeDefinitions-001
RO type definitions can only be defined in a path defined, if RO type definitions are defined outside a given path an error needs to be thrown

The rule pathTypeDefinitions is defined as follows:

```
pathTypeDefinitions:
    TYPE_DEFINTIONS eid_registration_list
;
```

A.1.4. Standard classes

The classes define locations of required classes in the EID tree:

- the top of the inheritance tree (ITOP),
- the root of the containment tree (CROOT), and
- the top of the inheritance tree for RO Policies (PTOP).

ITOP is the single root of the inheritance tree for RO Classes. In a given GDRO specification, each RO Class inherits from ITOP, either directly or indirectly. CROOT is the single root for RO Class containment, i.e. the root for instantiated RO Classes. PTOP is the single root for the inheritance tree of RO Policies. In a given GDRO specification, each RO Policy inherits from PTOP, either directly or indirectly.

The following conditions apply:

specStandardClasses-001
: the class names must differ, i.e. ITOP cannot be the same as PTOP or CROOT and so forth

specStandardClasses-002
: ITOP must be defined in a path declared for RO classes

specStandardClasses-003
: CROOT must be defined in a path declared for RO classes

specStandardClasses-004
: PTOP must be defined in a path declared for RO policies

The rule specStandardClasses is defined as follows:

```
specStandardClasses:
    AT_STANDARD_CLASSES classITop classCRoot classPTop
;
```

Required class Inheritance Top

GDRO requires a class starting the inheritance hierarchy. Commonly, this class is called Top. In this rule, one can define any RO class (with any name) as the inheritance hierarchy top level class.

The following conditions apply:

classITop-001
the class given as ITOP must be defined by the model

classITop-002
the ITOP class should be abstract, with no create/delete operation defined

classITop-003
the ITOP class should not define any policy, notification, or containment relationships

classITop-004
every RO class must inherit from ITOP, either directly or indirectly

classITop-005
ITOP does not inherit from any other RO class

The rule classITop is defined as follows:

```
classITop:
    ITOP eid_registration
;
```

Required class Containment Root

GDRO requires a class starting the containment hierarchy. Commonly, this class is called Root. In this rule, one can define any RO class (with any name) as the containment hierarchy top level class.

The following conditions apply:

classCRoot-001
the class given as CROOT must be defined by the model

classCRoot-002
the CROOT class should be abstract, with no create/delete operation defined

classCRoot-003
the CROOT class should not define any policy or notification

classCRoot-004
CROOT must inherit from ITOP, but not from any other RO class

The rule classCRoot is defined as follows:

```
classCRoot:
    CROOT eid_registration
;
```

Required class Policy Top

GDRO requires a class starting the inheritance hierarchy for RINA policies. There is no common name for this class. In this rule, one can define any RO class (with any name) as the inheritance hierarchy top level class for RINA policies.

The following conditions apply:

classPTop-001
the class given as PTOP must be defined by the model

classPTop-002
the PTOP class should be abstract

classPTop-003
the PTOP class should not inherit from any class but ITOP

classPTop-004
every RO policy must inherit from PTOP, either directly or indirectly

The rule classPTop is defined as follows:

```
classPTop:
    PTOP eid_registration
;
```

A.2. GDRO Definition

This is a definition, the core of any GDRO specification. The definition contains any number (0 or more) of the GDRO elements:

- roClass - a RIB Object class definition,
- roAttribute - a RIB Attribute definition,
- roNotification - a RIB Notification definition,
- roPolicy - a RIB Policy, and
- roTypeDefinition - a RIB Type Definition.

The order of the definitions is irrelevant. Forward declarations are allowed. For instance, we can define an RO Class using a particular RO Attribute before the RO Attribute is defined.

The following conditions apply:

gdroDefinition-001
A compiler must not assume any particular sequence for the non-terminals in the definition

gdroDefinition-002
A compiler must allow forward declarations, i.e. EID checks can only be done after the complete definition is parsed

The rule gdroDefinition is defined as follows:

```
gdroDefinition:
    (roClass | roAttribute | roNotification | roPolicy | roTypeDefinition)
;
```

A.2.1. Registered-as

Each RO element needs to be registered, effectively creating a tree of specified elements (classes, policies, notifications, etc.). A registration is realized by the name of the GDRO element plus plus an integer value. The name and the integer value must be unique in the given sub-path. This means that for instance in the path /a/b/c there can only be one GDRO class registered with the name 'd' and a previously not used integer value. The integer values can be used as substitution for the names, similar to GDMO and SMI.

A registration starts with EROOT followed by a path of registered elements and finally an integer value.

The rule registeredAs is defined as follows:

```
registeredAs:
    REGISTERED_AS eid_registration VAL_INTEGER
;
```

A.2.2. Documentation-text

This rule adds text that can be used to generate documentation for a GDRO language element (class, attribute, policy, notification, or type definition).

The rule documentationText is defined as follows:

```
documentationText:
    DOC_TEXT VAL_STRING+
;
```

A.3. RIB Object - RO

This is the definition of a RIB Object (RO). An RO has a mandatory identifier (ID) and a flexible class body. The class can be defined as abstract. An abstract class is part of the EID tree. It can be used in inheritance. However, abstract classes cannot be instantiated (at runtime) and thus not be used in a containment relationship.

The RO identifier is declared within the scope of the RO definition, only. The naming of the actual RO class is a combination of the ID plus the class registration.

The following conditions apply:

roClass-001
: the EID of an RO class (ID and registration) must be unique within the given GDRO model

The rule roClass is defined as follows:

```
roClass:
    ABSTRACT? RO RO_CLASS ID_F_UC roClassBody SEMICOLON
;
```

A.3.1. Class Body

This is the RO class body. It defines all elements of an RO:

- roClassBehavior - the mandatory behavior of the RO class (typically textual),
- roClassExtends - an optional inheritance relationship (other than to Top),
- roClassAttributes - optional RO attributes,
- roClassOperations - optional RO operations,
- roClassPolicies - optional policies used by the RO,
- roClassNtfc - optional notifications emitted by the RO,
- roClassContains - optional containment relationships defined by this RO, and
- registeredAs - the registration of this RO in the EID Tree

The following conditions apply:

roClassBody-001
if no roClassExtends is given, a compiler must assume inheritance from Top

roClassBody-002
if documentationText is used, it cannot be blank

The rule roClassBody is defined as follows:

```
roClassBody:
    documentationText? roClassBehavior roClassExtends? roClassAttributes?
    roClassOperations? roClassPolicies? roClassNtfc? roClassContains?
    registeredAs
;
```

A.3.2. Behavior

The behavior of the RO is expressed in natural language. This is done using the keyword BEHAVIOR (either 'behavior' or 'behaviour') and one or more STRING tokens. The strings inside the tokens are not further specified in this grammar. Instead, an GDRO compiler should define any markup or other formatting language for those strings.

The following conditions apply:

roClassBehavior-001
the behavior of an RO class cannot be blank

The rule roClassBehavior is defined as follows:

```
roClassBehavior:
    RO_BEHAVIOR VAL_STRING+
;
```

A.3.3. Inheritance Relationship (extends)

Every RO extends from ITOP (by definition, implicitly). ROs can inherit from any other RO (multiple inheritance).

If the RO inherits from ITOP only, no special inheritance definition is required.

The following conditions apply:

roClassExtends-001
each EID in an extends statement must point to a declared RO class

roClassExtends-002
each EID in the inheritance list should only be used once

roClassExtends-003
inherited inherit relationships should not be reused

The rule roClassExtends is defined as follows:

```
roClassExtends:
    EXTENDS eid_list
;
```

A.3.4. Attributes

An RO Class can have zero or more attributes. If this rule is used, at least one attribute needs to be declared. For RO Classes with zero attributes simply do not use the attribute rule.

The rule roClassAttributes is defined as follows:

```
roClassAttributes:
    ATTRIBUTES roClassAttribute+ SEMICOLON
;
```

RO Attribute

An attribute is defined by:

- an EID as a link to an attribute definition,
- an identifier to uniquely identify the attribute within the scope of the RO class,
- optional markers for permitted read and write operations, and
- an optional local description given in form of one or more strings.

The following conditions apply:

roClassAttribute-001
the EID it must point to an existing attribute definition

roClassAttribute-002
the attribute identifier must be unique within the scope of the RO

roClassAttribute-003
if no read and no write operation is given, a compiler must assume read and write being permitted (same as read and write being defined)

roClassAttribute-004
a given description overwrites the description of the original attribute definition

roClassAttribute-005
if a local description is used, it cannot be blank

roClassAttribute-006
all inherited attribute identifiers must not collide with local attribute identifiers

The rule roClassAttribute is defined as follows:

```
roClassAttribute:
```

```
    eid ID_F_LC OP_READ? OP_WRITE? (VAL_STRING+)?
;
```

A.3.5. Operations

ROs can have a number of standard CDAP operations defined. These operations are: create, delete, read, write, start, stop, and cancel-read. Each operation is defined for the RO class, i.e. not for individual RO class attributes. Each operation can only be declared once here, which means if it is defined it is applied to the entire RO.

The following conditions apply:

roClassOperations-001
: we should not assume any operation being available by default, i.e. every accepted operation must be defined explicitly

The rule roClassOperations is defined as follows:

```
roClassOperations:
    OPERATIONS opCreate? opDelete? opRead? opWrite? opStart? opStop?
    opCancelRead? SEMICOLON
;
```

Create

The create operation starts with the keyword CREATE ('create') and has the standard operation body.

The rule opCreate is defined as follows:

```
opCreate:
    OP_CREATE opBody
;
```

Delete

The delete operation starts with the keyword DELETE ('delete') and has the standard operation body.

The rule opDelete is defined as follows:

```
opDelete:
    OP_DELETE opBody
;
```

Read

The read operation starts with the keyword READ ('read') and has the standard operation body.

The rule opRead is defined as follows:

```
opRead:
    OP_READ opBody
;
```

Write

The write operation starts with the keyword WRITE ('write') and has the standard operation body.

The rule opWrite is defined as follows:

```
opWrite:
    OP_WRITE opBody
;
```

Start

The start operation starts with the keyword START ('start') and has the standard operation body.

The rule opStart is defined as follows:

```
opStart:
    OP_START opBody
;
```

Stop

The stop operation starts with the keyword STOP ('stop') and has the standard operation body.

The rule opStop is defined as follows:

```
opStop:
    OP_STOP opBody
;
```

Cancel-Read

The cancel-read operation starts with the keyword **CANCEL_READ** ('cancel-read') and has the standard operation body.

The rule opCancelRead is defined as follows:

```
opCancelRead:
    OP_CANCEL_READ opBody
;
```

Operation Body

This is the standard operation body, defined as:

- an optional flag for being protected,
- a description of one or more strings,
- zero or more input parameters, and
- zero or more output parameters.

The following conditions apply:

opBody-001
an operation marked as protected should only be called from within a RIB providing an application, e.g. the RIB DAEMON

opBody-002
the description of an RO operation cannot be blank

The rule opBody is defined as follows:

```
opBody:
    PROTECTED? VAL_STRING+ opParamIn* opParamOut*
;
```

Input Parameters

A standard parameter marked as an input parameter (for the operation).

The rule opParamIn is defined as follows:

```
opParamIn:
    IN opParam
;
```

Output Parameters

A standard parameter marked as an output parameter (for the operation).

The rule opParamOut is defined as follows:

```
opParamOut:
    OUT opParam
;
```

Standard Parameter

Each parameter of an operation (input our output) is defined by:

- a type,
- an identifier, and
- a description of one or more strings.

The following conditions apply:

opParam-001
: if the spec_type is an EID, it must be defined as a type, i.e. be a type definition

opParam-002
: the description of an RO operation parameter cannot be blank

The rule opParam is defined as follows:

```
opParam:
    spec_type ID_F_LC VAL_STRING+
;
```

A.3.6. Policies

An RO can have zero or more attached Policies.

The following conditions apply:

roClassPolicies-001
a policy identified by a EID must be defined as a policy

roClassPolicies-002
policies should only be referenced once (no policy used twice)

roClassPolicies-003
all inherited policies should only be referenced once (no policy used twice)

The rule roClassPolicies is defined as follows:

```
roClassPolicies:
    POLICIES eid_list SEMICOLON
;
```

A.3.7. Notifications

An RO can have zero or more attached RINA notifications.

The following conditions apply:

roClassNtfc-001
EIDs must point to defined notifications

roClassNtfc-002
notifications should only be referenced once (no policy used twice)

roClassNtfc-003
all inherited notifications should only be referenced once (no policy used twice)

The rule roClassNtfc is defined as follows:

```
roClassNtfc:
    NOTIFICATIONS eid_list SEMICOLON
;
```

A.3.8. Containment

An RO can have a containment relationship with zero or more other ROs.

The rule roClassContains is defined as follows:

```
roClassContains:
    CONTAINS containItem* SEMICOLON
;
```

Containment Item

An RO containment item is defined by

- an item EID,
- a create strategy, and
- a delete strategy.

The rule containItem is defined as follows:

```
containItem:
    containItemEid containItemStrategyCreate containItemStrategyDelete
;
```

Item EID The OID points to the class that is contained. There are then two options on how the containment is realize:

1. static using a particular string as identifier
2. dynamic, using a particular RO attribute as identifier

The following conditions apply:

containItemEid-001
: the EID must point to an existing RO class definition

containItemEid-002
: if the containment is realized as static, the given string cannot be blank

containItemEid-003
: if the containment is realized as dynamic, the given identifier must be an existing class attribute of the contained class

containItemEid-004
each containment EID should only be used once

containItemEid-005
inherited containments should not be reused (containment only used once)

The rule containItemEid is defined as follows:

```
containItemEid:
    eid ( (AS VAL_STRING) | (WITH_ATTRIBUTE ID_F_LC) )
;
```

Create Strategy The creation of containment items can be associated with particular strategies. The interpretation of the strategy is currently out of scope of the GDRO grammar.

The following conditions apply:

containItemStrategyCreate-001
the create strategy cannot be blank

The rule containItemStrategyCreate is defined as follows:

```
containItemStrategyCreate:
    CREATE_STRAT VAL_STRING
;
```

Delete Strategy The deletion of containment items can be associated with particular strategies. The interpretation of the strategy is currently out of scope of the GDRO grammar.

The following conditions apply:

containItemStrategyDelete-001
the delete strategy cannot be blank

The rule containItemStrategyDelete is defined as follows:

```
containItemStrategyDelete:
    DELETE_STRAT VAL_STRING
```

```
;
```

A.4. RIB Attribute Definition

A RIB attribute definition defines an attribute for use in ROs, Policies and other places. The attribute has a standard identifier (with upper case first character), a specification type, and a description. The definition must be finished with a semicolon;

The following conditions apply:

roAttribute-001
the EID of an RO attribute (ID and registration) must be unique within the given GDRO model

roAttribute-002
the description cannot be blank

roAttribute-003
if the type is an EID it must point to an existing type definition

roAttribute-004
if documentationText is used, it cannot be blank

The rule roAttribute is defined as follows:

```
roAttribute:
    RO RO_ATTR ID_F_UC documentationText? spec_type VAL_STRING+
    registeredAs SEMICOLON
;
```

A.5. RIB Notification Definition

This is the definition of a RIB Notification with a unique identifier, a mandatory behavior, an optional declaration for extending other notifications, attributes (here called notification objects), and a registration.

The following conditions apply:

roNotification-001
the EID of an RO notification (ID and registration) must be unique within the given GDRO model

roNotification-002
the behavior of a notification cannot be blank

roNotification-003
if documentationText is used, it cannot be blank

The rule roNotification is defined as follows:

```
roNotification:
    RO RO_NTFC ID_F_UC documentationText? RO_BEHAVIOR VAL_STRING+
    roNotificationExtends? ATTRIBUTES ntfcObject+ SEMICOLON registeredAs
    SEMICOLON
;
```

A.5.1. Inheritance Relationship (extends)

Notifications can inherit from any other notification (single inheritance).

The following conditions apply:

roNotificationExtends-001
the EID in an extends statement must point to a declared notification

roNotificationExtends-002
each EID in the inheritance list should only be used once

roNotificationExtends-003
inherited inherit relationships should not be reused

The rule roNotificationExtends is defined as follows:

```
roNotificationExtends:
    EXTENDS eid
;
```

A.5.2. Notification Object

h2: Notification Object A notification object is an attribute of a notification carrying information. It is defined by a type (which a specification type), an identifier and a textual description.

The following conditions apply:

ntfcObject-001
the notification object identifier must be unique within the notification definition

ntfcObject-002
the description of a notification object cannot be blank

ntfcObject-003
if the type is an EID it must point to an existing type definition

ntfcObject-004
all inherited object identifiers must not collide with local identifiers

The rule ntfcObject is defined as follows:

```
ntfcObject:
    spec_type ID_F_LC VAL_STRING+
;
```

A.6. RIB Policy Definition

This is the definition of a RIB Policy (otherwise known as a RINA policy) with: an identifier, and a body with all functional definitions for the policy.

The following conditions apply:

roPolicy-001
the EID of an RO policy (ID and registration) must be unique within the given GDRO model

The rule roPolicy is defined as follows:

```
roPolicy:
    ABSTRACT? RO RO_POLICY ID_F_UC roPolicyBody SEMICOLON
;
```

A.6.1. Policy Body

The body of a policy definition contains a mandatory behavior, optional inheritance and attribute definitions and a registration.

The following conditions apply:

roPolicyBody-001
if documentationText is used, it cannot be blank

The rule roPolicyBody is defined as follows:

```
roPolicyBody:
    documentationText? roPolicyBehavior roPolicyExtends?
    roPolicyAttributes? registeredAs
;
```

Behavior

The behavior of the policy is expressed in natural language. This is done using the keyword BEHAVIOR (either 'behavior' or 'behaviour') and one or more STRING tokens. The strings inside the tokens are not further specified in this grammar. Instead, a GDRO compiler should define any markup or other formatting language for those strings.

The following conditions apply:

roPolicyBehavior-001
the behavior of a policy cannot be blank

The rule roPolicyBehavior is defined as follows:

```
roPolicyBehavior:
    RO_BEHAVIOR VAL_STRING+
;
```

Inheritance Relationship (extends)

Every policy extends from PTOP (by definition, implicitly). Policies can inherit from any other policy (single inheritance).

If the policy inherits from PTOP only, no special inheritance definition is required.

The following conditions apply:

roPolicyExtends-001
the EID in an extends statement must point to a declared policy

roPolicyExtends-002
each EID in the inheritance list should only be used once

roPolicyExtends-003
inherited inherit relationships should not be reused

The rule roPolicyExtends is defined as follows:

```
roPolicyExtends:
    EXTENDS eid
;
```

A.6.2. Attributes

An RO Policy can have zero or more attributes. If this rule is used, at least one attribute needs to be declared. For RO Policies with zero attributes simply do not use the attribute rule.

The rule roPolicyAttributes is defined as follows:

```
roPolicyAttributes:
    ATTRIBUTES roPolicyAttribute+ SEMICOLON
;
```

Attribute An attribute is defined by:

- an EID as a link to an attribute definition,
- an identifier to uniquely identify the attribute within the scope of the RO class,
- an optional local description given in form of one or more strings.

The following conditions apply:

roPolicyAttribute-001
the EID it must point to an existing attribute definition

roPolicyAttribute-002
the attribute identifier must be unique within the scope of the policy

roPolicyAttribute-003
a given description overwrites the description of the original attribute definition

roPolicyAttribute-004
if a local description is used, it cannot be blank

roPolicyAttribute-005
all inherited attribute identifiers must not collide with local attribute identifiers

The rule roPolicyAttribute is defined as follows:

```
roPolicyAttribute:
    eid ID_F_LC (VAL_STRING+)?
;
```

A.7. RIB Type Definition

Type definitions define simple or complex type structures based on standard language types and/or other complex types. They can then be used for RO attributes, policies, and notifications.

The following conditions apply:

roTypeDefinition-001
the EID of an RO type definition (ID and registration) must be unique within the given GDRO model

roTypeDefinition-002
the description of a type definition cannot be blank

roTypeDefinition-003
if documentationText is used, it cannot be blank

The rule roTypeDefinition is defined as follows:

```
roTypeDefinition:
    RO RO_TYPEDEF ID_F_UC documentationText? VAL_STRING+
    roTypeDefinitionExtends? roType+ registeredAs SEMICOLON
;
```

A.7.1. Inheritance Relationship (extends)

Type definitions can inherit from any other type definition (single inheritance).

The following conditions apply:

roTypeDefinitionExtends-001
the EID in an extends statement must point to a declared type definition

roTypeDefinitionExtends-002
each EID in the inheritance list should only be used once

roTypeDefinitionExtends-003
inherited inherit relationships should not be reused

The rule roTypeDefinitionExtends is defined as follows:

```
roTypeDefinitionExtends:
    EXTENDS eid
;
```

A.7.2. Type member

A GDRO type member has an identifier (ID), a type, and a set of strings as non-formal description. The type itself can be either an EID to another (complex) type definition or one of the language base types.

The following conditions apply:

roType-001
the member identifier must be unique within the type definition

roType-002
the description of the member cannot be blank

roType-003
if the type is an EID it must point to another existing type definition

roType-004
all inherited type identifiers must not collide with local identifiers

The rule roType is defined as follows:

```
roType:
    spec_type ID_F_LC VAL_STRING+
;
```

A.8. Keywords

The GDRO grammar defines a number of keywords. Each keyword is defined by a lexer token (all upper case) followed by the keyword token. The example below defines a keyword DUMMY_KEYWORD with the lexer token #dummy-keyword.

The rule DUMMY_KEYWORD is defined as follows:

```
DUMMY_KEYWORD:
    '#dummy-keyword'
;
```

A.8.1. @date

Keyword preceding the date of a GDRO specification.

The rule AT_DATE is defined as follows:

```
AT_DATE:
    '@date'
;
```

A.8.2. @standard-classes

Keyword preceding the standard class definitions of a GDRO specification.

The rule AT_STANDARD_CLASSES is defined as follows:

```
AT_STANDARD_CLASSES:
    '@standard-classes'
;
```

A.8.3. @path-registrations

Keyword preceding the declaration of element paths of a GDRO specification.

The rule AT_PATH_REGISTRATIONS is defined as follows:

```
AT_PATH_REGISTRATIONS:
```

```
    '@path-registrations'
;
```

A.8.4. @version

Keyword preceding the version of a GDRO specification.

The rule AT_VERSION is defined as follows:

```
AT_VERSION:
    '@version'
;
```

A.8.5. abstract

Keyword preceding an abstract RO class.

The rule ABSTRACT is defined as follows:

```
ABSTRACT:
    'abstract'
;
```

A.8.6. as

Keyword for defining an RO containment item created as something.

The rule AS is defined as follows:

```
AS:
    'as'
;
```

A.8.7. attribute

Keyword preceding the GDRO element attribute.

The rule RO_ATTR is defined as follows:

```
RO_ATTR:
    'attribute'
```

```
;
```

A.8.8. attributes

Keyword for defining specification attribute paths or RO Attributes.

The rule ATTRIBUTES is defined as follows:

```
ATTRIBUTES:
    'attributes'
;
```

A.8.9. behavior (or behaviour)

Keyword preceding the behavior of an RO class.

The rule RO_BEHAVIOR is defined as follows:

```
RO_BEHAVIOR:
    'behavior' | 'behaviour'
;
```

A.8.10. cancel-read

Keyword for defining a 'cancel-read' operation.

The rule OP_CANCEL_READ is defined as follows:

```
OP_CANCEL_READ:
    'cancel-read'
;
```

A.8.11. Documentation-text

Text for documentation of a GDRO element (class, attribute, policy, notification, or type definition).

The rule DOC_TEXT is defined as follows:

```
DOC_TEXT:
```

```
    'documentation-text'
;
```

A.8.12. ERoot

The standard name of the root of a GDRO element tree.

The rule EROOT is defined as follows:

```
EROOT:
    'ERoot'
;
```

A.8.13. class

Keyword preceding the GDRO element class (RO Class).

The rule RO_CLASS is defined as follows:

```
RO_CLASS:
    'class'
;
```

A.8.14. classes

Keyword for defining specification classes EID paths.

The rule CLASSES is defined as follows:

```
CLASSES:
    'classes'
;
```

A.8.15. contains

Keyword for defining an RO containment relationship.

The rule CONTAINS is defined as follows:

```
CONTAINS:
```

```
    'contains'
;
```

A.8.16. create

Keyword for defining a 'create' operation.

The rule OP_CREATE is defined as follows:

```
OP_CREATE:
    'create'
;
```

A.8.17. create-strategy

Keyword for the create-strategy of an RO containment item.

The rule CREATE_STRAT is defined as follows:

```
CREATE_STRAT:
    'create-strategy'
;
```

A.8.18. CRoot

Keyword used in the containment root path declaration 'classCRoot'.

The rule CROOT is defined as follows:

```
CROOT:
    'CRoot'
;
```

A.8.19. delete

Keyword for defining a 'delete' operation.

The rule OP_DELETE is defined as follows:

```
OP_DELETE:
```

```
    'delete'
;
```

A.8.20. delete-strategy

Keyword for the delete-strategy of an RO containment item.

The rule DELETE_STRAT is defined as follows:

```
DELETE_STRAT:
    'delete-strategy'
;
```

A.8.21. extends

Keyword preceding an RO Class or an RO Policy extend definition.

The rule EXTENDS is defined as follows:

```
EXTENDS:
    'extends'
;
```

A.8.22. in

Keyword for defining the 'in' parameter of an operation.

The rule IN is defined as follows:

```
IN:
    'in'
;
```

A.8.23. ITop

Keyword used in the inheritance path declaration 'classITop'.

The rule ITOP is defined as follows:

```
ITOP:
```

```
    'ITop'
;
```

A.8.24. notification

Keyword preceding an RO Notification.

The rule RO_NTFC is defined as follows:

```
RO_NTFC:
    'notification'
;
```

A.8.25. notifications

Keyword for defining specification notification paths or RO Notifications.

The rule NOTIFICATIONS is defined as follows:

```
NOTIFICATIONS:
    'notifications'
;
```

A.8.26. operations

Keyword for defining RO class operations.

The rule OPERATIONS is defined as follows:

```
OPERATIONS:
    'operations'
;
```

A.8.27. out

Keyword for defining the 'out' parameter of an operation.

The rule OUT is defined as follows:

```
OUT:
```

```
    'out'
;
```

A.8.28. policy

Keyword preceding an RO Policy.

The rule RO_POLICY is defined as follows:

```
RO_POLICY:
    'policy'
;
```

A.8.29. policies

Keyword for defining specification policy paths or RO Policies.

The rule POLICIES is defined as follows:

```
POLICIES:
    'policies'
;
```

A.8.30. protected

Keyword for a protected RO Class.

The rule PROTECTED is defined as follows:

```
PROTECTED:
    'protected'
;
```

A.8.31. PTop

Keyword used in the policy inheritance path declaration 'classPTop'.

The rule PTOP is defined as follows:

```
PTOP:
    'PTop'
```

```
;
```

A.8.32. read

Keyword for defining a 'read' operation.

The rule OP_READ is defined as follows:

```
OP_READ:
    'read'
;
```

A.8.33. registered-as

Keyword for defining the registration of an GDRO element.

The rule REGISTERED_AS is defined as follows:

```
REGISTERED_AS:
    'registered-as'
;
```

A.8.34. ro

Keyword preceding every GDRO element (class, attribute, notification, policy, and type definition).

The rule RO is defined as follows:

```
RO:
    'ro'
;
```

A.8.35. start

Keyword for defining a 'start' operation.

The rule OP_START is defined as follows:

```
OP_START:
```

```
    'start'
;
```

A.8.36. stop

Keyword for defining a 'stop' operation.

The rule OP_STOP is defined as follows:

```
OP_STOP:
    'stop'
;
```

A.8.37. type definition

Keyword preceding an RO Type Definition.

The rule RO_TYPEDEF is defined as follows:

```
RO_TYPEDEF:
    'type definition'
;
```

A.8.38. type definitions

Keyword for defining specification type definition EID paths.

The rule TYPE_DEFINTIONS is defined as follows:

```
TYPE_DEFINTIONS:
    'type definitions'
;
```

A.8.39. with-attribute

Keyword for defining an RO containment item created as with-attribute.

The rule WITH_ATTRIBUTE is defined as follows:

```
WITH_ATTRIBUTE:
```

```
    'with-attribute'
;
```

A.8.40. write

Keyword for defining a 'write' operation.

The rule OP_WRITE is defined as follows:

```
OP_WRITE:
    'write'
;
```

A.9. Types

The GDRO grammar defines a number of grammar type definitions. Each grammar type definitions is defined by a token followed by a rule definition or lexer token (all upper case) followed by the keyword token. The example below defines a keyword DUMMY_TYPE with the lexer token #dummy-type.

The rule DUMMY_TYPE is defined as follows:

```
DUMMY_TYPE:
    '#dummy-type'
;
```

A.9.1. spec_type

A type of EID or base type.

The rule spec_type is defined as follows:

```
spec_type:
    base_type | eid
;
```

A.9.2. base_type

A base type (constant or sequence-of or set-of).

The rule base_type is defined as follows:

```
base_type:
    const_type | seq_of_type | set_of_type
;
```

seq_of_type

A special base type that marks a sequence of either a constant type or a type definition.

The following conditions apply:

seq_of_type-001
if an EID is used as type it must point to an existing type definition

The rule seq_of_type is defined as follows:

```
seq_of_type:
    T_SEQUENCE_OF (const_type | eid)
;
```

setq_of_type

A special base type that marks a set of either a constant type or a type definition.

The following conditions apply:

set_of_type-001
if an EID is used it must point to an existing type definition

The rule set_of_type is defined as follows:

```
set_of_type:
    T_SET_OF (const_type | eid)
;
```

const_type

Keyword for a constant type, i.e. a build-in type.

The rule const_type is defined as follows:

```
const_type:
    T_BOOLEAN | T_CHAR | T_DOUBLE | T_FLOAT | T_INT | T_LONG | T_STRING |
    T_SEQUENCE | T_SET | T_CHOICE
;
```

A.9.3. eid

Type for an element identifier (EID). It can, optionally, start with the keyword EROOT, followed by one or more elements.

The rule eid is defined as follows:

```
eid:
    EROOT? eid_element+
;
```

A.9.4. eid_element

This is an element (part) of a EID. Each element starts with a path separator. This separator can be a dot, a slash, a colon, or a greater than character. This is followed by either an identifier (first character must be an upper case character) or an integer value.

The rule eid_element is defined as follows:

```
eid_element:
    (DOT | SLASH | COLON | GT) (ID_F_UC | VAL_INTEGER)
;
```

A.9.5. eid_list

Type for a list of Class Identifiers.

The rule eid_list is defined as follows:

```
eid_list:
    eid (COMMA eid)*
;
```

A.9.6. eid_registration

Type for the registration of a EID or EID path. It can, optionally, start with the keyword EROOT, followed by one or more elements.

The rule eid_registration is defined as follows:

```
eid_registration:
    EROOT? eid_registration_element+
;
```

A.9.7. eid_registration_list

A list of eid registrations

The rule eid_registration_list is defined as follows:

```
eid_registration_list:
    eid_registration (COMMA eid_registration)*
;
```

A.9.8. eid_registration_element

This is an element (part) of a EID registration. Each element starts with an identifier (first character must be an upper case character). This identifier is the text variant of the element. This is followed by an integer value in brackets. This integer value is the numeric variant of the element.

The rule eid_registration_element is defined as follows:

```
eid_registration_element:
    ID_F_UC BL VAL_INTEGER BR
;
```

A.9.9. T_Boolean

Keyword T_Boolean for a boolean type.

The rule T_BOOLEAN is defined as follows:

```
T_BOOLEAN:
```

```
    'T_Boolean'
;
```

A.9.10. T_CHAR

Keyword T_CHAR for a character type.

The rule T_CHAR is defined as follows:

```
T_CHAR:
    'T_Char'
;
```

A.9.11. T_DOUBLE

Keyword T_DOUBLE for a double type.

The rule T_DOUBLE is defined as follows:

```
T_DOUBLE:
    'T_Double'
;
```

A.9.12. T_FLOAT

Keyword T_FLOAT for a float type.

The rule T_FLOAT is defined as follows:

```
T_FLOAT:
    'T_Float'
;
```

A.9.13. T_INT

Keyword T_INT for an integer type.

The rule T_INT is defined as follows:

```
T_INT:
```

```
    'T_Int'
;
```

A.9.14. T_LONG

Keyword T_LONG for a long type.

The rule T_LONG is defined as follows:

```
T_LONG:
    'T_Long'
;
```

A.9.15. T_STRING

Keyword T_STRING for a string type.

The rule T_STRING is defined as follows:

```
T_STRING:
    'T_String'
;
```

A.9.16. T_SEQUENCE

Keyword T_SEQUENCE for a not further qualified sequence.

The rule T_SEQUENCE is defined as follows:

```
T_SEQUENCE:
    'T_Sequence'
;
```

A.9.17. T_SEQUENCE_OF

Keyword T_SEQUENCE_OF for a sequence of a particular type.

The rule T_SEQUENCE_OF is defined as follows:

```
T_SEQUENCE_OF:
```

```
    'T_SequenceOf'
;
```

A.9.18. T_SET

Keyword T_SET marking a not further qualified set.

The rule T_SET is defined as follows:

```
T_SET:
    'T_Set'
;
```

A.9.19. T_SET_OF

Keyword T_SET_OF for a set of a particular type.

The rule T_SET_OF is defined as follows:

```
T_SET_OF:
    'T_SetOf'
;
```

A.9.20. T_CHOICE

Keyword T_CHOICE for a choice type.

The rule T_CHOICE is defined as follows:

```
T_CHOICE:
    'T_Choice'
;
```

A.9.21. VAL_INTEGER

Value of an integer.

The rule VAL_INTEGER is defined as follows:

```
VAL_INTEGER:
    [0-9]+
```

```
;
```

A.9.22. VAL_STRING

Value of a string, any character surrounded by double quotes.

The rule VAL_STRING is defined as follows:

```
VAL_STRING:
    '"' (ESC | ~["\\])* '"'
;
```

Escape fragment of a String

The escape code for a string value or the Unicode escape code.

The rule fragment ESC is defined as follows:

```
fragment ESC:
    '\\' (["\\/bfnrt] | UNICODE)
;
```

Unicode Escape fragment of a String

The escape for the Unicode encodings.

The rule fragment UNICODE is defined as follows:

```
fragment UNICODE:
    'u' HEX HEX HEX HEX
;
```

Hexadecimal codes for a Unicode Escape

The hexadecimal codes for a Unicode Escape.

The rule fragment HEX is defined as follows:

```
fragment HEX:
    [0-9a-fA-F]
;
```

A.10. Lexer

The GDRO grammar defines a number of lexer definitions. Each lexer definitions is defined by a lexer token (all upper case) followed by the keyword token. The example below defines a keyword DUMMY_LEXER with the lexer token #dummy-lexer.

The rule DUMMY_LEXER is defined as follows:

```
DUMMY_LEXER:
    '#dummy-lexer'
;
```

A.10.1. Character bracket left '('

Defines the character '(' for any rule.

The rule BL is defined as follows:

```
BL:
    '('
;
```

A.10.2. Character bracket right ')'

Defines the character ')' for any rule.

The rule BR is defined as follows:

```
BR:
    ')'
;
```

A.10.3. Character colon ':'

Defines the character ':' for any rule.

The rule COLON is defined as follows:

```
COLON:
```

```
    ':'
;
```

A.10.4. Character comma ','

Defines the character ',' for any rule.

The rule COMMA is defined as follows:

```
COMMA:
    ','
;
```

A.10.5. Character dot '.'

Defines the character '.' for any rule.

The rule DOT is defined as follows:

```
DOT:
    '.'
;
```

A.10.6. Character greater than '>'

Defines the character '>' for any rule.

The rule GT is defined as follows:

```
GT:
    '>'
;
```

A.10.7. Character semicolon ';'

Defines the character ';' for any rule.

The rule SEMICOLON is defined as follows:

```
SEMICOLON:
    ''
```

```
;
```

A.10.8. Character slash '/'

Defines the character '/' for any rule.

The rule SLASH is defined as follows:

```
SLASH:
    '/'
;
```

A.10.9. Identifiers starting with an upper case character

Defines an identifier starting with an upper case character.

The rule ID_F_UC is defined as follows:

```
ID_F_UC:
    [A-Z] ('_' | [A-Z] | [a-z] | [0-9])*
;
```

A.10.10. Identifiers starting with a lower case character

Defines an identifier starting with a lower case character.

The rule ID_F_LC is defined as follows:

```
ID_F_LC:
    [a-z] ('_' | [A-Z] | [a-z] | [0-9])*
;
```

A.10.11. Whitespace

Defines a whitespace (blank, tabulator, line feed, carriage return). All whitespaces are skipped, i.e. ignored

The rule WS is defined as follows:

```
WS:
    [ \t\n\r]+ -> skip
```

```
;
```

A.10.12. Single Line Comment

Defines a single line comment. All single line comments are skipped, i.e. ignored

The rule SL_COMMENT is defined as follows:

```
SL_COMMENT:
    ('//' | '--') .*? '\r'? '\n' -> skip
;
```

A.10.13. Multi Line Comment

Defines a multi line comment. All single line comments are skipped, i.e. ignored

The rule ML_COMMENT is defined as follows:

```
ML_COMMENT:
    '/*' .*? '*/' -> skip
;
```

B. Appendix: RIB object specification

B.1. Specification

GDRO Specification for the PRISTINE RIB Model Defined:

- date: "25-Apr-2016" and
- version: "0.0.2".

Specification path declarations:

- classes: Classes(1), Ericsson(99) Classes(1)
- attributes: Attributes(2), Ericsson(99) Attributes(2)
- notifications: Notifications(3)
- policies: Policies(4)
- type definitions: Types(5)

Specification standard classes:

- ITOP (class inheritance top): Classes(1) Top(1)
- CROOT (containment root): Classes(1) Root(2)
- PTOP (policy inheritance top): Policies(4) RINAPolicy(1)

B.2. RO Classes

B.2.1. Class IPCManagement

Behavior

This class represents an IPC Management component, managing the use of N-1 flows

Dependencies

- ERoot.Classes.IPCResourceManager
- ERoot.Classes.SDUProtection

Class Containment Relationships

- ERoot.Classes.IPCResourceManager
  - as "irm"
  - Create: This object is automatically created on IPCManagement creation.
  - Delete: This object is automatically deleted on IPCManagement destruction.
- ERoot.Classes.SDUProtection
  - as "sdup"
  - Create: This object is automatically created on IPCManagement creation.
  - Delete: This object is automatically deleted on IPCManagement destruction.

Registered As

ERoot Classes(1) 34

Specification

```
ro class IPCManagement
    behavior
        "This class represents an IPC Management component, managing the
 use of N-1 flows"

    contains
        ERoot.Classes.IPCResourceManager (as "irm")
            create-strategy "This object is automatically created on
 IPCManagement creation."
            delete-strategy "This object is automatically deleted on
 IPCManagement destruction."
        ERoot.Classes.SDUProtection (as "sdup")
            create-strategy "This object is automatically created on
 IPCManagement creation."
            delete-strategy "This object is automatically deleted on
 IPCManagement destruction."
    ;
```

```
    registered-as ERoot Classes(1) 34
;
```

B.2.2. Class ComputingSystem

Behavior

This class cannot be remotely created or deleted, since it represents the root of the containment subtree of a particular computing system. It is the root object of the Management Agent's RIB.

Dependencies

- ERoot.Attributes.ComputingSystemId
- ERoot.Classes.ProcessingSystem

Class Attributes

- computingSystemId → uniquely identifies the computing system within the Management Domain
  - defined by ERoot.Attributes.ComputingSystemId
  - read enabled
  - write enabled

Class Containment Relationships

- ERoot.Classes.ProcessingSystem
  - with attribute processingSystemId
  - Create: This object is automatically created on RIB creation.
  - Delete: This object cannot be deleted

Registered As

ERoot Classes(1) 3

Specification

```
ro class ComputingSystem
    behavior
        "This class cannot be remotely created or deleted, since it"
```

```
        " represents the root of the containment subtree of a particular"
        " computing system. It is the root object of the Management
Agent's RIB."

    attributes
        ERoot.Attributes.ComputingSystemId computingSystemId read write
          "uniquely identifies the computing system within the Management
Domain"
    ;

    contains
        ERoot.Classes.ProcessingSystem (with attribute processingSystemId)
            create-strategy "This object is automatically created on RIB
creation."
            delete-strategy "This object cannot be deleted"
    ;


    registered-as ERoot Classes(1) 3
;
```

B.2.3. Class SDUProtectionPolicySet

Behavior

This class represents a SDU Protection policy set.

Dependencies

- ERoot.Attributes.PortId
- ERoot.Policies.Protection.CompressionPolicy
- ERoot.Policies.Protection.ErrorCheckPolicy
- ERoot.Policies.Protection.LifetimeLimitingPolicy
- ERoot.Policies.Security.CryptographicProtectionPolicy

Class Attributes

- portId → The port-id of the N-1 flow associated to this SDU Protection policy set
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.Security.CryptographicProtectionPolicy
- ERoot.Policies.Protection.ErrorCheckPolicy
- ERoot.Policies.Protection.LifetimeLimitingPolicy
- ERoot.Policies.Protection.CompressionPolicy

Registered As

ERoot Classes(1) 43

Specification

```
ro class SDUProtectionPolicySet
    behavior
        "This class represents a SDU Protection policy set."

    attributes
        ERoot.Attributes.PortId portId read write
          "The port-id of the N-1 flow associated to this SDU Protection
 policy set"
    ;

    policies
        ERoot.Policies.Security.CryptographicProtectionPolicy
        ERoot.Policies.Protection.ErrorCheckPolicy
        ERoot.Policies.Protection.LifetimeLimitingPolicy
        ERoot.Policies.Protection.CompressionPolicy
    ;

    registered-as ERoot Classes(1) 43
;
```

B.2.4. Class PDUForwardingTable

Behavior

This class is the container for individual PDUForwardingTableEntry objects

Dependencies

- ERoot.Classes.PDUForwardingTableEntry

Class Containment Relationships

- ERoot.Classes.PDUForwardingTableEntry
  - with attribute key
  - Create: This object is created when a new entry is added to the PDU Forwarding Table
  - Delete: This object is removed when an entry is removed from the PDU Forwarding Table

Registered As

ERoot Classes(1) 17

Specification

```
ro class PDUForwardingTable
    behavior
        "This class is the container for individual
 PDUForwardingTableEntry objects"

    contains
        ERoot.Classes.PDUForwardingTableEntry (with attribute key)
            create-strategy "This object is created when a new entry is
 added to the PDU Forwarding Table"
            delete-strategy "This object is removed when an entry is
 removed from the PDU Forwarding Table"
    ;


    registered-as ERoot Classes(1) 17
;
```

B.2.5. Class SecurityManager

Behavior

This class represents a Security Manager Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName

- ERoot.Classes.ApplicationEntity
- ERoot.Policies.Security.AccessControlPolicy
- ERoot.Policies.Security.AuditingPolicy
- ERoot.Policies.Security.CredentialManagementPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Policies

- ERoot.Policies.Security.AccessControlPolicy
- ERoot.Policies.Security.AuditingPolicy
- ERoot.Policies.Security.CredentialManagementPolicy

Registered As

ERoot Classes(1) 23

Specification

```
ro class SecurityManager
    behavior
        "This class represents a Security Manager Application Entity."

    extends ERoot.Classes.ApplicationEntity

    policies
        ERoot.Policies.Security.AccessControlPolicy
        ERoot.Policies.Security.AuditingPolicy
        ERoot.Policies.Security.CredentialManagementPolicy
    ;

    registered-as ERoot Classes(1) 23
;
```

B.2.6. Class UnderlyingRegistration

Behavior

A registration of the App/IPCP to the N-1 DIF

Dependencies

- ERoot.Attributes.APNamingInfo
- ERoot.Attributes.DIFNameList
- ERoot.Notifications.CreateUnderlyingRegistration
- ERoot.Notifications.DeleteUnderlyingRegistration

Class Attributes

- applicationEntity → The naming information of this Application Entity
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled
- difNames → The names of the DIFs where this Application Entity is registered
  - defined by ERoot.Attributes.DIFNameList
  - read enabled
  - write enabled

Class Operations

- delete
  - invoked to cancel a registration to an N-1 DIF
  - in T_String difName
    - Unregister the AE from an N-1 DIF

Class Notifications

- ERoot.Notifications.CreateUnderlyingRegistration
- ERoot.Notifications.DeleteUnderlyingRegistration

Registered As

ERoot Classes(1) 39

Specification

```
ro class UnderlyingRegistration
    behavior
        "A registration of the App/IPCP to the N-1 DIF"

    attributes
        ERoot.Attributes.APNamingInfo applicationEntity read write
          "The naming information of this Application Entity"
        ERoot.Attributes.DIFNameList difNames read write
          "The names of the DIFs where this Application Entity is
 registered"
    ;

    operations
        delete
            "invoked to cancel a registration to an N-1 DIF"
            in T_String difName
                "Unregister the AE from an N-1 DIF"

    ;

    notifications
        ERoot.Notifications.CreateUnderlyingRegistration
        ERoot.Notifications.DeleteUnderlyingRegistration
    ;

    registered-as ERoot Classes(1) 39
;
```

B.2.7. Class ResourceAllocator

Behavior

This class represents a Resource Allocator Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.NextHopTable
- ERoot.Classes.PDUForwardingTable

- ERoot.Classes.QoSCubes
- ERoot.Policies.ResourceAllocation.PDUFTGenerationPolicy
- ERoot.Policies.ResourceAllocation.ResourceAllocatorPolicy
- ERoot.Policies.ResourceAllocation.RoutingPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Policies

- ERoot.Policies.ResourceAllocation.ResourceAllocatorPolicy
- ERoot.Policies.ResourceAllocation.RoutingPolicy
- ERoot.Policies.ResourceAllocation.PDUFTGenerationPolicy

Class Containment Relationships

- ERoot.Classes.NextHopTable
  - as "nhopt"
  - Create: This object is automatically created on ResourceAllocator creation.
  - Delete: This object is automatically deleted on ResourceAllocator destruction.
- ERoot.Classes.PDUForwardingTable
  - as "pduft"
  - Create: This object is automatically created on ResourceAllocator creation.
  - Delete: This object is automatically deleted on ResourceAllocator destruction.
- ERoot.Classes.QoSCubes
  - as "qoscubes"
  - Create: This object is automatically created on ResourceAllocator creation.
  - Delete: This object is automatically deleted on ResourceAllocator destruction.

Registered As

ERoot Classes(1) 15

Specification

```
ro class ResourceAllocator
    behavior
        "This class represents a Resource Allocator Application Entity."

    extends ERoot.Classes.ApplicationEntity

    policies
        ERoot.Policies.ResourceAllocation.ResourceAllocatorPolicy
        ERoot.Policies.ResourceAllocation.RoutingPolicy
        ERoot.Policies.ResourceAllocation.PDUFTGenerationPolicy
    ;

    contains
        ERoot.Classes.NextHopTable (as "nhopt")
            create-strategy "This object is automatically created on
 ResourceAllocator creation."
            delete-strategy "This object is automatically deleted on
 ResourceAllocator destruction."
        ERoot.Classes.PDUForwardingTable (as "pduft")
            create-strategy "This object is automatically created on
 ResourceAllocator creation."
            delete-strategy "This object is automatically deleted on
 ResourceAllocator destruction."
        ERoot.Classes.QoSCubes (as "qoscubes")
            create-strategy "This object is automatically created on
 ResourceAllocator creation."
            delete-strategy "This object is automatically deleted on
 ResourceAllocator destruction."
    ;


    registered-as ERoot Classes(1) 15
;
```

B.2.8. Class ApplicationConnections

Behavior

This class is the container for individual ApplicationConnection objects

Dependencies

- ERoot.Classes.ApplicationConnection

Class Containment Relationships

- ERoot.Classes.ApplicationConnection
  - with attribute portId
  - Create: This object is created when an application connection to a peer App/IPCP is established
  - Delete: This object is removed when an application connection to a peer App/IPCP is released

Registered As

ERoot Classes(1) 45

Specification

```
ro class ApplicationConnections
    behavior
        "This class is the container for individual ApplicationConnection
 objects"

    contains
        ERoot.Classes.ApplicationConnection (with attribute portId)
            create-strategy "This object is created when an application
 connection to a peer App/IPCP is established"
            delete-strategy "This object is removed when an application
 connection to a peer App/IPCP is released"
    ;


    registered-as ERoot Classes(1) 45
;
```

B.2.9. Class ProcessingSystem

Behavior

Models the hardware and software capable of executing programs instantiated as Application Processes that can coordinate with the

equivalent of a “test and set” instruction, i.e. the tasks can all atomically reference the same memory.

Dependencies

- ERoot.Attributes.ProcessingSystemId
- ERoot.Classes.Hardware
- ERoot.Classes.KernelApplicationProcess
- ERoot.Classes.Software

Class Attributes

- processingSystemId → uniquely identifies the Processing system within the Computing System
  - defined by ERoot.Attributes.ProcessingSystemId
  - read enabled
  - write enabled

Class Containment Relationships

- ERoot.Classes.Software
  - as "software"
  - Create: This object is automatically created on ProcessingSystem creation.
  - Delete: This object is automatically deleted on ProcessingSystem destruction
- ERoot.Classes.Hardware
  - as "hardware"
  - Create: This object is automatically created on ProcessingSystem creation.
  - Delete: This object is automatically deleted on ProcessingSystem destruction
- ERoot.Classes.KernelApplicationProcess
  - as "kernelApplicationProcess"
  - Create: This object is automatically created on ProcessingSystem creation.

- Delete: This object is automatically deleted on ProcessingSystem destruction

Registered As

ERoot Classes(1) 6

Specification

```
ro class ProcessingSystem
    behavior
        "Models the hardware and software capable of executing programs"
        " instantiated as Application Processes that can coordinate with"
        " the equivalent of a “test and set” instruction, i.e. the tasks"
        " can all atomically reference the same memory."

    attributes
        ERoot.Attributes.ProcessingSystemId processingSystemId read write
          "uniquely identifies the Processing system within the Computing
 System"
    ;

    contains
        ERoot.Classes.Software (as "software")
            create-strategy "This object is automatically created on
 ProcessingSystem creation."
            delete-strategy "This object is automatically deleted on
 ProcessingSystem destruction"
        ERoot.Classes.Hardware (as "hardware")
            create-strategy "This object is automatically created on
 ProcessingSystem creation."
            delete-strategy "This object is automatically deleted on
 ProcessingSystem destruction"
        ERoot.Classes.KernelApplicationProcess (as
 "kernelApplicationProcess")
            create-strategy "This object is automatically created on
 ProcessingSystem creation."
            delete-strategy "This object is automatically deleted on
 ProcessingSystem destruction"
    ;


    registered-as ERoot Classes(1) 6
;
```

B.2.10. Class IPCProcess

Behavior

This class represents a running instance of an IPCProcess.

Dependencies

- ERoot.Attributes.DAPInstance
- ERoot.Attributes.DAPName
- ERoot.Attributes.IpcProcessId
- ERoot.Attributes.SynonymList
- ERoot.Classes.ApplicationProcess
- ERoot.Classes.DIFManagement
- ERoot.Classes.DataTransfer
- ERoot.Classes.FlowAllocator
- ERoot.Classes.IPCManagement
- ERoot.Classes.RIBDaemon
- ERoot.Classes.RelayingAndMultiplexing
- ERoot.Classes.ResourceAllocator
- ERoot.Classes.SDUDelimiting
- ERoot.Notifications.CreateIPCProcess

Super Classes

- ERoot.Classes.ApplicationProcess

Class Attributes

- processId → uniquely identifies the IPC Process within the processing system
  - defined by ERoot.Attributes.IpcProcessId
  - read enabled
  - write enabled

Class Operations

- create

- invoked when someone requests the Management Agent to instantiate a new IPC process
- in ERoot.Types.T_IPCPConfig instantiateDifData
  - The configuration data to instantiate an IPC Process in a processing sytem
- out T_Int processId
  - The id of the IPC Process

- delete
  - invoked when someone requests the Management Agent to terminate a running IPC process
  - in T_Boolean hardDelete
    - If true the IPC Process must be killed right away
- read
  - read IPC Process naming information
  - out ERoot.Types.T_IPCPInfo ipcpInfo
    - The IPC Process naming information (naming, configuration)
- cancel-read
  - cancel ongoing read operation

Class Notifications

- ERoot.Notifications.CreateIPCProcess

Class Containment Relationships

- ERoot.Classes.DataTransfer
  - as "dt"
  - Create: This object is automatically created on IPCProcess creation.
  - Delete: This object is automatically deleted on IPCProcess destruction.
- ERoot.Classes.DIFManagement
  - as "difmanagement"

- Create: This object is automatically created on IPCProcess creation.
- Delete: This object is automatically deleted on IPCProcess destruction.

- ERoot.Classes.FlowAllocator
  - as "fa"
  - Create: This object is automatically created on IPCProcess creation.
  - Delete: This object is automatically deleted on IPCProcess destruction.
- ERoot.Classes.IPCManagement
  - as "ipcmanagement"
  - Create: This object is automatically created on IPCProcess creation.
  - Delete: This object is automatically deleted on IPCProcess destruction.
- ERoot.Classes.RelayingAndMultiplexing
  - as "rmt"
  - Create: This object is automatically created on IPCProcess creation.
  - Delete: This object is automatically deleted on IPCProcess destruction.
- ERoot.Classes.RIBDaemon
  - as "ribdaemon"
  - Create: This object is automatically created on IPCProcess creation.
  - Delete: This object is automatically deleted on IPCProcess destruction.
- ERoot.Classes.ResourceAllocator
  - as "resalloc"
  - Create: This object is automatically created on IPCProcess creation.
  - Delete: This object is automatically deleted on IPCProcess destruction.
- ERoot.Classes.SDUDelimiting
  - as "sdudel"
  - Create: This object is automatically created on IPCProcess creation.

- Delete: This object is automatically deleted on IPCProcess destruction.

Registered As

ERoot Classes(1) 12

Specification

```
ro class IPCProcess
    behavior
        "This class represents a running instance of an IPCProcess."

    extends ERoot.Classes.ApplicationProcess

    attributes
        ERoot.Attributes.IpcProcessId processId read write
          "uniquely identifies the IPC Process within the processing
 system"
    ;

    operations
        create
            "invoked when someone requests the Management Agent to
 instantiate a new IPC process"
            in ERoot.Types.T_IPCPConfig instantiateDifData
                "The configuration data to instantiate an IPC Process in a
 processing sytem"
            out T_Int processId
                "The id of the IPC Process"

        delete
            "invoked when someone requests the Management Agent to
 terminate a running IPC process"
            in T_Boolean hardDelete
                "If true the IPC Process must be killed right away"

        read
            "read IPC Process naming information"
            out ERoot.Types.T_IPCPInfo ipcpInfo
                "The IPC Process naming information (naming,
 configuration)"

        cancel-read
            "cancel ongoing read operation"
```

```
    ;

    notifications
        ERoot.Notifications.CreateIPCProcess
    ;

    contains
        ERoot.Classes.DataTransfer (as "dt")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.DIFManagement (as "difmanagement")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.FlowAllocator (as "fa")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.IPCManagement (as "ipcmanagement")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.RelayingAndMultiplexing (as "rmt")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.RIBDaemon (as "ribdaemon")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.ResourceAllocator (as "resalloc")
            create-strategy "This object is automatically created on
IPCProcess creation."
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
        ERoot.Classes.SDUDelimiting (as "sdudel")
            create-strategy "This object is automatically created on
IPCProcess creation."
```

```
            delete-strategy "This object is automatically deleted on
IPCProcess destruction."
   ;


   registered-as ERoot Classes(1) 12
;
```

B.2.11. Class PDUForwardingTableEntry

Behavior

Entry in the next hop table. Maps destination address and qos-id to the N-1 port-ids where the PDU will be forwarded

Dependencies

- ERoot.Attributes.Address
- ERoot.Attributes.PortIdList
- ERoot.Attributes.QoSId
- ERoot.Attributes.TableKey

Class Attributes

- key → Unique key of this entry in the table
  - defined by ERoot.Attributes.TableKey
  - read enabled
  - write enabled
- destAddress → Address of the destination IPC Process
  - defined by ERoot.Attributes.Address
  - read enabled
  - write enabled
- qosId → Id of the QoS-cube the PDU belongs to
  - defined by ERoot.Attributes.QoSId
  - read enabled
  - write enabled

- portIds → N-1 port-ids where the PDU will be forwarded
  - defined by ERoot.Attributes.PortIdList
  - read enabled
  - write enabled

Class Operations

- create
  - invoked to add a static entry to the PDU forwarding table
  - in ERoot.Types.T_PDUForwardingTableEntry pdufTableEntry
    - The data of the PDU forwarding table entry
- delete
  - invoked to remove a static entry from the next hop table
- read
  - read information from the table
  - out ERoot.Types.T_PDUForwardingTableEntry pdufTableEntry
    - The data of the PDU forwarding table entry
- cancel-read
  - cancel ongoing read operation

Registered As

ERoot Classes(1) 20

Specification

```
ro class PDUForwardingTableEntry
    behavior
        "Entry in the next hop table. Maps destination address and qos-id
 "
        " to the N-1 port-ids where the PDU will be forwarded"

    attributes
        ERoot.Attributes.TableKey key read write
          "Unique key of this entry in the table"
        ERoot.Attributes.Address destAddress read write
          "Address of the destination IPC Process"
```

```
        ERoot.Attributes.QoSId qosId read write
          "Id of the QoS-cube the PDU belongs to"
        ERoot.Attributes.PortIdList portIds read write
          "N-1 port-ids where the PDU will be forwarded"
    ;

    operations
        create
            "invoked to add a static entry to the PDU forwarding table"
            in ERoot.Types.T_PDUForwardingTableEntry pdufTableEntry
                "The data of the PDU forwarding table entry"

        delete
            "invoked to remove a static entry from the next hop table"

        read
            "read information from the table"
            out ERoot.Types.T_PDUForwardingTableEntry pdufTableEntry
                "The data of the PDU forwarding table entry"

        cancel-read
            "cancel ongoing read operation"

    ;

    registered-as ERoot Classes(1) 20
;
```

B.2.12. Class Top

Behavior

The top-level object for the inheritance tree.

Dependencies

- ERoot.Attributes.ObjectClass
- ERoot.Attributes.ObjectInstance
- ERoot.Attributes.ObjectName

Class Attributes

- objectClass → name of the managed object class

  - defined by ERoot.Attributes.ObjectClass
  - read enabled
  - write enabled
- objectName → name of the managed object instance (uniquely the MO instance within the containment tree)
  - defined by ERoot.Attributes.ObjectName
  - read enabled
  - write enabled
- objectInstance → uniquely identifies the object instance within the containment tree)
  - defined by ERoot.Attributes.ObjectInstance
  - read enabled
  - write enabled

Registered As

ERoot Classes(1) 2

Specification

```
abstract ro class Top
    behavior
        "The top-level object for the inheritance tree."

    attributes
        ERoot.Attributes.ObjectClass objectClass read write
          "name of the managed object class"
        ERoot.Attributes.ObjectName objectName read write
          "name of the managed object instance (uniquely the MO instance
 within the containment tree)"
        ERoot.Attributes.ObjectInstance objectInstance read write
          "uniquely identifies the object instance within the containment
 tree)"
    ;

    registered-as ERoot Classes(1) 2
;
```

B.2.13. Class DirectoryForwardingTable

Behavior

This class is the container for individual DirectoryForwardingTableEntry objects

Dependencies

- ERoot.Classes.DirectoryForwardingTableEntry

Class Containment Relationships

- ERoot.Classes.DirectoryForwardingTableEntry
  - with attribute key
  - Create: This object is created when a new entry is added to the Directory Forwarding Table
  - Delete: This object is removed when an entry is removed from the Directory Forwarding Table

Registered As

ERoot Classes(1) 25

Specification

```
ro class DirectoryForwardingTable
    behavior
        "This class is the container for individual
 DirectoryForwardingTableEntry objects"

    contains
        ERoot.Classes.DirectoryForwardingTableEntry (with attribute key)
            create-strategy "This object is created when a new entry is
 added to the Directory Forwarding Table"
            delete-strategy "This object is removed when an entry is
 removed from the Directory Forwarding Table"
    ;


    registered-as ERoot Classes(1) 25
;
```

B.2.14. Class RMTQueuePair

Behavior

A pair of input/output queues (where one of the directions can be null), attached to N-1 ports by the RMT

Dependencies

- ERoot.Attributes.QueueId
- ERoot.Attributes.RxQueueInfo
- ERoot.Attributes.TxQueueInfo

Class Attributes

- queueId → The id of the queue (unique within N-1 port)
  - defined by ERoot.Attributes.QueueId
  - read enabled
  - write enabled
- rxInfo → Information about the reception queue of this pair
  - defined by ERoot.Attributes.RxQueueInfo
  - read enabled
  - write enabled
- txInfo → Information about the transmission queue of this pair
  - defined by ERoot.Attributes.TxQueueInfo
  - read enabled
  - write enabled

Class Operations

- read
  - read information about an RMT queue pair
  - out ERoot.Types.T_RMTQueuePairState queueState
    - Information about the queue pair
- cancel-read

- cancel ongoing read operation

Registered As

ERoot Classes(1) 33

Specification

```
ro class RMTQueuePair
    behavior
        "A pair of input/output queues (where one of the directions can be
 null), attached to N-1 ports by the RMT"

    attributes
        ERoot.Attributes.QueueId queueId read write
          "The id of the queue (unique within N-1 port)"
        ERoot.Attributes.RxQueueInfo rxInfo read write
          "Information about the reception queue of this pair"
        ERoot.Attributes.TxQueueInfo txInfo read write
          "Information about the transmission queue of this pair"
    ;

    operations
        read
            "read information about an RMT queue pair"
            out ERoot.Types.T_RMTQueuePairState queueState
                "Information about the queue pair"

        cancel-read
            "cancel ongoing read operation"

    ;

    registered-as ERoot Classes(1) 33
;
```

B.2.15. Class Root

Behavior

This class cannot be remotely created or deleted, since it represents the root of the containment subtree of a particular computing system. It is the root object of the RIB containment tree.

Dependencies

- ERoot.Classes.ComputingSystem
- ERoot.Classes.DAF
- ERoot.Classes.DIF

Class Containment Relationships

- ERoot.Classes.ComputingSystem
  - with attribute computingSystemId
  - Create: This object is automatically created on RIB creation.
  - Delete: This object cannot be deleted
- ERoot.Classes.DAF
  - with attribute dafName
  - Create: This object is created when the DAF is created.
  - Delete: This object is destroyed when the DAF is destroyed.
- ERoot.Classes.DIF
  - with attribute difName
  - Create: This object is created when the DIF is created.
  - Delete: This object is destroyed when the DIF is destroyed.

Registered As

ERoot Classes(1) 1

Specification

```
abstract ro class Root
    behavior
        "This class cannot be remotely created or deleted, since it"
        "represents the root of the containment subtree of a particular
 computing"
        "system. It is the root object of the RIB containment tree."

    contains
        ERoot.Classes.ComputingSystem (with attribute computingSystemId)
```

```
            create-strategy "This object is automatically created on RIB
 creation."
            delete-strategy "This object cannot be deleted"
        ERoot.Classes.DAF (with attribute dafName)
            create-strategy "This object is created when the DAF is
 created."
            delete-strategy "This object is destroyed when the DAF is
 destroyed."
        ERoot.Classes.DIF (with attribute difName)
            create-strategy "This object is created when the DIF is
 created."
            delete-strategy "This object is destroyed when the DIF is
 destroyed."
    ;


    registered-as ERoot Classes(1) 1
;
```

B.2.16. Class KernelApplicationProcess

Behavior

Models the AP which manages the hardware resources of a processing system . It is the AP at the lowest level of a processing system.

Dependencies

- ERoot.Classes.OSApplicationProcess

Class Containment Relationships

- ERoot.Classes.OSApplicationProcess
  - as "osApplicationProcess"
  - Create: This object is automatically created on KernelApplicationProcess creation.
  - Delete: This object is automatically deleted on KernelApplicationProcess destruction

Registered As

ERoot Classes(1) 10

Specification

```
ro class KernelApplicationProcess
    behavior
        "Models the AP which manages the hardware resources of a
 processing system"
        ". It is the AP at the lowest level of a processing system."

    contains
        ERoot.Classes.OSApplicationProcess (as "osApplicationProcess")
            create-strategy "This object is automatically created on
 KernelApplicationProcess creation."
            delete-strategy "This object is automatically deleted on
 KernelApplicationProcess destruction"
    ;


    registered-as ERoot Classes(1) 10
;
```

B.2.17. Class UnderlyingDIF

Behavior

The properties of an N-1 DIF known by the application/IPC Process

Dependencies

- ERoot.Attributes.DAPName
- ERoot.Attributes.QoSCubeList
- ERoot.Attributes.SDUSize

Class Attributes

- difName → The name of the N-1 DIF
  - defined by ERoot.Attributes.DAPName
  - read enabled
  - write enabled
- maxSDUSize → The maximum SDU size allowed by the DIF
  - defined by ERoot.Attributes.SDUSize

  - read enabled
  - write enabled
- qosCubes → The list of QoS cubes and their properties
  - defined by ERoot.Attributes.QoSCubeList
  - read enabled
  - write enabled

Registered As

ERoot Classes(1) 37

Specification

```
ro class UnderlyingDIF
    behavior
        "The properties of an N-1 DIF known by the application/IPC
 Process"

    attributes
        ERoot.Attributes.DAPName difName read write
          "The name of the N-1 DIF"
        ERoot.Attributes.SDUSize maxSDUSize read write
          "The maximum SDU size allowed by the DIF"
        ERoot.Attributes.QoSCubeList qosCubes read write
          "The list of QoS cubes and their properties"
    ;

    registered-as ERoot Classes(1) 37
;
```

B.2.18. Class DIF

Behavior

This class represents a Distributed Inter Process Communication Facility

Dependencies

- ERoot.Attributes.DAPName
- ERoot.Classes.IPCProcess

Class Attributes

- difName → uniquely identifies the DIF
  - defined by ERoot.Attributes.DAPName
  - read enabled
  - write enabled

Class Containment Relationships

- ERoot.Classes.IPCProcess
  - with attribute processName
  - Create: This object is created when an IPC process joins the DIF
  - Delete: This object is destroyed when an IPC process leaves the DIF

Registered As

ERoot Classes(1) *5*

Specification

```
ro class DIF
    behavior
        "This class represents a Distributed Inter Process Communication
 Facility"

    attributes
        ERoot.Attributes.DAPName difName read write
          "uniquely identifies the DIF"
    ;

    contains
        ERoot.Classes.IPCProcess (with attribute processName)
            create-strategy "This object is created when an IPC process
 joins the DIF"
            delete-strategy "This object is destroyed when an IPC process
 leaves the DIF"
    ;


    registered-as ERoot Classes(1) 5
;
```

B.2.19. Class ForwardingDiscriminator

Behavior

Manages a group of notifications to a subscriber

Dependencies

- ERoot.Attributes.APNamingInfo
- ERoot.Attributes.ForwardingDiscriminatorId
- ERoot.Policies.NotificationManagement.NotificationFilteringPolicy
- ERoot.Policies.NotificationManagement.ReportArchivePolicy

Class Attributes

- fwDiscriminatorId → The identifier of the forwarding discriminator
  - defined by ERoot.Attributes.ForwardingDiscriminatorId
  - read enabled
  - write enabled
- subscriber → The application that has subscribed to notifications
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled

Class Operations

- delete
  - invoked to cancel the subscription to certain sets of notifications

Class Policies

- ERoot.Policies.NotificationManagement.ReportArchivePolicy
- ERoot.Policies.NotificationManagement.NotificationFilteringPolicy

Registered As

ERoot Classes(1) 62

Specification

```
ro class ForwardingDiscriminator
    behavior
        "Manages a group of notifications to a subscriber"

    attributes
        ERoot.Attributes.ForwardingDiscriminatorId fwDiscriminatorId read
 write
          "The identifier of the forwarding discriminator"
        ERoot.Attributes.APNamingInfo subscriber read write
          "The application that has subscribed to notifications"
    ;

    operations
        delete
            "invoked to cancel the subscription to certain sets of
 notifications"

    ;

    policies
        ERoot.Policies.NotificationManagement.ReportArchivePolicy
        ERoot.Policies.NotificationManagement.NotificationFilteringPolicy
    ;

    registered-as ERoot Classes(1) 62
;
```

B.2.20. Class Flows

Behavior

This class is the container for individual Flow objects

Dependencies

- ERoot.Classes.Flow

Class Containment Relationships

- ERoot.Classes.Flow
  - with attribute localPortId

- Create: This object is created when a flow is allocated in the DIF
- Delete: This object is removed when a flow is deallocated in the DIF

Registered As

ERoot Classes(1) 59

Specification

```
ro class Flows
    behavior
        "This class is the container for individual Flow objects"

    contains
        ERoot.Classes.Flow (with attribute localPortId)
            create-strategy "This object is created when a flow is
 allocated in the DIF"
            delete-strategy "This object is removed when a flow is
 deallocated in the DIF"
    ;


    registered-as ERoot Classes(1) 59
;
```

B.2.21. Class DAF

Behavior

This class represents a Distributed Application Facility

Dependencies

- ERoot.Attributes.DAPName
- ERoot.Classes.ApplicationProcess

Class Attributes

- dafName → uniquely identifies the DAF
  - defined by ERoot.Attributes.DAPName

- read enabled
- write enabled

Class Containment Relationships

- ERoot.Classes.ApplicationProcess
  - with attribute processName
  - Create: This object is created when an application process joins the DAF
  - Delete: This object is destroyed when an application process leaves the DAF

Registered As

ERoot Classes(1) 4

Specification

```
ro class DAF
    behavior
        "This class represents a Distributed Application Facility"

    attributes
        ERoot.Attributes.DAPName dafName read write
          "uniquely identifies the DAF"
    ;

    contains
        ERoot.Classes.ApplicationProcess (with attribute processName)
            create-strategy "This object is created when an application
 process joins the DAF"
            delete-strategy "This object is destroyed when an application
 process leaves the DAF"
    ;


    registered-as ERoot Classes(1) 4
;
```

B.2.22. Class DTCP

Behavior

The DTCP instance of an EFCP connection

Dependencies

- ERoot.Classes.DTCPStateVector
- ERoot.Classes.FlowControl
- ERoot.Classes.RetransmissionControl
- ERoot.Policies.DataTransfer.LostControlPDUPolicy
- ERoot.Policies.DataTransfer.RTTEstimatorPolicy

Class Policies

- ERoot.Policies.DataTransfer.LostControlPDUPolicy
- ERoot.Policies.DataTransfer.RTTEstimatorPolicy

Class Containment Relationships

- ERoot.Classes.DTCPStateVector
  - as "dtcpsv"
  - Create: This object is automatically created when DTCP is created
  - Delete: This object is automatically destroyed when DTCP is destroyed
- ERoot.Classes.FlowControl
  - as "flowCtrl"
  - Create: This object is automatically created when DTCP is creation, if the connection supports flow control
  - Delete: This object is automatically destroyed when DTCP is destroyed
- ERoot.Classes.RetransmissionControl

- as "rtxCtrl"
- Create: This object is automatically created when DTCP is creation, if the connection supports retransmission control
- Delete: This object is automatically destroyed when DTCP is destroyed

Registered As

ERoot Classes(1) 52

Specification

```
ro class DTCP
    behavior
        "The DTCP instance of an EFCP connection"

    policies
        ERoot.Policies.DataTransfer.LostControlPDUPolicy
        ERoot.Policies.DataTransfer.RTTEstimatorPolicy
    ;

    contains
        ERoot.Classes.DTCPStateVector (as "dtcpsv")
            create-strategy "This object is automatically created when
 DTCP is created"
            delete-strategy "This object is automatically destroyed when
 DTCP is destroyed"
        ERoot.Classes.FlowControl (as "flowCtrl")
            create-strategy "This object is automatically created when
 DTCP is creation, if the connection supports flow control"
            delete-strategy "This object is automatically destroyed when
 DTCP is destroyed"
        ERoot.Classes.RetransmissionControl (as "rtxCtrl")
            create-strategy "This object is automatically created when
 DTCP is creation, if the connection supports retransmission control"
            delete-strategy "This object is automatically destroyed when
 DTCP is destroyed"
    ;


    registered-as ERoot Classes(1) 52
;
```

B.2.23. Class DataTransfer

Behavior

This class represents a Data Transfer Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Attributes.DataTransferConstants
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.EFCPConnections
- ERoot.Policies.DataTransfer.UnknownFlowPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Attributes

- dtContstants → DIF-wide parameters that define the concrete syntax of EFCP for this DIF and other DIF-wide values
  - defined by ERoot.Attributes.DataTransferConstants
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.DataTransfer.UnknownFlowPolicy

Class Containment Relationships

- ERoot.Classes.EFCPConnections
  - as "connections"
  - Create: This object is automatically created when the DataTransfer object is created
  - Delete: This object is automatically destroyed when the DataTransfer object is destroyed

Registered As

ERoot Classes(1) 48

Specification

```
ro class DataTransfer
    behavior
        "This class represents a Data Transfer Application Entity."

    extends ERoot.Classes.ApplicationEntity

    attributes
        ERoot.Attributes.DataTransferConstants dtContstants read write
          "DIF-wide parameters that define the concrete syntax of EFCP for
 this DIF and other DIF-wide values"
    ;

    policies
        ERoot.Policies.DataTransfer.UnknownFlowPolicy
    ;

    contains
        ERoot.Classes.EFCPConnections (as "connections")
            create-strategy "This object is automatically created when the
 DataTransfer object is created"
            delete-strategy "This object is automatically destroyed when
 the DataTransfer object is destroyed"
    ;


    registered-as ERoot Classes(1) 48
;
```

B.2.24. Class DirectoryForwardingTableEntry

Behavior

Entry in the directory forwarding table. Maps destination Ap name and qos-id to IPC Process address

Dependencies

- ERoot.Attributes.APNamingInfo

- ERoot.Attributes.Address
- ERoot.Attributes.TableKey

Class Attributes

- key → Unique key of this entry in the table
  - defined by ERoot.Attributes.TableKey
  - read enabled
  - write enabled
- appName → Destination application process name/instance - optionally including entity/instance
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled
- destAddress → Address of the destination IPC Process
  - defined by ERoot.Attributes.Address
  - read enabled
  - write enabled

Class Operations

- create
  - invoked to add a static entry to the Directory forwarding table
  - in ERoot.Types.T_DirectoryForwardingTableEntry dfTableEntry
    - The data of the Directory forwarding table entry
- delete
  - invoked to remove a static entry from the directory forwarding table
- read
  - read information from the table
  - out ERoot.Types.T_DirectoryForwardingTableEntry dfTableEntry
    - The data of the Directory forwarding table entry
- cancel-read

- cancel ongoing read operation

Registered As

ERoot Classes(1) 26

Specification

```
ro class DirectoryForwardingTableEntry
    behavior
        "Entry in the directory forwarding table. Maps destination Ap name
 and qos-id to IPC Process address"

    attributes
        ERoot.Attributes.TableKey key read write
          "Unique key of this entry in the table"
        ERoot.Attributes.APNamingInfo appName read write
          "Destination application process name/instance - optionally
 including entity/instance"
        ERoot.Attributes.Address destAddress read write
          "Address of the destination IPC Process"
    ;

    operations
        create
            "invoked to add a static entry to the Directory forwarding
 table"
            in ERoot.Types.T_DirectoryForwardingTableEntry dfTableEntry
                "The data of the Directory forwarding table entry"

        delete
            "invoked to remove a static entry from the directory
 forwarding table"

        read
            "read information from the table"
            out ERoot.Types.T_DirectoryForwardingTableEntry dfTableEntry
                "The data of the Directory forwarding table entry"

        cancel-read
            "cancel ongoing read operation"

    ;

    registered-as ERoot Classes(1) 26
```

```
;
```

B.2.25. Class UnderlyingRegistrations

Behavior

This class is the container for individual UnderlyingRegistration objects

Dependencies

- ERoot.Classes.UnderlyingRegistration

Class Operations

- create
  - invoked to register an AE to an N-1 DIF
  - in ERoot.Types.T_DIFRegistrationRequest request
    - Register the AE to an N-1 DIF

Class Containment Relationships

- ERoot.Classes.UnderlyingRegistration
  - with attribute applicationEntity
  - Create: This object is created when the app/IPCP registers to the N-1 DIF
  - Delete: This object is removed when the app/IPCP unregisters from the N-1 DIF

Registered As

ERoot Classes(1) 38

Specification

```
ro class UnderlyingRegistrations
    behavior
        "This class is the container for individual UnderlyingRegistration
 objects"
```

```
    operations
        create
            "invoked to register an AE to an N-1 DIF"
            in ERoot.Types.T_DIFRegistrationRequest request
                "Register the AE to an N-1 DIF"

    ;

    contains
        ERoot.Classes.UnderlyingRegistration (with attribute
applicationEntity)
            create-strategy "This object is created when the app/IPCP
registers to the N-1 DIF"
            delete-strategy "This object is removed when the app/IPCP
unregisters from the N-1 DIF"
    ;


    registered-as ERoot Classes(1) 38
;
```

B.2.26. Class Neighbors

Behavior

This class is the container for individual Neighbor objects

Dependencies

- ERoot.Classes.Neighbor

Class Containment Relationships

- ERoot.Classes.Neighbor
  - with attribute processName
  - Create: This object is created when a new neighbor is acquired
  - Delete: This object is removed when a neighbor is lost

Registered As

ERoot Classes(1) 28

Specification

```
ro class Neighbors
    behavior
        "This class is the container for individual Neighbor objects"

    contains
        ERoot.Classes.Neighbor (with attribute processName)
            create-strategy "This object is created when a new neighbor is
 acquired"
            delete-strategy "This object is removed when a neighbor is
 lost"
    ;


    registered-as ERoot Classes(1) 28
;
```

B.2.27. Class RetransmissionControl

Behavior

Retransmission control state of a DTCP instance

Dependencies

- ERoot.Attributes.DataRtxMax
- ERoot.Attributes.Time
- ERoot.Policies.DataTransfer.RcvrAckPolicy
- ERoot.Policies.DataTransfer.RcvrControlAckPolicy
- ERoot.Policies.DataTransfer.ReceivingAckListPolicy
- ERoot.Policies.DataTransfer.RetransmissionTimerExpiryPolicy
- ERoot.Policies.DataTransfer.SenderAckPolicy
- ERoot.Policies.DataTransfer.SendingAckPolicy

Class Attributes

- maxTimeToRetry → Maximum time to attempt the retransmission of a packet, this is R.

  - defined by ERoot.Attributes.Time
  - read enabled
  - write enabled
- dataRtxMax → Indicates the number of times the retransmission of a PDU will be attempted before some other action must be taken
  - defined by ERoot.Attributes.DataRtxMax
  - read enabled
  - write enabled
- initialRtxTime → Indicates the time to wait before retransmitting a PDU (tr)
  - defined by ERoot.Attributes.Time
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.DataTransfer.RetransmissionTimerExpiryPolicy
- ERoot.Policies.DataTransfer.SenderAckPolicy
- ERoot.Policies.DataTransfer.ReceivingAckListPolicy
- ERoot.Policies.DataTransfer.RcvrAckPolicy
- ERoot.Policies.DataTransfer.SendingAckPolicy
- ERoot.Policies.DataTransfer.RcvrControlAckPolicy

Registered As

ERoot Classes(1) *57*

Specification

```
ro class RetransmissionControl
    behavior
        "Retransmission control state of a DTCP instance"
```

```
    attributes
        ERoot.Attributes.Time maxTimeToRetry read write
          "Maximum time to attempt the retransmission of a packet, this is
R."
        ERoot.Attributes.DataRtxMax dataRtxMax read write
          "Indicates the number of times the retransmission of a PDU will
be attempted before some other action must be taken"
        ERoot.Attributes.Time initialRtxTime read write
          "Indicates the time to wait before retransmitting a PDU (tr)"
    ;

    policies
        ERoot.Policies.DataTransfer.RetransmissionTimerExpiryPolicy
        ERoot.Policies.DataTransfer.SenderAckPolicy
        ERoot.Policies.DataTransfer.ReceivingAckListPolicy
        ERoot.Policies.DataTransfer.RcvrAckPolicy
        ERoot.Policies.DataTransfer.SendingAckPolicy
        ERoot.Policies.DataTransfer.RcvrControlAckPolicy
    ;

    registered-as ERoot Classes(1) 57
;
```

B.2.28. Class DIFManagement

Behavior

This class groups together a number of DIF Management functions.

Dependencies

- ERoot.Classes.Enrollment
- ERoot.Classes.NamespaceManager
- ERoot.Classes.SecurityManager

Class Containment Relationships

- ERoot.Classes.SecurityManager
  - as "secman"
  - Create: This object is automatically created on DIFManagement creation.

  - Delete: This object is automatically deleted on DIFManagement destruction.

- ERoot.Classes.NamespaceManager
  - as "nsm"
  - Create: This object is automatically created on DIFManagement creation.
  - Delete: This object is automatically deleted on DIFManagement destruction.
- ERoot.Classes.Enrollment
  - as "enrollment"
  - Create: This object is automatically created on DIFManagement creation.
  - Delete: This object is automatically deleted on DIFManagement destruction.

Registered As

ERoot Classes(1) 22

Specification

```
ro class DIFManagement
    behavior
        "This class groups together a number of DIF Management functions."

    contains
        ERoot.Classes.SecurityManager (as "secman")
            create-strategy "This object is automatically created on
 DIFManagement creation."
            delete-strategy "This object is automatically deleted on
 DIFManagement destruction."
        ERoot.Classes.NamespaceManager (as "nsm")
            create-strategy "This object is automatically created on
 DIFManagement creation."
            delete-strategy "This object is automatically deleted on
 DIFManagement destruction."
        ERoot.Classes.Enrollment (as "enrollment")
            create-strategy "This object is automatically created on
 DIFManagement creation."
```

```
            delete-strategy "This object is automatically deleted on
DIFManagement destruction."
    ;


    registered-as ERoot Classes(1) 22
;
```

B.2.29. Class Neighbor

Behavior

Represents a neighbor IPCP with whom we are sharing state

Dependencies

- ERoot.Attributes.Address
- ERoot.Attributes.DAPInstance
- ERoot.Attributes.DAPName
- ERoot.Attributes.UnderlyingDIFs
- ERoot.Attributes.UnderlyingFlows
- ERoot.Notifications.CreateNeighbor
- ERoot.Notifications.DeleteNeighbor

Class Attributes

- processName → The neighbor IPCP's name
  - defined by ERoot.Attributes.DAPName
  - read enabled
  - write enabled
- processInstance → The neighbor IPCP's instance
  - defined by ERoot.Attributes.DAPInstance
  - read enabled
  - write enabled
- address → Address of the neighbor IPCP

  - defined by ERoot.Attributes.Address
  - read enabled
  - write enabled
- underDIFs → The names of the N-1 DIFs in common with the neighbor IPC Process
  - defined by ERoot.Attributes.UnderlyingDIFs
  - read enabled
  - write enabled
- underFlows → The port-id of the N-1 flow used to talk to the neighbor
  - defined by ERoot.Attributes.UnderlyingFlows
  - read enabled
  - write enabled

Class Operations

- create
  - acquire a new neighbor IPCP
  - in ERoot.Types.T_NeighborConfig neighReq
    - The data required to add a neighbor
- delete
  - disconnect from neighbor IPCP
- read
  - read information from the neighbor
  - out ERoot.Types.T_NeighborConfig neighInfo
    - Data about the neighbor
- cancel-read
  - cancel ongoing read operation

Class Notifications

- ERoot.Notifications.CreateNeighbor

- ERoot.Notifications.DeleteNeighbor

Registered As

ERoot Classes(1) 29

Specification

```
ro class Neighbor
    behavior
        "Represents a neighbor IPCP with whom we are sharing state"

    attributes
        ERoot.Attributes.DAPName processName read write
          "The neighbor IPCP's name"
        ERoot.Attributes.DAPInstance processInstance read write
          "The neighbor IPCP's instance"
        ERoot.Attributes.Address address read write
          "Address of the neighbor IPCP"
        ERoot.Attributes.UnderlyingDIFs underDIFs read write
          "The names of the N-1 DIFs in common with the neighbor IPC
 Process"
        ERoot.Attributes.UnderlyingFlows underFlows read write
          "The port-id of the N-1 flow used to talk to the neighbor"
    ;

    operations
        create
            "acquire a new neighbor IPCP"
            in ERoot.Types.T_NeighborConfig neighReq
                "The data required to add a neighbor"

        delete
            "disconnect from neighbor IPCP"

        read
            "read information from the neighbor"
            out ERoot.Types.T_NeighborConfig neighInfo
                "Data about the neighbor"

        cancel-read
            "cancel ongoing read operation"

    ;
```

```
    notifications
        ERoot.Notifications.CreateNeighbor
        ERoot.Notifications.DeleteNeighbor
    ;

    registered-as ERoot Classes(1) 29
;
```

B.2.30. Class DTCPStateVector

Behavior

The Data Transfer Control Protocol State Vector

Registered As

ERoot Classes(1) 53

Specification

```
ro class DTCPStateVector
    behavior
        "The Data Transfer Control Protocol State Vector"

    registered-as ERoot Classes(1) 53
;
```

B.2.31. Class RIBDaemon

Behavior

This class represents a RIB Daemon Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Attributes.RIBVersionList
- ERoot.Classes.ApplicationConnections
- ERoot.Classes.ApplicationEntity

- ERoot.Classes.Discriminators
- ERoot.Policies.RIBDaemon.RIBLoggingPolicy
- ERoot.Policies.RIBDaemon.RIBReplicationPolicy
- ERoot.Policies.RIBDaemon.RIBSubscriptionPolicy
- ERoot.Policies.RIBDaemon.RIBUpdatePolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Attributes

- supportedRIBVersions → The list of RIB versions supported by this App/IPCP
  - defined by ERoot.Attributes.RIBVersionList
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.RIBDaemon.RIBUpdatePolicy
- ERoot.Policies.RIBDaemon.RIBReplicationPolicy
- ERoot.Policies.RIBDaemon.RIBSubscriptionPolicy
- ERoot.Policies.RIBDaemon.RIBLoggingPolicy

Class Containment Relationships

- ERoot.Classes.ApplicationConnections
  - as "appConnections"
  - Create: This object is automatically created when the RIBDaemon object is created
  - Delete: This object is automatically destroyed when the RIBDaemon object is destroyed
- ERoot.Classes.Discriminators

- as "discriminators"
- Create: This object is automatically created when the RIBDaemon object is created
- Delete: This object is automatically destroyed when the RIBDaemon object is destroyed

Registered As

ERoot Classes(1) 44

Specification

```
ro class RIBDaemon
    behavior
        "This class represents a RIB Daemon Application Entity."

    extends ERoot.Classes.ApplicationEntity

    attributes
        ERoot.Attributes.RIBVersionList supportedRIBVersions read write
          "The list of RIB versions supported by this App/IPCP"
    ;

    policies
        ERoot.Policies.RIBDaemon.RIBUpdatePolicy
        ERoot.Policies.RIBDaemon.RIBReplicationPolicy
        ERoot.Policies.RIBDaemon.RIBSubscriptionPolicy
        ERoot.Policies.RIBDaemon.RIBLoggingPolicy
    ;

    contains
        ERoot.Classes.ApplicationConnections (as "appConnections")
            create-strategy "This object is automatically created when the
RIBDaemon object is created"
            delete-strategy "This object is automatically destroyed when
the RIBDaemon object is destroyed"
        ERoot.Classes.Discriminators (as "discriminators")
            create-strategy "This object is automatically created when the
RIBDaemon object is created"
            delete-strategy "This object is automatically destroyed when
the RIBDaemon object is destroyed"
    ;
```

```
    registered-as ERoot Classes(1) 44
;
```

B.2.32. Class Discriminators

Behavior

This class is the container for individual ForwardingDiscrimnator objects

Dependencies

- ERoot.Classes.ForwardingDiscriminator

Class Operations

- create
  - invoked to request the allocation of an N-1 flow
  - in ERoot.Types.T_NotificationSubscriptionRequest request
    - Request to subscribe to a certain set of notifications

Class Containment Relationships

- ERoot.Classes.ForwardingDiscriminator
  - with attribute fwDiscriminatorId
  - Create: This object is created when a forwarding discriminator is created
  - Delete: This object is removed when a forwarding discrminator is removed

Registered As

ERoot Classes(1) 61

Specification

```
ro class Discriminators
    behavior
```

```
        "This class is the container for individual ForwardingDiscrimnator
objects"

    operations
        create
            "invoked to request the allocation of an N-1 flow"
            in ERoot.Types.T_NotificationSubscriptionRequest request
                "Request to subscribe to a certain set of notifications"

    ;

    contains
        ERoot.Classes.ForwardingDiscriminator (with attribute
fwDiscriminatorId)
            create-strategy "This object is created when a forwarding
discriminator is created"
            delete-strategy "This object is removed when a forwarding
discrminator is removed"
    ;


    registered-as ERoot Classes(1) 61
;
```

B.2.33. Class RMTN1Flows

Behavior

This class is the container for individual RMTN1Flow objects

Dependencies

- ERoot.Classes.RMTN1Flow

Class Containment Relationships

- ERoot.Classes.RMTN1Flow
  - with attribute portId
  - Create: This object is created when a new N-1 Flow with the IPCP as source/target is created
  - Delete: This object is removed when a N-1 Flow with the IPCP as a source/target is destroyed

Registered As

ERoot Classes(1) 31

Specification

```
ro class RMTN1Flows
    behavior
        "This class is the container for individual RMTN1Flow objects"

    contains
        ERoot.Classes.RMTN1Flow (with attribute portId)
            create-strategy "This object is created when a new N-1 Flow
 with the IPCP as source/target is created"
            delete-strategy "This object is removed when a N-1 Flow with
 the IPCP as a source/target is destroyed"
    ;


    registered-as ERoot Classes(1) 31
;
```

B.2.34. Class EFCPConnection

Behavior

An EFCP connection (DTP instance), which may contain a DTCP instance

Dependencies

- ERoot.Attributes.ATimer
- ERoot.Attributes.Address
- ERoot.Attributes.CEPId
- ERoot.Attributes.PortId
- ERoot.Attributes.QoSId
- ERoot.Attributes.SequenceNumber
- ERoot.Classes.DTCP
- ERoot.Classes.DTPStateVector
- ERoot.Notifications.CreateEFCPConnection

- ERoot.Notifications.DeleteEFCPConnection
- ERoot.Policies.DataTransfer.InitialSequenceNumberPolicy
- ERoot.Policies.DataTransfer.ReceiverTimerInactivityPolicy
- ERoot.Policies.DataTransfer.SenderTimerInactivityPolicy

Class Attributes

- srcCepId → The source connection endpoint id
  - defined by ERoot.Attributes.CEPId
  - read enabled
  - write enabled
- destCepId → The destination connection endpoint id
  - defined by ERoot.Attributes.CEPId
  - read enabled
  - write enabled
- srcAddress → The address of the source IPCP of this connection
  - defined by ERoot.Attributes.Address
  - read enabled
  - write enabled
- destAddress → The address of the destination IPCP of this connection
  - defined by ERoot.Attributes.Address
  - read enabled
  - write enabled
- qosId → The id of the QoS cube this connection belongs to
  - defined by ERoot.Attributes.QoSId
  - read enabled
  - write enabled
- portId → The id of the flow supported by this EFCP connection
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled

- initialATimer → The initial A timer for this connection
  - defined by ERoot.Attributes.ATimer
  - read enabled
  - write enabled
- seqNumThreshold → The sequence number rollover threshold for this connection
  - defined by ERoot.Attributes.SequenceNumber
  - read enabled
  - write enabled

Class Operations

- delete
  - invoked to request the destruction of the EFCP connection

Class Policies

- ERoot.Policies.DataTransfer.ReceiverTimerInactivityPolicy
- ERoot.Policies.DataTransfer.SenderTimerInactivityPolicy
- ERoot.Policies.DataTransfer.InitialSequenceNumberPolicy

Class Notifications

- ERoot.Notifications.CreateEFCPConnection
- ERoot.Notifications.DeleteEFCPConnection

Class Containment Relationships

- ERoot.Classes.DTPStateVector
  - as "dtpsv"
  - Create: This object is automatically created when the EFCPConnection is setup
  - Delete: This object is automatically destroyed when the EFCPConnection is destroyed
- ERoot.Classes.DTCP

- as "dtcp"
- Create: This object is automatically created when the EFCPConnection is setup, if the connection supports DTCP
- Delete: This object is automatically destroyed when the EFCPConnection is destroyed

Registered As

ERoot Classes(1) 50

Specification

```
ro class EFCPConnection
    behavior
        "An EFCP connection (DTP instance), which may contain a DTCP
 instance"

    attributes
        ERoot.Attributes.CEPId srcCepId read write
          "The source connection endpoint id"
        ERoot.Attributes.CEPId destCepId read write
          "The destination connection endpoint id"
        ERoot.Attributes.Address srcAddress read write
          "The address of the source IPCP of this connection"
        ERoot.Attributes.Address destAddress read write
          "The address of the destination IPCP of this connection"
        ERoot.Attributes.QoSId qosId read write
          "The id of the QoS cube this connection belongs to"
        ERoot.Attributes.PortId portId read write
          "The id of the flow supported by this EFCP connection"
        ERoot.Attributes.ATimer initialATimer read write
          "The initial A timer for this connection"
        ERoot.Attributes.SequenceNumber seqNumThreshold read write
          "The sequence number rollover threshold for this connection"
    ;

    operations
        delete
            "invoked to request the destruction of the EFCP connection"

    ;

    policies
```

```
        ERoot.Policies.DataTransfer.ReceiverTimerInactivityPolicy
        ERoot.Policies.DataTransfer.SenderTimerInactivityPolicy
        ERoot.Policies.DataTransfer.InitialSequenceNumberPolicy
    ;

    notifications
        ERoot.Notifications.CreateEFCPConnection
        ERoot.Notifications.DeleteEFCPConnection
    ;

    contains
        ERoot.Classes.DTPStateVector (as "dtpsv")
            create-strategy "This object is automatically created when the
EFCPConnection is setup"
            delete-strategy "This object is automatically destroyed when
the EFCPConnection is destroyed"
        ERoot.Classes.DTCP (as "dtcp")
            create-strategy "This object is automatically created when the
EFCPConnection is setup, if the connection supports DTCP"
            delete-strategy "This object is automatically destroyed when
the EFCPConnection is destroyed"
    ;


    registered-as ERoot Classes(1) 50
;
```

B.2.35. Class ManagementAgent

Behavior

This class represents a Management Agent Instance.

Dependencies

- ERoot.Attributes.DAPInstance
- ERoot.Attributes.DAPName
- ERoot.Attributes.ManagementAgentId
- ERoot.Attributes.MasterAgent
- ERoot.Attributes.SynonymList
- ERoot.Classes.ApplicationProcess
- ERoot.Classes.DIFManagement

- ERoot.Classes.IPCManagement
- ERoot.Classes.RIBDaemon

Super Classes

- ERoot.Classes.ApplicationProcess

Class Attributes

- agentId → uniquely identifies the Management Agent within the Processing System
  - defined by ERoot.Attributes.ManagementAgentId
  - read enabled
  - write enabled
- masterAgent → True if the Management Agent is the master of this processing system
  - defined by ERoot.Attributes.MasterAgent
  - read enabled
  - write enabled

Class Operations

- read
  - read Management Agent naming information
  - out T_String managementAgentInfo
    - The Management Agent naming information: DAP name/ instance, list of synonyms and ipc process id; plus the flag indicating if it is the master
- cancel-read
  - cancel ongoing read operation

Class Containment Relationships

- ERoot.Classes.DIFManagement
  - as "dafmanagement"

  - Create: This object is automatically created on ManagementAgent creation.
  - Delete: This object is automatically deleted on ManagementAgent destruction.
- ERoot.Classes.IPCManagement
  - as "ipcmanagement"
  - Create: This object is automatically created on ManagementAgent creation.
  - Delete: This object is automatically deleted on ManagementAgent destruction.
- ERoot.Classes.RIBDaemon
  - as "ribdaemon"
  - Create: This object is automatically created on ManagementAgent creation.
  - Delete: This object is automatically deleted on ManagementAgent destruction.

Registered As

ERoot Classes(1) 13

Specification

```
ro class ManagementAgent
    behavior
        "This class represents a Management Agent Instance."

    extends ERoot.Classes.ApplicationProcess

    attributes
        ERoot.Attributes.ManagementAgentId agentId read write
          "uniquely identifies the Management Agent within the Processing
 System"
        ERoot.Attributes.MasterAgent masterAgent read write
          "True if the Management Agent is the master of this processing
 system"
    ;

    operations
```

```
        read
            "read Management Agent naming information"
            out T_String managementAgentInfo
                "The Management Agent naming information: DAP name/
instance, list of "
                " synonyms and ipc process id; plus the flag indicating if
 it is the master"

        cancel-read
            "cancel ongoing read operation"

    ;

    contains
        ERoot.Classes.DIFManagement (as "dafmanagement")
            create-strategy "This object is automatically created on
 ManagementAgent creation."
            delete-strategy "This object is automatically deleted on
 ManagementAgent destruction."
        ERoot.Classes.IPCManagement (as "ipcmanagement")
            create-strategy "This object is automatically created on
 ManagementAgent creation."
            delete-strategy "This object is automatically deleted on
 ManagementAgent destruction."
        ERoot.Classes.RIBDaemon (as "ribdaemon")
            create-strategy "This object is automatically created on
 ManagementAgent creation."
            delete-strategy "This object is automatically deleted on
 ManagementAgent destruction."
    ;


    registered-as ERoot Classes(1) 13
;
```

B.2.36. Class IPCResourceManager

Behavior

This class represents an IPC Resource Manager component, manages registrations to N-1 DIFs and allocation of N-1 flows

Dependencies

- ERoot.Attributes.AEInstance

- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.UnderlyingDIFs
- ERoot.Classes.UnderlyingFlows
- ERoot.Classes.UnderlyingRegistrations

Super Classes

- ERoot.Classes.ApplicationEntity

Class Containment Relationships

- ERoot.Classes.UnderlyingDIFs
  - as "irm"
  - Create: This object is automatically created on IPCResourceManager creation.
  - Delete: This object is automatically deleted on IPCResourceManager destruction.
- ERoot.Classes.UnderlyingRegistrations
  - as "underregs"
  - Create: This object is automatically created on IPCResourceManager creation.
  - Delete: This object is automatically deleted on IPCResourceManager destruction.
- ERoot.Classes.UnderlyingFlows
  - as "underflows"
  - Create: This object is automatically created on IPCResourceManager creation.
  - Delete: This object is automatically deleted on IPCResourceManager destruction.

Registered As

ERoot Classes(1) 35

Specification

```
ro class IPCResourceManager
    behavior
        "This class represents an IPC Resource Manager component, manages
 registrations to N-1 DIFs and allocation of N-1 flows"

    extends ERoot.Classes.ApplicationEntity

    contains
        ERoot.Classes.UnderlyingDIFs (as "irm")
            create-strategy "This object is automatically created on
 IPCResourceManager creation."
            delete-strategy "This object is automatically deleted on
 IPCResourceManager destruction."
        ERoot.Classes.UnderlyingRegistrations (as "underregs")
            create-strategy "This object is automatically created on
 IPCResourceManager creation."
            delete-strategy "This object is automatically deleted on
 IPCResourceManager destruction."
        ERoot.Classes.UnderlyingFlows (as "underflows")
            create-strategy "This object is automatically created on
 IPCResourceManager creation."
            delete-strategy "This object is automatically deleted on
 IPCResourceManager destruction."
    ;


    registered-as ERoot Classes(1) 35
;
```

B.2.37. Class NamespaceManager

Behavior

This class represents a Namespace Manager Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.DirectoryForwardingTable

- ERoot.Policies.NamespaceManagement.AddressManagementPolicy
- ERoot.Policies.NamespaceManagement.DFTGenerationPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Policies

- ERoot.Policies.NamespaceManagement.AddressManagementPolicy
- ERoot.Policies.NamespaceManagement.DFTGenerationPolicy

Class Containment Relationships

- ERoot.Classes.DirectoryForwardingTable
  - as "dft"
  - Create: This object is automatically created on NamespaceManager creation.
  - Delete: This object is automatically deleted on NamespaceManager destruction.

Registered As

ERoot Classes(1) 24

Specification

```
ro class NamespaceManager
    behavior
        "This class represents a Namespace Manager Application Entity."

    extends ERoot.Classes.ApplicationEntity

    policies
        ERoot.Policies.NamespaceManagement.AddressManagementPolicy
        ERoot.Policies.NamespaceManagement.DFTGenerationPolicy
    ;

    contains
        ERoot.Classes.DirectoryForwardingTable (as "dft")
```

```
            create-strategy "This object is automatically created on
 NamespaceManager creation."
            delete-strategy "This object is automatically deleted on
 NamespaceManager destruction."
    ;


    registered-as ERoot Classes(1) 24
;
```

B.2.38. Class FlowAllocator

Behavior

This class represents a Flow Allocator Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Attributes.FlowAllocatorStats
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.Flows
- ERoot.Policies.FlowAllocation.AllocateNotifyPolicy
- ERoot.Policies.FlowAllocation.AllocateRetryPolicy
- ERoot.Policies.FlowAllocation.NewFlowRequestPolicy
- ERoot.Policies.FlowAllocation.SeqNumRolloverPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Attributes

- flowStats → Statistics on accepted and rejected incoming/outgoing flow requests
  - defined by ERoot.Attributes.FlowAllocatorStats
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.FlowAllocation.AllocateNotifyPolicy
- ERoot.Policies.FlowAllocation.AllocateRetryPolicy
- ERoot.Policies.FlowAllocation.NewFlowRequestPolicy
- ERoot.Policies.FlowAllocation.SeqNumRolloverPolicy

Class Containment Relationships

- ERoot.Classes.Flows
  - as "flows"
  - Create: This object is automatically created when the FlowAllocator object is created
  - Delete: This object is automatically destroyed when the FlowAllocator object is destroyed

Registered As

ERoot Classes(1) 58

Specification

```
ro class FlowAllocator
    behavior
        "This class represents a Flow Allocator Application Entity."

    extends ERoot.Classes.ApplicationEntity

    attributes
        ERoot.Attributes.FlowAllocatorStats flowStats read write
          "Statistics on accepted and rejected incoming/outgoing flow
 requests"
    ;

    policies
        ERoot.Policies.FlowAllocation.AllocateNotifyPolicy
        ERoot.Policies.FlowAllocation.AllocateRetryPolicy
        ERoot.Policies.FlowAllocation.NewFlowRequestPolicy
        ERoot.Policies.FlowAllocation.SeqNumRolloverPolicy
    ;
```

```
    contains
        ERoot.Classes.Flows (as "flows")
            create-strategy "This object is automatically created when the
 FlowAllocator object is created"
            delete-strategy "This object is automatically destroyed when
 the FlowAllocator object is destroyed"
    ;


    registered-as ERoot Classes(1) 58
;
```

B.2.39. Class QoSCube

Behavior

A class of service supported by the DIF

Dependencies

- ERoot.Attributes.AverageBW
- ERoot.Attributes.AverageSDUBW
- ERoot.Attributes.DTCPConfig
- ERoot.Attributes.DTPConfig
- ERoot.Attributes.Delay
- ERoot.Attributes.Jitter
- ERoot.Attributes.QoSCubeName
- ERoot.Attributes.QoSId
- ERoot.Notifications.CreateQoSCube
- ERoot.Notifications.DeleteQoSCube

Class Attributes

- name → The name of the QoS cube
  - defined by ERoot.Attributes.QoSCubeName
  - read enabled
  - write enabled
- qosId → Id of the QoS cube
  - defined by ERoot.Attributes.QoSId

  - read enabled
  - write enabled
- averageBW → Average bandwidth in bytes/s. A value of 0 means don't care.
  - defined by ERoot.Attributes.AverageBW
  - read enabled
  - write enabled
- averageSDUBW → Average bandwidth in SDUs/s. A value of 0 means don't care
  - defined by ERoot.Attributes.AverageSDUBW
  - read enabled
  - write enabled
- delay → In milliseconds, indicates the maximum delay allowed in this flow. A value of 0 indicates 'do not care'
  - defined by ERoot.Attributes.Delay
  - read enabled
  - write enabled
- jitter → In milliseconds, indicates the maximum jitter allowed in this flow. A value of 0 indicates 'do not care'
  - defined by ERoot.Attributes.Jitter
  - read enabled
  - write enabled
- dtpConfig → The configuration of DTP for this QoS cube
  - defined by ERoot.Attributes.DTPConfig
  - read enabled
  - write enabled
- dtcpConfig → The configuration of DTCP for this QoS cube
  - defined by ERoot.Attributes.DTCPConfig
  - read enabled
  - write enabled

Class Operations

- create
  - invoked to add support for a new QoS cube
  - in ERoot.Types.T_QoSCubeConfig newCube
    - Add support for a new QoS cube
- delete
  - invoked to stop supporting a specific QoS cube
- read
  - read information about a QoS cube
  - out ERoot.Types.T_QoSCubeConfig qosCube
    - Information about the QoS cube
- cancel-read
  - cancel ongoing read operation

Class Notifications

- ERoot.Notifications.CreateQoSCube
- ERoot.Notifications.DeleteQoSCube

Registered As

ERoot Classes(1) 21

Specification

```
ro class QoSCube
    behavior
        "A class of service supported by the DIF"

    attributes
        ERoot.Attributes.QoSCubeName name read write
          "The name of the QoS cube"
        ERoot.Attributes.QoSId qosId read write
          "Id of the QoS cube"
        ERoot.Attributes.AverageBW averageBW read write
          "Average bandwidth in bytes/s. A value of 0 means don't care."
        ERoot.Attributes.AverageSDUBW averageSDUBW read write
          "Average bandwidth in SDUs/s. A value of 0 means don't care"
```

```
        ERoot.Attributes.Delay delay read write
          "In milliseconds, indicates the maximum delay allowed in this
  flow. A value of 0 indicates 'do not care'"
        ERoot.Attributes.Jitter jitter read write
          "In milliseconds, indicates the maximum jitter allowed in this
  flow. A value of 0 indicates 'do not care'"
        ERoot.Attributes.DTPConfig dtpConfig read write
          "The configuration of DTP for this QoS cube"
        ERoot.Attributes.DTCPConfig dtcpConfig read write
          "The configuration of DTCP for this QoS cube"
    ;

    operations
        create
            "invoked to add support for a new QoS cube"
            in ERoot.Types.T_QoSCubeConfig newCube
                "Add support for a new QoS cube"

        delete
            "invoked to stop supporting a specific QoS cube"

        read
            "read information about a QoS cube"
            out ERoot.Types.T_QoSCubeConfig qosCube
                "Information about the QoS cube"

        cancel-read
            "cancel ongoing read operation"

    ;

    notifications
        ERoot.Notifications.CreateQoSCube
        ERoot.Notifications.DeleteQoSCube
    ;

    registered-as ERoot Classes(1) 21
;
```

B.2.40. Class NextHopTableEntry

Behavior

Entry in the next hop table. Maps destination address and qos-id to the IPCP @ that is the next hop.

Dependencies

- ERoot.Attributes.Address
- ERoot.Attributes.AddressList
- ERoot.Attributes.QoSId
- ERoot.Attributes.TableKey

Class Attributes

- key → Unique key of this entry in the table
  - defined by ERoot.Attributes.TableKey
  - read enabled
  - write enabled
- destAddress → Address of the destination IPC Process
  - defined by ERoot.Attributes.Address
  - read enabled
  - write enabled
- qosId → Id of the QoS-cube the PDU belongs to
  - defined by ERoot.Attributes.QoSId
  - read enabled
  - write enabled
- nextHops → Address(es) of the IPCP(s) that are the next hop(s) towards destination
  - defined by ERoot.Attributes.AddressList
  - read enabled
  - write enabled

Class Operations

- create
  - invoked to add a static entry to the next hop table
  - in ERoot.Types.T_NextHopTableEntry nHopTableEntry

    - The data of the next hop table entry
- delete
  - invoked to remove a static entry from the next hop table
- read
  - read information from the table
  - out ERoot.Types.T_NextHopTableEntry nHopTableEntry
    - The data of the next hop table entry
- cancel-read
  - cancel ongoing read operation

Registered As

ERoot Classes(1) 19

Specification

```
ro class NextHopTableEntry
    behavior
        "Entry in the next hop table. Maps destination address"
        " and qos-id to the IPCP @ that is the next hop."

    attributes
        ERoot.Attributes.TableKey key read write
          "Unique key of this entry in the table"
        ERoot.Attributes.Address destAddress read write
          "Address of the destination IPC Process"
        ERoot.Attributes.QoSId qosId read write
          "Id of the QoS-cube the PDU belongs to"
        ERoot.Attributes.AddressList nextHops read write
          "Address(es) of the IPCP(s) that are the next hop(s) towards
 destination"
    ;

    operations
        create
            "invoked to add a static entry to the next hop table"
            in ERoot.Types.T_NextHopTableEntry nHopTableEntry
                "The data of the next hop table entry"

        delete
```

```
            "invoked to remove a static entry from the next hop table"

        read
            "read information from the table"
            out ERoot.Types.T_NextHopTableEntry nHopTableEntry
                "The data of the next hop table entry"

        cancel-read
            "cancel ongoing read operation"

    ;

    registered-as ERoot Classes(1) 19
;
```

B.2.41. Class Enrollment

Behavior

This class represents a Enrollment Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.Neighbors
- ERoot.Policies.Enrollment.EnrollmentPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Policies

- ERoot.Policies.Enrollment.EnrollmentPolicy

Class Containment Relationships

- ERoot.Classes.Neighbors

- as "neighbors"
- Create: This object is automatically created on Enrollment creation.
- Delete: This object is automatically deleted on Enrollment destruction.

Registered As

ERoot Classes(1) 27

Specification

```
ro class Enrollment
    behavior
        "This class represents a Enrollment Application Entity."

    extends ERoot.Classes.ApplicationEntity

    policies
        ERoot.Policies.Enrollment.EnrollmentPolicy
    ;

    contains
        ERoot.Classes.Neighbors (as "neighbors")
            create-strategy "This object is automatically created on
 Enrollment creation."
            delete-strategy "This object is automatically deleted on
 Enrollment destruction."
    ;


    registered-as ERoot Classes(1) 27
;
```

B.2.42. Class DTPStateVector

Behavior

The Data Transfer Protocol State Vector

Dependencies

- ERoot.Attributes.EFCPConnectionStats

- ERoot.Attributes.SequenceNumber

Class Attributes

- statistics → The statistics on the data sent/received in this connection
  - defined by ERoot.Attributes.EFCPConnectionStats
  - read enabled
  - write enabled
- maxSeqNumRx → The maximum sequence number received
  - defined by ERoot.Attributes.SequenceNumber
  - read enabled
  - write enabled
- lastSeqNumTx → The last sequence number sent
  - defined by ERoot.Attributes.SequenceNumber
  - read enabled
  - write enabled

Registered As

ERoot Classes(1) 51

Specification

```
ro class DTPStateVector
    behavior
        "The Data Transfer Protocol State Vector"

    attributes
        ERoot.Attributes.EFCPConnectionStats statistics read write
          "The statistics on the data sent/received in this connection"
        ERoot.Attributes.SequenceNumber maxSeqNumRx read write
          "The maximum sequence number received"
        ERoot.Attributes.SequenceNumber lastSeqNumTx read write
          "The last sequence number sent"
    ;

    registered-as ERoot Classes(1) 51
;
```

B.2.43. Class RateBasedFlowControl

Behavior

Rate-based flow control state of a DTCP instance

Dependencies

- ERoot.Attributes.Rate
- ERoot.Attributes.TimePeriod
- ERoot.Policies.DataTransfer.NoOverrideDefaultPeakPolicy
- ERoot.Policies.DataTransfer.NoRateSlowDownPolicy
- ERoot.Policies.DataTransfer.RateReductionPolicy

Class Attributes

- sendingRate → Indicates the number of PDUs that may be sent in a TimePeriod. Used with rate-based flow control
  - defined by ERoot.Attributes.Rate
  - read enabled
  - write enabled
- timePeriod → Indicates the length of time in microseconds for pacing rate-based flow control
  - defined by ERoot.Attributes.TimePeriod
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.DataTransfer.NoRateSlowDownPolicy
- ERoot.Policies.DataTransfer.NoOverrideDefaultPeakPolicy
- ERoot.Policies.DataTransfer.RateReductionPolicy

Registered As

ERoot Classes(1) 56

Specification

```
ro class RateBasedFlowControl
    behavior
        "Rate-based flow control state of a DTCP instance"

    attributes
        ERoot.Attributes.Rate sendingRate read write
          "Indicates the number of PDUs that may be sent in a TimePeriod.
 Used with rate-based flow control"
        ERoot.Attributes.TimePeriod timePeriod read write
          "Indicates the length of time in microseconds for pacing rate-
based flow control"
    ;

    policies
        ERoot.Policies.DataTransfer.NoRateSlowDownPolicy
        ERoot.Policies.DataTransfer.NoOverrideDefaultPeakPolicy
        ERoot.Policies.DataTransfer.RateReductionPolicy
    ;

    registered-as ERoot Classes(1) 56
;
```

B.2.44. Class Hardware

Behavior

This class represents top class of a processing's system hardware management subtree

Registered As

ERoot Classes(1) 8

Specification

```
ro class Hardware
    behavior
        "This class represents top class of a processing's system hardware
 management subtree"

    registered-as ERoot Classes(1) 8
```

;

B.2.45. Class ApplicationEntity

Behavior

This class represents an entity of an application process.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName

Class Attributes

- entityName → Name of the application entity
  - defined by ERoot.Attributes.AEName
  - read enabled
  - write enabled
- entityInstance → Instance of the application entity
  - defined by ERoot.Attributes.AEInstance
  - read enabled
  - write enabled

Registered As

ERoot Classes(1) 14

Specification

```
ro class ApplicationEntity
    behavior
        "This class represents an entity of an application process."

    attributes
        ERoot.Attributes.AEName entityName read write
          "Name of the application entity"
        ERoot.Attributes.AEInstance entityInstance read write
          "Instance of the application entity"
```

```
    ;

    registered-as ERoot Classes(1) 14
;
```

B.2.46. Class EFCPConnections

Behavior

This class is the container for individual ApplicationConnection objects

Dependencies

- ERoot.Classes.EFCPConnection

Class Containment Relationships

- ERoot.Classes.EFCPConnection
  - with attribute srcCepId
  - Create: This object is created when an application connection to a peer App/IPCP is established
  - Delete: This object is removed when an application connection to a peer App/IPCP is released

Registered As

ERoot Classes(1) 49

Specification

```
ro class EFCPConnections
    behavior
        "This class is the container for individual ApplicationConnection
 objects"

    contains
        ERoot.Classes.EFCPConnection (with attribute srcCepId)
            create-strategy "This object is created when an application
 connection to a peer App/IPCP is established"
            delete-strategy "This object is removed when an application
 connection to a peer App/IPCP is released"
    ;
```

```
    registered-as ERoot Classes(1) 49
;
```

B.2.47. Class SDUDelimiting

Behavior

This class represents a SDU Delimiting Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Policies.Delimiting.ConcatenationPolicy
- ERoot.Policies.Delimiting.FragmentationPolicy
- ERoot.Policies.Delimiting.ReassemblyAndSeparationPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Policies

- ERoot.Policies.Delimiting.ConcatenationPolicy
- ERoot.Policies.Delimiting.FragmentationPolicy
- ERoot.Policies.Delimiting.ReassemblyAndSeparationPolicy

Registered As

ERoot Classes(1) 47

Specification

```
ro class SDUDelimiting
    behavior
        "This class represents a SDU Delimiting Application Entity."
```

```
    extends ERoot.Classes.ApplicationEntity

    policies
        ERoot.Policies.Delimiting.ConcatenationPolicy
        ERoot.Policies.Delimiting.FragmentationPolicy
        ERoot.Policies.Delimiting.ReassemblyAndSeparationPolicy
    ;

    registered-as ERoot Classes(1) 47
;
```

B.2.48. Class Flow

Behavior

A flow provided by the DIF

Dependencies

- ERoot.Attributes.APNamingInfo
- ERoot.Attributes.CEPId
- ERoot.Attributes.CEPIdList
- ERoot.Attributes.FlowProperties
- ERoot.Attributes.FlowState
- ERoot.Attributes.PortId
- ERoot.Attributes.Retries
- ERoot.Notifications.CreateFlow
- ERoot.Notifications.DeleteFlow
- ERoot.Notifications.FlowQoSViolated

Class Attributes

- localPortId → The port-id of the flow
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled
- localAppName → The local application entity that is using the N flow.
  - defined by ERoot.Attributes.APNamingInfo

  - read enabled
  - write enabled
- remotePortId → The portId of the flow at the remote IPC Process
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled
- remoteAppName → The remote application entity that is using the N flow
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled
- flowSpec → The characteristics of the N flow (loss, delay, reliability, in order-delivery of SDUs, etc)
  - defined by ERoot.Attributes.FlowProperties
  - read enabled
  - write enabled
- currentCEPId → The connection-endpoint currently used by the EFCP connection supporting this flow
  - defined by ERoot.Attributes.CEPId
  - read enabled
  - write enabled
- reservedCEPIds → The connection-endpoint ids reserved for the connections that will support this flow in this IPC Process
  - defined by ERoot.Attributes.CEPIdList
  - read enabled
  - write enabled
- state → 0 allocation in progress, 1 allocated, 2, deallocation in progress, 3 deallocated
  - defined by ERoot.Attributes.FlowState
  - read enabled
  - write enabled

- createFlowRetries → The current number of attempts for allocating this flow
  - defined by ERoot.Attributes.Retries
  - read enabled
  - write enabled
- maxCreateFlowRetries → The maximum number of attempts for allocating the flows
  - defined by ERoot.Attributes.Retries
  - read enabled
  - write enabled

Class Operations

- delete
  - invoked to request the deallocation of the flow

Class Notifications

- ERoot.Notifications.CreateFlow
- ERoot.Notifications.DeleteFlow
- ERoot.Notifications.FlowQoSViolated

Registered As

ERoot Classes(1) 60

Specification

```
ro class Flow
    behavior
        "A flow provided by the DIF"

    attributes
        ERoot.Attributes.PortId localPortId read write
          "The port-id of the flow"
        ERoot.Attributes.APNamingInfo localAppName read write
          "The local application entity that is using the N flow."
        ERoot.Attributes.PortId remotePortId read write
```

```
          "The portId of the flow at the remote IPC Process"
        ERoot.Attributes.APNamingInfo remoteAppName read write
          "The remote application entity that is using the N flow"
        ERoot.Attributes.FlowProperties flowSpec read write
          "The characteristics of the N flow (loss, delay, reliability, in
order-delivery of SDUs, etc)"
        ERoot.Attributes.CEPId currentCEPId read write
          "The connection-endpoint currently used by the EFCP connection
supporting this flow"
        ERoot.Attributes.CEPIdList reservedCEPIds read write
          "The connection-endpoint ids reserved for the connections that
will support this flow in this IPC Process"
        ERoot.Attributes.FlowState state read write
          "0 allocation in progress, 1 allocated, 2, deallocation in
progress, 3 deallocated"
        ERoot.Attributes.Retries createFlowRetries read write
          "The current number of attempts for allocating this flow"
        ERoot.Attributes.Retries maxCreateFlowRetries read write
          "The maximum number of attempts for allocating the flows"
    ;

    operations
        delete
            "invoked to request the deallocation of the flow"

    ;

    notifications
        ERoot.Notifications.CreateFlow
        ERoot.Notifications.DeleteFlow
        ERoot.Notifications.FlowQoSViolated
    ;

    registered-as ERoot Classes(1) 60
;
```

B.2.49. Class SDUProtection

Behavior

This class represents a SDU Protection Application Entity.

Dependencies

- ERoot.Attributes.AEInstance

- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.SDUProtectionPolicySet

Super Classes

- ERoot.Classes.ApplicationEntity

Class Containment Relationships

- ERoot.Classes.SDUProtectionPolicySet
  - with attribute portId
  - Create: This object is automatically created when an N-1 flow is created
  - Delete: This object is automatically destroyed when an N-1 flow is destroyed

Registered As

ERoot Classes(1) 42

Specification

```
ro class SDUProtection
    behavior
        "This class represents a SDU Protection Application Entity."

    extends ERoot.Classes.ApplicationEntity

    contains
        ERoot.Classes.SDUProtectionPolicySet (with attribute portId)
            create-strategy "This object is automatically created when an
 N-1 flow is created"
            delete-strategy "This object is automatically destroyed when
 an N-1 flow is destroyed"
    ;


    registered-as ERoot Classes(1) 42
;
```

B.2.50. Class OSApplicationProcess

Behavior

Models the AP which manages other APs in the processing system.

Dependencies

- ERoot.Classes.IPCProcess
- ERoot.Classes.ManagementAgent

Class Containment Relationships

- ERoot.Classes.IPCProcess
  - with attribute processId
  - Create: This object is created when a new IPC Process is instantiated. The creation of the IPC Process may involve its registration in an N-1 DIF, and/or its assignment to a DIF and enrollment to this DIF via a neighbor IPC Process.
  - Delete: This object is deleted when the IPC Process is terminated. Termination of an IPCP will remove it from the DIF it was a member of, and unregister it from all N-1 DIFs the IPCP was registered at.
- ERoot.Classes.ManagementAgent
  - with attribute agentId
  - Create: This object is created when a management agent is instantiated at the processing system. All systems contain a master Management Agent that has access to the full processing sytem's resources and may contain other Management Agents that have access to only subsets of the processing sytem's resources.
  - Delete: The master Management Agent is deleted when the processing system is shut down. The other agents can be deleted dynamically.

Registered As

ERoot Classes(1) 11

Specification

```
ro class OSApplicationProcess
    behavior
        "Models the AP which manages other APs in the processing system."

    contains
        ERoot.Classes.IPCProcess (with attribute processId)
            create-strategy "This object is created when a new IPC
 Process is instantiated. The creation of the IPC Process may involve its
 registration in an N-1 DIF, and/or its assignment to a DIF and enrollment
 to this DIF via a neighbor IPC Process."
            delete-strategy "This object is deleted when the IPC Process
 is terminated. Termination of an IPCP will remove it from the DIF it was
 a member of, and unregister it from all N-1 DIFs the IPCP was registered
 at."
        ERoot.Classes.ManagementAgent (with attribute agentId)
            create-strategy "This object is created when a management
 agent is instantiated at the processing system. All systems contain a
 master Management Agent that has access to the full processing sytem's
 resources and may contain other Management Agents that have access to
 only subsets of the processing sytem's resources."
            delete-strategy "The master Management Agent is deleted when
 the processing system is shut down. The other agents can be deleted
 dynamically."
    ;


    registered-as ERoot Classes(1) 11
;
```

B.2.51. Class NextHopTable

Behavior

This class is the container for individual NextHopTableEntry objects

Dependencies

- ERoot.Classes.NextHopTableEntry

Class Containment Relationships

- ERoot.Classes.NextHopTableEntry

- with attribute key
- Create: This object is created when a new entry is added to the Next Hop Table
- Delete: This object is removed when an entry is removed from the Next Hop Table

Registered As

ERoot Classes(1) 18

Specification

```
ro class NextHopTable
    behavior
        "This class is the container for individual NextHopTableEntry
 objects"

    contains
        ERoot.Classes.NextHopTableEntry (with attribute key)
            create-strategy "This object is created when a new entry is
 added to the Next Hop Table"
            delete-strategy "This object is removed when an entry is
 removed from the Next Hop Table"
    ;


    registered-as ERoot Classes(1) 18
;
```

B.2.52. Class RMTN1Flow

Behavior

A N-1 flow used by the RMT

Dependencies

- ERoot.Attributes.N1FlowStats
- ERoot.Attributes.PortId
- ERoot.Attributes.PortStarted
- ERoot.Classes.RMTQueuePair

- ERoot.Notifications.CreateRMTN1Flow
- ERoot.Notifications.DeleteRMTN1Flow

Class Attributes

- portId → The portId of the N-1 flow
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled
- portUp → True if the N-1 flow is started, false otherwise
  - defined by ERoot.Attributes.PortStarted
  - read enabled
  - write enabled
- portStats → Statistics of the N-1 flow
  - defined by ERoot.Attributes.N1FlowStats
  - read enabled
  - write enabled

Class Operations

- read
  - read information about an N-1 flow
  - out ERoot.Types.T_N1FlowState n1FlowState
    - Information about the N-1 flow
- start
  - The RMT can use the N-1 flow again
- stop
  - The RMT can no longer use the N-1 flow
- cancel-read
  - cancel ongoing read operation

Class Notifications

- ERoot.Notifications.CreateRMTN1Flow

- ERoot.Notifications.DeleteRMTN1Flow

Class Containment Relationships

- ERoot.Classes.RMTQueuePair
  - with attribute queueId
  - Create: This object is created when a new queue pair is added to the N-1 port
  - Delete: This object is removed when a queue pair is removed from the N-1 port

Registered As

ERoot Classes(1) 32

Specification

```
ro class RMTN1Flow
    behavior
        "A N-1 flow used by the RMT"

    attributes
        ERoot.Attributes.PortId portId read write
          "The portId of the N-1 flow"
        ERoot.Attributes.PortStarted portUp read write
          "True if the N-1 flow is started, false otherwise"
        ERoot.Attributes.N1FlowStats portStats read write
          "Statistics of the N-1 flow"
    ;

    operations
        read
            "read information about an N-1 flow"
            out ERoot.Types.T_N1FlowState n1FlowState
                "Information about the N-1 flow"

        start
            "The RMT can use the N-1 flow again"

        stop
            "The RMT can no longer use the N-1 flow"

        cancel-read
```

```
            "cancel ongoing read operation"

    ;

    notifications
        ERoot.Notifications.CreateRMTN1Flow
        ERoot.Notifications.DeleteRMTN1Flow
    ;

    contains
        ERoot.Classes.RMTQueuePair (with attribute queueId)
            create-strategy "This object is created when a new queue pair
 is added to the N-1 port"
            delete-strategy "This object is removed when a queue pair is
 removed from the N-1 port"
    ;


    registered-as ERoot Classes(1) 32
;
```

B.2.53. Class UnderlyingDIFs

Behavior

This class is the container for individual UnderlyingDIF objects

Dependencies

- ERoot.Classes.UnderlyingDIF

Class Containment Relationships

- ERoot.Classes.UnderlyingDIF
  - with attribute difName
  - Create: This object is created when a new N-1 DIF is discovered by the application / IPC Process
  - Delete: This object is removed when a known N-1 DIF is destroyed

Registered As

ERoot Classes(1) 36

Specification

```
ro class UnderlyingDIFs
    behavior
        "This class is the container for individual UnderlyingDIF objects"

    contains
        ERoot.Classes.UnderlyingDIF (with attribute difName)
            create-strategy "This object is created when a new N-1 DIF is
 discovered by the application / IPC Process"
            delete-strategy "This object is removed when a known N-1 DIF
 is destroyed"
    ;


    registered-as ERoot Classes(1) 36
;
```

B.2.54. Class UnderlyingFlows

Behavior

This class is the container for individual UnderlyingFlow objects

Dependencies

- ERoot.Classes.UnderlyingFlow

Class Operations

- create
  - invoked to request the allocation of an N-1 flow
  - in ERoot.Types.T_FlowAllocationRequest difName
    - Flow allocation request

Class Containment Relationships

- ERoot.Classes.UnderlyingFlow
  - with attribute portId
  - Create: This object is created when the app/IPCP has successfully been allocated an N-1 flow

- Delete: This object is removed when the N-1 flow is removed

Registered As

ERoot Classes(1) 40

Specification

```
ro class UnderlyingFlows
    behavior
        "This class is the container for individual UnderlyingFlow
 objects"

    operations
        create
            "invoked to request the allocation of an N-1 flow"
            in ERoot.Types.T_FlowAllocationRequest difName
                "Flow allocation request"

    ;

    contains
        ERoot.Classes.UnderlyingFlow (with attribute portId)
            create-strategy "This object is created when the app/IPCP has
 successfully been allocated an N-1 flow"
            delete-strategy "This object is removed when the N-1 flow is
 removed"
    ;


    registered-as ERoot Classes(1) 40
;
```

B.2.55. Class UnderlyingFlow

Behavior

An N-1 flow that can be used by the App/IPCP

Dependencies

- ERoot.Attributes.APNamingInfo
- ERoot.Attributes.DAPName

- ERoot.Attributes.FlowProperties
- ERoot.Attributes.PortId
- ERoot.Notifications.CreateUnderlyingFlow
- ERoot.Notifications.DeleteUnderlyingFlow

Class Attributes

- portId → The port-id of the N-1 flow
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled
- localAppName → The names of the application that requested the flow
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled
- remoteAppNAme → The names of the application that is the target of the flow
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled
- flowSpec → The characteristics of this flow
  - defined by ERoot.Attributes.FlowProperties
  - read enabled
  - write enabled
- difName → The name of the N-1 DIF providing the flow
  - defined by ERoot.Attributes.DAPName
  - read enabled
  - write enabled

Class Operations

- delete

- invoked to request the de-allocation of an N-1 flow

Class Notifications

- ERoot.Notifications.CreateUnderlyingFlow
- ERoot.Notifications.DeleteUnderlyingFlow

Registered As

ERoot Classes(1) 41

Specification

```
ro class UnderlyingFlow
    behavior
        "An N-1 flow that can be used by the App/IPCP"

    attributes
        ERoot.Attributes.PortId portId read write
          "The port-id of the N-1 flow"
        ERoot.Attributes.APNamingInfo localAppName read write
          "The names of the application that requested the flow"
        ERoot.Attributes.APNamingInfo remoteAppNAme read write
          "The names of the application that is the target of the flow"
        ERoot.Attributes.FlowProperties flowSpec read write
          "The characteristics of this flow"
        ERoot.Attributes.DAPName difName read write
          "The name of the N-1 DIF providing the flow"
    ;

    operations
        delete
            "invoked to request the de-allocation of an N-1 flow"

    ;

    notifications
        ERoot.Notifications.CreateUnderlyingFlow
        ERoot.Notifications.DeleteUnderlyingFlow
    ;

    registered-as ERoot Classes(1) 41
;
```

B.2.56. Class RelayingAndMultiplexing

Behavior

This class represents a Relaying And Multiplexing Application Entity.

Dependencies

- ERoot.Attributes.AEInstance
- ERoot.Attributes.AEName
- ERoot.Classes.ApplicationEntity
- ERoot.Classes.RMTN1Flows
- ERoot.Policies.RelayingAndMultiplexing.PDUForwardingPolicy
- ERoot.Policies.RelayingAndMultiplexing.PDUSchedulingPolicy

Super Classes

- ERoot.Classes.ApplicationEntity

Class Policies

- ERoot.Policies.RelayingAndMultiplexing.PDUSchedulingPolicy
- ERoot.Policies.RelayingAndMultiplexing.PDUForwardingPolicy

Class Containment Relationships

- ERoot.Classes.RMTN1Flows
  - as "n1flows"
  - Create: This object is automatically created on RelayingAndMultiplexing creation.
  - Delete: This object is automatically deleted on RelayingAndMultiplexing destruction.

Registered As

ERoot Classes(1) 30

Specification

```
ro class RelayingAndMultiplexing
    behavior
        "This class represents a Relaying And Multiplexing Application
 Entity."

    extends ERoot.Classes.ApplicationEntity

    policies
        ERoot.Policies.RelayingAndMultiplexing.PDUSchedulingPolicy
        ERoot.Policies.RelayingAndMultiplexing.PDUForwardingPolicy
    ;

    contains
        ERoot.Classes.RMTN1Flows (as "n1flows")
            create-strategy "This object is automatically created on
 RelayingAndMultiplexing creation."
            delete-strategy "This object is automatically deleted on
 RelayingAndMultiplexing destruction."
    ;


    registered-as ERoot Classes(1) 30
;
```

B.2.57. Class ApplicationProcess

The instantiation of a program executing in a processing system intended to accomplish some purpose. An Application Process contains one or more tasks or Application-Entities, as well as functions for managing the resources (processor, storage, and IPC) allocated to this AP.

Behavior

This class represents a running instance of an Application Process.

Dependencies

- ERoot.Attributes.DAPInstance
- ERoot.Attributes.DAPName
- ERoot.Attributes.SynonymList

Class Attributes

- processName → Name of the application process
  - defined by ERoot.Attributes.DAPName
  - read enabled
  - write enabled
- processInstance → Instance of the application process
  - defined by ERoot.Attributes.DAPInstance
  - read enabled
  - write enabled
- synonymList → A set of synonyms for the process whose scope is restricted to the DAF it is a member of and may be structured to facilitate its use with in the DAF.
  - defined by ERoot.Attributes.SynonymList
  - read enabled
  - write enabled

Registered As

ERoot Classes(1) 9

Specification

```
ro class ApplicationProcess
    documentation-text
        "The instantiation of a program executing in a processing system
 intended"
        "to accomplish some purpose. An Application Process contains one
 or"
        "more tasks or Application-Entities, as well as functions for
 managing the"
        "resources (processor, storage, and IPC) allocated to this AP."

    behavior
        "This class represents a running instance of an Application
 Process."

    attributes
```

```
        ERoot.Attributes.DAPName processName read write
          "Name of the application process"
        ERoot.Attributes.DAPInstance processInstance read write
          "Instance of the application process"
        ERoot.Attributes.SynonymList synonymList read write
          "A set of synonyms for the process whose scope is restricted to
 the DAF"
          " it is a member of and may be structured to facilitate its use
 with in the DAF."
    ;

    registered-as ERoot Classes(1) 9
;
```

B.2.58. Class QoSCubes

Behavior

This class is the container for individual QoSCube objects

Dependencies

- ERoot.Classes.QoSCube

Class Containment Relationships

- ERoot.Classes.QoSCube
  - with attribute qosId
  - Create: This object is created when a new QoS cube is supported by the DIF
  - Delete: This object is removed when a QoS cube stops being supported by the DIF

Registered As

ERoot Classes(1) 16

Specification

```
ro class QoSCubes
    behavior
        "This class is the container for individual QoSCube objects"
```

```
    contains
        ERoot.Classes.QoSCube (with attribute qosId)
            create-strategy "This object is created when a new QoS cube is
 supported by the DIF"
            delete-strategy "This object is removed when a QoS cube stops
 being supported by the DIF"
    ;


    registered-as ERoot Classes(1) 16
;
```

B.2.59. Class Software

Behavior

This class represents top class of a processing's system software management subtree

Registered As

ERoot Classes(1) 7

Specification

```
ro class Software
    behavior
        "This class represents top class of a processing's system software
 management subtree"

    registered-as ERoot Classes(1) 7
;
```

B.2.60. Class ApplicationConnection

Behavior

An application connection with a peer App/IPCP

Dependencies

- ERoot.Attributes.APNamingInfo

- ERoot.Attributes.CDAPEncoding
- ERoot.Attributes.CDAPSyntax
- ERoot.Attributes.PortId
- ERoot.Attributes.RIBVersion
- ERoot.Policies.Security.AuthenticationPolicy

Class Attributes

- portId → The port-id of the N-1 flow over which the Application connection is established
  - defined by ERoot.Attributes.PortId
  - read enabled
  - write enabled
- abstractSyntax → The version of the CDAP protocol used over this application connection
  - defined by ERoot.Attributes.CDAPSyntax
  - read enabled
  - write enabled
- concreteSyntax → The concrete encoding of the CDAP abstract syntax (GBP, JSON, XML, etc.)
  - defined by ERoot.Attributes.CDAPEncoding
  - read enabled
  - write enabled
- ribVersion → The RIB class/version exposed over this application connection
  - defined by ERoot.Attributes.RIBVersion
  - read enabled
  - write enabled
- localAE → The naming information of the local AP/AE
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled

- remoteAE → The naming information of the remote AP/AE
  - defined by ERoot.Attributes.APNamingInfo
  - read enabled
  - write enabled

Class Operations

- delete
  - invoked to request the termination of the application connection

Class Policies

- ERoot.Policies.Security.AuthenticationPolicy

Registered As

ERoot Classes(1) 46

Specification

```
ro class ApplicationConnection
    behavior
        "An application connection with a peer App/IPCP"

    attributes
        ERoot.Attributes.PortId portId read write
          "The port-id of the N-1 flow over which the Application
 connection is established"
        ERoot.Attributes.CDAPSyntax abstractSyntax read write
          "The version of the CDAP protocol used over this application
 connection"
        ERoot.Attributes.CDAPEncoding concreteSyntax read write
          "The concrete encoding of the CDAP abstract syntax (GBP, JSON,
 XML, etc.)"
        ERoot.Attributes.RIBVersion ribVersion read write
          "The RIB class/version exposed over this application connection"
        ERoot.Attributes.APNamingInfo localAE read write
          "The naming information of the local AP/AE"
        ERoot.Attributes.APNamingInfo remoteAE read write
          "The naming information of the remote AP/AE"
    ;
```

```
    operations
        delete
            "invoked to request the termination of the application
 connection"

    ;

    policies
        ERoot.Policies.Security.AuthenticationPolicy
    ;

    registered-as ERoot Classes(1) 46
;
```

B.2.61. Class WindowBasedFlowControl

Behavior

Window-based flow control state of a DTCP instance

Dependencies

- ERoot.Attributes.Credit
- ERoot.Attributes.QueueLength
- ERoot.Policies.DataTransfer.ReceiverFlowControlPolicy
- ERoot.Policies.DataTransfer.TransmissionControlPolicy

Class Attributes

- initialCredit → Added to the initial sequence number to get right window edge
  - defined by ERoot.Attributes.Credit
  - read enabled
  - write enabled
- maxClosedWindowQLength → Added to the initial sequence number to get right window edge
  - defined by ERoot.Attributes.QueueLength
  - read enabled
  - write enabled

Class Policies

- ERoot.Policies.DataTransfer.TransmissionControlPolicy
- ERoot.Policies.DataTransfer.ReceiverFlowControlPolicy

Registered As

ERoot Classes(1) *55*

Specification

```
ro class WindowBasedFlowControl
    behavior
        "Window-based flow control state of a DTCP instance"

    attributes
        ERoot.Attributes.Credit initialCredit read write
          "Added to the initial sequence number to get right window edge"
        ERoot.Attributes.QueueLength maxClosedWindowQLength read write
          "Added to the initial sequence number to get right window edge"
    ;

    policies
        ERoot.Policies.DataTransfer.TransmissionControlPolicy
        ERoot.Policies.DataTransfer.ReceiverFlowControlPolicy
    ;

    registered-as ERoot Classes(1) 55
;
```

B.2.62. Class FlowControl

Behavior

Flow control state of a DTCP instance

Dependencies

- ERoot.Classes.RateBasedFlowControl
- ERoot.Classes.WindowBasedFlowControl
- ERoot.Policies.DataTransfer.ClosedWindowPolicy
- ERoot.Policies.DataTransfer.FlowControlOverrunPolicy

- ERoot.Policies.DataTransfer.ReceivingFlowControlPolicy
- ERoot.Policies.DataTransfer.ReconcileFlowConflictPolicy

Class Policies

- ERoot.Policies.DataTransfer.ClosedWindowPolicy
- ERoot.Policies.DataTransfer.FlowControlOverrunPolicy
- ERoot.Policies.DataTransfer.ReconcileFlowConflictPolicy
- ERoot.Policies.DataTransfer.ReceivingFlowControlPolicy

Class Containment Relationships

- ERoot.Classes.WindowBasedFlowControl
  - as "wbFlowCtrl"
  - Create: This object is automatically created when FlowControl is created if window-based
  - Delete: This object is automatically destroyed when FlowControl is destroyed
- ERoot.Classes.RateBasedFlowControl
  - as "rbFlowCtrl"
  - Create: This object is automatically created when FlowControl is created if rate-based
  - Delete: This object is automatically destroyed when FlowControl is destroyed

Registered As

ERoot Classes(1) 54

Specification

```
ro class FlowControl
    behavior
        "Flow control state of a DTCP instance"

    policies
        ERoot.Policies.DataTransfer.ClosedWindowPolicy
```

```
        ERoot.Policies.DataTransfer.FlowControlOverrunPolicy
        ERoot.Policies.DataTransfer.ReconcileFlowConflictPolicy
        ERoot.Policies.DataTransfer.ReceivingFlowControlPolicy
    ;

    contains
        ERoot.Classes.WindowBasedFlowControl (as "wbFlowCtrl")
            create-strategy "This object is automatically created when
 FlowControl is created if window-based"
            delete-strategy "This object is automatically destroyed when
 FlowControl is destroyed"
        ERoot.Classes.RateBasedFlowControl (as "rbFlowCtrl")
            create-strategy "This object is automatically created when
 FlowControl is created if rate-based"
            delete-strategy "This object is automatically destroyed when
 FlowControl is destroyed"
    ;


    registered-as ERoot Classes(1) 54
;
```

B.3. RO Attributes

B.3.1. Attribute Jitter

Behavior

In milliseconds, indicates the maximum jitter allowed in this flow. A value of 0 indicates 'do not care'

Type

T_Int

Registered As

ERoot Attributes(2) 15

Specification

```
ro attribute Jitter
    T_Int
```

```
    "In milliseconds, indicates the maximum jitter allowed in this flow. A
 value of 0 indicates 'do not care'"

    registered-as ERoot Attributes(2) 15
;
```

B.3.2. Attribute AddressList

Behavior

A list of IPCP process addresses

Type

T_SetOf T_Int

Registered As

ERoot Attributes(2) 2

Specification

```
ro attribute AddressList
    T_SetOf T_Int

    "A list of IPCP process addresses"

    registered-as ERoot Attributes(2) 2
;
```

B.3.3. Attribute ObjectName

Behavior

name of the managed object instance (uniquely the MO instance within the containment tree)

Type

T_String

Registered As

ERoot Attributes(2) 21

Specification

```
ro attribute ObjectName
    T_String

    "name of the managed object instance (uniquely the MO instance within
 the containment tree)"

    registered-as ERoot Attributes(2) 21
;
```

B.3.4. Attribute FlowState

Behavior

State of a flow. 0 allocation in progress, 1 allocated, 2, deallocation in progress, 3 deallocated

Type

T_Int

Registered As

ERoot Attributes(2) 57

Specification

```
ro attribute FlowState
    T_Int

    "State of a flow. 0 allocation in progress, 1 allocated, 2,
 deallocation in progress, 3 deallocated"

    registered-as ERoot Attributes(2) 57
;
```

B.3.5. Attribute QoSCubeList

Behavior

The list of QoS cubes provided by a DIF and their properties

Type

T_SetOf ERoot.Types.T_QoSCubeData

Dependencies

- ERoot.Types.T_QoSCubeData

Registered As

ERoot Attributes(2) 36

Specification

```
ro attribute QoSCubeList
    T_SetOf ERoot.Types.T_QoSCubeData

    "The list of QoS cubes provided by a DIF and their properties"

    registered-as ERoot Attributes(2) 36
;
```

B.3.6. Attribute TableKey

Behavior

Unique key of an entry in a table

Type

T_Int

Registered As

ERoot Attributes(2) 32

Specification

```
ro attribute TableKey
    T_Int

    "Unique key of an entry in a table"
```

```
    registered-as ERoot Attributes(2) 32
;
```

B.3.7. Attribute DIFNameList

Behavior

A list of DIF names

Type

T_SetOf T_String

Registered As

ERoot Attributes(2) 38

Specification

```
ro attribute DIFNameList
    T_SetOf T_String

    "A list of DIF names"

    registered-as ERoot Attributes(2) 38
;
```

B.3.8. Attribute DAPName

Behavior

uniquely identifies a distributed application member, or the whole distributed application

Type

T_String

Registered As

ERoot Attributes(2) 10

Specification

```
ro attribute DAPName
    T_String

    "uniquely identifies a distributed application member, or the whole
 distributed application"

    registered-as ERoot Attributes(2) 10
;
```

B.3.9. Attribute Rate

Behavior

Rate of a rate-based flow control EFCP connection

Type

T_Int

Registered As

ERoot Attributes(2) 51

Specification

```
ro attribute Rate
    T_Int

    "Rate of a rate-based flow control EFCP connection"

    registered-as ERoot Attributes(2) 51
;
```

B.3.10. Attribute AverageSDUBW

Behavior

Average bandwidth in SDUs/s. A value of 0 means don't care.

Type

T_Int

Registered As

ERoot Attributes(2) 7

Specification

```
ro attribute AverageSDUBW
    T_Int

    "Average bandwidth in SDUs/s. A value of 0 means don't care."

    registered-as ERoot Attributes(2) 7
;
```

B.3.11. Attribute ComputingSystemId

Behavior

uniquely identifies the computing system within the Management Domain

Type

T_Int

Registered As

ERoot Attributes(2) 8

Specification

```
ro attribute ComputingSystemId
    T_Int

    "uniquely identifies the computing system within the Management
 Domain"

    registered-as ERoot Attributes(2) 8
;
```

B.3.12. Attribute QueueLength

Behavior

Length of a queue (in PDUs)

Type

T_Int

Registered As

ERoot Attributes(2) 50

Specification

```
ro attribute QueueLength
    T_Int

    "Length of a queue (in PDUs)"

    registered-as ERoot Attributes(2) 50
;
```

B.3.13. Attribute DataRtxMax

Behavior

Indicates the number of times the retransmission of a PDU will be attempted before some other action must be taken

Type

T_Int

Registered As

ERoot Attributes(2) 54

Specification

```
ro attribute DataRtxMax
    T_Int

    "Indicates the number of times the retransmission of a PDU will be
 attempted before some other action must be taken"

    registered-as ERoot Attributes(2) 54
;
```

B.3.14. Attribute SDUSize

Behavior

Size of SDUs written to /read from an N-1 flow

Type

T_Int

Registered As

ERoot Attributes(2) 37

Specification

```
ro attribute SDUSize
    T_Int

    "Size of SDUs written to /read from an N-1 flow"

    registered-as ERoot Attributes(2) 37
;
```

B.3.15. Attribute ATimer

Behavior

Value of the A Timer (maximum time to acknowledge)

Type

T_Int

Registered As

ERoot Attributes(2) 46

Specification

```
ro attribute ATimer
    T_Int
```

```
    "Value of the A Timer (maximum time to acknowledge)"

    registered-as ERoot Attributes(2) 46
;
```

B.3.16. Attribute AEName

Behavior

name of the application entity

Type

T_String

Registered As

ERoot Attributes(2) 4

Specification

```
ro attribute AEName
    T_String

    "name of the application entity"

    registered-as ERoot Attributes(2) 4
;
```

B.3.17. Attribute DAPInstance

Behavior

uniquely identifies an instance of an application process

Type

T_String

Registered As

ERoot Attributes(2) 9

Specification

```
ro attribute DAPInstance
    T_String

    "uniquely identifies an instance of an application process"

    registered-as ERoot Attributes(2) 9
;
```

B.3.18. Attribute ObjectInstance

Behavior

uniquely identifies the object instance within the containment tree)

Type

T_String

Registered As

ERoot Attributes(2) 20

Specification

```
ro attribute ObjectInstance
    T_String

    "uniquely identifies the object instance within the containment tree)"

    registered-as ERoot Attributes(2) 20
;
```

B.3.19. Attribute Address

Behavior

Address of an IPC Process

Type

T_Int

Registered As

ERoot Attributes(2) 1

Specification

```
ro attribute Address
    T_Int

    "Address of an IPC Process"

    registered-as ERoot Attributes(2) 1
;
```

B.3.20. Attribute IpcProcessId

Behavior

uniquely identifies an IPC Process within a Processing System

Type

T_Int

Registered As

ERoot Attributes(2) 14

Specification

```
ro attribute IpcProcessId
    T_Int

    "uniquely identifies an IPC Process within a Processing System"

    registered-as ERoot Attributes(2) 14
;
```

B.3.21. Attribute SequenceNumber

Behavior

Value of the sequence number rollover threshold

Type

T_Int

Registered As

ERoot Attributes(2) 47

Specification

```
ro attribute SequenceNumber
    T_Int

    "Value of the sequence number rollover threshold"

    registered-as ERoot Attributes(2) 47
;
```

B.3.22. Attribute RIBVersionList

Behavior

A list of RIB versions

Type

T_SetOf T_String

Registered As

ERoot Attributes(2) 43

Specification

```
ro attribute RIBVersionList
    T_SetOf T_String

    "A list of RIB versions"

    registered-as ERoot Attributes(2) 43
;
```

B.3.23. Attribute AEInstance

Behavior

instance of the application entity

Type

T_String

Registered As

ERoot Attributes(2) 3

Specification

```
ro attribute AEInstance
    T_String

    "instance of the application entity"

    registered-as ERoot Attributes(2) 3
;
```

B.3.24. Attribute CEPIdList

Behavior

List of connection endpoint ids (the id of an endpoint of an EFCP connection)

Type

T_SetOf T_Int

Registered As

ERoot Attributes(2) 56

Specification

```
ro attribute CEPIdList
```

```
    T_SetOf T_Int

    "List of connection endpoint ids (the id of an endpoint of an EFCP
 connection)"

    registered-as ERoot Attributes(2) 56
;
```

B.3.25. Attribute APNamingInfo

Behavior

Naming information of an application - process name/instance + entity name/instance

Type

ERoot.Types.T_APNamingInfo

Dependencies

- ERoot.Types.T_APNamingInfo

Registered As

ERoot Attributes(2) *5*

Specification

```
ro attribute APNamingInfo
    ERoot.Types.T_APNamingInfo

    "Naming information of an application - process name/instance + entity
 name/instance"

    registered-as ERoot Attributes(2) 5
;
```

B.3.26. Attribute Time

Behavior

A certain time expressed in integer units

Type

T_Int

Registered As

ERoot Attributes(2) 53

Specification

```
ro attribute Time
    T_Int

    "A certain time expressed in integer units"

    registered-as ERoot Attributes(2) 53
;
```

B.3.27. Attribute ProcessingSystemId

Behavior

uniquely identifies the Processing system within the Computing System

Type

T_String

Registered As

ERoot Attributes(2) 26

Specification

```
ro attribute ProcessingSystemId
    T_String

    "uniquely identifies the Processing system within the Computing
 System"

    registered-as ERoot Attributes(2) 26
;
```

B.3.28. Attribute CDAPSyntax

Behavior

CDAP Abstract syntax, version of the CDAP protocol

Type

T_String

Registered As

ERoot Attributes(2) 39

Specification

```
ro attribute CDAPSyntax
    T_String

    "CDAP Abstract syntax, version of the CDAP protocol"

    registered-as ERoot Attributes(2) 39
;
```

B.3.29. Attribute AverageBW

Behavior

Average bandwidth in bytes/s. A value of 0 means don't care.

Type

T_Int

Registered As

ERoot Attributes(2) 6

Specification

```
ro attribute AverageBW
    T_Int
```

```
    "Average bandwidth in bytes/s. A value of 0 means don't care."

    registered-as ERoot Attributes(2) 6
;
```

B.3.30. Attribute CDAPEncoding

Behavior

The concrete encoding of the CDAP abstract syntax (GBP, JSON, XML, etc.)

Type

T_String

Registered As

ERoot Attributes(2) 40

Specification

```
ro attribute CDAPEncoding
    T_String

    "The concrete encoding of the CDAP abstract syntax (GBP, JSON, XML,
 etc.)"

    registered-as ERoot Attributes(2) 40
;
```

B.3.31. Attribute FlowAllocatorStats

Behavior

Statistics on accepted and rejected incoming/outgoing flow requests

Type

ERoot.Types.T_FlowAllocatorStats

Dependencies

- ERoot.Types.T_FlowAllocatorStats

Registered As

ERoot Attributes(2) *55*

Specification

```
ro attribute FlowAllocatorStats
    ERoot.Types.T_FlowAllocatorStats

    "Statistics on accepted and rejected incoming/outgoing flow requests"

    registered-as ERoot Attributes(2) 55
;
```

B.3.32. Attribute PolicyConfig

Behavior

Represents the configuration of a policy

Type

ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Attributes(2) 22

Specification

```
ro attribute PolicyConfig
    ERoot.Types.T_PolicyConfig

    "Represents the configuration of a policy"

    registered-as ERoot Attributes(2) 22
;
```

B.3.33. Attribute PortIdList

Behavior

A list of port-ids

Type

T_SetOf T_Int

Registered As

ERoot Attributes(2) 24

Specification

```
ro attribute PortIdList
    T_SetOf T_Int

    "A list of port-ids"

    registered-as ERoot Attributes(2) 24
;
```

B.3.34. Attribute SynonymList

Behavior

A set of synonyms for the process whose scope is restricted to the DAF it is a member of and may be structured to facilitate its use with in the DAF.

Type

T_SetOf T_String

Registered As

ERoot Attributes(2) 31

Specification

```
ro attribute SynonymList
```

```
    T_SetOf T_String

    "A set of synonyms for the process whose scope is restricted to the
 DAF"
    " it is a member of and may be structured to facilitate its use with
 in the DAF."

    registered-as ERoot Attributes(2) 31
;
```

B.3.35. Attribute QoSCubeName

Behavior

The name of a QoS-cube

Type

T_String

Registered As

ERoot Attributes(2) 27

Specification

```
ro attribute QoSCubeName
    T_String

    "The name of a QoS-cube"

    registered-as ERoot Attributes(2) 27
;
```

B.3.36. Attribute N1FlowStats

Behavior

Statistics of data sent and received through an N-1 flow

Type

ERoot.Types.T_N1FlowStats

Dependencies

- ERoot.Types.T_N1FlowStats

Registered As

ERoot Attributes(2) 18

Specification

```
ro attribute N1FlowStats
    ERoot.Types.T_N1FlowStats

    "Statistics of data sent and received through an N-1 flow"

    registered-as ERoot Attributes(2) 18
;
```

B.3.37. Attribute QoSId

Behavior

An identifier of a QoS-cube

Type

T_Int

Registered As

ERoot Attributes(2) 28

Specification

```
ro attribute QoSId
    T_Int

    "An identifier of a QoS-cube"

    registered-as ERoot Attributes(2) 28
;
```

B.3.38. Attribute ObjectClass

Behavior

name of the managed object class

Type

T_String

Registered As

ERoot Attributes(2) 19

Specification

```
ro attribute ObjectClass
    T_String

    "name of the managed object class"

    registered-as ERoot Attributes(2) 19
;
```

B.3.39. Attribute RxQueueInfo

Behavior

Information about the rx queue

Type

ERoot.Types.T_QueueInfo

Dependencies

- ERoot.Types.T_QueueInfo

Registered As

ERoot Attributes(2) 30

Specification

```
ro attribute RxQueueInfo
    ERoot.Types.T_QueueInfo

    "Information about the rx queue"

    registered-as ERoot Attributes(2) 30
;
```

B.3.40. Attribute CEPId

Behavior

Connection endpoint id (the id of an endpoint of an EFCP connection)

Type

T_Int

Registered As

ERoot Attributes(2) 45

Specification

```
ro attribute CEPId
    T_Int

    "Connection endpoint id (the id of an endpoint of an EFCP connection)"

    registered-as ERoot Attributes(2) 45
;
```

B.3.41. Attribute DataTransferConstants

Behavior

DIF-wide parameters that define the concrete syntax of EFCP for this DIF and other DIF-wide values

Type

ERoot.Types.T_DataTransferConstants

Dependencies

- ERoot.Types.T_DataTransferConstants

Registered As

ERoot Attributes(2) 44

Specification

```
ro attribute DataTransferConstants
    ERoot.Types.T_DataTransferConstants

    "DIF-wide parameters that define the concrete syntax of EFCP for this
 DIF and other DIF-wide values"

    registered-as ERoot Attributes(2) 44
;
```

B.3.42. Attribute UnderlyingDIFs

Behavior

The names of the N-1 DIFs in common with the neighbor IPC Process

Type

T_SetOf T_String

Registered As

ERoot Attributes(2) 34

Specification

```
ro attribute UnderlyingDIFs
    T_SetOf T_String
```

```
    "The names of the N-1 DIFs in common with the neighbor IPC Process"

    registered-as ERoot Attributes(2) 34
;
```

B.3.43. Attribute Retries

Behavior

Counter on the number of times an action may be / has been retried

Type

T_Int

Registered As

ERoot Attributes(2) 58

Specification

```
ro attribute Retries
    T_Int

    "Counter on the number of times an action may be / has been retried"

    registered-as ERoot Attributes(2) 58
;
```

B.3.44. Attribute QueueId

Behavior

Identity of a queue

Type

T_Int

Registered As

ERoot Attributes(2) 29

Specification

```
ro attribute QueueId
    T_Int

    "Identity of a queue"

    registered-as ERoot Attributes(2) 29
;
```

B.3.45. Attribute UnderlyingFlows

Behavior

The port-id of the N-1 flow used to talk to the neighbor

Type

T_SetOf T_Int

Registered As

ERoot Attributes(2) 35

Specification

```
ro attribute UnderlyingFlows
    T_SetOf T_Int

    "The port-id of the N-1 flow used to talk to the neighbor"

    registered-as ERoot Attributes(2) 35
;
```

B.3.46. Attribute DTPConfig

Behavior

Configuration of a DTP instance

Type

ERoot.Types.T_DTPConfig

Dependencies

- ERoot.Types.T_DTPConfig

Registered As

ERoot Attributes(2) 13

Specification

```
ro attribute DTPConfig
    ERoot.Types.T_DTPConfig

    "Configuration of a DTP instance"

    registered-as ERoot Attributes(2) 13
;
```

B.3.47. Attribute PortId

Behavior

Port-id of a flow

Type

T_Int

Registered As

ERoot Attributes(2) 23

Specification

```
ro attribute PortId
    T_Int

    "Port-id of a flow"

    registered-as ERoot Attributes(2) 23
;
```

B.3.48. Attribute EFCPConnectionStats

Behavior

Statistics of an EFCP connection

Type

ERoot.Types.T_EFCPConnectionStats

Dependencies

- ERoot.Types.T_EFCPConnectionStats

Registered As

ERoot Attributes(2) 48

Specification

```
ro attribute EFCPConnectionStats
    ERoot.Types.T_EFCPConnectionStats

    "Statistics of an EFCP connection"

    registered-as ERoot Attributes(2) 48
;
```

B.3.49. Attribute DTCPConfig

Behavior

Configuration of a DTCP instance

Type

ERoot.Types.T_DTCPConfig

Dependencies

- ERoot.Types.T_DTCPConfig

Registered As

ERoot Attributes(2) 12

Specification

```
ro attribute DTCPConfig
    ERoot.Types.T_DTCPConfig

    "Configuration of a DTCP instance"

    registered-as ERoot Attributes(2) 12
;
```

B.3.50. Attribute TxQueueInfo

Behavior

Information about the tx queue

Type

ERoot.Types.T_QueueInfo

Dependencies

- ERoot.Types.T_QueueInfo

Registered As

ERoot Attributes(2) 33

Specification

```
ro attribute TxQueueInfo
    ERoot.Types.T_QueueInfo

    "Information about the tx queue"

    registered-as ERoot Attributes(2) 33
;
```

B.3.51. Attribute ForwardingDiscriminatorId

Behavior

Identity of a forwarding discriminator

Type

T_Int

Registered As

ERoot Attributes(2) 59

Specification

```
ro attribute ForwardingDiscriminatorId
    T_Int

    "Identity of a forwarding discriminator"

    registered-as ERoot Attributes(2) 59
;
```

B.3.52. Attribute TimePeriod

Behavior

Indicates the length of time in microseconds for pacing rate-based flow control

Type

T_Int

Registered As

ERoot Attributes(2) 52

Specification

```
ro attribute TimePeriod
    T_Int
```

```
    "Indicates the length of time in microseconds for pacing rate-based
  flow control"

    registered-as ERoot Attributes(2) 52
;
```

B.3.53. Attribute MasterAgent

Behavior

True if the Management Agent is the master of this processing system

Type

T_Boolean

Registered As

ERoot Attributes(2) 17

Specification

```
ro attribute MasterAgent
    T_Boolean

    "True if the Management Agent is the master of this processing system"

    registered-as ERoot Attributes(2) 17
;
```

B.3.54. Attribute RIBVersion

Behavior

The RIB class/version exposed over an application connection

Type

T_String

Registered As

ERoot Attributes(2) 41

Specification

```
ro attribute RIBVersion
    T_String

    "The RIB class/version exposed over an application connection"

    registered-as ERoot Attributes(2) 41
;
```

B.3.55. Attribute Credit

Behavior

Credit of a window-based flow control EFCP connection

Type

T_Int

Registered As

ERoot Attributes(2) 49

Specification

```
ro attribute Credit
    T_Int

    "Credit of a window-based flow control EFCP connection"

    registered-as ERoot Attributes(2) 49
;
```

B.3.56. Attribute FlowProperties

Behavior

The properties of an N-1 flow (capacity, delay, loss, etc.)

Type

ERoot.Types.T_FlowProperties

Dependencies

- ERoot.Types.T_FlowProperties

Registered As

ERoot Attributes(2) 42

Specification

```
ro attribute FlowProperties
    ERoot.Types.T_FlowProperties

    "The properties of an N-1 flow (capacity, delay, loss, etc.)"

    registered-as ERoot Attributes(2) 42
;
```

B.3.57. Attribute ManagementAgentId

Behavior

uniquely identifies a Management Agent within a Processing System

Type

T_Int

Registered As

ERoot Attributes(2) 16

Specification

```
ro attribute ManagementAgentId
    T_Int

    "uniquely identifies a Management Agent within a Processing System"

    registered-as ERoot Attributes(2) 16
;
```

B.3.58. Attribute PortStarted

Behavior

True if the port is started, false if it is stopped

Type

T_Boolean

Registered As

ERoot Attributes(2) 25

Specification

```
ro attribute PortStarted
    T_Boolean

    "True if the port is started, false if it is stopped"

    registered-as ERoot Attributes(2) 25
;
```

B.3.59. Attribute Delay

Behavior

In milliseconds, indicates the maximum delay allowed in this flow. A value of 0 indicates 'do not care'

Type

T_Int

Registered As

ERoot Attributes(2) 11

Specification

```
ro attribute Delay
```

```
    T_Int

    "In milliseconds, indicates the maximum delay allowed in this flow. A
 value of 0 indicates 'do not care'"

    registered-as ERoot Attributes(2) 11
;
```

B.4. RO Notifications

B.4.1. Notification CreateFlow

Behavior

Triggered when an N-Flow is allocated by the IPC Process

Attributes

- localPortId → The port-id of the flow
  - as type T_Int
- localAppName → The local application entity that is using the N flow.
  - as type ERoot.Types.T_APNamingInfo
- remotePortId → The portId of the flow at the remote IPC Process
  - as type T_Int
- remoteAppName → The remote application entity that is using the N flow
  - as type ERoot.Types.T_APNamingInfo
- flowSpec → The characteristics of the N flow (loss, delay, reliability, in order-delivery of SDUs, etc)
  - as type ERoot.Types.T_FlowProperties
- currentCEPId → The connection-endpoint currently used by the EFCP connection supporting this flow
  - as type T_Int
- reservedCEPIds → The connection-endpoint ids reserved for the connections that will support this flow in this IPC Process

    - as type T_SetOf T_Int
- state → 0 allocation in progress, 1 allocated, 2, deallocation in progress, 3 deallocated
    - as type T_Int
- createFlowRetries → The current number of attempts for allocating this flow
    - as type T_Int
- maxCreateFlowRetries → The maximum number of attempts for allocating the flows
    - as type T_Int

Registered As

ERoot Notifications(3) *5*

Specification

```
ro notification CreateFlow
    behavior
        "Triggered when an N-Flow is allocated by the IPC Process"

    attributes
        localPortId T_Int
            "The port-id of the flow"
        localAppName ERoot.Types.T_APNamingInfo
            "The local application entity that is using the N flow."
        remotePortId T_Int
            "The portId of the flow at the remote IPC Process"
        remoteAppName ERoot.Types.T_APNamingInfo
            "The remote application entity that is using the N flow"
        flowSpec ERoot.Types.T_FlowProperties
            "The characteristics of the N flow (loss, delay, reliability,
 in order-delivery of SDUs, etc)"
        currentCEPId T_Int
            "The connection-endpoint currently used by the EFCP connection
 supporting this flow"
        reservedCEPIds T_SetOf T_Int
            "The connection-endpoint ids reserved for the connections that
 will support this flow in this IPC Process"
        state T_Int
```

```
            "0 allocation in progress, 1 allocated, 2, deallocation in
 progress, 3 deallocated"
        createFlowRetries T_Int
            "The current number of attempts for allocating this flow"
        maxCreateFlowRetries T_Int
            "The maximum number of attempts for allocating the flows"
    ;

    registered-as ERoot Notifications(3) 5
;
```

B.4.2. Notification DeleteNeighbor

Behavior

Triggered when the App / IPC Process looses contact with a neighbor

Attributes

- processName → The neighbor IPCP's name
  - as type T_String
- processInstance → The neighbor IPCP's instance
  - as type T_String
- address → Address of the neighbor IPCP
  - as type T_Int

Registered As

ERoot Notifications(3) 8

Specification

```
ro notification DeleteNeighbor
    behavior
        "Triggered when the App / IPC Process looses contact with a
 neighbor"

    attributes
        processName T_String
            "The neighbor IPCP's name"
```

```
        processInstance T_String
            "The neighbor IPCP's instance"
        address T_Int
            "Address of the neighbor IPCP"
    ;

    registered-as ERoot Notifications(3) 8
;
```

B.4.3. Notification DeleteUnderlyingRegistration

Behavior

Triggered when the App/IPCP unregisters from an N-1 DIF

Attributes

- applicationEntity → The naming information of this Application Entity
  - as type ERoot.Types.T_APNamingInfo
- difName → The name of the DIF where this Application Entity has unregistered
  - as type T_String

Registered As

ERoot Notifications(3) 14

Specification

```
ro notification DeleteUnderlyingRegistration
    behavior
        "Triggered when the App/IPCP unregisters from an N-1 DIF"

    attributes
        applicationEntity ERoot.Types.T_APNamingInfo
            "The naming information of this Application Entity"
        difName T_String
            "The name of the DIF where this Application Entity has
 unregistered"
    ;

    registered-as ERoot Notifications(3) 14
```

```
;
```

B.4.4. Notification DeleteUnderlyingFlow

Behavior

Triggered when an N-1 flow used by an App/IPCP has been deallocated

Attributes

- portId → The id of the N-1 flow
  - as type T_Int

Registered As

ERoot Notifications(3) 16

Specification

```
ro notification DeleteUnderlyingFlow
    behavior
        "Triggered when an N-1 flow used by an App/IPCP has been
 deallocated"

    attributes
        portId T_Int
            "The id of the N-1 flow"
    ;

    registered-as ERoot Notifications(3) 16
;
```

B.4.5. Notification DeleteQoSCube

Behavior

Triggered when the App / IPC Process stops supporting a QoS Cube

Attributes

- qosId → If of the QoS cube
  - as type T_Int

Registered As

ERoot Notifications(3) 10

Specification

```
ro notification DeleteQoSCube
    behavior
        "Triggered when the App / IPC Process stops supporting a QoS Cube"

    attributes
        qosId T_Int
            "If of the QoS cube"
    ;

    registered-as ERoot Notifications(3) 10
;
```

B.4.6. Notification DeleteFlow

Behavior

Triggered when an N-Flow is deallocated by the IPC Process

Attributes

- portId → The port-id of the flow deallocated
  - as type T_Int

Registered As

ERoot Notifications(3) 6

Specification

```
ro notification DeleteFlow
    behavior
        "Triggered when an N-Flow is deallocated by the IPC Process"

    attributes
        portId T_Int
            "The port-id of the flow deallocated"
```

```
    ;

    registered-as ERoot Notifications(3) 6
;
```

B.4.7. Notification CreateRMTN1Flow

Behavior

Triggered when the RMT creates an N-1 flow data structure and associated queues

Attributes

- portId → The portId of the N-1 flow
  - as type T_Int
- portUp → True if the N-1 flow is started, false otherwise
  - as type T_Boolean
- portStats → Statistics of the N-1 flow
  - as type ERoot.Types.T_N1FlowStats
- queueConfigs → Configurations of the RMT queues for this port
  - as type T_SetOf ERoot.Types.T_RMTQueuePairState

Registered As

ERoot Notifications(3) 11

Specification

```
ro notification CreateRMTN1Flow
    behavior
        "Triggered when the RMT creates an N-1 flow data structure and
 associated queues"

    attributes
        portId T_Int
            "The portId of the N-1 flow"
        portUp T_Boolean
            "True if the N-1 flow is started, false otherwise"
        portStats ERoot.Types.T_N1FlowStats
```

```
            "Statistics of the N-1 flow"
        queueConfigs T_SetOf ERoot.Types.T_RMTQueuePairState
            "Configurations of the RMT queues for this port"
    ;

    registered-as ERoot Notifications(3) 11
;
```

B.4.8. Notification CreateQoSCube

Behavior

Triggered when the IPCP acquires support for a new QoS cube

Attributes

- name → The name of the QoS cube
  - as type T_String
- qosId → Id of the QoS cube
  - as type T_Int
- averageBW → Average bandwidth in bytes/s. A value of 0 means don't care.
  - as type T_Int
- averageSDUBW → Average bandwidth in SDUs/s. A value of 0 means don't care
  - as type T_Int
- delay → In milliseconds, indicates the maximum delay allowed in this flow. A value of 0 indicates 'do not care'
  - as type T_Int
- jitter → In milliseconds, indicates the maximum jitter allowed in this flow. A value of 0 indicates 'do not care'
  - as type T_Int
- dtpConfig → The configuration of DTP for this QoS cube
  - as type ERoot.Types.T_DTPConfig
- dtcpConfig → The configuration of DTCP for this QoS cube
  - as type ERoot.Types.T_DTCPConfig

Dependencies

- ERoot.Types.T_DTCPConfig

Registered As

ERoot Notifications(3) 9

Specification

```
ro notification CreateQoSCube
    behavior
        "Triggered when the IPCP acquires support for a new QoS cube"

    attributes
        name T_String
            "The name of the QoS cube"
        qosId T_Int
            "Id of the QoS cube"
        averageBW T_Int
            "Average bandwidth in bytes/s. A value of 0 means don't care."
        averageSDUBW T_Int
            "Average bandwidth in SDUs/s. A value of 0 means don't care"
        delay T_Int
            "In milliseconds, indicates the maximum delay allowed in this
 flow. A value of 0 indicates 'do not care'"
        jitter T_Int
            "In milliseconds, indicates the maximum jitter allowed in this
 flow. A value of 0 indicates 'do not care'"
        dtpConfig ERoot.Types.T_DTPConfig
            "The configuration of DTP for this QoS cube"
        dtcpConfig ERoot.Types.T_DTCPConfig
            "The configuration of DTCP for this QoS cube"
    ;

    registered-as ERoot Notifications(3) 9
;
```

B.4.9. Notification DeleteRMTN1Flow

Behavior

Triggered when the RMT destroys the N-1 port structure and associated queues

Attributes

- portId → If of N-1 port/flow
  - as type T_Int

Registered As

ERoot Notifications(3) 12

Specification

```
ro notification DeleteRMTN1Flow
    behavior
        "Triggered when the RMT destroys the N-1 port structure and
 associated queues"

    attributes
        portId T_Int
            "If of N-1 port/flow"
    ;

    registered-as ERoot Notifications(3) 12
;
```

B.4.10. Notification CreateNeighbor

Behavior

Triggered when the App/IPC Process obtains a new neighbor

Attributes

- processName → The neighbor IPCP's name
  - as type T_String
- processInstance → The neighbor IPCP's instance
  - as type T_String
- address → Address of the neighbor IPCP
  - as type T_Int
- underDIFs → The names of the N-1 DIFs in common with the neighbor IPC Process

    - as type T_SetOf T_String
- underFlows → The port-id of the N-1 flow used to talk to the neighbor
    - as type T_SetOf T_Int

Registered As

ERoot Notifications(3) 7

Specification

```
ro notification CreateNeighbor
    behavior
        "Triggered when the App/IPC Process obtains a new neighbor"

    attributes
        processName T_String
            "The neighbor IPCP's name"
        processInstance T_String
            "The neighbor IPCP's instance"
        address T_Int
            "Address of the neighbor IPCP"
        underDIFs T_SetOf T_String
            "The names of the N-1 DIFs in common with the neighbor IPC
 Process"
        underFlows T_SetOf T_Int
            "The port-id of the N-1 flow used to talk to the neighbor"
    ;

    registered-as ERoot Notifications(3) 7
;
```

B.4.11. Notification CreateEFCPConnection

Behavior

Triggered when an EFCP connection is created

Attributes

- srcCepId → The source connection endpoint id
    - as type T_Int

- destCepId → The destination connection endpoint id
  - as type T_Int
- srcAddress → The address of the source IPCP of this connection
  - as type T_Int
- destAddress → The address of the destination IPCP of this connection
  - as type T_Int
- qosId → The id of the QoS cube this connection belongs to
  - as type T_Int
- portId → The id of the flow supported by this EFCP connection
  - as type T_Int
- initialATimer → The initial A timer for this connection
  - as type T_Int
- seqNumThreshold → The sequence number rollover threshold for this connection
  - as type T_Int
- rcvTimeInacPolicy → Configuration of the receiver timer inactivity policy
  - as type ERoot.Types.T_PolicyConfig
- sndTimeInacPolicy → Configuration of the sender timer inactivity policy
  - as type ERoot.Types.T_PolicyConfig
- initSeqNumPolicy → Configuration of the initial sequence number policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Notifications(3) 2

Specification

```
ro notification CreateEFCPConnection
    behavior
        "Triggered when an EFCP connection is created"

    attributes
        srcCepId T_Int
            "The source connection endpoint id"
        destCepId T_Int
            "The destination connection endpoint id"
        srcAddress T_Int
            "The address of the source IPCP of this connection"
        destAddress T_Int
            "The address of the destination IPCP of this connection"
        qosId T_Int
            "The id of the QoS cube this connection belongs to"
        portId T_Int
            "The id of the flow supported by this EFCP connection"
        initialATimer T_Int
            "The initial A timer for this connection"
        seqNumThreshold T_Int
            "The sequence number rollover threshold for this connection"
        rcvTimeInacPolicy ERoot.Types.T_PolicyConfig
            "Configuration of the receiver timer inactivity policy"
        sndTimeInacPolicy ERoot.Types.T_PolicyConfig
            "Configuration of the sender timer inactivity policy"
        initSeqNumPolicy ERoot.Types.T_PolicyConfig
            "Configuration of the initial sequence number policy"
    ;

    registered-as ERoot Notifications(3) 2
;
```

B.4.12. Notification CreateUnderlyingRegistration

Behavior

Triggered when the App/IPCP registers to an N-1 DIF

Attributes

- applicationEntity → The naming information of this Application Entity
  - as type ERoot.Types.T_APNamingInfo

- difName → The name of the DIF where this Application Entity has registered
  - as type T_String

Registered As

ERoot Notifications(3) 13

Specification

```
ro notification CreateUnderlyingRegistration
    behavior
        "Triggered when the App/IPCP registers to an N-1 DIF"

    attributes
        applicationEntity ERoot.Types.T_APNamingInfo
            "The naming information of this Application Entity"
        difName T_String
            "The name of the DIF where this Application Entity has
 registered"
    ;

    registered-as ERoot Notifications(3) 13
;
```

B.4.13. Notification CreateUnderlyingFlow

Behavior

Triggered when the App/IPCP is allocated an N-1 flow

Attributes

- portId → The port-id of the N-1 flow
  - as type T_Int
- localAppName → The names of the application that requested the flow
  - as type ERoot.Types.T_APNamingInfo
- remoteAppNAme → The names of the application that is the target of the flow

  - as type ERoot.Types.T_APNamingInfo
- flowSpec → The characteristics of this flow
  - as type ERoot.Types.T_FlowProperties
- difName → The name of the N-1 DIF providing the flow
  - as type T_String

Registered As

ERoot Notifications(3) 15

Specification

```
ro notification CreateUnderlyingFlow
    behavior
        "Triggered when the App/IPCP is allocated an N-1 flow"

    attributes
        portId T_Int
            "The port-id of the N-1 flow"
        localAppName ERoot.Types.T_APNamingInfo
            "The names of the application that requested the flow"
        remoteAppNAme ERoot.Types.T_APNamingInfo
            "The names of the application that is the target of the flow"
        flowSpec ERoot.Types.T_FlowProperties
            "The characteristics of this flow"
        difName T_String
            "The name of the N-1 DIF providing the flow"
    ;

    registered-as ERoot Notifications(3) 15
;
```

B.4.14. Notification DeleteEFCPConnection

Behavior

Triggered when an EFCP connection is destroyed

Attributes

- srcCepId → The source connection endpoint id

- as type T_Int

Registered As

ERoot Notifications(3) 3

Specification

```
ro notification DeleteEFCPConnection
    behavior
        "Triggered when an EFCP connection is destroyed"

    attributes
        srcCepId T_Int
            "The source connection endpoint id"
    ;

    registered-as ERoot Notifications(3) 3
;
```

B.4.15. Notification CreateIPCProcess

Behavior

Triggered when an IPC Process is created

Attributes

- namingInfo → The naming information of the IPC Process
  - as type ERoot.Types.T_APNamingInfo
- ipcpId → The id of the IPC Process within the computing system
  - as type T_Int

Registered As

ERoot Notifications(3) 1

Specification

```
ro notification CreateIPCProcess
```

```
    behavior
        "Triggered when an IPC Process is created"

    attributes
        namingInfo ERoot.Types.T_APNamingInfo
            "The naming information of the IPC Process"
        ipcpId T_Int
            "The id of the IPC Process within the computing system"
    ;

    registered-as ERoot Notifications(3) 1
;
```

B.4.16. Notification FlowQoSViolated

Behavior

Triggered when an EFCP connection can't maintain the QoS of a flow

Attributes

- portId → The port-id of the flow
  - as type T_Int
- errorCode → Error code indicating more information about the QoS violation
  - as type T_Int
- flowSpec → Characteristics of the flow service
  - as type ERoot.Types.T_FlowProperties

Dependencies

- ERoot.Types.T_FlowProperties

Registered As

ERoot Notifications(3) 4

Specification

```
ro notification FlowQoSViolated
```

```
    behavior
        "Triggered when an EFCP connection can't maintain the QoS of a
 flow"

    attributes
        portId T_Int
            "The port-id of the flow"
        errorCode T_Int
            "Error code indicating more information about the QoS
 violation"
        flowSpec ERoot.Types.T_FlowProperties
            "Characteristics of the flow service"
    ;

    registered-as ERoot Notifications(3) 4
;
```

B.5. RO Policies

B.5.1. Policy Protection

Behavior

The root of protection policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy Protection
    behavior
        "The root of protection policies, has no explicit behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 7
```

```
;
```

B.5.2. Policy Security

Behavior

The root of security policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy Security
    behavior
        "The root of security policies, has no explicit behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 2
;
```

B.5.3. Policy CredentialManagementPolicy

Behavior

Policy to manage the credentials of an IPC Process

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Security

Super Class

ERoot.Policies.Security

Specification

```
ro policy CredentialManagementPolicy
    behavior
        "Policy to manage the credentials of an IPC Process"

    extends ERoot.Policies.Security
    registered-as ERoot Policies(4) Security(2) 3
;
```

B.5.4. Policy AccessControlPolicy

Behavior

Policy to determine the access control to a DIF.

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Security

Super Class

ERoot.Policies.Security

Specification

```
ro policy AccessControlPolicy
    behavior
        "Policy to determine the access control to a DIF."

    extends ERoot.Policies.Security
    registered-as ERoot Policies(4) Security(2) 1
;
```

B.5.5. Policy AuditingPolicy

Behavior

Policy to audit the actions happening in a DIF

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Security

Super Class

ERoot.Policies.Security

Specification

```
ro policy AuditingPolicy
    behavior
        "Policy to audit the actions happening in a DIF"

    extends ERoot.Policies.Security
    registered-as ERoot Policies(4) Security(2) 2
;
```

B.5.6. Policy CryptographicProtectionPolicy

Behavior

Policy to manage the cyrptographically protect (encryption, HMAC) a PDU before sending it to the lower DIF

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Security

Super Class

ERoot.Policies.Security

Specification

```
ro policy CryptographicProtectionPolicy
    behavior
        "Policy to manage the cyrptographically protect (encryption, HMAC)
 a PDU before sending it to the lower DIF"
```

```
    extends ERoot.Policies.Security
    registered-as ERoot Policies(4) Security(2) 4
;
```

B.5.7. Policy AuthenticationPolicy

Behavior

Policy to authentication to a peer application/IPCP

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Security

Super Class

ERoot.Policies.Security

Specification

```
ro policy AuthenticationPolicy
    behavior
        "Policy to authentication to a peer application/IPCP"

    extends ERoot.Policies.Security
    registered-as ERoot Policies(4) Security(2) 5
;
```

B.5.8. Policy PDUForwardingPolicy

Behavior

The PDU forwarding policy, used to decide how to forward PDUs to N-1 port-ids

Dependencies

- ERoot.Attributes.PolicyConfig

- ERoot.Policies.RelayingAndMultiplexing

Super Class

ERoot.Policies.RelayingAndMultiplexing

Specification

```
ro policy PDUForwardingPolicy
    behavior
        "The PDU forwarding policy, used to decide how to forward PDUs to
 N-1 port-ids"

    extends ERoot.Policies.RelayingAndMultiplexing
    registered-as ERoot Policies(4) RelayingAndMultiplexing(5) 2
;
```

B.5.9. Policy PDUSchedulingPolicy

Behavior

The PDU scheduling policy, used to decide how to process PDUs queued in the RMT queues

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RelayingAndMultiplexing

Super Class

ERoot.Policies.RelayingAndMultiplexing

Specification

```
ro policy PDUSchedulingPolicy
    behavior
        "The PDU scheduling policy, used to decide how to process PDUs
 queued in the RMT queues"

    extends ERoot.Policies.RelayingAndMultiplexing
```

```
    registered-as ERoot Policies(4) RelayingAndMultiplexing(5) 1
;
```

B.5.10. Policy NamespaceManagement

Behavior

The root of namespace management policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy NamespaceManagement
    behavior
        "The root of namespace management policies, has no explicit
 behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 3
;
```

B.5.11. Policy LifetimeLimitingPolicy

Behavior

The lifetime limiting policy, makes sure PDUs respect the maximum PDU lifetime in the DIF (via hopcounts, etc.)

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Protection

Super Class

ERoot.Policies.Protection

Specification

```
ro policy LifetimeLimitingPolicy
    behavior
        "The lifetime limiting policy, makes sure PDUs respect the maximum
 PDU lifetime in the DIF (via hopcounts, etc.)"

    extends ERoot.Policies.Protection
    registered-as ERoot Policies(4) Protection(7) 2
;
```

B.5.12. Policy CompressionPolicy

Behavior

The compression policy, to compress a PDU before sending it to the layer below

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Protection

Super Class

ERoot.Policies.Protection

Specification

```
ro policy CompressionPolicy
    behavior
        "The compression policy, to compress a PDU before sending it to
 the layer below"

    extends ERoot.Policies.Protection
    registered-as ERoot Policies(4) Protection(7) 3
;
```

B.5.13. Policy ErrorCheckPolicy

Behavior

The error check policy, verifies that a received PDU contains no errors (via CRCs, FECs, etc.)

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Protection

Super Class

ERoot.Policies.Protection

Specification

```
ro policy ErrorCheckPolicy
    behavior
        "The error check policy, verifies that a received PDU contains no
 errors (via CRCs, FECs, etc.)"

    extends ERoot.Policies.Protection
    registered-as ERoot Policies(4) Protection(7) 1
;
```

B.5.14. Policy RoutingPolicy

Behavior

The routing policy, used to compute next hop IPCPs to destination addresses in the DIF

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.ResourceAllocation

Super Class

ERoot.Policies.ResourceAllocation

Specification

```
ro policy RoutingPolicy
    behavior
        "The routing policy, used to compute next hop IPCPs to destination
 addresses in the DIF"

    extends ERoot.Policies.ResourceAllocation
    registered-as ERoot Policies(4) ResourceAllocation(4) 2
;
```

B.5.15. Policy PDUFTGenerationPolicy

Behavior

Computes N-1 port-ids to forward PDUs addressed to any IPCP in the DIF

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.ResourceAllocation

Super Class

ERoot.Policies.ResourceAllocation

Specification

```
ro policy PDUFTGenerationPolicy
    behavior
        "Computes N-1 port-ids to forward PDUs addressed to any IPCP in
 the DIF"

    extends ERoot.Policies.ResourceAllocation
    registered-as ERoot Policies(4) ResourceAllocation(4) 3
;
```

B.5.16. Policy ResourceAllocatorPolicy

Behavior

Main resource allocation policy

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.ResourceAllocation

Super Class

ERoot.Policies.ResourceAllocation

Specification

```
ro policy ResourceAllocatorPolicy
    behavior
        "Main resource allocation policy"

    extends ERoot.Policies.ResourceAllocation
    registered-as ERoot Policies(4) ResourceAllocation(4) 1
;
```

B.5.17. Policy SeqNumRolloverPolicy

Behavior

This policy is used when the SeqRollOverThres event occurs and action may be required by the Flow Allocator to modify the bindings between connection-endpoint-ids and port-ids

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.FlowAllocation

Super Class

ERoot.Policies.FlowAllocation

Specification

```
ro policy SeqNumRolloverPolicy
    behavior
```

```
        "This policy is used when the SeqRollOverThres event occurs and
 action may be required by the Flow Allocator to modify the bindings
 between connection-endpoint-ids and port-ids"

    extends ERoot.Policies.FlowAllocation
    registered-as ERoot Policies(4) FlowAllocation(11) 4
;
```

B.5.18. Policy AllocateNotifyPolicy

Behavior

This policy determines when the requesting application is given an Allocate_Response primitive

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.FlowAllocation

Super Class

ERoot.Policies.FlowAllocation

Specification

```
ro policy AllocateNotifyPolicy
    behavior
        "This policy determines when the requesting application is given
 an Allocate_Response primitive"

    extends ERoot.Policies.FlowAllocation
    registered-as ERoot Policies(4) FlowAllocation(11) 1
;
```

B.5.19. Policy NewFlowRequestPolicy

Behavior

This policy is used when converting an Allocate Request into a create_flow request

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.FlowAllocation

Super Class

ERoot.Policies.FlowAllocation

Specification

```
ro policy NewFlowRequestPolicy
    behavior
        "This policy is used when converting an Allocate Request into a
 create_flow request"

    extends ERoot.Policies.FlowAllocation
    registered-as ERoot Policies(4) FlowAllocation(11) 3
;
```

B.5.20. Policy AllocateRetryPolicy

Behavior

This policy is used when the destination has refused the create_flow request, and the FAI can overcome the cause for refusal and try again

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.FlowAllocation

Super Class

ERoot.Policies.FlowAllocation

Specification

```
ro policy AllocateRetryPolicy
    behavior
```

```
        "This policy is used when the destination has refused the
 create_flow request, and the FAI can overcome the cause for refusal and
 try again"

    extends ERoot.Policies.FlowAllocation
    registered-as ERoot Policies(4) FlowAllocation(11) 2
;
```

B.5.21. Policy SenderAckPolicy

Behavior

This policy is executed by the Sender and provides the Sender with some discretion on when PDUs may be deleted from the ReTransmissionQ

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy SenderAckPolicy
    behavior
        "This policy is executed by the Sender and provides the
 Sender with some discretion on when PDUs may be deleted from the
 ReTransmissionQ"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 17
;
```

B.5.22. Policy ReceivingAckListPolicy

Behavior

This policy is executed by the Sender and provides the Sender with some discretion on when PDUs may be deleted from the ReTransmissionQ

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy ReceivingAckListPolicy
    behavior
        "This policy is executed by the Sender and provides the
 Sender with some discretion on when PDUs may be deleted from the
 ReTransmissionQ"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 18
;
```

B.5.23. Policy ReceiverTimerInactivityPolicy

Behavior

If no PDUs arrive in this time period, the receiver should expect a DRF (Data Run Flag) in the next Transfer PDU. If not, something is very wrong. The timeout value should generally be set to 3(MPL+R+A

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy ReceiverTimerInactivityPolicy
    behavior
```

```
        "If no PDUs arrive in this time period, the receiver should expect
 a DRF (Data Run Flag) in the next Transfer PDU. If not, something is very
 wrong. The timeout value should generally be set to 3(MPL+R+A"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 2
;
```

B.5.24. Policy TransmissionControlPolicy

Behavior

This policy is used when there are conditions that warrant sending fewer PDUs than allowed by the sliding window flow control, e.g. the ECN bit is set

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy TransmissionControlPolicy
    behavior
        "This policy is used when there are conditions that warrant
 sending fewer PDUs than allowed by the sliding window flow control, e.g.
 the ECN bit is set"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 11
;
```

B.5.25. Policy ReceiverFlowControlPolicy

Behavior

This policy is invoked when a Transfer PDU is received to give the receiving PM an opportunity to update the flow control allocations

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy ReceiverFlowControlPolicy
    behavior
        "This policy is invoked when a Transfer PDU is received to give
 the receiving PM an opportunity to update the flow control allocations"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 12
;
```

B.5.26. Policy RTTEstimatorPolicy

Behavior

This policy is executed by the sender to estimate the duration of the retransmission timer

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy RTTEstimatorPolicy
    behavior
```

```
        "This policy is executed by the sender to estimate the duration of
 the retransmission timer"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 6
;
```

B.5.27. Policy RateReductionPolicy

Behavior

This policy allows an alternate action when using rate-based flow control and the number of free buffers is getting low

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy RateReductionPolicy
    behavior
        "This policy allows an alternate action when using rate-based flow
 control and the number of free buffers is getting low"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 15
;
```

B.5.28. Policy ClosedWindowPolicy

Behavior

This policy is used with flow control to determine the action to be taken when the receiver has not extended more credit to allow the sender to send more PDUs

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy ClosedWindowPolicy
    behavior
        "This policy is used with flow control to determine the action
 to be taken when the receiver has not extended more credit to allow the
 sender to send more PDUs"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 7
;
```

B.5.29. Policy NoRateSlowDownPolicy

Behavior

This policy is used to momentarily lower the send rate below the rate allowed

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy NoRateSlowDownPolicy
```

```
    behavior
        "This policy is used to momentarily lower the send rate below the
 rate allowed"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 13
;
```

B.5.30. Policy ReceivingFlowControlPolicy

Behavior

This policy allows some discretion in when to send a Flow Control PDU when there is no Retransmission Control

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy ReceivingFlowControlPolicy
    behavior
        "This policy allows some discretion in when to send a Flow Control
 PDU when there is no Retransmission Control"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 10
;
```

B.5.31. Policy RetransmissionTimerExpiryPolicy

Behavior

This policy is executed by the sender when a Retransmission Timer Expires.

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy RetransmissionTimerExpiryPolicy
    behavior
        "This policy is executed by the sender when a Retransmission Timer
 Expires."

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 16
;
```

B.5.32. Policy SenderTimerInactivityPolicy

Behavior

This timer is used to detect long periods of no traffic, indicating that a DRF should be sent. If not, something is very wrong. The timeout value should generally be set to 2(MPL+R+A)

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy SenderTimerInactivityPolicy
```

```
    behavior
        "This timer is used to detect long periods of no traffic,
 indicating that a DRF should be sent. If not, something is very wrong.
 The timeout value should generally be set to 2(MPL+R+A)"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 3
;
```

B.5.33. Policy ReconcileFlowConflictPolicy

Behavior

This policy is invoked when both Credit and Rate based flow control are in use and they disagree on whether the PM can send or receive data

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy ReconcileFlowConflictPolicy
    behavior
        "This policy is invoked when both Credit and Rate based flow
 control are in use and they disagree on whether the PM can send or
 receive data"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 9
;
```

B.5.34. Policy RcvrControlAckPolicy

Behavior

This policy allows an alternate action when a Control Ack PDU is received.

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy RcvrControlAckPolicy
    behavior
        "This policy allows an alternate action when a Control Ack PDU is
 received."

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 21
;
```

B.5.35. Policy SendingAckPolicy

Behavior

This policy allows an alternate action when the A-Timer expires when DTCP is present.

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy SendingAckPolicy
    behavior
        "This policy allows an alternate action when the A-Timer expires
 when DTCP is present."
```

```
    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 20
;
```

B.5.36. Policy NoOverrideDefaultPeakPolicy

Behavior

This policy allows rate-based flow control to exceed its nominal rate

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy NoOverrideDefaultPeakPolicy
    behavior
        "This policy allows rate-based flow control to exceed its nominal
 rate"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 14
;
```

B.5.37. Policy FlowControlOverrunPolicy

Behavior

This policy determines what action to take if the receiver receives PDUs but the credit or rate has been exceeded

Dependencies

- ERoot.Attributes.PolicyConfig

- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy FlowControlOverrunPolicy
    behavior
        "This policy determines what action to take if the receiver
 receives PDUs but the credit or rate has been exceeded"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 8
;
```

B.5.38. Policy InitialSequenceNumberPolicy

Behavior

This policy allows some discretion in selecting the initial sequence number, when DRF is going to be sent

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy InitialSequenceNumberPolicy
    behavior
        "This policy allows some discretion in selecting the initial
 sequence number, when DRF is going to be sent"

    extends ERoot.Policies.DataTransfer
```

```
    registered-as ERoot Policies(4) DataTransfer(10) 4
;
```

B.5.39. Policy RcvrAckPolicy

Behavior

This policy is executed by the receiver of the PDU and provides some discretion in the action taken.

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy RcvrAckPolicy
    behavior
        "This policy is executed by the receiver of the PDU and provides
 some discretion in the action taken."

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 19
;
```

B.5.40. Policy LostControlPDUPolicy

Behavior

This policy determines what action to take when the PM detects that a control PDU (Ack or Flow Control) may have been lost

Dependencies

- ERoot.Attributes.PolicyConfig

- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy LostControlPDUPolicy
    behavior
        "This policy determines what action to take when the PM detects
 that a control PDU (Ack or Flow Control) may have been lost"

    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 5
;
```

B.5.41. Policy UnknownFlowPolicy

Behavior

When a PDU arrives for a Data Transfer Flow terminating in this IPC-Process and there is no active DTSV, this policy consults the ResourceAllocator to determine what to do

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.DataTransfer

Super Class

ERoot.Policies.DataTransfer

Specification

```
ro policy UnknownFlowPolicy
    behavior
        "When a PDU arrives for a Data Transfer Flow terminating in
 this IPC-Process and there is no active DTSV, this policy consults the
 ResourceAllocator to determine what to do"
```

```
    extends ERoot.Policies.DataTransfer
    registered-as ERoot Policies(4) DataTransfer(10) 1
;
```

B.5.42. Policy DataTransfer

Behavior

The root of the Data Transfer policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy DataTransfer
    behavior
        "The root of the Data Transfer policies, has no explicit behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 10
;
```

B.5.43. Policy ResourceAllocation

Behavior

The root of resource allocation policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy ResourceAllocation
    behavior
        "The root of resource allocation policies, has no explicit
 behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 4
;
```

B.5.44. Policy FragmentationPolicy

Behavior

Policy that may partition a complete SDU into multiple fragments, in order to comply with the limitation of the maximum PDU size in a given layer

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Delimiting

Super Class

ERoot.Policies.Delimiting

Specification

```
ro policy FragmentationPolicy
    behavior
        "Policy that may partition a complete SDU into multiple fragments,
 in order to comply with the limitation of the maximum PDU size in a given
 layer"

    extends ERoot.Policies.Delimiting
    registered-as ERoot Policies(4) Delimiting(9) 1
;
```

B.5.45. Policy ConcatenationPolicy

Behavior

Policy that creates a complete user data field (used as the payload of a DTP PDU) out of SDU fragments and complete SDUs. It generates SDU sequence numbers as described in the General Delimiting Module specification

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Delimiting

Super Class

ERoot.Policies.Delimiting

Specification

```
ro policy ConcatenationPolicy
    behavior
        "Policy that creates a complete user data field (used as the
 payload of a DTP PDU) out of SDU fragments and complete SDUs. It
 generates SDU sequence numbers as described in the General Delimiting
 Module specification"

    extends ERoot.Policies.Delimiting
    registered-as ERoot Policies(4) Delimiting(9) 2
;
```

B.5.46. Policy ReassemblyAndSeparationPolicy

Behavior

Processes the elements in the SDU reassembly queue in order to generate SDUs to be consumed by the user of the flow. The SDUs may not be complete if incomplete delivery is allowed

Dependencies

- ERoot.Attributes.PolicyConfig

- ERoot.Policies.Delimiting

Super Class

ERoot.Policies.Delimiting

Specification

```
ro policy ReassemblyAndSeparationPolicy
    behavior
        "Processes the elements in the SDU reassembly queue in order to
 generate SDUs to be consumed by the user of the flow. The SDUs may not be
 complete if incomplete delivery is allowed"

    extends ERoot.Policies.Delimiting
    registered-as ERoot Policies(4) Delimiting(9) 3
;
```

B.5.47. Policy RIBDaemon

Behavior

The root of the RIB Daemon policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy RIBDaemon
    behavior
        "The root of the RIB Daemon policies, has no explicit behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 8
;
```

B.5.48. Policy EnrollmentPolicy

Behavior

The enrollment policy, used to exchange information with a peer when joining a DIF

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.Enrollment

Super Class

ERoot.Policies.Enrollment

Specification

```
ro policy EnrollmentPolicy
    behavior
        "The enrollment policy, used to exchange information with a peer
 when joining a DIF"

    extends ERoot.Policies.Enrollment
    registered-as ERoot Policies(4) Enrollment(6) 1
;
```

B.5.49. Policy FlowAllocation

Behavior

The root of the Flow Allocation policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy FlowAllocation
    behavior
        "The root of the Flow Allocation policies, has no explicit
 behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 11
;
```

B.5.50. Policy Enrollment

Behavior

The root of enrollment policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy Enrollment
    behavior
        "The root of enrollment policies, has no explicit behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 6
;
```

B.5.51. Policy RIBUpdatePolicy

Behavior

Defines how often a set of objects in the RIB need to be updated, performing what remote operations and on which IPC Processes

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RIBDaemon

Super Class

ERoot.Policies.RIBDaemon

Specification

```
ro policy RIBUpdatePolicy
    behavior
        "Defines how often a set of objects in the RIB need to be updated,
 performing what remote operations and on which IPC Processes"

    extends ERoot.Policies.RIBDaemon
    registered-as ERoot Policies(4) RIBDaemon(8) 1
;
```

B.5.52. Policy RIBLoggingPolicy

Behavior

Defines what events (operations on remote RIB objects) should be logged, how (what information of the event should be stored) and where

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RIBDaemon

Super Class

ERoot.Policies.RIBDaemon

Specification

```
ro policy RIBLoggingPolicy
    behavior
        "Defines what events (operations on remote RIB objects) should be
 logged, how (what information of the event should be stored) and where"
```

```
    extends ERoot.Policies.RIBDaemon
    registered-as ERoot Policies(4) RIBDaemon(8) 4
;
```

B.5.53. Policy RIBSubscriptionPolicy

Behavior

Links a series of remote operations on one or more objects in the RIB to layer management tasks that want to be informed when these remote operations occur

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RIBDaemon

Super Class

ERoot.Policies.RIBDaemon

Specification

```
ro policy RIBSubscriptionPolicy
    behavior
        "Links a series of remote operations on one or more objects in the
 RIB to layer management tasks that want to be informed when these remote
 operations occur"

    extends ERoot.Policies.RIBDaemon
    registered-as ERoot Policies(4) RIBDaemon(8) 3
;
```

B.5.54. Policy RIBReplicationPolicy

Behavior

Defines how a set of objects in the RIB are replicated (example: fully replicated, partially replicated, not replicated), and over which IPC Processes these set of objects are replicated.

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RIBDaemon

Super Class

ERoot.Policies.RIBDaemon

Specification

```
ro policy RIBReplicationPolicy
    behavior
        "Defines how a set of objects in the RIB are replicated (example:
 fully replicated, partially replicated, not replicated), and over which
 IPC Processes these set of objects are replicated."

    extends ERoot.Policies.RIBDaemon
    registered-as ERoot Policies(4) RIBDaemon(8) 2
;
```

B.5.55. Policy RINAPolicy

Behavior

Represents the state of a RINA policy

Dependencies

- ERoot.Attributes.PolicyConfig

Attributes

- ERoot.Attributes.PolicyConfig policyConfig → Represents the configuration of a policy

Specification

```
abstract ro policy RINAPolicy
```

```
    behavior
        "Represents the state of a RINA policy"

    attributes
        ERoot.Attributes.PolicyConfig policyConfig
            "Represents the configuration of a policy"

    registered-as ERoot Policies(4) 1
;
```

B.5.56. Policy NotificationManagement

Behavior

The root of the Notification Management policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy NotificationManagement
    behavior
        "The root of the Notification Management policies, has no explicit
 behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 12
;
```

B.5.57. Policy RelayingAndMultiplexing

Behavior

The root of relaying and multiplexing policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy RelayingAndMultiplexing
    behavior
        "The root of relaying and multiplexing policies, has no explicit
 behavior"

    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 5
;
```

B.5.58. Policy Delimiting

Behavior

The root of the Delimiting policies, has no explicit behavior

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.RINAPolicy

Super Class

ERoot.Policies.RINAPolicy

Specification

```
abstract ro policy Delimiting
    behavior
        "The root of the Delimiting policies, has no explicit behavior"
```

```
    extends ERoot.Policies.RINAPolicy
    registered-as ERoot Policies(4) 9
;
```

B.5.59. Policy AddressManagementPolicy

Behavior

Policy to assign and validate address of IPC Processes

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.NamespaceManagement

Super Class

ERoot.Policies.NamespaceManagement

Specification

```
ro policy AddressManagementPolicy
    behavior
        "Policy to assign and validate address of IPC Processes"

    extends ERoot.Policies.NamespaceManagement
    registered-as ERoot Policies(4) NamespaceManagement(3) 1
;
```

B.5.60. Policy DFTGenerationPolicy

Behavior

Policy to generate the Directory Forwarding Table

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.NamespaceManagement

Super Class

ERoot.Policies.NamespaceManagement

Specification

```
ro policy DFTGenerationPolicy
    behavior
        "Policy to generate the Directory Forwarding Table"

    extends ERoot.Policies.NamespaceManagement
    registered-as ERoot Policies(4) NamespaceManagement(3) 2
;
```

B.5.61. Policy ReportArchivePolicy

Behavior

This policy how to store notifications belonging to a specific forwarding discriminator

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.NotificationManagement

Super Class

ERoot.Policies.NotificationManagement

Specification

```
ro policy ReportArchivePolicy
    behavior
        "This policy how to store notifications belonging to a specific
 forwarding discriminator"

    extends ERoot.Policies.NotificationManagement
    registered-as ERoot Policies(4) NotificationManagement(12) 1
;
```

B.5.62. Policy NotificationFilteringPolicy

Behavior

This policy decides which notifications should be forwarded to the subscriber and which not

Dependencies

- ERoot.Attributes.PolicyConfig
- ERoot.Policies.NotificationManagement

Super Class

ERoot.Policies.NotificationManagement

Specification

```
ro policy NotificationFilteringPolicy
    behavior
        "This policy decides which notifications should be forwarded to
 the subscriber and which not"

    extends ERoot.Policies.NotificationManagement
    registered-as ERoot Policies(4) NotificationManagement(12) 2
;
```

B.6. RO Type Definitions

B.6.1. Type Definition T_PolicyConfig

Behavior

The configuration of a policy

Members

- name → The name of the policy
  - as type T_String
- version → The version of the policy
  - as type T_String

- parameters → Parameters required to configure the policy
  - as type T_SetOf ERoot.Types.T_Parameter

Registered As

ERoot Types(5) 28

Specification

```
ro type definition T_PolicyConfig
    "The configuration of a policy"

    name T_String
        "The name of the policy"
    version T_String
        "The version of the policy"
    parameters T_SetOf ERoot.Types.T_Parameter
        "Parameters required to configure the policy"


    registered-as ERoot Types(5) 28
;
```

B.6.2. Type Definition T_QoSCubeData

Behavior

The data of a QoSCube

Members

- name → The name of the QoS cube
  - as type T_String
- id → The id of the QoS cube
  - as type T_Int
- averageBW → Average bandwidth in bytes/s. A value of 0 means don't care.
  - as type T_Int
- averageSDUBW → Average bandwidth in SDUs/s. A value of 0 means don't care

    - as type T_Int
- peakBWDuration → In milliseconds. A value of 0 means don't care
    - as type T_Int
- peakSDUBWDuration → In milliseconds. A value of 0 means don't care
    - as type T_Int
- undetectedBER → A value of 0 indicates 'do not care'
    - as type T_Int
- partialDelivery → Indicates if partial delivery of SDUs is allowed or not
    - as type T_Boolean
- orderedDelivery → Indicates if SDUs have to be delivered in order
    - as type T_Boolean
- maxAllowableGap → Indicates the maximum gap allowed among SDUs, a gap of N SDUs is considered the same as all SDUs delivered. A value of -1 indicates 'Any'
    - as type T_Int
- delay → In milliseconds, indicates the maximum delay allowed in this flow. A value of 0 indicates 'do not care'
    - as type T_Int
- jitter → In milliseconds, indicates the maximum jitter allowed in this flow. A value of 0 indicates 'do not care'
    - as type T_Int

Registered As

ERoot Types(5) 31

Specification

```
ro type definition T_QoSCubeData
    "The data of a QoSCube"

    name T_String
        "The name of the QoS cube"
    id T_Int
```

```
        "The id of the QoS cube"
    averageBW T_Int
        "Average bandwidth in bytes/s. A value of 0 means don't care."
    averageSDUBW T_Int
        "Average bandwidth in SDUs/s. A value of 0 means don't care"
    peakBWDuration T_Int
        "In milliseconds. A value of 0 means don't care"
    peakSDUBWDuration T_Int
        "In milliseconds. A value of 0 means don't care"
    undetectedBER T_Int
        "A value of 0 indicates 'do not care'"
    partialDelivery T_Boolean
        "Indicates if partial delivery of SDUs is allowed or not"
    orderedDelivery T_Boolean
        "Indicates if SDUs have to be delivered in order"
    maxAllowableGap T_Int
        "Indicates the maximum gap allowed among SDUs, a gap of N SDUs is
 considered the same as all SDUs delivered. A value of -1 indicates 'Any'"
    delay T_Int
        "In milliseconds, indicates the maximum delay allowed in this
 flow. A value of 0 indicates 'do not care'"
    jitter T_Int
        "In milliseconds, indicates the maximum jitter allowed in this
 flow. A value of 0 indicates 'do not care'"


    registered-as ERoot Types(5) 31
;
```

B.6.3. Type Definition T_RAConfig

Behavior

The configuration of the Resource Allocator

Members

- pduftgPolicy → Configuration of the PDU Forwarding Table Generator policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 33

Specification

```
ro type definition T_RAConfig
    "The configuration of the Resource Allocator"

    pduftgPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the PDU Forwarding Table Generator policy"


    registered-as ERoot Types(5) 33
;
```

B.6.4. Type Definition T_PolicyState

Behavior

The state of the policy

Members

- policyConfig → Configuration of the policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 29

Specification

```
ro type definition T_PolicyState
    "The state of the policy"

    policyConfig ERoot.Types.T_PolicyConfig
```

```
        "Configuration of the policy"


    registered-as ERoot Types(5) 29
;
```

B.6.5. Type Definition T_DirectoryForwardingTableEntry

Behavior

Contents of a Directory Forwarding table entry

Members

- key → Unique key of this entry in the table
  - as type T_Int
- destAddress → Address of the destination IPC Process
  - as type T_Int
- destAppName → Naming information of the destination application
  - as type ERoot.Types.T_APNamingInfo

Dependencies

- ERoot.Types.T_APNamingInfo

Registered As

ERoot Types(5) 8

Specification

```
ro type definition T_DirectoryForwardingTableEntry
    "Contents of a Directory Forwarding table entry"

    key T_Int
        "Unique key of this entry in the table"
    destAddress T_Int
        "Address of the destination IPC Process"
    destAppName ERoot.Types.T_APNamingInfo
        "Naming information of the destination application"
```

```
    registered-as ERoot Types(5) 8
;
```

B.6.6. Type Definition T_RoutingConfig

Behavior

The configuration of the Routing Task

Members

- policySet → Set of policies to define routing's behaviour
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 36

Specification

```
ro type definition T_RoutingConfig
    "The configuration of the Routing Task"

    policySet ERoot.Types.T_PolicyConfig
        "Set of policies to define routing's behaviour"


    registered-as ERoot Types(5) 36
;
```

B.6.7. Type Definition T_N1FlowState

Behavior

State of an RMT N-1 Flow

Members

- portId → The port id of the N-1 flow
  - as type T_Int
- started → True if the port is started, false if the port is stopped
  - as type T_Boolean
- statistics → Statistics on the data received and transmitted
  - as type ERoot.Types.T_N1FlowStats

Dependencies

- ERoot.Types.T_N1FlowStats

Registered As

ERoot Types(5) 20

Specification

```
ro type definition T_N1FlowState
    "State of an RMT N-1 Flow"

    portId T_Int
        "The port id of the N-1 flow"
    started T_Boolean
        "True if the port is started, false if the port is stopped"
    statistics ERoot.Types.T_N1FlowStats
        "Statistics on the data received and transmitted"


    registered-as ERoot Types(5) 20
;
```

B.6.8. Type Definition T_DataTransferConstants

Behavior

DIF-wide parameters that define the concrete syntax of EFCP for this DIF and other DIF-wide values

Members

- qosIdLength → The length of the QoS-id field in the DTP PCI, in bytes
  - as type T_Int
- cepIdLength → The length of the CEP-id field in the DTP PCI, in bytes
  - as type T_Int
- seqNumLength → The length of the sequence number field in the DTP PCI, in bytes
  - as type T_Int
- addressLength → The length of the address field in the DTP PCI, in bytes
  - as type T_Int
- lengthLength → The length of the length field in the DTP PCI, in bytes
  - as type T_Int
- ctrlSeqNumLength → The length of the control sequence number field in the DTCP PCI, in bytes
  - as type T_Int
- rateLength → The length of the rate field in the DTCP PCI, in bytes
  - as type T_Int
- frameLength → The length of the frame field in the DTCP PCI, in bytes
  - as type T_Int
- maxPDUSize → The maximum length allowed for a PDU in this DIF, in bytes
  - as type T_Int
- maxPDULifetime → The maximum PDU lifetime in this DIF, in milliseconds. This is MPL in delta-T
  - as type T_Int
- seqRollOverThres → The sequence number after which the Flow Allocator instance should create a new EFCP connection
  - as type T_Int
- difConcatenation → True if multiple SDUs can be delimited and concatenated within a single PDU

    - as type T_Boolean
- difFragmentation → This is true if multiple SDUs can be fragmented and reassembled within a single PDU
    - as type T_Boolean
- maxTimeToKeepRtx → The maximum time DTCP will try to keep retransmitting a PDU, before discarding it. This is R in delta-T
    - as type T_Int
- maxTimeToAck → The maximum time the receiving side of a DTCP connection will take to ACK a PDU once it has received it. This is A in delta-T
    - as type T_Int

Registered As

ERoot Types(5) *5*

Specification

```
ro type definition T_DataTransferConstants
    "DIF-wide parameters that define the concrete syntax of EFCP for this
 DIF and other DIF-wide values"

    qosIdLength T_Int
        "The length of the QoS-id field in the DTP PCI, in bytes"
    cepIdLength T_Int
        "The length of the CEP-id field in the DTP PCI, in bytes"
    seqNumLength T_Int
        "The length of the sequence number field in the DTP PCI, in bytes"
    addressLength T_Int
        "The length of the address field in the DTP PCI, in bytes"
    lengthLength T_Int
        "The length of the length field in the DTP PCI, in bytes"
    ctrlSeqNumLength T_Int
        "The length of the control sequence number field in the DTCP PCI,
 in bytes"
    rateLength T_Int
        "The length of the rate field in the DTCP PCI, in bytes"
    frameLength T_Int
        "The length of the frame field in the DTCP PCI, in bytes"
    maxPDUSize T_Int
```

```
        "The maximum length allowed for a PDU in this DIF, in bytes"
    maxPDULifetime T_Int
        "The maximum PDU lifetime in this DIF, in milliseconds. This is
MPL in delta-T"
    seqRollOverThres T_Int
        "The sequence number after which the Flow Allocator instance
should create a new EFCP connection"
    difConcatenation T_Boolean
        "True if multiple SDUs can be delimited and concatenated within a
single PDU"
    difFragmentation T_Boolean
        "This is true if multiple SDUs can be fragmented and reassembled
within a single PDU"
    maxTimeToKeepRtx T_Int
        "The maximum time DTCP will try to keep retransmitting a PDU,
before discarding it. This is R in delta-T"
    maxTimeToAck T_Int
        "The maximum time the receiving side of a DTCP connection will
take to ACK a PDU once it has received it. This is A in delta-T"


    registered-as ERoot Types(5) 5
;
```

B.6.9. Type Definition T_NotificationSubscriptionRequest

Behavior

Request to subscribe to a certain set of notifications

Members

- subscriber → Naming information of the subscriber
  - as type ERoot.Types.T_APNamingInfo
- reportArchPolicy → Configuration of the report archive policy
  - as type ERoot.Types.T_PolicyConfig
- filterPolicy → Configuration of the notification filtering policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_APNamingInfo

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 44

Specification

```
ro type definition T_NotificationSubscriptionRequest
    "Request to subscribe to a certain set of notifications"

    subscriber ERoot.Types.T_APNamingInfo
        "Naming information of the subscriber"
    reportArchPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the report archive policy"
    filterPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the notification filtering policy"


    registered-as ERoot Types(5) 44
;
```

B.6.10. Type Definition T_NextHopTableEntry

Behavior

Contents of a next hop table entry

Members

- key → Unique key of this entry in the table
  - as type T_Int
- destAddress → Address of the destination IPC Process
  - as type T_Int
- qosId → Id of the QoS-cube the PDU belongs to
  - as type T_Int
- nextHops → Address(es) of the IPCP(s) that are the next hop(s) towards destination
  - as type T_SetOf T_Int

Registered As

ERoot Types(5) 23

Specification

```
ro type definition T_NextHopTableEntry
    "Contents of a next hop table entry"

    key T_Int
        "Unique key of this entry in the table"
    destAddress T_Int
        "Address of the destination IPC Process"
    qosId T_Int
        "Id of the QoS-cube the PDU belongs to"
    nextHops T_SetOf T_Int
        "Address(es) of the IPCP(s) that are the next hop(s) towards
 destination"


    registered-as ERoot Types(5) 23
;
```

B.6.11. Type Definition T_ETConfig

Behavior

The configuration of the Enrollment Task

Members

- policySet → Configuration of the set of policies to define ET's behaviour
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 16

Specification

```
ro type definition T_ETConfig
    "The configuration of the Enrollment Task"

    policySet ERoot.Types.T_PolicyConfig
        "Configuration of the set of policies to define ET's behaviour"


    registered-as ERoot Types(5) 16
;
```

B.6.12. Type Definition T_DTCPWBFlowCtrlConfig

Behavior

The configuration of window-based flow control for a DTCP instance

Members

- maxClosedWQLength → The maximum length of the closed window queue
  - as type T_Int
- initialCredit → The initial credit (in PDUs)
  - as type T_Int
- recvrFlowCtrlPolicy → The configuration of the receiver flow control policy
  - as type ERoot.Types.T_PolicyConfig
- txCtrlPolicy → The configuration of the transmission control policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 13

Specification

```
ro type definition T_DTCPWBFlowCtrlConfig
    "The configuration of window-based flow control for a DTCP instance"

    maxClosedWQLength T_Int
        "The maximum length of the closed window queue"
    initialCredit T_Int
        "The initial credit (in PDUs)"
    recvrFlowCtrlPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the receiver flow control policy"
    txCtrlPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the transmission control policy"


    registered-as ERoot Types(5) 13
;
```

B.6.13. Type Definition T_PDUForwardingTableEntry

Behavior

Contents of a PDU Forwarding table entry

Members

- key → Unique key of this entry in the table
  - as type T_Int
- destAddress → Address of the destination IPC Process
  - as type T_Int
- qosId → Id of the QoS-cube the PDU belongs to
  - as type T_Int
- portIds → N-1 port-ids where the PDU will be forwarded
  - as type T_SetOf T_Int

Registered As

ERoot Types(5) 26

Specification

```
ro type definition T_PDUForwardingTableEntry
    "Contents of a PDU Forwarding table entry"

    key T_Int
        "Unique key of this entry in the table"
    destAddress T_Int
        "Address of the destination IPC Process"
    qosId T_Int
        "Id of the QoS-cube the PDU belongs to"
    portIds T_SetOf T_Int
        "N-1 port-ids where the PDU will be forwarded"


    registered-as ERoot Types(5) 26
;
```

B.6.14. Type Definition T_NSMConfig

Behavior

The configuration of the NameSpace Manager

Members

- policySet → Set of policies to define NSM's behaviour
  - as type ERoot.Types.T_PolicyConfig
- addressingConfig → configuration of the assignment of addresses to IPC Processes
  - as type ERoot.Types.T_AddressingConfig

Dependencies

- ERoot.Types.T_AddressingConfig
- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 24

Specification

```
ro type definition T_NSMConfig
    "The configuration of the NameSpace Manager"

    policySet ERoot.Types.T_PolicyConfig
        "Set of policies to define NSM's behaviour"
    addressingConfig ERoot.Types.T_AddressingConfig
        "configuration of the assignment of addresses to IPC Processes"


    registered-as ERoot Types(5) 24
;
```

B.6.15. Type Definition T_StaticAddress

Behavior

A mapping of an IPCP name to an static address

Members

- apName → The IPCP process name
  - as type T_String
- apInstance → The IPCP process instance
  - as type T_String
- address → The IPCP address
  - as type T_Int

Registered As

ERoot Types(5) 38

Specification

```
ro type definition T_StaticAddress
    "A mapping of an IPCP name to an static address"

    apName T_String
        "The IPCP process name"
```

```
    apInstance T_String
        "The IPCP process instance"
    address T_Int
        "The IPCP address"


    registered-as ERoot Types(5) 38
;
```

B.6.16. Type Definition T_DTCPRBFlowCtrlConfig

Behavior

The configuration of rate-based flow control for a DTCP instance

Members

- sendingRate → The number of PDUs that may be sent in a TimePeriod. Used with rate-based flow control.
  - as type T_Int
- timePeriod → The length of time in microseconds for pacing rate-based flow control
  - as type T_Int
- noRateSlowDownPolicy → The configuration of the no rate slow down policy
  - as type ERoot.Types.T_PolicyConfig
- noOverrDefPeakPolicy → The configuration of the no override default peak policy
  - as type ERoot.Types.T_PolicyConfig
- rateRedPolicy → The configuration of the rate reduction policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 11

Specification

```
ro type definition T_DTCPRBFlowCtrlConfig
    "The configuration of rate-based flow control for a DTCP instance"

    sendingRate T_Int
        "The number of PDUs that may be sent in a TimePeriod. Used with
 rate-based flow control."
    timePeriod T_Int
        "The length of time in microseconds for pacing rate-based flow
 control"
    noRateSlowDownPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the no rate slow down policy"
    noOverrDefPeakPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the no override default peak policy"
    rateRedPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the rate reduction policy"


    registered-as ERoot Types(5) 11
;
```

B.6.17. Type Definition T_AuthSDUProfile

Behavior

The configuration of an authentication and SDU Protection profile

Members

- n1DIFName → Name of the N-1 DIF this profile refers to
  - as type T_String
- authPolicy → Configuration of the authentication policy
  - as type ERoot.Types.T_PolicyConfig
- encryptPolicy → Configuration of the encryption policy
  - as type ERoot.Types.T_PolicyConfig
- crcPolicy → Configuration of the error check policy
  - as type ERoot.Types.T_PolicyConfig
- ttlPolicy → Configuration of the lifetime limiting policy

- as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 4

Specification

```
ro type definition T_AuthSDUProfile
    "The configuration of an authentication and SDU Protection profile"

    n1DIFName T_String
        "Name of the N-1 DIF this profile refers to"
    authPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the authentication policy"
    encryptPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the encryption policy"
    crcPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the error check policy"
    ttlPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the lifetime limiting policy"


    registered-as ERoot Types(5) 4
;
```

B.6.18. Type Definition T_IPCPConfig

Behavior

The configuration of an IPC Process

Members

- name → The IPC Process name/instance information
  - as type ERoot.Types.T_APNamingInfo
- difInfo → The DIF to which the IPC Process will be assigned (optional)
  - as type ERoot.Types.T_DIFInfo

- difsToRegister → The N-1 DIFs where the IPC Process should register at
  - as type T_SetOf ERoot.Types.T_APNamingInfo
- neighborsList → The list of neighbors to contact for enrollment
  - as type T_SetOf ERoot.Types.T_NeighborConfig

Dependencies

- ERoot.Types.T_APNamingInfo
- ERoot.Types.T_DIFInfo

Registered As

ERoot Types(5) 18

Specification

```
ro type definition T_IPCPConfig
    "The configuration of an IPC Process"

    name ERoot.Types.T_APNamingInfo
        "The IPC Process name/instance information"
    difInfo ERoot.Types.T_DIFInfo
        "The DIF to which the IPC Process will be assigned (optional)"
    difsToRegister T_SetOf ERoot.Types.T_APNamingInfo
        "The N-1 DIFs where the IPC Process should register at"
    neighborsList T_SetOf ERoot.Types.T_NeighborConfig
        "The list of neighbors to contact for enrollment"


    registered-as ERoot Types(5) 18
;
```

B.6.19. Type Definition T_RMTQueuePairState

Behavior

State of a queue pair

Members

- queueId → Id of the queue

  - as type T_Int
- txQueueState → State of tx queue
  - as type ERoot.Types.T_QueueInfo
- rxQueueState → State of rx queue
  - as type ERoot.Types.T_QueueInfo

Dependencies

- ERoot.Types.T_QueueInfo

Registered As

ERoot Types(5) 35

Specification

```
ro type definition T_RMTQueuePairState
    "State of a queue pair"

    queueId T_Int
        "Id of the queue"
    txQueueState ERoot.Types.T_QueueInfo
        "State of tx queue"
    rxQueueState ERoot.Types.T_QueueInfo
        "State of rx queue"


    registered-as ERoot Types(5) 35
;
```

B.6.20. Type Definition T_QoSCubeConfig

Behavior

The configuration of a QoS cube, including the supporting EFCP policies

Members

- data → The data of the QoS cube
  - as type ERoot.Types.T_QoSCubeData

- dtpConfig → The configuration of DTP for this QoS cube
  - as type ERoot.Types.T_DTPConfig
- dtcpConfig → The configuration of DTCP for this QoS cube
  - as type ERoot.Types.T_DTCPConfig

Dependencies

- ERoot.Types.T_DTCPConfig
- ERoot.Types.T_DTPConfig
- ERoot.Types.T_QoSCubeData

Registered As

ERoot Types(5) 30

Specification

```
ro type definition T_QoSCubeConfig
    "The configuration of a QoS cube, including the supporting EFCP
 policies"

    data ERoot.Types.T_QoSCubeData
        "The data of the QoS cube"
    dtpConfig ERoot.Types.T_DTPConfig
        "The configuration of DTP for this QoS cube"
    dtcpConfig ERoot.Types.T_DTCPConfig
        "The configuration of DTCP for this QoS cube"


    registered-as ERoot Types(5) 30
;
```

B.6.21. Type Definition T_AddressPrefix

Behavior

A mapping of an IPCP organization to an address prefix

Members

- organization → The organization where the IPCP belongs to

  - as type T_String
- addressPrefix → The IPCP address prefix
  - as type T_Int

Registered As

ERoot Types(5) 2

Specification

```
ro type definition T_AddressPrefix
    "A mapping of an IPCP organization to an address prefix"

    organization T_String
        "The organization where the IPCP belongs to"
    addressPrefix T_Int
        "The IPCP address prefix"


    registered-as ERoot Types(5) 2
;
```

B.6.22. Type Definition T_EFCPConnectionStats

Behavior

Statistics of an EFCP connection

Members

- bytesTx → Number of bytes transmitted
  - as type T_Int
- pdusTx → Number of PDUs transmitted
  - as type T_Int
- bytesRx → Number of bytes received
  - as type T_Int
- pdusRx → Number of PDUs received
  - as type T_Int

- pdusRtx → Number of PDUs retransmitted
  - as type T_Int

Registered As

ERoot Types(5) 42

Specification

```
ro type definition T_EFCPConnectionStats
    "Statistics of an EFCP connection"

    bytesTx T_Int
        "Number of bytes transmitted"
    pdusTx T_Int
        "Number of PDUs transmitted"
    bytesRx T_Int
        "Number of bytes received"
    pdusRx T_Int
        "Number of PDUs received"
    pdusRtx T_Int
        "Number of PDUs retransmitted"


    registered-as ERoot Types(5) 42
;
```

B.6.23. Type Definition T_IPCPInfo

Behavior

Information about the IPC Process

Members

- apName → Application process name
  - as type T_String
- apInstance → Application process instance
  - as type T_String
- processId → Id of the IPC Process within the system

    - as type T_Int
- synonymList → The IPC Process naming information: DAP name/instance, list of synonyms and ipc process id.
    - as type T_SetOf T_String
- difInfo → The DIF information if this IPC Process is a member of a DIF
    - as type ERoot.Types.T_DIFInfo

Dependencies

- ERoot.Types.T_DIFInfo

Registered As

ERoot Types(5) 19

Specification

```
ro type definition T_IPCPInfo
    "Information about the IPC Process"

    apName T_String
        "Application process name"
    apInstance T_String
        "Application process instance"
    processId T_Int
        "Id of the IPC Process within the system"
    synonymList T_SetOf T_String
        "The IPC Process naming information: DAP name/instance, list of
 synonyms and ipc process id."
    difInfo ERoot.Types.T_DIFInfo
        "The DIF information if this IPC Process is a member of a DIF"


    registered-as ERoot Types(5) 19
;
```

B.6.24. Type Definition T_QueueInfo

Behavior

State of an queue

Members

- capacityBytes → Capacity of the queue in bytes
  - as type T_Int
- capacityPDUs → Capacity of the queue in PDUs
  - as type T_Int
- sizeBytes → Size of the queue in bytes
  - as type T_Int
- sizePDUs → Size of the queue in PDUs
  - as type T_Int
- processedPDUs → PDUs processed
  - as type T_Int
- processedBytes → Bytes processed
  - as type T_Int
- droppedPDUs → PDUs dropped
  - as type T_Int
- droppedBytes → Bytes dropped
  - as type T_Int

Registered As

ERoot Types(5) 32

Specification

```
ro type definition T_QueueInfo
    "State of an queue"

    capacityBytes T_Int
        "Capacity of the queue in bytes"
    capacityPDUs T_Int
        "Capacity of the queue in PDUs"
    sizeBytes T_Int
        "Size of the queue in bytes"
    sizePDUs T_Int
```

```
        "Size of the queue in PDUs"
    processedPDUs T_Int
        "PDUs processed"
    processedBytes T_Int
        "Bytes processed"
    droppedPDUs T_Int
        "PDUs dropped"
    droppedBytes T_Int
        "Bytes dropped"


    registered-as ERoot Types(5) 32
;
```

B.6.25. Type Definition T_FAConfig

Behavior

The configuration of the Flow Allocator

Members

- policySet → Set of policies to define FA's behaviour
  - as type ERoot.Types.T_PolicyConfig
- maxCreateFlowRetries → Maximum number of retries to create a flow
  - as type T_Int

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 17

Specification

```
ro type definition T_FAConfig
    "The configuration of the Flow Allocator"

    policySet ERoot.Types.T_PolicyConfig
```

```
        "Set of policies to define FA's behaviour"
    maxCreateFlowRetries T_Int
        "Maximum number of retries to create a flow"


    registered-as ERoot Types(5) 17
;
```

B.6.26. Type Definition T_FlowProperties

Behavior

A request to allocate a flow

Members

- averageBW → Average bandwidth in bytes/s. A value of 0 means don't care.
  - as type T_Int
- averageSDUBW → Average bandwidth in SDUs/s. A value of 0 means don't care
  - as type T_Int
- peakBWDuration → In milliseconds. A value of 0 means don't care
  - as type T_Int
- undetectedBER → A value of 0 indicates 'do not care'
  - as type T_Int
- partialDelivery → Indicates if partial delivery of SDUs is allowed or not
  - as type T_Boolean
- orderedDelivery → Indicates if SDUs have to be delivered in order
  - as type T_Boolean
- maxAllowableGap → Indicates the maximum gap allowed among SDUs,
  - as type T_Int
- delay → In milliseconds, indicates the maximum delay allowed in this flow. A value of 0 indicates 'do not care
  - as type T_Int

- jitter → In milliseconds, indicates the maximum jitter allowed in this flow. A value of 0 indicates 'do not care'
  - as type T_Int
- maxSDUSize → The maximum SDU size for the flow. May influence the choice of the DIF where the flow will be created
  - as type T_Int

Registered As

ERoot Types(5) 41

Specification

```
ro type definition T_FlowProperties
    "A request to allocate a flow"

    averageBW T_Int
        "Average bandwidth in bytes/s. A value of 0 means don't care."
    averageSDUBW T_Int
        "Average bandwidth in SDUs/s. A value of 0 means don't care"
    peakBWDuration T_Int
        "In milliseconds. A value of 0 means don't care"
    undetectedBER T_Int
        "A value of 0 indicates 'do not care'"
    partialDelivery T_Boolean
        "Indicates if partial delivery of SDUs is allowed or not"
    orderedDelivery T_Boolean
        "Indicates if SDUs have to be delivered in order"
    maxAllowableGap T_Int
        "Indicates the maximum gap allowed among SDUs,"
    delay T_Int
        "In milliseconds, indicates the maximum delay allowed in this
 flow. A value of 0 indicates 'do not care"
    jitter T_Int
        "In milliseconds, indicates the maximum jitter allowed in this
 flow. A value of 0 indicates 'do not care'"
    maxSDUSize T_Int
        "The maximum SDU size for the flow. May influence the choice of
 the DIF where the flow will be created"


    registered-as ERoot Types(5) 41
;
```

B.6.27. Type Definition T_FlowAllocationRequest

Behavior

A request to allocate a flow

Members

- localAppName → The local application name (flow requestor)
  - as type ERoot.Types.T_APNamingInfo
- remoteAppName → The remote application name
  - as type ERoot.Types.T_APNamingInfo
- flowSpec → The properties requested for the flow (capacity, delay, jitter, etc..)
  - as type ERoot.Types.T_FlowProperties

Dependencies

- ERoot.Types.T_APNamingInfo
- ERoot.Types.T_FlowProperties

Registered As

ERoot Types(5) 40

Specification

```
ro type definition T_FlowAllocationRequest
    "A request to allocate a flow"

    localAppName ERoot.Types.T_APNamingInfo
        "The local application name (flow requestor)"
    remoteAppName ERoot.Types.T_APNamingInfo
        "The remote application name"
    flowSpec ERoot.Types.T_FlowProperties
        "The properties requested for the flow (capacity, delay, jitter,
 etc..)"


    registered-as ERoot Types(5) 40
```

```
;
```

B.6.28. Type Definition T_Parameter

Behavior

A parameter with a name and value

Members

- name → The name of the parameter
  - as type T_String
- value → The value of the parameter
  - as type T_String

Registered As

ERoot Types(5) 25

Specification

```
ro type definition T_Parameter
    "A parameter with a name and value"

    name T_String
        "The name of the parameter"
    value T_String
        "The value of the parameter"


    registered-as ERoot Types(5) 25
;
```

B.6.29. Type Definition T_DIFInfo

Behavior

The information of a DIF, including its configuration

Members

- type → The type of DIF (normal or one of the shims)

  - as type T_String
- name → The name of the DIF
  - as type ERoot.Types.T_APNamingInfo
- difConfig → The DIF configuration
  - as type ERoot.Types.T_DIFConfig

Dependencies

- ERoot.Types.T_APNamingInfo
- ERoot.Types.T_DIFConfig

Registered As

ERoot Types(5) 7

Specification

```
ro type definition T_DIFInfo
    "The information of a DIF, including its configuration"

    type T_String
        "The type of DIF (normal or one of the shims)"
    name ERoot.Types.T_APNamingInfo
        "The name of the DIF"
    difConfig ERoot.Types.T_DIFConfig
        "The DIF configuration"


    registered-as ERoot Types(5) 7
;
```

B.6.30. Type Definition T_DTCPRtxCtrlConfig

Behavior

The configuration of retransmission control for a DTCP instance

Members

- maxTimeRetry → Maximum time to attempt the retransmission of a packet, this is R

  - as type T_Int
- dataRtxMax → The number of times the retransmission of a PDU will be attempted before some other action must be taken
  - as type T_Int
- initialRtxTime → Initial retransmission time: Tr. R = tr*data_rxms_max_
  - as type T_Int
- rtxTimeExpiryPolicy → Configuration of the retransmission timer expiry policy
  - as type ERoot.Types.T_PolicyConfig
- sdrAckPolicy → The configuration of the sender ACK policy
  - as type ERoot.Types.T_PolicyConfig
- recvingAckListPolicy → The configuration of the receiving ACK list policy
  - as type ERoot.Types.T_PolicyConfig
- rcvrAckPolicy → The configuration of the receiver ACK policy
  - as type ERoot.Types.T_PolicyConfig
- sendingAckPolicy → The configuration of the sending ACK policy
  - as type ERoot.Types.T_PolicyConfig
- rcvrCtrlAckPolicy → The configuration of the receiver control ACK policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 12

Specification

```
ro type definition T_DTCPRtxCtrlConfig
```

```
    "The configuration of retransmission control for a DTCP instance"

    maxTimeRetry T_Int
        "Maximum time to attempt the retransmission of a packet, this is
R"
    dataRtxMax T_Int
        "The number of times the retransmission of a PDU will be attempted
before some other action must be taken"
    initialRtxTime T_Int
        "Initial retransmission time: Tr. R = tr*data_rxms_max_"
    rtxTimeExpiryPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the retransmission timer expiry policy"
    sdrAckPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the sender ACK policy"
    recvingAckListPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the receiving ACK list policy"
    rcvrAckPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the receiver ACK policy"
    sendingAckPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the sending ACK policy"
    rcvrCtrlAckPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the receiver control ACK policy"


    registered-as ERoot Types(5) 12
;
```

B.6.31. Type Definition T_AddressingConfig

Behavior

The configuration of the assignment of addresses to IPC Processes

Members

- staticAddresses → static address mappings
  - as type T_SetOf ERoot.Types.T_StaticAddress
- addressPrefix → address prefixes
  - as type T_SetOf ERoot.Types.T_AddressPrefix

Registered As

ERoot Types(5) 1

Specification

```
ro type definition T_AddressingConfig
    "The configuration of the assignment of addresses to IPC Processes"

    staticAddresses T_SetOf ERoot.Types.T_StaticAddress
        "static address mappings"
    addressPrefix T_SetOf ERoot.Types.T_AddressPrefix
        "address prefixes"


    registered-as ERoot Types(5) 1
;
```

B.6.32. Type Definition T_DIFConfig

Behavior

The configuration of a DIF

Members

- address → The address of the IPC Process in the DIF
  - as type T_Int
- efcpConfig → Configuration of the Error and Flow Control Protocol
  - as type ERoot.Types.T_EFCPConfig
- rmtConfig → Configuration of the Relaying and Multiplexing Task
  - as type ERoot.Types.T_RMTConfig
- faConfig → Configuration of the Flow Allocator
  - as type ERoot.Types.T_FAConfig
- etConfig → Configuration of the enrollment task
  - as type ERoot.Types.T_ETConfig
- nsmConfig → Configuration of the Namespace manager
  - as type ERoot.Types.T_NSMConfig
- routingConfig → Configuration of the routing policy
  - as type ERoot.Types.T_RoutingConfig

- raConfig → Configuration of the resource allocator
  - as type ERoot.Types.T_RAConfig
- smConfig → Configuration of the Security Manager
  - as type ERoot.Types.T_SMConfig
- params → Extra configuration parameters
  - as type T_SetOf ERoot.Types.T_Parameter

Dependencies

- ERoot.Types.T_EFCPConfig
- ERoot.Types.T_ETConfig
- ERoot.Types.T_FAConfig
- ERoot.Types.T_NSMConfig
- ERoot.Types.T_RAConfig
- ERoot.Types.T_RMTConfig
- ERoot.Types.T_RoutingConfig
- ERoot.Types.T_SMConfig

Registered As

ERoot Types(5) 6

Specification

```
ro type definition T_DIFConfig
    "The configuration of a DIF"

    address T_Int
        "The address of the IPC Process in the DIF"
    efcpConfig ERoot.Types.T_EFCPConfig
        "Configuration of the Error and Flow Control Protocol"
    rmtConfig ERoot.Types.T_RMTConfig
        "Configuration of the Relaying and Multiplexing Task"
    faConfig ERoot.Types.T_FAConfig
        "Configuration of the Flow Allocator"
    etConfig ERoot.Types.T_ETConfig
        "Configuration of the enrollment task"
```

```
    nsmConfig ERoot.Types.T_NSMConfig
        "Configuration of the Namespace manager"
    routingConfig ERoot.Types.T_RoutingConfig
        "Configuration of the routing policy"
    raConfig ERoot.Types.T_RAConfig
        "Configuration of the resource allocator"
    smConfig ERoot.Types.T_SMConfig
        "Configuration of the Security Manager"
    params T_SetOf ERoot.Types.T_Parameter
        "Extra configuration parameters"


    registered-as ERoot Types(5) 6
;
```

B.6.33. Type Definition T_APNamingInfo

Behavior

The naming information of an Application process/instance or and Application Entity/instance

Members

- apName → The application process name
  - as type T_String
- apInstance → The application process instance
  - as type T_String
- aeName → The application entity name
  - as type T_String
- aeInstance → The application entity instance
  - as type T_String

Registered As

ERoot Types(5) 3

Specification

```
ro type definition T_APNamingInfo
```

```
    "The naming information of an Application process/instance or and
 Application Entity/instance"

    apName T_String
        "The application process name"
    apInstance T_String
        "The application process instance"
    aeName T_String
        "The application entity name"
    aeInstance T_String
        "The application entity instance"


    registered-as ERoot Types(5) 3
;
```

B.6.34. Type Definition T_DTCPFlowCtrlConfig

Behavior

The configuration of flow control for a DTCP instance

Members

- windowBased → Indicates if window based flow control is active
  - as type T_Boolean
- rateBased → Indicates if rate-based flow control is active
  - as type T_Boolean
- dtcpWBFlowCtrlPolicy → Configuration of the window-based flow control policies
  - as type ERoot.Types.T_DTCPWBFlowCtrlConfig
- dtcpRBFlowCtrlPolicy → Configuration of the rate-based flow control policies
  - as type ERoot.Types.T_DTCPRBFlowCtrlConfig
- closedWindowPolicy → Configuration of the closed window policy
  - as type ERoot.Types.T_PolicyConfig
- flowCtrlOvrPolicy → The configuration of the flow control overrun policy

  - as type ERoot.Types.T_PolicyConfig
- recFlowCtrlPolicy → The configuration of the reconcile flow control policy
  - as type ERoot.Types.T_PolicyConfig
- rcvFlowCtrlPolicy → The configuration of the receiving flow control policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_DTCPRBFlowCtrlConfig
- ERoot.Types.T_DTCPWBFlowCtrlConfig
- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 10

Specification

```
ro type definition T_DTCPFlowCtrlConfig
    "The configuration of flow control for a DTCP instance"

    windowBased T_Boolean
        "Indicates if window based flow control is active"
    rateBased T_Boolean
        "Indicates if rate-based flow control is active"
    dtcpWBFlowCtrlPolicy ERoot.Types.T_DTCPWBFlowCtrlConfig
        "Configuration of the window-based flow control policies"
    dtcpRBFlowCtrlPolicy ERoot.Types.T_DTCPRBFlowCtrlConfig
        "Configuration of the rate-based flow control policies"
    closedWindowPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the closed window policy"
    flowCtrlOvrPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the flow control overrun policy"
    recFlowCtrlPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the reconcile flow control policy"
    rcvFlowCtrlPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the receiving flow control policy"
```

```
    registered-as ERoot Types(5) 10
;
```

B.6.35. Type Definition T_FlowAllocatorStats

Behavior

Statistics of the Flow Allocator

Members

- inFlowRequests → Number of incoming flow requests
  - as type T_Int
- inFlowRequestsRej → Number of rejected incoming flow requests
  - as type T_Int
- outFlowRequests → Number of outgoing flow requests
  - as type T_Int
- outFlowRequestsRej → Number of rejected outgoing flow requests
  - as type T_Int

Registered As

ERoot Types(5) 43

Specification

```
ro type definition T_FlowAllocatorStats
    "Statistics of the Flow Allocator"

    inFlowRequests T_Int
        "Number of incoming flow requests"
    inFlowRequestsRej T_Int
        "Number of rejected incoming flow requests"
    outFlowRequests T_Int
        "Number of outgoing flow requests"
    outFlowRequestsRej T_Int
        "Number of rejected outgoing flow requests"


    registered-as ERoot Types(5) 43
```

;

B.6.36. Type Definition T_PFFConfig

Behavior

The configuration of the PDU Forwarding Function

Members

- policySet → Set of policies to define PFF behavior
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 27

Specification

```
ro type definition T_PFFConfig
    "The configuration of the PDU Forwarding Function"

    policySet ERoot.Types.T_PolicyConfig
        "Set of policies to define PFF behavior"


    registered-as ERoot Types(5) 27
;
```

B.6.37. Type Definition T_DTCPConfig

Behavior

The configuration of a DTCP instance

Members

- isFlowControl → Indicates if flow control is enabled

  - as type T_Boolean
- isRtxControl → Indicates if retransmission control is enabled
  - as type T_Boolean
- dtcpFlowCtrlConfig → Configuration of the DTCP flow control policies
  - as type ERoot.Types.T_DTCPFlowCtrlConfig
- dtcpRtxCtrlConfig → Configuration of the DTCP retransmission control policies
  - as type ERoot.Types.T_DTCPRtxCtrlConfig
- dtcpPolicySet → Configuration of DTCP policy set
  - as type ERoot.Types.T_PolicyConfig
- lostCtrlPDUPolicy → The configuration of the lost control PDU policy
  - as type ERoot.Types.T_PolicyConfig
- rttEstimatorPolicy → The configuration of the round-trip time estimator policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_DTCPFlowCtrlConfig
- ERoot.Types.T_DTCPRtxCtrlConfig
- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 9

Specification

```
ro type definition T_DTCPConfig
    "The configuration of a DTCP instance"

    isFlowControl T_Boolean
        "Indicates if flow control is enabled"
    isRtxControl T_Boolean
        "Indicates if retransmission control is enabled"
```

```
    dtcpFlowCtrlConfig ERoot.Types.T_DTCPFlowCtrlConfig
        "Configuration of the DTCP flow control policies"
    dtcpRtxCtrlConfig ERoot.Types.T_DTCPRtxCtrlConfig
        "Configuration of the DTCP retransmission control policies"
    dtcpPolicySet ERoot.Types.T_PolicyConfig
        "Configuration of DTCP policy set"
    lostCtrlPDUPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the lost control PDU policy"
    rttEstimatorPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the round-trip time estimator policy"


    registered-as ERoot Types(5) 9
;
```

B.6.38. Type Definition T_RMTConfig

Behavior

The configuration of the Relaying and Multiplexing Task

Members

- policySet → Set of policies to define RMT's behaviour. QMonitor Policy, MaxQ Policy and Scheduling Policy
  - as type ERoot.Types.T_PolicyConfig
- pffConfig → Configuration of the PDU Forwarding Function
  - as type ERoot.Types.T_PFFConfig

Dependencies

- ERoot.Types.T_PFFConfig
- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 34

Specification

```
ro type definition T_RMTConfig
```

```
    "The configuration of the Relaying and Multiplexing Task"

    policySet ERoot.Types.T_PolicyConfig
        "Set of policies to define RMT's behaviour. QMonitor Policy, MaxQ
 Policy and Scheduling Policy"
    pffConfig ERoot.Types.T_PFFConfig
        "Configuration of the PDU Forwarding Function"


    registered-as ERoot Types(5) 34
;
```

B.6.39. Type Definition T_DTPConfig

Behavior

The configuration of a DTP instance

Members

- dtcpPresent → Indicates if DTCP is required
  - as type T_Boolean
- seqRollOverThres → The sequence number rollover threshold
  - as type T_Int
- initialATimer → Indicates the maximum time that a receiver will wait before sending an Ack. Some DIFs may wish to set a maximum value for the DIF.
  - as type T_Int
- partialDelivery → True if partial delivery of an SDU is allowed, false otherwise
  - as type T_Boolean
- incompleteDelivery → True if incomplete delivery is allowed (one fragment of SDU delivered is the same as all the SDU delivered), false otherwise
  - as type T_Boolean
- inOrderDelivery → True if in order delivery of SDUs is mandatory, false otherwise

  - as type T_Boolean
- maxSDUGap → The maximum gap of SDUs allowed
  - as type T_Int
- dtpPolicySet → Configuration of DTP policy set
  - as type ERoot.Types.T_PolicyConfig
- rcvrTimerInPolicy → The configuration of the receiver timer inactivity policy
  - as type ERoot.Types.T_PolicyConfig
- sndrTimerInPolicy → The configuration of the sender timer inactivity policy
  - as type ERoot.Types.T_PolicyConfig
- initSeqNumPolicy → The configuration of the initial sequence number policy
  - as type ERoot.Types.T_PolicyConfig

Dependencies

- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 14

Specification

```
ro type definition T_DTPConfig
    "The configuration of a DTP instance"

    dtcpPresent T_Boolean
        "Indicates if DTCP is required"
    seqRollOverThres T_Int
        "The sequence number rollover threshold"
    initialATimer T_Int
        "Indicates the maximum time that a receiver will wait before
 sending an Ack. Some DIFs may wish to set a maximum value for the DIF."
    partialDelivery T_Boolean
        "True if partial delivery of an SDU is allowed, false otherwise"
```

```
    incompleteDelivery T_Boolean
        "True if incomplete delivery is allowed (one fragment of SDU
 delivered is the same as all the SDU delivered), false otherwise"
    inOrderDelivery T_Boolean
        "True if in order delivery of SDUs is mandatory, false otherwise"
    maxSDUGap T_Int
        "The maximum gap of SDUs allowed"
    dtpPolicySet ERoot.Types.T_PolicyConfig
        "Configuration of DTP policy set"
    rcvrTimerInPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the receiver timer inactivity policy"
    sndrTimerInPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the sender timer inactivity policy"
    initSeqNumPolicy ERoot.Types.T_PolicyConfig
        "The configuration of the initial sequence number policy"


    registered-as ERoot Types(5) 14
;
```

B.6.40. Type Definition T_N1FlowStats

Behavior

Statistics about data send and received through an N-1 flow

Members

- pdusRx → The number of PDUs received
  - as type T_Int
- bytesRx → The number of bytes received
  - as type T_Int
- pdusTx → The number of PDUs transmitted
  - as type T_Int
- bytesTx → The number of bytes transmitted
  - as type T_Int
- pdusDropped → The number of PDUs dropped
  - as type T_Int
- pdusErrors → The number of PDUs with errors

- as type T_Int

Registered As

ERoot Types(5) 21

Specification

```
ro type definition T_N1FlowStats
    "Statistics about data send and received through an N-1 flow"

    pdusRx T_Int
        "The number of PDUs received"
    bytesRx T_Int
        "The number of bytes received"
    pdusTx T_Int
        "The number of PDUs transmitted"
    bytesTx T_Int
        "The number of bytes transmitted"
    pdusDropped T_Int
        "The number of PDUs dropped"
    pdusErrors T_Int
        "The number of PDUs with errors"


    registered-as ERoot Types(5) 21
;
```

B.6.41. Type Definition T_SMConfig

Behavior

The configuration of the Security Manager

Members

- policySet → Configuration of the Security Manager Policy Set
  - as type ERoot.Types.T_PolicyConfig
- defaultAuthProfile → Configuration of the default authentication/SDU Protection profile
  - as type ERoot.Types.T_AuthSDUProfile

- specificAuthProfiles → N-1 DIF specific authentication/SDU Protection profiles
  - as type T_SetOf ERoot.Types.T_AuthSDUProfile

Dependencies

- ERoot.Types.T_AuthSDUProfile
- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 37

Specification

```
ro type definition T_SMConfig
    "The configuration of the Security Manager"

    policySet ERoot.Types.T_PolicyConfig
        "Configuration of the Security Manager Policy Set"
    defaultAuthProfile ERoot.Types.T_AuthSDUProfile
        "Configuration of the default authentication/SDU Protection
 profile"
    specificAuthProfiles T_SetOf ERoot.Types.T_AuthSDUProfile
        "N-1 DIF specific authentication/SDU Protection profiles"


    registered-as ERoot Types(5) 37
;
```

B.6.42. Type Definition T_DIFRegistrationRequest

Behavior

A request to register an AE to an N-1 DIF

Members

- aeNamingInfo → The AP/AE naming information
  - as type ERoot.Types.T_APNamingInfo
- difName → The DIF Name

- as type T_String

Dependencies

- ERoot.Types.T_APNamingInfo

Registered As

ERoot Types(5) 39

Specification

```
ro type definition T_DIFRegistrationRequest
    "A request to register an AE to an N-1 DIF"

    aeNamingInfo ERoot.Types.T_APNamingInfo
        "The AP/AE naming information"
    difName T_String
        "The DIF Name"


    registered-as ERoot Types(5) 39
;
```

B.6.43. Type Definition T_NeighborConfig

Behavior

The configuration to contact a neighbor of the IPCP

Members

- neighborName → The name of the neighbor IPCP
  - as type ERoot.Types.T_APNamingInfo
- underDIFName → The name of the underlying DIF used to contact the neighbor IPCP
  - as type ERoot.Types.T_APNamingInfo
- difName → The name of the DIF to which the neighbor IPCP belongs
  - as type ERoot.Types.T_APNamingInfo

Dependencies

- ERoot.Types.T_APNamingInfo

Registered As

ERoot Types(5) 22

Specification

```
ro type definition T_NeighborConfig
    "The configuration to contact a neighbor of the IPCP"

    neighborName ERoot.Types.T_APNamingInfo
        "The name of the neighbor IPCP"
    underDIFName ERoot.Types.T_APNamingInfo
        "The name of the underlying DIF used to contact the neighbor IPCP"
    difName ERoot.Types.T_APNamingInfo
        "The name of the DIF to which the neighbor IPCP belongs"


    registered-as ERoot Types(5) 22
;
```

B.6.44. Type Definition T_EFCPConfig

Behavior

The configuration of a the Error and Flow Control Protocol

Members

- dtCons → DIF-wide parameters that define the concrete syntax of EFCP for this DIF and other DIF-wide values
  - as type ERoot.Types.T_DataTransferConstants
- unknwonFlowPolicy → Configuration of the unknown flow policy (optional)
  - as type ERoot.Types.T_PolicyConfig
- qosCubes → The QoS cubes supported by the DIF, and its associated EFCP policies

- as type T_SetOf ERoot.Types.T_QoSCubeConfig

Dependencies

- ERoot.Types.T_DataTransferConstants
- ERoot.Types.T_PolicyConfig

Registered As

ERoot Types(5) 15

Specification

```
ro type definition T_EFCPConfig
    "The configuration of a the Error and Flow Control Protocol"

    dtCons ERoot.Types.T_DataTransferConstants
        "DIF-wide parameters that define the concrete syntax of EFCP for
 this DIF and other DIF-wide values"
    unknwonFlowPolicy ERoot.Types.T_PolicyConfig
        "Configuration of the unknown flow policy (optional)"
    qosCubes T_SetOf ERoot.Types.T_QoSCubeConfig
        "The QoS cubes supported by the DIF, and its associated EFCP
 policies"


    registered-as ERoot Types(5) 15
;
```

C. DIF template schema specification

C.1. DIF Template schema

The annotations in this XML schema are optional, and used to automatically generate an HTML form that facilitates generating XML DIF specifications that conform to this schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://
org.pouzinsociety/2016/dif-template" xmlns:i="http://moten.david.org/xsd-
forms" targetNamespace="http://org.pouzinsociety/2016/dif-template">
   <xs:annotation i:numberItems="true">
      <xs:appinfo>
         <i:header><![CDATA[<h2>Specification Template for a DIF</h2>]]></
i:header>
         <i:footer><![CDATA[<p>Thanks for your time.</p>]]></i:footer>
      </xs:appinfo>
   </xs:annotation>
   <xs:element name="dif-template" type="dif-template" />
   <xs:complexType name="dif-template">
      <xs:sequence>
         <xs:element name="introduction" type="intro" />
         <xs:element name="references" type="spec-
ref" maxOccurs="unbounded">
            <xs:annotation i:nonRepeatingTitle="2. References to other
 specifications" i:label="Specification" i:before="This should list the
 versions of the specifications this assumes and the other more detailed
 specifications that are cited below." />
         </xs:element>
         <xs:element name="servicesprovided" type="serviceprov" />
         <xs:element name="servicesrequired" type="servicereq" />
         <xs:element name="datatransfer" type="datatransfer" />
         <xs:element name="layermanagement" type="layermgmt" />
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="intro">
      <xs:sequence>
         <xs:element name="name" type="non-blank-string">
            <xs:annotation i:title="1. Introduction" i:before="This
 section provides a brief narrative that characterizes this proforma and
 its field of application.One should expect several forms of proforma from
 these templates.  The simple case is a DIF Specification that specifies
 everything.  Others might specify some elements but only put bounds
```

```
on some policies. Allowing flexibility in behavior within limits, or
detailed to be filled in for specific networks.  Some might require a
minimum set of policies be supported, etc. Some DIF Specifications may
be intended as frameworks to simplify configuring more specialized DIFs,
etc." i:label="Name of the DIF template" />
        </xs:element>
        <xs:element name="version" type="non-blank-string">
           <xs:annotation i:label="Version of the DIF template" />
        </xs:element>
        <xs:element name="description" type="non-blank-string">
           <xs:annotation i:text="textarea" />
        </xs:element>
     </xs:sequence>
  </xs:complexType>
  <xs:complexType name="serviceprov">
     <xs:sequence>
        <xs:element name="description" type="xs:string" minOccurs="0">
           <xs:annotation i:title="3. Services
provided" i:text="textarea" i:label="Description (optional)" />
        </xs:element>
        <xs:element name="maxCapacity" type="xs:integer">
           <xs:annotation i:label="Maximum aggregated capacity (Mbps)" />
        </xs:element>
        <xs:element name="maxSDUSize" type="xs:integer">
           <xs:annotation i:label="Maximum SDU Size (bytes)" />
        </xs:element>
        <xs:element name="undetectedBER" type="xs:integer" minOccurs="0">
           <xs:annotation i:label="Undetected bit error rate (10^-x)" />
        </xs:element>
        <xs:element name="qos-cubes" type="qos-
cube" maxOccurs="unbounded">
           <xs:annotation i:nonRepeatingTitle="3.1. QoS cubes
supported" i:label="QoS cube" i:before="This section gives a precise
definition of the QoS-cubes supported by this proforma and their QoS-
ids." />
        </xs:element>
        <xs:element name="system-apis" type="spec-
ref" maxOccurs="unbounded">
           <xs:annotation i:nonRepeatingTitle="3.2. System-specific
APIs" i:label="System API" i:before="This section specifies one or more
definitions of the API for specific system environments that conform to
the Service Definition. This should also specify the API flow control
policy for this QoS-cube. Primarily an indication of when the user of the
flow will be blocked. If there are no APIs in the implementation, this
section is N/A." />
        </xs:element>
```

```
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="servicereq">
      <xs:sequence>

    <xs:element name="description" i:label="Description" type="xs:string" i:before="This
 section lists the ranges of  QoS that this DIF requires.  Note that while
 these ranges of QoS form a QoS-cube, they need not be precisely the same
 as the QoS-cubes provided by the (N-1)-DIF. They would not necessarily be
 assigned a QoS-cube-id.">
            <xs:annotation i:title="4. Services
 required" i:text="textarea" />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="datatransfer">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:title="5. Data transfer and data transfer
 control" i:label="Description (optional)" i:text="textarea" />
         </xs:element>
         <xs:element name="difparams" type="dif-params" />
         <xs:element name="efcpsyntax" type="efcp-syntax" />
         <xs:element name="delimiting" type="policy-
conf" maxOccurs="unbounded">
            <xs:annotation i:nonRepeatingTitle="5.3. Delimiting (refs to
 delimiting policies)" i:label="5.3.x Delimiting policy spec" />
         </xs:element>
         <xs:element name="qoscubespolicies" type="qos-cube-
policy" maxOccurs="unbounded">
            <xs:annotation i:nonRepeatingTitle="5.4. QoS-Cubes
 policies" i:label="5.4.x QoS Cube policy" />
         </xs:element>
         <xs:element name="mtpolicies" type="rmt-policies" />
         <xs:element name="sduprotection" type="policy-
conf" maxOccurs="unbounded">
            <xs:annotation i:nonRepeatingTitle="5.6. SDU Protection (refs
 to delimiting policies)" i:label="5.6.x SDU protection policy spec" />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="layermgmt">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:title="6. Layer
 Management" i:label="Description (optional)" i:text="textarea" />
```

```
            </xs:element>
            <xs:element name="cacep" type="cacept" />
            <xs:element name="cdap" type="cdapt" />
            <xs:element name="ribdefinition" type="rib-definition" />
            <xs:element name="ribdaemon" type="rib-daemon" />
            <xs:element name="enrollmen" type="enrollment" />
            <xs:element name="nsm" type="nsmt" />
            <xs:element name="flowallocator" type="flow-allocator" />
            <xs:element name="resourceallocator" type="res-allocator" />
            <xs:element name="routing" type="routingt" />
            <xs:element name="secmanager" type="sec-manager" />
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="spec-ref">
         <xs:sequence>
            <xs:element name="specref">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="name" type="xs:string" default="N/A">
                        <xs:annotation i:label="Name of the spec" />
                     </xs:element>
                     <xs:element name="version" type="xs:string" default="N/
A">
                        <xs:annotation i:label="Version of the spec" />
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="param">
         <xs:sequence>
            <xs:element name="parameter">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="name" type="xs:string" default="N/A">
                        <xs:annotation i:label="Name of the parameter" />
                     </xs:element>
                     <xs:element name="value" type="xs:string" default="N/A">
                        <xs:annotation i:label="Value of the parameter" />
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
```

```
   <xs:complexType name="policy-conf">
      <xs:sequence>
         <xs:element name="policyconf">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="name" type="spec-ref" default="N/A">
                     <xs:annotation i:label="Name of the policy spec" />
                  </xs:element>
                  <xs:element name="config-
params" type="param" maxOccurs="unbounded">
                     <xs:annotation i:nonRepeatingTitle="Policy
 parameters" i:label="Parameter" />
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="qos-cube">
      <xs:sequence>
         <xs:element name="qoscube">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="id" type="xs:integer">
                     <xs:annotation i:label="Id of the QoS Cube" />
                  </xs:element>
                  <xs:element name="name" type="xs:string">
                     <xs:annotation i:label="Name of the QoS Cube" />
                  </xs:element>

 <xs:element name="maxAvgCp" type="xs:integer" minOccurs="0">
                     <xs:annotation i:label="Maximum average capacity per
 flow (bits/s)" />
                  </xs:element>

 <xs:element name="avgCpGran" type="xs:integer" minOccurs="0">
                     <xs:annotation i:label="Granularity of average
 capacity per flow (bits/s)" />
                  </xs:element>

 <xs:element name="maxBurstCp" type="xs:integer" minOccurs="0">
                     <xs:annotation i:label="Maximum capacity during
 bursts (bits/s) - flow peak capacity -" />
                  </xs:element>

 <xs:element name="maxBurstDuration" type="xs:integer" minOccurs="0">
```

```
                    <xs:annotation i:label="Maximum duration of burst
 (us)" />
                </xs:element>

 <xs:element name="maxSDUSize" type="xs:integer" minOccurs="0">
                    <xs:annotation i:label="Maximum SDU Size (bytes), if
 different from DIF max. SDU size" />
                </xs:element>
                <xs:element name="partialDelivery" type="xs:boolean">
                   <xs:annotation i:label="Partial Delivery" />
                </xs:element>
                <xs:element name="incompleteDelivery" type="xs:boolean">
                   <xs:annotation i:label="Incomplete Delivery" />
                </xs:element>
                <xs:element name="inOderDelivery" type="xs:boolean">
                   <xs:annotation i:label="In Order Delivery" />
                </xs:element>

 <xs:element name="maxDelay" type="xs:integer" minOccurs="0">
                    <xs:annotation i:label="Maximum Delay (us)" />
                </xs:element>

 <xs:element name="maxJitter" type="xs:integer" minOccurs="0">
                    <xs:annotation i:label="Maximum Jitter (us)" />
                </xs:element>

 <xs:element name="maxSDULoss" type="xs:integer" minOccurs="0">
                    <xs:annotation i:label="Maximum SDU loss probability
 (10^-x) - leave it blank if no losses available -" />
                </xs:element>

 <xs:element name="maxSDUGap" type="xs:integer" minOccurs="0">
                    <xs:annotation i:label="Maximum SDU Gap (num SDUs)" /
>
                </xs:element>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="dif-params">
     <xs:sequence>
        <xs:element name="msdu" type="xs:integer">
           <xs:annotation i:label="Maximum SDU size (bytes)" i:title="5.1
 DIF Parameters" />
        </xs:element>
```

```
        <xs:element name="mpdu" type="xs:integer">
           <xs:annotation i:label="Maximum PDU size (bytes)" />
        </xs:element>
        <xs:element name="seqrovthres" type="xs:integer">
           <xs:annotation i:label="Sequence number rollover threshold" />
        </xs:element>
        <xs:element name="mpdult" type="xs:integer">
           <xs:annotation i:label="Maximum PDU lifetime (ms)" />
        </xs:element>
        <xs:element name="ta" type="xs:integer">
           <xs:annotation i:label="TA: Maximum time an ACK is delayed
before sending (ms)" />
        </xs:element>
        <xs:element name="tg" type="xs:integer">
           <xs:annotation i:label="TG: Maximum time to exhaust
retransmission retries (ms)" />
        </xs:element>
        <xs:element name="tunit" type="xs:integer">
           <xs:annotation i:label="TimeUnit for rate-based flow control
(PDUs sent per time unit) (ms)" />
        </xs:element>
        <xs:element name="sdurtper" type="xs:integer">
           <xs:annotation i:label="SDU Reassembly Timer period (ms)" />
        </xs:element>
        <xs:element name="sdugtper" type="xs:integer">
           <xs:annotation i:label="SDU Gap Timer period (ms)" />
        </xs:element>
     </xs:sequence>
  </xs:complexType>
  <xs:complexType name="efcp-syntax">
     <xs:sequence>
        <xs:element name="addrl" type="xs:integer">
           <xs:annotation i:label="Address length (bytes)" i:title="5.2
EFCP Syntax" />
        </xs:element>
        <xs:element name="cepidl" type="xs:integer">
           <xs:annotation i:label="CEP-id length (bytes)" />
        </xs:element>
        <xs:element name="qosidl" type="xs:integer">
           <xs:annotation i:label="QoS-id length (bytes)" />
        </xs:element>
        <xs:element name="seqnl" type="xs:integer">
           <xs:annotation i:label="Sequence number lenth (bytes)" />
        </xs:element>
        <xs:element name="lengthl" type="xs:integer">
           <xs:annotation i:label="Length length(bytes)" />
```

```
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="qos-cube-policy">
      <xs:sequence>
         <xs:element name="qoscubepolicy">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="id" type="xs:integer">
                     <xs:annotation i:label="Id of the QoS Cube" />
                  </xs:element>
                  <xs:element name="dtppolicies">
                     <xs:complexType>
                        <xs:sequence>

  <xs:element name="atimer" type="xs:integer" default="0">
                              <xs:annotation i:title="5.4.x.1 EFCP
 policies (DTP)" i:label="Initial A Timer" />
                           </xs:element>
                           <xs:element name="flowinitpolicy" type="policy-
conf">
                              <xs:annotation i:label="Flow Init
 policy" i:description="This policy performs any initialization for Data
 Transfer Control for this flow." />
                           </xs:element>
                           <xs:element name="svupdatepolicy" type="policy-
conf">
                              <xs:annotation i:label="State Vector Update
 policy" i:description="This policy is invoked to update Data Transfer
 Control state when a PDU arrives." />
                           </xs:element>

  <xs:element name="lostctrlpdupolicy" type="policy-conf">
                              <xs:annotation i:label="Lost Control PDU
 policy" i:description="This policy determines what to do when a lost
 Control PDU is detected." />
                           </xs:element>

  <xs:element name="rttestimatorpolicy" type="policy-conf">
                              <xs:annotation i:label="RTT estimator
 policy" i:description="This is the algorithm for estimating RTT." />
                           </xs:element>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="dtcprtxcpolicies" minOccurs="0">
```

```
                        <xs:complexType>
                           <xs:sequence>

   <xs:element name="rtxctlpsname" type="xs:string" default="N/A">
                              <xs:annotation i:label="If all the rtx ctl
  policies belong to the same group, name of the set" i:title="5.4.x.2 EFCP
  Retransmission Control policies (DTCP)" />
                           </xs:element>

   <xs:element name="rtxtimerexpirypolicy" type="policy-conf">
                              <xs:annotation i:label="Retransmission timer
  expiry policy" i:description="This policy determines what to do when a
  Retransmission Timer Expires, if the action is not retransmit all PDUs
  with sequence numbers less than this." />
                           </xs:element>
                           <xs:element name="rcvrrtxpolicy" type="policy-
conf">
                              <xs:annotation i:label="Receiver
  retransmission policy" i:description="This policy is executed by the
  receiver to determine when to positively or negatively ack PDUs." />
                           </xs:element>

   <xs:element name="sendingackpolicy" type="policy-conf">
                              <xs:annotation i:label="Sending ACK
  policy" i:description="provides some discretion on when PDUs may be
  deleted from the ReTransmissionQ.  This is useful for multicast and
  similar situations where one might want to delay discarding PDUs from the
  retransmission queue." />
                           </xs:element>

   <xs:element name="sendingacklistpolicy" type="policy-conf">
                              <xs:annotation i:label="Sending ACK list
  policy" i:description="similar to the previous one for selective ack/
nack." />
                           </xs:element>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="dtcprfccpolicies" minOccurs="0">
                     <xs:complexType>
                        <xs:sequence>

   <xs:element name="fctlpsname" type="xs:string" default="N/A">
                              <xs:annotation i:label="If all
  the flow control policies belong to the same group, name of the
  set" i:title="5.4.x.3 EFCP Flow Control policies (DTCP)" />
```

```
                        </xs:element>

  <xs:element name="initcreditpolicy" type="policy-conf">
                           <xs:annotation i:label="Initial credit
 policy" i:description="sets the initial amount of credit on the flow." />
                        </xs:element>
                        <xs:element name="initratepolicy" type="policy-
conf">
                           <xs:annotation i:label="Initial rate
 policy" i:description="sets the initial sending rate to be allowed on the
 flow." />
                        </xs:element>
                        <xs:element name="rcvingfcpolicy" type="policy-
conf">
                           <xs:annotation i:label="Receiving flow
 control policy" i:description="on receipt of a Transfer PDU can to update
 the flow control allocations." />
                        </xs:element>

  <xs:element name="updatecreditpolicy" type="policy-conf">
                           <xs:annotation i:label="Update credit
 policy" i:description="determines how to update the RightWindowEdge, i.e.
 whether the value is absolute or relative to the sequence number." />
                        </xs:element>

  <xs:element name="fcoverrunpoilcy" type="policy-conf">
                           <xs:annotation i:label="Flow control overrun
 policy" i:description="what action to take if the credit or rate has been
 exceeded." />
                        </xs:element>

  <xs:element name="rcfclconflictpolicy" type="policy-conf">
                           <xs:annotation i:label="Reconcile flow
 control conflict policy" i:description="when both Credit and Rate based
 flow control are in use and they disagree on whether the PM can send or
 receive data." />
                        </xs:element>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="rmt-policies">
```

```
      <xs:sequence>
         <xs:element name="rmtpolicies">
            <xs:complexType>
               <xs:sequence>

 <xs:element name="description" type="xs:string" minOccurs="0">
                     <xs:annotation i:label="Description
(optional)" i:title="5.5 Relaying and Multiplexing" />
                  </xs:element>
                  <xs:element name="rmtqmonpolicy" type="policy-conf">
                     <xs:annotation i:label="RMT queue monitor
policy" i:description="Policy for monitoring the status of the RMT and
the QoS being provided by the (N-1)-DIF from data available in the RMT."
/>
                  </xs:element>
                  <xs:element name="rmtschpoicy" type="policy-conf">
                     <xs:annotation i:label="RMT scheduling
policy" i:description="Policy determines what flows are mapped to what
RMT queues and how the queues are serviced." />
                  </xs:element>
                  <xs:element name="rmtmaxqpolicy" type="policy-conf">
                     <xs:annotation i:label="RMT max queue
policy" i:description="Policy invoked if a queue reaches its limit." />
                  </xs:element>
                  <xs:element name="pdufwdingpolicy" type="policy-conf">
                     <xs:annotation i:label="PDU forwarding
policy" i:description="Policy invoked to obtain one or  more N-1 port-ids
through which the PDU should be forwarded." />
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="cacept">
      <xs:sequence>
         <xs:element name="concretesyntax" type="xs:string">
            <xs:annotation i:label="Concrete Syntax" i:title="6.1 Common
Application Connection Establishment Phase" />
         </xs:element>
         <xs:element name="authpolicies" type="policy-
conf" maxOccurs="unbounded">
            <xs:annotation i:nonRepeatingTitle="Authentication
policies" i:label="6.1.x Authentication policy spec" />
         </xs:element>
      </xs:sequence>
```

```
  </xs:complexType>
  <xs:complexType name="cdapt">
     <xs:sequence>
        <xs:element name="concretesyntaxid" type="xs:string">
           <xs:annotation i:label="Concrete Syntax id" i:title="6.2
Common Distributed Application Protocol" />
        </xs:element>
     </xs:sequence>
  </xs:complexType>
  <xs:complexType name="rib-definition">
     <xs:sequence>
        <xs:element name="description" type="xs:string" minOccurs="0">
           <xs:annotation i:label="Description (optional)" i:title="6.3
RIB Definition" />
        </xs:element>
        <xs:element name="ribobjectmodel" type="policy-conf">
           <xs:annotation i:label="RIB Object Model" />
        </xs:element>
     </xs:sequence>
  </xs:complexType>
  <xs:complexType name="rib-daemon">
     <xs:sequence>
        <xs:element name="description" type="xs:string" minOccurs="0">
           <xs:annotation i:label="Description (optional)" i:title="6.4
RIB Daemon" />
        </xs:element>
        <xs:element name="updatepolicy" type="policy-conf">
           <xs:annotation i:label="Update policy" i:description="Specify
a policy with either a reference to a specification or inline." />
        </xs:element>
        <xs:element name="replicationpolicy" type="policy-conf">
           <xs:annotation i:label="Replication
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
        </xs:element>
        <xs:element name="subspolicy" type="policy-conf">
           <xs:annotation i:label="Subscription
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
        </xs:element>
        <xs:element name="logpolicy" type="policy-conf">
           <xs:annotation i:label="Logging policy" i:description="Specify
a policy with either a reference to a specification or inline. Logging
requirements may be different for different IPCPs in the DIF." />
        </xs:element>
        <xs:element name="accpolicy" type="policy-conf">
```

```
            <xs:annotation i:label="RIB access control
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="enrollment">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:label="Description (optional)" i:title="6.5
Enrollment" />
         </xs:element>
         <xs:element name="enrollment" type="policy-conf">
            <xs:annotation i:label="Enrollment specification" />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="nsmt">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:label="Description (optional)" i:title="6.6
Namespace management" />
         </xs:element>
         <xs:element name="addressvalpolicy" type="policy-conf">
            <xs:annotation i:label="Address validation
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="addressasspolicy" type="policy-conf">
            <xs:annotation i:label="Address assignment
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="dirfwpolicy" type="policy-conf">
            <xs:annotation i:label="Directory forwarding
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="dirfwgenpolicy" type="policy-conf">
            <xs:annotation i:label="Directory forwarding table generator
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="flow-allocator">
```

```
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:label="Description (optional)" i:title="6.7
Flow Allocator" />
         </xs:element>
         <xs:element name="allocnotifpolicy" type="policy-conf">
            <xs:annotation i:label="Allocate notify
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="allocretrypolicy" type="policy-conf">
            <xs:annotation i:label="Allocate retry
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="newflowreqpolicy" type="policy-conf">
            <xs:annotation i:label="New flow request
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="seqrolloverpolicy" type="policy-conf">
            <xs:annotation i:label="Sequence rollover
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="flowmonpolicy" type="policy-conf">
            <xs:annotation i:label="Flow monitoring
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="newflowaccpolicy" type="policy-conf">
            <xs:annotation i:label="New Flow access control
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="res-allocator">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:label="Description (optional)" i:title="6.8
Resource allocator" />
         </xs:element>
         <xs:element name="pduftgpolicy" type="policy-conf">
```

```
            <xs:annotation i:label="PDU Forwarding Table generator
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="qosmgmtpolicy" type="policy-conf">
            <xs:annotation i:label="QoS Management
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="congestionmgmtpolicy" type="policy-conf">
            <xs:annotation i:label="Congestion management
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="routingt">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:label="Description (optional)" i:title="6.9
Routing" />
         </xs:element>
         <xs:element name="routing" type="policy-conf">
            <xs:annotation i:label="Routing specification" />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="sec-manager">
      <xs:sequence>
         <xs:element name="description" type="xs:string" minOccurs="0">
            <xs:annotation i:label="Description (optional)" i:title="6.10
Security Manager" />
         </xs:element>
         <xs:element name="credmgmtpolicy" type="policy-conf">
            <xs:annotation i:label="Credential management
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
         <xs:element name="auditpolicy" type="policy-conf">
            <xs:annotation i:label="Auditing
policy" i:description="Specify a policy with either a reference to a
specification or inline." />
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="non-blank-string">
```

```
      <xs:restriction base="xs:string">
         <xs:minLength value="1" />
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

C.2. Wired point to point DIF specification

```
<?xml version="1.0" encoding="utf-8"?>
<dif-template xmlns="http://org.pouzinsociety/2016/dif-template">
  <introduction>
    <name>Wired ptp DIF</name>
    <version>1</version>
    <description>Point to point DIF over wired physical media (real
 standard should be more specific: short reach optics, cat-5 cable,
 etc.) </description>
  </introduction>
  <servicesprovided>
    <maxCapacity>10737418240</maxCapacity>
    <maxSDUSize>9000</maxSDUSize>
    <undetectedBER>12</undetectedBER>
    <qos-cubes>
      <qoscube>
        <id>1</id>
        <name>Low delay</name>
        <maxAvgCp>10737418240</maxAvgCp>
        <avgCpGran>1048576</avgCpGran>
        <maxBurstCp>10737418240</maxBurstCp>
        <partialDelivery>false</partialDelivery>
        <incompleteDelivery>true</incompleteDelivery>
        <inOderDelivery>false</inOderDelivery>
        <maxDelay>2</maxDelay>
        <maxJitter>1</maxJitter>
        <maxSDULoss>8</maxSDULoss>
      </qoscube>
      <qoscube>
        <id>2</id>
        <name>Best effort</name>
        <maxAvgCp>10737418240</maxAvgCp>
        <avgCpGran>1048576</avgCpGran>
        <maxBurstCp>10737418240</maxBurstCp>
        <partialDelivery>false</partialDelivery>
        <incompleteDelivery>true</incompleteDelivery>
        <inOderDelivery>false</inOderDelivery>
        <maxDelay>10</maxDelay>
```

```
        <maxJitter>5</maxJitter>
        <maxSDULoss>6</maxSDULoss>
      </qoscube>
    </qos-cubes>
  </servicesprovided>
  <datatransfer>
    <difparams>
      <msdu>9000</msdu>
      <mpdu>9011</mpdu>
      <seqrovthres>0</seqrovthres>
      <mpdult>1</mpdult>
      <ta>0</ta>
      <tg>0</tg>
      <tunit>1</tunit>
      <sdurtper>0</sdurtper>
      <sdugtper>0</sdugtper>
    </difparams>
    <efcpsyntax>
      <addrl>0</addrl>
      <cepidl>1</cepidl>
      <qosidl>1</qosidl>
      <seqnl>4</seqnl>
      <lengthl>2</lengthl>
    </efcpsyntax>
    <qoscubespolicies>
      <qoscubepolicy>
        <id>1, 2</id>
        <dtppolicies>
          <atimer>0</atimer>
          <flowinitpolicy>
            <policyconf>
              <name>
                <specref>
                  <name>PSOC-default</name>
                  <version>1</version>
                </specref>
              </name>
            </policyconf>
          </flowinitpolicy>
          <svupdatepolicy>
            <policyconf>
              <name>
                <specref>
                  <name>PSOC-default</name>
                  <version>1</version>
                </specref>
```

```
        </name>
      </policyconf>
    </svupdatepolicy>
    <lostctrlpdupolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </lostctrlpdupolicy>
    <rttestimatorpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </rttestimatorpolicy>
  </dtppolicies>
  <dtcprfccpolicies>
    <fctlpsname>PSOC-rate-based-default</fctlpsname>
    <initratepolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-rate-based-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </initratepolicy>
    <fcoverrunpoilcy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-rate-based-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </fcoverrunpoilcy>
```

```
      </dtcprfccpolicies>
    </qoscubepolicy>
  </qoscubespolicies>
  <mtpolicies>
    <rmtpolicies>
      <description>QTA Mux scheduling</description>
      <rmtqmonpolicy>
        <policyconf>
          <name>
            <specref>
              <name>PSOC-QTAMux</name>
              <version>1</version>
            </specref>
          </name>
        </policyconf>
      </rmtqmonpolicy>
      <rmtschpoicy>
        <policyconf>
          <name>
            <specref>
              <name>PSOC-QTAMux</name>
              <version>1</version>
            </specref>
          </name>
          <config-params>
            <parameter>
              <name>C_U_Mux_Order</name>
              <value>2</value>
            </parameter>
          </config-params>
        </policyconf>
      </rmtschpoicy>
      <rmtmaxqpolicy>
        <policyconf>
          <name>
            <specref>
              <name>PSOC-QTAMux</name>
              <version>1</version>
            </specref>
          </name>
        </policyconf>
      </rmtmaxqpolicy>
    </rmtpolicies>
  </mtpolicies>
</datatransfer>
<layermanagement>
```

```
<cacep>
  <concretesyntax>1</concretesyntax>
  <authpolicies>
    <policyconf>
      <name>
        <specref>
          <name>PSOC-Auth-none</name>
          <version>1</version>
        </specref>
      </name>
    </policyconf>
  </authpolicies>
</cacep>
<cdap>
  <concretesyntaxid>1</concretesyntaxid>
</cdap>
<ribdefinition>
  <description>PSOC-default-lite</description>
  <ribobjectmodel>
    <policyconf>
      <name>
        <specref>
          <name>PSOC-default-lite</name>
          <version>1</version>
        </specref>
      </name>
    </policyconf>
  </ribobjectmodel>
</ribdefinition>
<ribdaemon>
  <logpolicy>
    <policyconf>
      <name>
        <specref>
          <name>PSOC-default</name>
          <version>1</version>
        </specref>
      </name>
    </policyconf>
  </logpolicy>
</ribdaemon>
<enrollmen>
  <enrollment>
    <policyconf>
      <name>
        <specref>
```

```
            <name>PSOC-ptp-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </enrollment>
  </enrollmen>
  <nsm>
    <dirfwpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-ptp-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </dirfwpolicy>
  </nsm>
  <flowallocator>
    <description>Default policy for point to point links</description>
    <allocnotifpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-ptp-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </allocnotifpolicy>
    <allocretrypolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-ptp-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </allocretrypolicy>
    <newflowreqpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-ptp-default</name>
```

```
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </newflowreqpolicy>
    <seqrolloverpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </seqrolloverpolicy>
    <flowmonpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-default</name>
            <version>1</version>
          </specref>
        </name>
      </policyconf>
    </flowmonpolicy>
  </flowallocator>
  <resourceallocator>
    <qosmgmtpolicy>
      <policyconf>
        <name>
          <specref>
            <name>PSOC-default-ptp-qtamux</name>
            <version>1</version>
          </specref>
        </name>
        <config-params>
          <parameter>
            <name>QTAMuxOrder</name>
            <value>2</value>
          </parameter>
        </config-params>
      </policyconf>
    </qosmgmtpolicy>
  </resourceallocator>
 </layermanagement>
</dif-template>
```

C.3. Data centre fabric DIF specification

```
<?xml version="1.0" encoding="utf-8"?>
<dif-template xmlns="http://org.pouzinsociety/2016/dif-template">
  <introduction>
    <name>DC Fabric DIF</name>
    <version>1</version>
    <description>DIF template for a large-scale DCN fabric using
 topological addressing</description>
  </introduction>
  <references>
    <specref>
      <name>N/A</name>
      <version>N/A</version>
    </specref>
  </references>
  <servicesprovided>
    <description>4 QoS cubes: i) loss and delay sensitive; ii) delay
 sensitive; iii) loss sensitive; iV) best effort.</description>
    <qos-cubes>
      <qoscube>
        <id>1</id>
        <name>loss-delay sensitive</name>
        <avgBw>0</avgBw>
        <avgSDUBw>0</avgSDUBw>
        <peakBw>0</peakBw>
        <peakSDUBw>0</peakSDUBw>
        <undetectedBER>12</undetectedBER>
        <maxSDUSize>9000</maxSDUSize>
        <partialDelivery>false</partialDelivery>
        <incompleteDelivery>false</incompleteDelivery>
        <inOderDelivery>false</inOderDelivery>
        <maxDelay>5</maxDelay>
        <maxJitter>1</maxJitter>
        <maxSDUGap>0</maxSDUGap>
      </qoscube>
      <qoscube>
        <id>2</id>
        <name>loss sensitive</name>
        <avgBw>0</avgBw>
        <avgSDUBw>0</avgSDUBw>
        <peakBw>0</peakBw>
        <peakSDUBw>0</peakSDUBw>
        <undetectedBER>12</undetectedBER>
        <maxSDUSize>9000</maxSDUSize>
```

```
      <partialDelivery>false</partialDelivery>
      <incompleteDelivery>false</incompleteDelivery>
      <inOderDelivery>false</inOderDelivery>
      <maxDelay>20</maxDelay>
      <maxJitter>10</maxJitter>
      <maxSDUGap>0</maxSDUGap>
    </qoscube>
    <qoscube>
      <id>3</id>
      <name>delay sensitive</name>
      <avgBw>0</avgBw>
      <avgSDUBw>0</avgSDUBw>
      <peakBw>0</peakBw>
      <peakSDUBw>0</peakSDUBw>
      <undetectedBER>9</undetectedBER>
      <maxSDUSize>9000</maxSDUSize>
      <partialDelivery>false</partialDelivery>
      <incompleteDelivery>false</incompleteDelivery>
      <inOderDelivery>false</inOderDelivery>
      <maxDelay>5</maxDelay>
      <maxJitter>1</maxJitter>
      <maxSDUGap>0</maxSDUGap>
    </qoscube>
    <qoscube>
      <id>4</id>
      <name>best effort</name>
      <avgBw>0</avgBw>
      <avgSDUBw>0</avgSDUBw>
      <peakBw>0</peakBw>
      <peakSDUBw>0</peakSDUBw>
      <undetectedBER>9</undetectedBER>
      <maxSDUSize>9000</maxSDUSize>
      <partialDelivery>false</partialDelivery>
      <incompleteDelivery>false</incompleteDelivery>
      <inOderDelivery>false</inOderDelivery>
      <maxDelay>20</maxDelay>
      <maxJitter>10</maxJitter>
      <maxSDUGap>0</maxSDUGap>
    </qoscube>
  </qos-cubes>
  <system-apis>
    <specref>
      <name>N/A</name>
      <version>N/A</version>
    </specref>
  </system-apis>
```

```
  </servicesprovided>
  <servicesrequired>
    <description>Several N-1 DIFs which provided 9000 bytes MTU, 10^-9 bit
 error rate, 2 ms delay.</description>
  </servicesrequired>
  <datatransfer>
    <description>No rtx. control, just flow control. No fragmentation/
reassembly.</description>
    <difparams>
      <msdu>9000</msdu>
      <mpdu>9017</mpdu>
      <seqrovthres>0</seqrovthres>
      <mpdult>1000</mpdult>
      <ta>0</ta>
      <tg>0</tg>
      <tunit>0</tunit>
      <sdurtper>0</sdurtper>
      <sdugtper>0</sdugtper>
    </difparams>
    <efcpsyntax>
      <addrl>3</addrl>
      <cepidl>2</cepidl>
      <qosidl>1</qosidl>
      <seqnl>4</seqnl>
      <lengthl>2</lengthl>
    </efcpsyntax>
    <delimiting>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </delimiting>
    <qoscubespolicies>
      <qoscubepolicy>
        <id>0</id>
        <dtppolicies>
          <atimer>0</atimer>
          <flowinitpolicy>
            <specref>
              <name>defaut</name>
              <version>default</version>
            </specref>
          </flowinitpolicy>
          <svupdatepolicy>
            <specref>
              <name>default</name>
```

```
        <version>default</version>
      </specref>
    </svupdatepolicy>
    <lostctrlpdupolicy>
      <specref>
        <name>default</name>
        <version>default</version>
      </specref>
    </lostctrlpdupolicy>
    <rttestimatorpolicy>
      <specref>
        <name>default</name>
        <version>default</version>
      </specref>
    </rttestimatorpolicy>
  </dtppolicies>
  <dtcprfccpolicies>
    <fctlpsname>congestion-avoidance</fctlpsname>
    <initcreditpolicy>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </initcreditpolicy>
    <initratepolicy>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </initratepolicy>
    <rcvingfcpolicy>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </rcvingfcpolicy>
    <updatecreditpolicy>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </updatecreditpolicy>
    <fcoverrunpoilcy>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
```

```
            </specref>
          </fcoverrunpoilcy>
          <rcfclconflictpolicy>
            <specref>
              <name>N/A</name>
              <version>N/A</version>
            </specref>
          </rcfclconflictpolicy>
        </dtcprfccpolicies>
      </qoscubepolicy>
    </qoscubespolicies>
    <mtpolicies>
      <rmtpolicies>
        <rmtqmonpolicy>
          <specref>
            <name>qtamux</name>
            <version>1</version>
          </specref>
        </rmtqmonpolicy>
        <rmtschpoicy>
          <specref>
            <name>qtamux</name>
            <version>q</version>
          </specref>
        </rmtschpoicy>
        <rmtmaxqpolicy>
          <specref>
            <name>qtamux</name>
            <version>1</version>
          </specref>
        </rmtmaxqpolicy>
        <pdufwdingpolicy>
          <specref>
            <name>ecmp</name>
            <version>1</version>
          </specref>
        </pdufwdingpolicy>
      </rmtpolicies>
    </mtpolicies>
    <sduprotection>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </sduprotection>
  </datatransfer>
```

```
<layermanagement>
  <cacep>
    <concretesyntax>GPB</concretesyntax>
    <authpolicies>
      <specref>
        <name>N/A</name>
        <version>N/A</version>
      </specref>
    </authpolicies>
  </cacep>
  <cdap>
    <concretesyntaxid>1</concretesyntaxid>
  </cdap>
  <ribdefinition>
    <description>Uses default RIB</description>
    <ribobjectmodel>
      <specref>
        <name>default</name>
        <version>default</version>
      </specref>
    </ribobjectmodel>
  </ribdefinition>
  <ribdaemon>
    <updatepolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </updatepolicy>
    <replicationpolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </replicationpolicy>
    <subspolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </subspolicy>
    <logpolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
```

```
    </logpolicy>
    <accpolicy>
      <specref>
        <name>none</name>
        <version>N/A</version>
      </specref>
    </accpolicy>
  </ribdaemon>
  <enrollmen>
    <enrollment>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </enrollment>
  </enrollmen>
  <nsm>
    <description>centralized address mgmt</description>
    <addressvalpolicy>
      <specref>
        <name>central</name>
        <version>1</version>
      </specref>
    </addressvalpolicy>
    <addressasspolicy>
      <specref>
        <name>central</name>
        <version>1</version>
      </specref>
    </addressasspolicy>
    <dirfwpolicy>
      <specref>
        <name>central</name>
        <version>1</version>
      </specref>
    </dirfwpolicy>
    <dirfwgenpolicy>
      <specref>
        <name>central</name>
        <version>1</version>
      </specref>
    </dirfwgenpolicy>
  </nsm>
  <flowallocator>
    <allocnotifpolicy>
      <specref>
```

```
        <name>default</name>
        <version>1</version>
      </specref>
    </allocnotifpolicy>
    <allocretrypolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </allocretrypolicy>
    <newflowreqpolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </newflowreqpolicy>
    <seqrolloverpolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </seqrolloverpolicy>
    <flowmonpolicy>
      <specref>
        <name>default</name>
        <version>1</version>
      </specref>
    </flowmonpolicy>
    <newflowaccpolicy>
      <specref>
        <name>none</name>
        <version>N/A</version>
      </specref>
    </newflowaccpolicy>
  </flowallocator>
  <resourceallocator>
    <pduftgpolicy>
      <specref>
        <name>fb-dcn-topological</name>
        <version>1</version>
      </specref>
    </pduftgpolicy>
    <qosmgmtpolicy>
      <specref>
        <name>qta</name>
        <version>1</version>
```

```
        </specref>
      </qosmgmtpolicy>
      <congestionmgmtpolicy>
        <specref>
          <name>congestion-avoidance</name>
          <version>1</version>
        </specref>
      </congestionmgmtpolicy>
    </resourceallocator>
    <routing>
      <description>Link-state that just disseminates information on broken
 N-1 flows</description>
      <routing>
        <specref>
          <name>link-state-errors</name>
          <version>1</version>
        </specref>
      </routing>
    </routing>
    <secmanager>
      <credmgmtpolicy>
        <specref>
          <name>N/A</name>
          <version>N/A</version>
        </specref>
      </credmgmtpolicy>
      <auditpolicy>
        <specref>
          <name>N/A</name>
          <version>N/A</version>
        </specref>
      </auditpolicy>
    </secmanager>
  </layermanagement>
</dif-template>
```