DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2018

S. Matsushima
SoftBank
L. Bertz
Sprint
M. Liebsch
NEC
S. Gundavelli
Cisco
D. Moses
Intel Corporation
C. Perkins
Futurewei
June 18, 2018

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-12

Abstract

This document describes a way, called Forwarding Policy Configuration (FPC) to manage the separation of data-plane and control-plane. FPC defines a flexible mobility management system using FPC agent and FPC client functions. A FPC agent provides an abstract interface to the data-plane. The FPC client configures data-plane nodes by using the functions and abstractions provided by the FPC agent for the data-plane nodes. The data-plane abstractions presented in this document are extensible in order to support many different types of mobility management systems and data-plane functions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 20, 2018.

Table of Contents

1. Introduction

This document describes Forwarding Policy Configuration (FPC), a system for managing the separation of control-plane and data-plane. FPC enables flexible mobility management using FPC client and FPC agent functions. A FPC agent exports an abstract interface representing the data-plane. To configure data-plane nodes and functions, the FPC client uses the interface to the data-plane offered by the FPC agent.

Control planes of mobility management systems, or related applications which require data-plane control, can utilize the FPC client at various levels of abstraction. FPC operations are capable of directly configuring a single Data-Plane Node (DPN), as well as multiple DPNs, as determined by the data-plane models exported by the FPC agent.

A FPC agent represents the data-plane operation according to several basic information models. A FPC agent also provides access to Monitors, which produce reports when triggered by events or FPC Client requests regarding Mobility Contexts, DPNs or the Agent.

To manage mobility sessions, the FPC client assembles applicable sets of forwarding policies from the data model, and configures them on the appropriate FPC Agent. The Agent then renders those policies into specific configurations for each DPN at which mobile nodes are attached. The specific protocols and configurations to configure a DPN from a FPC Agent are outside the scope of this document.

A DPN is a logical entity that performs data-plane operations (packet movement and management). It may represent a physical DPN unit, a sub-function of a physical DPN or a collection of physical DPNs (i.e., a "virtual DPN"). A DPN may be virtual -- it may export the FPC DPN Agent interface, but be implemented as software that controls other data-plane hardware or modules that may or may not be FPC-compliant. In this document, DPNs are specified without regard for whether the implementation is virtual or physical. DPNs are connected to provide mobility management systems such as access networks, anchors and domains. The FPC agent interface enables establishment of a topology for the forwarding plane.

When a DPN is mapped to physical data-plane equipment, the FPC client can have complete knowledge of the DPN architecture, and use that information to perform DPN selection for specific sessions. On the other hand, when a virtual DPN is mapped to a collection of physical DPNs, the FPC client cannot select a specific physical DPN because it is hidden by the abstraction; only the FPC Agent can address the specific associated physical DPNs. Network architects have the flexibility to determine which DPN-selection capabilities are performed by the FPC Agent (distributed) and which by the FPC client (centralized). In this way, overlay networks can be configured without disclosing detailed knowledge of the underlying hardware to the FPC client and applications.

The abstractions in this document are designed to support many different mobility management systems and data-plane functions. The architecture and protocol design of FPC is not tied to specific types of access technologies and mobility protocols.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Attribute Expression: The definition of a template Property. This includes setting the type, current value, default value and if the attribute is static, i.e. can no longer be changed.

Domain:	One or more DPNs that form a logical partition of network resources (e.g., a data-plane network under common network administration). A FPC client (e.g., a mobility management system) may utilize a single or multiple domains.
DPN:	A data-plane node (DPN) is capable of performing data-plane features. For example, DPNs may be switches or routers, regardless of whether they are realized as hardware or purely in software.
FPC Client:	A FPC Client is integrated with a mobility management system or related application, enabling control over forwarding policy, mobility sessions and DPNs via a FPC Agent.
Mobility Context:	A Mobility Context contains the data-plane information necessary to efficiently send and receive traffic from a mobile node. This includes policies that are created or modified during the network's operation - in most cases, on a per-flow or per session basis. A Mobility-Context represents the mobility sessions (or flows) which are active on a mobile node. This includes associated runtime attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Mobility-Contexts are associated to specific DPNs. Some pre-defined Policies may apply during mobility signaling requests. The Mobility Context supplies information about the policy settings specific to a mobile node and its flows; this information is often quite dynamic.
Mobility Session:	Traffic to/from a mobile node that is expected to survive reconnection events.
Monitor:	A reporting mechanism for a list of events that trigger notification messages from a FPC Agent to a FPC Client.
Policy:	A Policy determines the mechanisms for managing specific traffic flows or packets. Policies specify QoS, rewriting rules for

	packet processing, etc. A Policy consists of one or more rules. Each rule is composed of a Descriptor and Actions. The Descriptor in a rule identifies packets (e.g., traffic flows), and the Actions apply treatments to packets that match the Descriptor in the rule. Policies can apply to Domains, DPNs, Mobile Nodes, Service-Groups, or particular Flows on a Mobile Node.
Property:	An attribute-value pair for an instance of a FPC entity.
Service-Group:	A set of DPN interfaces that support a specific data-plane purpose, e.g. inbound/ outbound, roaming, subnetwork with common specific configuration, etc.
Template:	A recipe for instantiating FPC entities. Template definitions are accessible (by name or by a key) in an indexed set. A Template is used to create specific instances (e.g., specific policies) by assigning appropriate values into the Template definition via Attribute Expression.
Template Configuration	The process by which a Template is referenced (by name or by key) and Attribute Expressions are created that change the value, default value or static nature of the Attribute, if permitted. If the Template is Extensible, new attributes MAY be added.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions. A Tenant defines a global namespace for all entities owned by the Tenant enabling its entities to be used by multiple FPC Clients across multiple FPC Agents.
Topology:	The DPNs and the links between them. For example, access nodes may be assigned to a Service-Group which peers to a Service-Group of anchor nodes.

3. FPC Design Objectives and Deployment

Using FPC, mobility control-planes and applications can configure DPNs to perform various mobility management roles as described in [I-D.ietf-dmm-deployment-models]. This fulfills the requirements described in [RFC7333].

This document defines FPC Agent and FPC Client, as well as the information models that they use. The attributes defining those models serve as the protocol elements for the interface between the FPC Agent and the FPC Client.

Mobility control-plane applications integrate features offered by the FPC Client. The FPC Client connects to FPC Agent functions. The Client and the Agent communicate based on information models described in Section 4. The models allow the control-plane to configure forwarding policies on the Agent for data-plane communications with mobile nodes.

Once the Topology of DPN(s) and domains are defined on an Agent for a data plane, the DPNs in the topology are available for further configuration. The FPC Agent connects those DPNs to manage their configurations.

A FPC Agent configures and manages its DPN(s) according to forwarding policies requested and Attributes provided by the FPC Client. Configuration commands used by the FPC agent to configure its DPN node(s) may be specific to the DPN implementation; consequently the method by which the FPC Agent carries out the specific configuration for its DPN(s) is out of scope for this document. Along with the data models, the FPC Client (on behalf of control-plane and applications) requests that the Agent configures Policies prior to the time when the DPNs start forwarding data for their mobility sessions.

This architecture is illustrated in Figure 1. A FPC Agent may be implemented in a network controller that handles multiple DPNs, or (more simply) an FPC Agent may itself be integrated into a DPN.

This document does not specify a protocol for the FPC interface; it is out of scope. However, an implementation must support the FPC transactions described in Section 5.

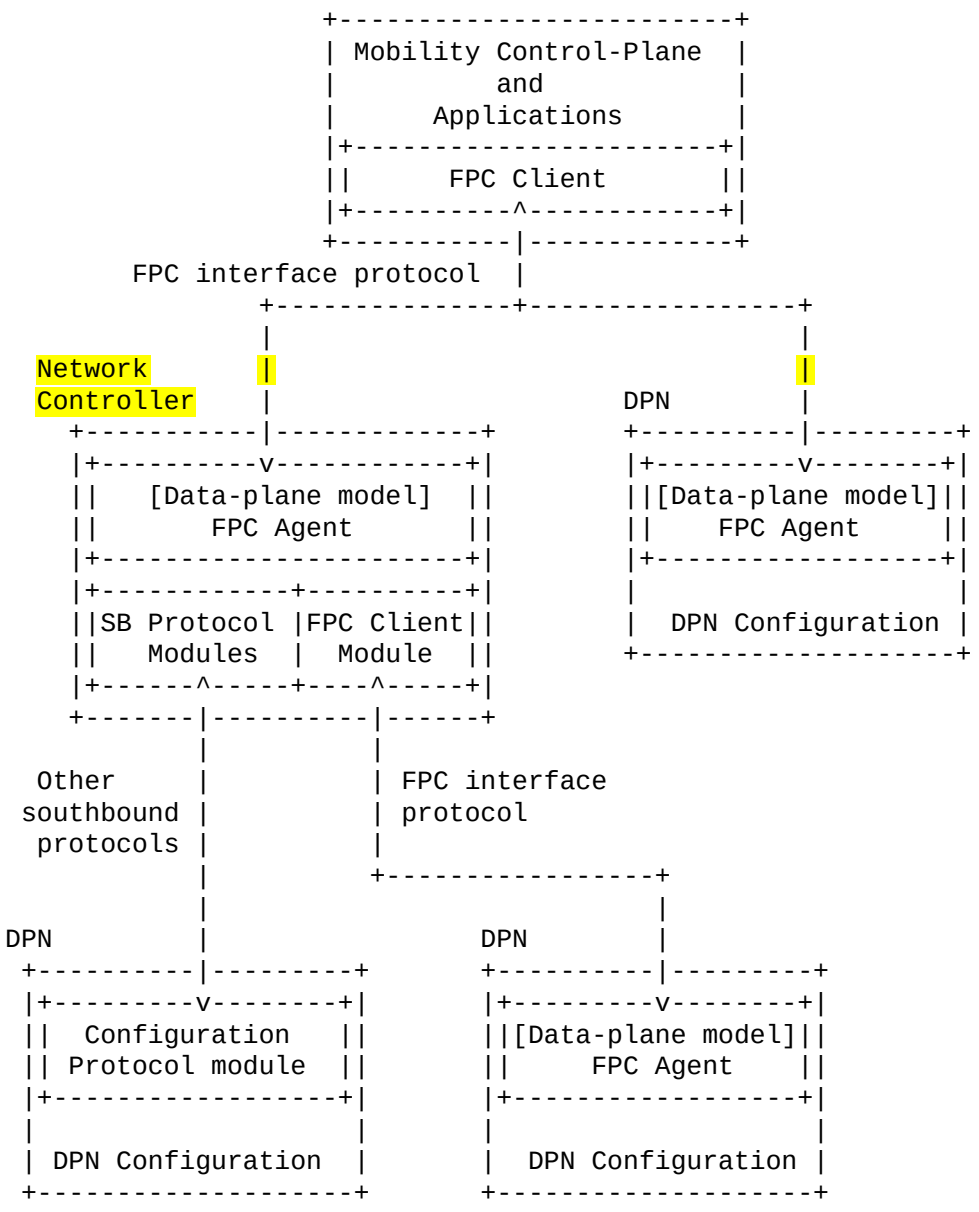


Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

The FPC architecture supports multi-tenancy; a FPC enabled data-plane supports tenants of multiple mobile operator networks and/or applications. It means that the FPC Client of each tenant connects to the FPC Agent and it MUST partition namespace and data for their data-planes. DPNs on the data-plane may fulfill multiple data-plane roles which are defined per session, domain and tenant.

Multi-tenancy permits the paritioning of data-plane entities as well as a common namespace requirement upon FPC Agents and Clients when they use the same Tenant for a common data-plane entity.

FPC information models often configuration to fit the specific needs for DPN management of a mobile node's traffic. The FPC interfaces in Figure 1 are the only interfaces required to handle runtime data in a Mobility Context. The Topology and some Policy FPC models MAY be pre-configured; in that case real-time protocol exchanges are not required for them.

The information model provides an extensibility mechanism through Templates that permits specialization for the needs of a particular vendor's equipment or future extension of the model presented in this specification.

4. FPC Mobility Information Model

The FPC information model includes the following components:

DPN Information Model,
Topology Information Model,
Policy Information Model,
Mobility-Context, and
Monitor, as illustrated in Figure 2.

```
                  :
                  |
                  +-[FPC Mobility Information Model]
                  |           |
                  |           +-[Topology Information Model]
                  |           |
                  |           +-[Policy Information Model]
                  |           |
                  |           +-[Mobility-Context]
                  |           |
                  |           +-[Monitor]
                  |
```

Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

The following conventions are used to describe the FPC information models.

Information model entities (e.g. DPNs, Rules, etc.) are defined in a hierarchical notation where all entities at the same hierarchical

level are located on the same left-justified vertical position sequentially. When entities are composed of sub-entities, the sub-entities appear shifted to the right, as shown in Figure 3.

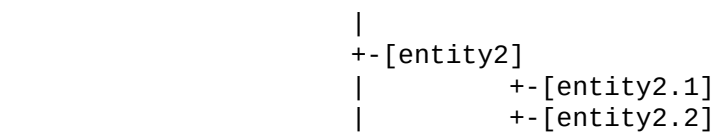


Figure 3: Model Notation - An Example

Some entities have one or more qualifiers placed on the right hand side of the element definition in angle-brackets. Common types include:

List: A collection of entities (some could be duplicated)

Set: A nonempty collection of entities without duplications

Name: A human-readable string

Key: A unique value. We distinguish 3 types of keys:

U-Key: A key unique across all Tenants. U-Key spaces typically involve the use of registries or language specific mechanisms that guarantee universal uniqueness of values.

G-Key: A key unique within a Tenant

L-Key: A key unique within a local namespace. For example, there may exist interfaces with the same name, e.g. "if0", in two different DPNs but there can only be one "if0" within each DPN (i.e. its local Interface-Key L-Key space).

Each entity or attribute may be optional (O) or mandatory (M). Entities that are not marked as optional are mandatory.

```
The following example shows 3 entities:
    -- Entity1 is a globally unique key, and optionally can have
               an associated Name
    -- Entity2 is a list
    -- Entity3 is a set and is optional
               +
               |
               +-[entity1] <G-Key> (M), <Name> (O)
               +-[entity2] <List>
               +-[entity3] <Set> (O)
               |
               +
```

Figure 4

When expanding entity1 into a modeling language such as YANG it would result in two values: entity1-Key and entity1-Name.

To encourage re-use, FPC defines indexed sets of various entity Templates. Other model elements that need access to an indexed model entity contain an attribute which is always denoted as "entity-Key". When a Key attribute is encountered, the referencing model element may supply attribute values for use when the referenced entity model is instantiated. For example: Figure 5 shows 2 entities:

EntityA definition references an entityB model element.

EntityB model elements are indexed by entityB-Key.

Each EntityB model element has an entityB-Key which allows it to be uniquely identified, and a list of Attributes (or, alternatively, a Type) which specifies its form. This allows a referencing entity to create an instance by supplying entityB-Values to be inserted, in a Settings container.

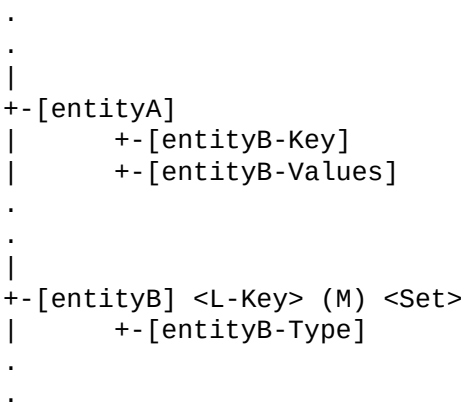


Figure 5: Indexed sets of entities

Indexed sets are specified for each of the following kinds of entities:

- Domain (See Section 4.9.3)
- DPN (See Section 4.9.4)
- Policy (See Section 4.9.5)
- Rule (See Section 4.9.5)
- Descriptor (See Figure 12)
- Action (See Figure 12)
- Service-Group (See Section 4.9.2, and
- Mobility-Context (See Section 4.9.6)

As an example, for a Domain entity, there is a corresponding attribute denoted as "Domain-Key" whose value can be used to determine a reference to the Domain.

4.2. Templates and Attributes

In order to simplify development and maintenance of the needed policies and other objects used by FPC, the Information Models which are presented often have attributes that are not initialized with their final values. When an FPC entity is instantiated according to a template definition, specific values need to be configured for each such attribute. For instance, suppose an entity Template has an Attribute named "IPv4-Address", and also suppose that a FPC Client instantiates the entity and requests that it be installed on a DPN. An IPv4 address will be needed for the value of that Attribute before the entity can be used.

```
+-[Template] <U-Key, Name> (M) <Set>
|      +-[Attributes] <Set> (M)
|      +-[Extensible ~ FALSE]
|      +-[Entity-State ~ Initial]
|      +-[Version]
```

Figure 6: Template entities

Attributes: A set of Attribute names MAY be included when defining a Template for instantiating FPC entities.

Extensible: Determines whether or not entities instantiated from the Template can be extended with new non-mandatory Attributes not originally defined for the Template. Default value is FALSE. If a Template does not explicitly specify this attribute, the default value is considered to be in effect.

Entity-State: Either Initial, PartiallyConfigured, Configured, or Active. Default value is Initial. See Section 4.6 for more information about how the Entity-Status changes during the configuration steps of the Entity.

Version: Provides a version tag for the Template.

The Attributes in an Entity Template may be either mandatory or non-mandatory. Attribute values may also be associated with the attributes in the Entity Template. If supplied, the value may be either assigned with a default value that can be reconfigured later, or the value can be assigned with a static value that cannot be reconfigured later (see Section 4.3).

It is possible for a Template to provide values for all of its Attributes, so that no additional values are needed before the entity can made Active. Any instantiation from a Template MUST have at least one Attribute in order to be a useful entity unless the Template has none.

4.3. Attribute-Expressions

The syntax of the Attribute definition is formatted to make it clear. For every Attribute in the Entity Template, six possibilities are specified as follows:

'[Att-Name:]' Mandatory Attribute is defined, but template does not provide any configured value.

'[Att-Name: Att-Value]' Mandatory Attribute is defined, and has a statically configured value.

'[Att-Name: ~ Att-Value]' Mandatory Attribute is defined, and has a default value.

'[Att-Name]' Non-mandatory Attribute may be included but template does not provide any configured value.

'[Att-Name = Att-Value]' Non-mandatory Attribute may be included and has a statically configured value.

'[Att-Name ~ Att-Value]' Non-mandatory Attribute may be included and has a default value.

So, for example, a default value for a non-mandatory IPv4-Address attribute would be denoted by [IPv4-Address ~ 127.0.0.1].

After a FPC Client identifies which additional Attributes have been configured to be included in an instantiated entity, those configured Attributes MUST NOT be deleted by the FPC Agent. Similarly, any statically configured value for an entity Attribute MUST NOT be changed by the FPC Agent.

Whenever there is danger of confusion, the fully qualified Attribute name MUST be used when supplying needed Attribute Values for a structured Attribute.

4.4. Attribute Value Types

For situations in which the type of an attribute value is required, the following syntax is recommended. To declare than an attribute has data type "foo", typecast the attribute name by using the parenthesized data type (foo). So, for instance, [(float) Max-Latency-in-ms:] would indicate that the mandatory Attribute "Max-Latency-in-ms" requires to be configured with a floating point value before the instantiated entity could be used. Similarly, [(float) Max-Latency-in-ms: 9.5] would statically configure a floating point value of 9.5 to the mandatory Attribute "Max-Latency-in-ms".

4.5. Namespace and Format

The identifiers and names in FPC models which reside in the same Tenant must be unique. That uniqueness must be maintained by all Clients, Agents and DPNs that support the Tenant. The Tenant namespace uniqueness MUST be applied to all elements of the tenant model, i.e. Topology, Policy and Mobility models.

When a Policy needs to be applied to Mobility-Contexts in all Tenants on an Agent, the Agent SHOULD define that policy to be visible by all Tenants. In this case, the Agent assigns a unique identifier in the

Agent namespace and copies the values to each Tenant. This effectively creates a U-Key although only a G-Key is required within the Tenant.

The notation for identifiers can utilize any format with agreement between data-plane agent and client operators. The formats include but are not limited to Globally Unique IDentifiers (GUIDs), Universally Unique IDentifiers (UUIDs), Fully Qualified Domain Names (FQDNs), Fully Qualified Path Names (FQPNs) and Uniform Resource Identifiers (URIs). The FPC model does not limit the format, which could dictate the choice of FPC protocol. Nevertheless, the identifiers which are used in a Mobility model should be considered to efficiently handle runtime parameters.

There are identifiers reserved for Protocol Operation. See Section 5.1.1.5 for details.

4.6. Configuring Attribute Values

Attributes of Information Model components such as policy templates are configured with values as part of FPC configuration operations. There may be several such configuration operations before the template instantiation is fully configured.

Entity-Status indicates when an Entity is usable within a DPN. This permits DPN design tradeoffs amongst local storage (or other resources), over the wire request size and the speed of request processing. For example, DPN designers with constrained systems MAY only house entities whose status is Active which may result in sending over all policy information with a Mobility-Context request. Storing information elements with an entity status of "PartiallyConfigured" on the DPN requires more resources but can result in smaller over the wire FPC communication and request processing efficiency.

When the FPC Client instantiates a Policy from a Template, the Policy-Status is "Initial". When the FPC Client sends the policy to a FPC Agent for installation on a DPN, the Client often will configure appropriate attribute values for the installation, and accordingly changes the Policy-Status to "PartiallyConfigured" or "Configured". The FPC Agent will also configure Domain-specific policies and DPN-specific policies on the DPN. When configured to provide particular services for mobile nodes, the FPC Agent will apply whatever service-specific policies are needed on the DPN. When a mobile node attaches to the network data-plane within the topology under the jurisdiction of a FPC Agent, the Agent may apply policies and settings as appropriate for that mobile node. Finally, when the mobile node launches new flows, or quenches existing flows, the FPC

Agent, on behalf of the FPC Client, applies or deactivates whatever policies and attribute values are appropriate for managing the flows of the mobile node. When a "Configured" policy is de-activated, Policy-Status is changed to be "Active". When an "Active" policy is activated, Policy-Status is changed to be "Configured".

Attribute values in DPN resident Policies may be configured by the FPC Agent as follows:

Domain-Policy-Configuration: Values for Policy attributes that are required for every DPN in the domain.

DPN-Policy-Configuration: Values for Policy attributes that are required for every policy configured on this DPN.

Service-Group-Policy-Configuration: Values for Policy attributes that are required to carry out the intended Service of the Service Group.

MN-Policy-Configuration: Values for Policy attributes that are required for all traffic to/from a particular mobile node.

Service-Data-Flow-Policy-Configuration: Values for Policy attributes that are required for traffic belonging to a particular set of flows on the mobile node.

Any configuration changes MAY also supply updated values for existing default attribute values that may have been previously configured on the DPN resident policy.

Entity blocks describe the format of the policy configurations.

4.7. Entity Configuration Blocks

As described in Section 4.6, a Policy Template may be configured in several stages by configuring default or missing values for Attributes that do not already have statically configured values. A Policy-Configuration is the combination of a Policy-Key (to identify the Policy Template defining the Attributes) and the currently configured Attribute Values to be applied to the Policy Template. Policy-Configurations MAY add attributes to a Template if Extensible is True. They MAY also refine existing attributes by:

assign new values if the Attribute is not static

make attributes static if they were not

make an attribute mandatory

A Policy-Configuration MUST NOT define or refine an attribute twice. More generally, an Entity-Configuration can be defined for any configurable Indexed Set to be the combination of the Entity-Key along with a set of Attribute-Expressions that supply configuration information for the entity's Attributes. Figure 7 shows a schematic representation for such Entity Configuration Blocks.

```
[Entity Configuration Block]
|       +-[Entity-Key] (M)
|       +-[Attribute-Expression] <Set> (M)
```

Figure 7: Entity Configuration Block

This document makes use of the following kinds of Entity Configuration Blocks:

Descriptor-Configuration

Action-Configuration

Rule-Configuration

Interface-Configuration

Service-Group-Configuration

Domain-Policy-Configuration

DPN-Policy-Configuration

Policy-Configuration

MN-Policy-Configuration

Service-Data-Flow-Policy-Configuration

4.8. Information Model Checkpoint

The Information Model Checkpoint permits Clients and Tenants with common scopes, referred to in this specification as Checkpoint BaseNames, to track the state of provisioned information on an Agent. The Agent records the Checkpoint BaseName and Checkpoint value set by a Client. When a Client attaches to the Agent it can query to determine the amount of work that must be executed to configure the Agent to a specific BaseName / checkpoint revision.

Checkpoints are defined for the following information model components:

Service-Group

DPN Information Model

Domain Information Model

Policy Information Model

4.9. Information Model Components

4.9.1. Topology Information Model

The Topology structure specifies DPNs and the communication paths between them. A network management system can use the Topology to select the most appropriate DPN resources for handling specific session flows.

The Topology structure is illustrated in Figure 8 (for definitions see Section 2):

```
                    |
                    +-[Topology Information Model]
                    |          +-[Extensible: FALSE]
                    |          +-[Service-Group]
                    |          +-[DPN] <Set>
                    |          +-[Domain] <Set>
```

Figure 8: Topology Structure

4.9.2. Service-Group

Service-Group-Set is collection of DPN interfaces serving some data-plane purpose including but not limited to DPN Interface selection to fulfill a Mobility-Context. Each Group contains a list of DPNs (referenced by DPN-Key) and selected interfaces (referenced by Interface-Key). The Interfaces are listed explicitly (rather than referred implicitly by its specific DPN) so that every Interface of a DPN is not required to be part of a Group. The information provided is sufficient to ensure that the Protocol, Settings (stored in the Service-Group-Configuration) and Features relevant to successful interface selection is present in the model.

```
|
+-[Service-Group] <G-Key>, <Name> (O) <Set>
|           +-[Extensible: FALSE]
|           +-[Role] <U-Key>
|           +-[Protocol] <Set>
|           +-[Feature] <Set> (O)
|           +-[Service-Group-Configuration] <Set> (O)
|           +-[DPN-Key] <Set>
|           |           +-[Referenced-Interface] <Set>
|           |           |       +-[Interface-Key] <L-Key>
|           |           |       +-[Peer-Service-Group-Key] <Set> (O)
```

Figure 9: Service Group

Each Service-Group element contains the following information:

Service-Group-Key: A unique ID of the Service-Group.

Service-Group-Name: A human-readable display string.

Role: The role (MAG, LMA, etc.) of the device hosting the interfaces of the DPN Group.

Protocol-Set: The set of protocols supported by this interface (e.g., PMIP, S5-GTP, S5-PMIP etc.). The protocol MAY be only its name, e.g. 'gtp', but many protocols implement specific message sets, e.g. s5-pmip, s8-pmip. When the Service-Group supports specific protocol message sub-subsets the Protocol value MUST include this information.

Feature-Set: An optional set of static features which further determine the suitability of the interface to the desired operation.

Service-Group-Configuration-Set: An optional set of configurations that further determine the suitability of an interface for the specific request. For example: SequenceNumber=ON/OFF.

DPN-Key-Set: A key used to identify the DPN.

Referenced-Interface-Set: The DPN Interfaces and peer Service-Groups associated with them. Each entry contains

Interface-Key: A key that is used together with the DPN-Key, to create a key that is refers to a specific DPN interface definition.

Peer-Service-Group-Key: Enables location of the peer Service-Group for this Interface.

4.9.3. Domain Information Model

A Domain-Set represents a group of heterogeneous Topology resources typically sharing a common administrative authority. Other models, outside of the scope of this specification, provide the details for the Domain.

```
|
+-[Domain] <G-Key>, <Name> (O) <Set>
|       +-[Domain-Policy-Configuration] (O) <Set>
|
```

Figure 10: Domain Information Model

Each Domain entry contains the following information:

Domain-Key: Identifies and enables reference to the Domain.

Domain-Name: A human-readable display string naming the Domain.

4.9.4. DPN Information Model

A DPN-Set contains some or all of the DPNs in the Tenant's network. Some of the DPNs in the Set may be identical in functionality and only differ by their Key.

```
|
+-[DPN] <G-Key>, <Name> (O) <Set>
|     +-[Extensible: FALSE]
|     +-[Interface] <L-Key> <Set>
|     |      +-[Role] <U-Key>
|     |      +-[Protocol] <Set>
|     |      +-[Interface-Configuration] <Set> (O)
|     +-[Domain-Key]
|     +-[Service-Group-Key] <Set> (O)
|     +-[DPN-Policy-Configuration] <List> (M)
|     +-[DPN-Resource-Mapping-Reference] (O)
```

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

DPN-Name: A human-readable display string.

Domain-Key: A Key providing access to the Domain information about the Domain in which the DPN resides.

Interface-Set: The Interface-Set references all interfaces (through which data packets are received and transmitted) available on the DPN. Each Interface makes use of attribute values that are specific to that interface, for example, the MTU size. These do not affect the DPN selection of active or enabled interfaces. Interfaces contain the following information:

Role: The role (MAG, LMA, PGW, AMF, etc.) of the DPN.

Protocol (Set): The set of protocols supported by this interface (e.g., PMIP, S5-GTP, S5-PMIP etc.). The protocol MAY implement specific message sets, e.g. s5-pmip, s8-pmip. When a protocol implements such message sub-subsets the Protocol value MUST include this information.

Interface-Configuration-Set: Configurable settings that further determine the suitability of an interface for the specific request. For example: SequenceNumber=ON/OFF.

Service-Group-Set: The Service-Group-Set references all of the Service-Groups which have been configured using Interfaces hosted on this DPN. The purpose of a Service-Group is not to describe each interface of each DPN, but rather to indicate interface types for use during the DPN selection process, when a DPN with specific interface capabilities is required.

DPN-Policy-Configuration: A list of Policies that have been configured on this DPN. Some may have values for all attributes, and some may require further configuration. Each Policy-Configuration has a key to enable reference to its Policy-Template. Each Policy-Configuration also has been configured to supply missing and non-default values to the desired Attributes defined within the Policy-Template.

DPN-Resource-Mapping-Reference (O): A reference to the underlying implementation, e.g. physical node, software module, etc. that supports this DPN. Further specification of this attribute is out of scope for this document.

4.9.5. Policy Information Model

The Policy Information Model defines and identifies Rules for enforcement at DPNs. A Policy is basically a set of Rules that are to be applied to each incoming or outgoing packet at a DPN interface. Rules comprise Descriptors and a set of Actions. The Descriptors,

when evaluated, determine whether or not a set of Actions will be performed on the packet. The Policy structure is independent of a policy context.

In addition to the Policy structure, the Information Model (per Section 4.9.6) defines Mobility-Context. Each Mobility-Context may be configured with appropriate Attribute values, for example depending on the identity of a mobile node.

Traffic descriptions are defined in Descriptors, and treatments are defined separately in Actions. A Rule-Set binds Descriptors and associated Actions by reference, using Descriptor-Key and Action-Key. A Rule-Set is bound to a policy in the Policy-Set (using Policy-Key), and the Policy references the Rule definitions (using Rule-Key).

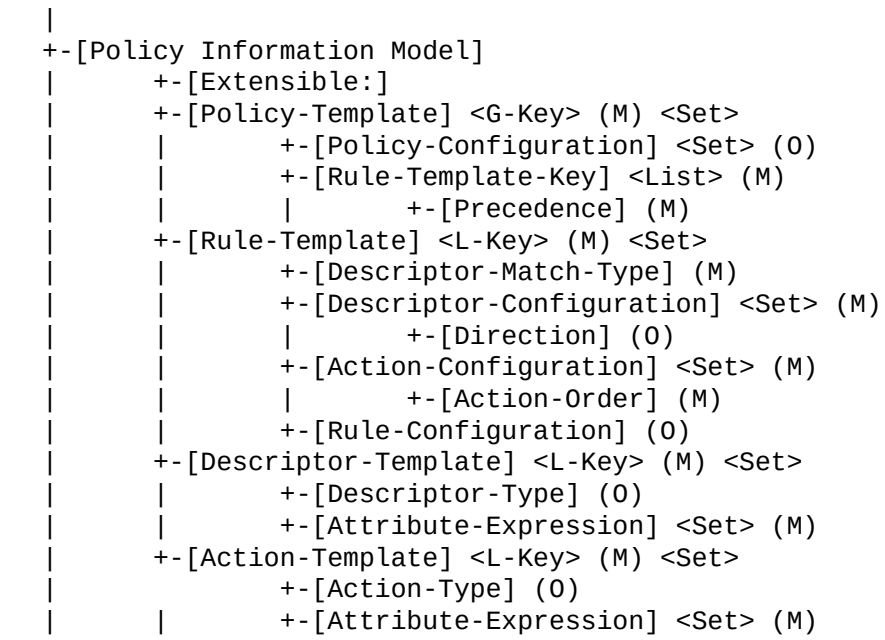


Figure 12: Policy Information Model

The Policy structure defines Policy-Set, Rule-Set, Descriptor-Set, and Action-Set, as follows:

Policy-Template: <Set> A set of Policy structures, indexed by Policy-Key, each of which is determined by a list of Rules referenced by their Rule-Key. Each Policy structure contains the following:

Policy-Key: Identifies and enables reference to this Policy definition.

Rule-Template-Key: Enables reference to a Rule template definition.

Rule-Precedence: For each Rule identified by a Rule-Template-Key in the Policy, specifies the order in which that Rule must be applied. The lower the numerical value of Precedence, the higher the rule precedence. Rules with equal precedence MAY be executed in parallel if supported by the DPN. If this value is absent, the rules SHOULD be applied in the order in which they appear in the Policy.

Rule-Template-Set: A set of Rule Template definitions indexed by Rule-Key. Each Rule is defined by a list of Descriptors (located by Descriptor-Key) and a list of Actions (located by Action-Key) as follows:

Rule-Template-Key: Identifies and enables reference to this Rule definition.

Descriptor-Match-Type Indicates whether the evaluation of the Rule proceeds by using conditional-AND, or conditional-OR, on the list of Descriptors.

Descriptor-Configuration: References a Descriptor template definition, along with an expression which names the Attributes for this instantiation from the Descriptor-Template and also specifies whether each Attribute of the Descriptor has a default value or a statically configured value, according to the syntax specified in Section 4.2.

Direction: Indicates if a rule applies to uplink traffic, to downlink traffic, or to both uplink and downlink traffic. Applying a rule to both uplink and downlink traffic, in case of symmetric rules, eliminates the requirement for a separate entry for each direction. When not present, the direction is implied by the Descriptor’s values.

Action-Configuration: References an Action Template definition, along with an expression which names the Attributes for this instantiation from the Action-Template and also specifies whether each Attribute of the Action has a default value or a statically configured value, according to the syntax specified in Section 4.2.

Action-Order: Defines the order in which actions are executed when the associated traffic descriptor selects the packet.

Descriptor-Template-Set: A set of traffic Descriptor Templates, each of which can be evaluated on the incoming or outgoing packet, returning a TRUE or FALSE value, defined as follows:

Descriptor-Template-Key: Identifies and enables reference to this descriptor template definition.

Attribute-Expression: An expression which defines an Attribute in the Descriptor-Template and also specifies whether the Template also defines a default value or a statically configured value for the Attribute of the Descriptor has, according to the syntax specified in Section 4.2.

Descriptor-Type: Identifies the type of descriptor, e.g. an IPv6 traffic selector per [RFC6088].

Action-Template-Set: A set of Action Templates defined as follows:

Action-Template-Key: Identifies and enables reference to this action template definition.

Attribute-Expression: An expression which defines an Attribute in the Action-Template and also specifies whether the Template also defines a default value or a statically configured value for the Attribute of the Action has, according to the syntax specified in Section 4.2.

Action-Type: Identifies the type of an action for unambiguous interpretation of an Action-Value entry.

4.9.6. Mobility-Context Information Model

The Mobility-Context structure holds entries associated with a mobile node and its mobility sessions (flows). It is created on a DPN during the mobile node’s registration to manage the mobile node’s flows. Flow information is added or deleted from the Mobility-Context as needed to support new flows or to deallocate resources for flows that are deactivated. Descriptors are used to characterize the nature and resource requirement for each flow.

Termination of a Mobility-Context implies termination of all flows represented in the Mobility-Context, e.g. after deregistration of a mobile node. If any Child-Contexts are defined, they are also terminated.

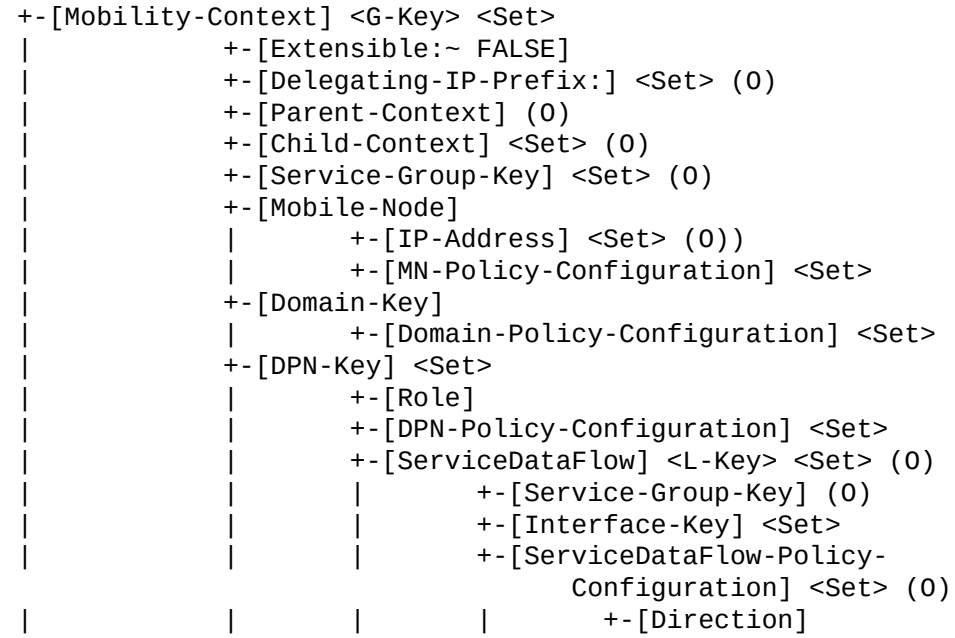


Figure 13: Mobility-Context Information Model

The Mobility-Context Substructure holds the following entries:

Mobility-Context-Key: Identifies a Mobility-Context

Delegating-IP-Prefix-Set: Delegated IP Prefixes assigned to the Mobility-Context

Parent-Context: If present, a Mobility Context from which the Attributes and Attribute Values of this Mobility Context are inherited.

Child-Context-Set: A set of Mobility Contexts which inherit the Attributes and Attribute Values of this Mobility Context.

Service-Group-Key: Service-Group(s) used during DPN assignment and re-assignment.

Mobile-Node: Attributes specific to the Mobile Node. It contains the following

IP-Address-Set IP addresses assigned to the Mobile Node.

MN-Policy-Configuration-Set For each MN-Policy in the set, a key and relevant information for the Policy Attributes.

Domain-Key: Enables access to a Domain instance.

Domain-Policy-Configuration-Set: For each Domain-Policy in the set, a key and relevant information for the Policy Attributes.

DPN-Key-Set: Enables access to a DPN instance assigned to a specific role, i.e. this is a Set that uses DPN-Key and Role as a compound key to access specific set instances.

Role: Role this DPN fulfills in the Mobility-Context.

DPN-Policy-Configuration-Set: For each DPN-Policy in the set, a key and relevant information for the Policy Attributes.

ServiceDataFlow-Key-Set: Characterizes a traffic flow that has been configured (and provided resources) on the DPN to support data-plane traffic to and from the mobile device.

Service-Group-Key: Enables access to a Service-Group instance.

Interface-Key-Set: Assigns the selected interface of the DPN.

ServiceDataFlow-Policy-Configuration-Set: For each Policy in the set, a key and relevant information for the Policy Attributes.

Direction: Indicates if the reference Policy applies to uplink or downlink traffic, or to both, uplink- and downlink traffic. Applying a rule to both, uplink- and downlink traffic, in case of symmetric rules, allows omitting a separate entry for each direction. When not present the value is assumed to apply to both directions.

4.9.7. Monitor Information Model

Monitors provide a mechanism to produce reports when events occur. A Monitor will have a target that specifies what is to be watched.

The attribute/entity to be monitored places certain constraints on the configuration that can be specified. For example, a Monitor using a Threshold configuration cannot be applied to a Mobility-Context, because it does not have a threshold. Such a monitor configuration could be applied to a numeric threshold property of a Context.

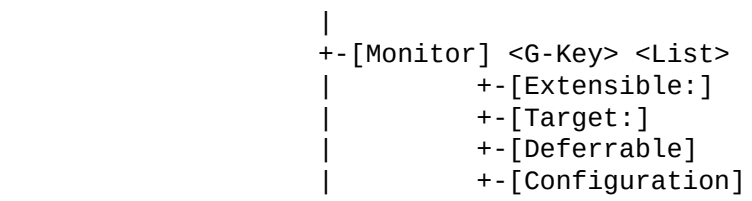


Figure 14: Monitor Substructure

Monitor-Key: Identifies the Monitor.

Target: Description of what is to be monitored. This can be a Service Data Flow, a Policy installed upon a DPN, values of a Mobility-Context, etc. The target name is the absolute information model path (separated by '//') to the attribute / entity to be monitored.

Deferrable: Indicates that a monitoring report can be delayed up to a defined maximum delay, set in the Agent, for possible bundling with other reports.

Configuration: Determined by the Monitor subtype. The monitor report is specified by the Configuration. Four report types are defined:

* "Periodic" reporting specifies an interval by which a notification is sent.

* "Event-List" reporting specifies a list of event types that, if they occur and are related to the monitored attribute, will result in sending a notification.

* "Scheduled" reporting specifies the time (in seconds since Jan 1, 1970) when a notification for the monitor should be sent. Once this Monitor's notification is completed the Monitor is automatically de-registered.

* "Threshold" reporting specifies one or both of a low and high threshold. When these values are crossed a corresponding notification is sent.

5. Protocol

5.1. Protocol Messages and Semantics

Four Client to Agent messages are supported.

Message	Description
Configure	A Configure message includes multiple edits to one or more information model entities. Edits are executed according to their Edit-Id in ascending order. The global status of the operation and the status of individual edits are returned. Partial failures, i.e. individual edit failures, are allowed.
Register-Monitors	Register monitors at an Agent. The message includes the Monitor information as specified in Section 4.9.7.
Deregister-Monitors	Deregister monitors from an Agent. An optional boolean, Send-Data, indicates if a successful deregistration triggers a Notify with final data from the Agent for the corresponding Monitor.
Probe	Probe the status of registered monitors. This triggers a Notify with current data from the Agent for the corresponding Monitors.

Table 1: Client to Agent Messages

Each message contains a header with the following information:

Client Identifier: An Identifier used by the Agent to associate specific configuration characteristics, e.g. options used by the Client when communicating with the Agent, the association of the Client and tenant in the information model as well as tracking operations and notifications.

Delay: An optional time (in ms) to delay the execution of the operation on the DPN once it is received by the Agent.

Operation Identifier: A unique identifier created by the Client to correlate responses and notifications

An Agent will respond with an ERROR, indicating one or more Errors have occurred, or an OK.

For Configure messages, an OK status for an edit MAY include subsequent edits in the response that were required to properly execute the edit. It MAY also indicate that the final status and any final edits required to fulfill the request will be sent via a

Configure Result Notification from the Agent to the Client, see Section 5.1.1.4.2.

If errors occur, they MUST be returned as a list in responses and each Error contains the following information:

Error-type: The specific error type. Values are TRANSPORT (0), RPC (1), PROTOCOL(2) or APPLICATION (3).

Error-Tag: An error tag.

Error-App-Tag: Application specific error tag.

Error-Message: A message describing the error.

Error-Info: Any data required for the response.

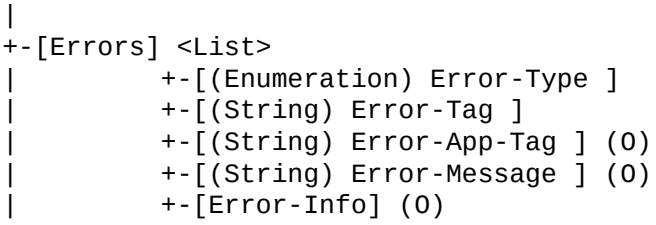


Figure 15: Error Information Model

Two Agent to Client notifications are supported.

Message	Description
Configure-Result-Notification	An asynchronous notification from Agent to Client based upon a previous Configure request.
Notify	An asynchronous notification from Agent to Client based upon a registered Monitor's configuration, a Monitor deregistration or Probe.

Table 2: Agent to Client Messages (notifications)

5.1.1. Configure Message

The Configure message follows edit formats proposed by [RFC8072] with more fields in each edit, an extra operation (clone) and a different response format.

5.1.1.1. Edit Operation Types

Operation	Description
create	Creates a new data resource or Entity. If the resource exists an error is returned.
delete	Deletes a resource. If it does not exist an error is returned.
insert	Inserts data in a list or user ordered list.
merge	Merges the edit value with the target data resource; the resource is created if it does not exist.
move	Moves the target data resource.
replace	Replace the target data resource with the edit value.
remove	Removes a data resource if it already exists.
clone	Clones a data resource and places the copy at the new location. If the resource does not exist an error is returned.

Table 3: Configure Edit Operations

5.1.1.2. Edit Operation

Each Configure includes one or more edits. These edits include the following information:

Edit-Id: Uniquely specifies the identifier of the edit within the operation.

Edit-Type: Specifies the type of operation (see Section 5.1.1.1).

Command-Set: The Command-Set is a technology-specific bitset that allows for a single entity to be sent in an edit with multiple requested, technology specific sub-transactions to be completed. It can also provide clarity for a request. For example, a Mobility-Context could have the Home Network Prefix absent but it is unclear if the Client would like the address to be assigned by the Agent or if this is an error. Rather than creating a specific command for assigning the IP, a bit position in a Command-Set can be used to indicate Agent based IP assignment requests.

Reference-Scope: If supported, specifies the Reference Scope (see Section 5.1.1.3)

Target: Specifies the Target node (Data node path or FPC Identity) for the edit operation. This MAY be a resource, e.g. Mobility-Context, Descriptor-Template, etc., or a data node within a resource as specified by its path.

Point: The absolute URL path for the data node that is being used as the insertion point, clone point or move point for the target of this 'edit' entry.

Where: Identifies where a data resource will be inserted, cloned to or moved. Only allowed these for lists and lists of data nodes that are 'ordered-by user'. The values are 'before', 'after', 'first', 'last' (default value).

Value The value used for this edit operation. In this message it MUST NOT be a MONITOR entity.

```
          |
          +-[Configure]
          |         +-[Client-Id:]
          |         +-[(Unsigned 32) Execution-Delay]
          |         +-[Operation-Id:]
          |         +-[Edit:] <List>
          |         |    +-[Edit-Id:] <L-Key>
          |         |    +-[(Enumeration) Edit-Type:]
          |         |    +-[(BitSet) Command-Set]
          |         |    +-[(Enumeration) Reference-Scope]
          |         |    +-[Target:]
          |         |    +-[Point]
          |         |    +-[(Enumeration) Where]
          |         |    +-[Value]
```

Figure 16: Configure Request

Edits sent to the Agent provided in an operation SHOULD be sent in the following order to avoid errors:

1. Action Templates

2. Descriptor Templates

3. Rule Templates

4. Policy Templates

5. DPN Templates

6. Mobility Contexts

5.1.1.3. Reference Scope

The Reference Scope is an optional feature that provides the scope of references used in a configuration command. These scopes are defined as:

- none - All edits have no references to other entities or within edits.
- edit - All references are contained within each edit body (intra-edit/intra-operation)
- operation - All references exist in the operation (inter-edit/ intra-operation).
- storage - One or more references exist outside of the operation. A lookup to cache / storage is required.
- unknown - the location of the references are unknown. This is treated as a 'storage' type.

An Agent that only accepts 'edit' or 'operation' reference scope messages is referred to as 'stateless' as it has no direct memory of references outside messages themselves. This permits low memory footprint Agents/DPNs. Even when an Agent supports all message types an 'edit' or 'operation' scoped message can be processed quickly by the Agent/DPN as it does not require storage access.

Figure 17 shows an example containment hierarchy provided for all caches.

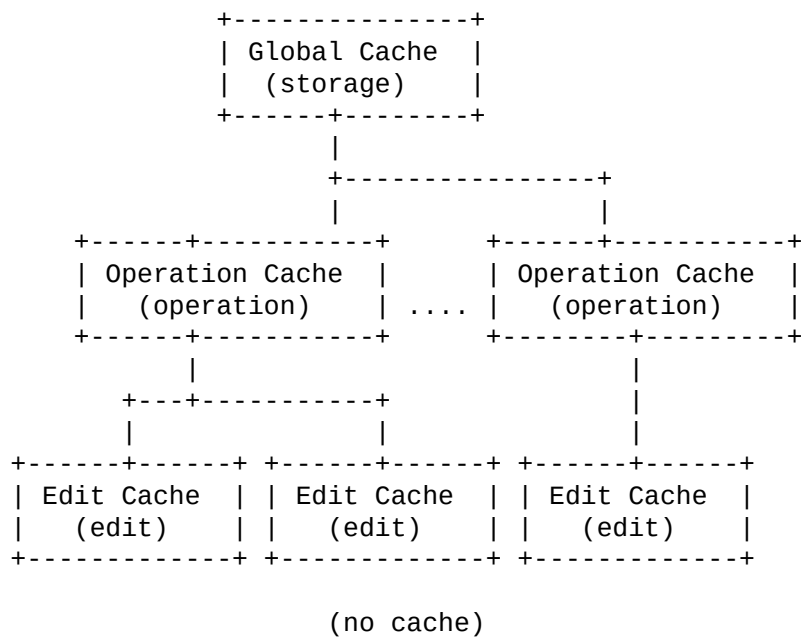


Figure 17: Example Hierarchical Cache

5.1.1.4. Operation Response

5.1.1.4.1. Immediate Response

The Response MUST include the following:

Operation Identifier of the corresponding request.

Global Status for the operation (see Table 1).

A list of Edit results (described below).

An edit response, Edit-Status, is comprised of the following:

Edit-Id: Edit Identifier.

Edit-Status: OK.

When the Edit-Status is OK the following values MAY be present

Notify-Follows - A boolean indicator that the edit has been accepted by the Agent but further processing is required. A Configure-Result-Notification will be sent once the processing has succeeded or failed.

Subsequent-Edits-List: This is a list of Edits that were required to fulfill the request. It follows the edit request semantics (see Section 5.1.1.2).

Errors-List: When the Edit-Status is ERROR the following values are present. See Table 1 for details.

The response will minimally contain an Edit-Status implying 'OK' or a list of errors.

```
|
+-[Operation-Id:]
+-[Result-Status:]
+-[Errors] <List>
|         +-[(Enumeration) Error-Type:]
|         +-[(String) Error-Tag:]
|         +-[(String) Error-App-Tag]
|         +-[(String) Error-Message]
|         +-[Error-Info]
+-[Edit-Status]
|         +-[Edit-Id:]
|         +-[Edit-Status: ~ OK]
|         +-[Notify-Follows]
|         +-[Subsequent-Edits] <List>
|         |    +-[Edit-Id:] <L-Key>
|         |    +-[(Enumeration) Edit-Type:]
|         |    +-[Target:]
|         |    +-[Point]
|         |    +-[(Enumeration) Where]
|         |    +-[Value]
|         +-[Errors] <List>
|         |         +-[(Enumeration) Error-Type:]
|         |         +-[(String) Error-Tag:]
|         |         +-[(String) Error-App-Tag]
|         |         +-[(String) Error-Message]
|         |         +-[Error-Info]
|
```

Figure 18: Configure Operation Response

5.1.1.4.2. Asynchronous Notification

A Configure-Result-Notification occurs after the Agent has completed processing related to a Configure request. It is an asynchronous communication from the Agent to the Client.

It is identical to the immediate response with the exception that the Notify-Follows, if present, MUST be false. As this value is unnecessary it SHOULD be omitted.

5.1.1.5. Reserved Identities

Several identities are reserved in the Policy Information Model and Mobility-Context to facilitate specific uses cases.

Agents and tenants express their support for descriptors and actions using the following Key patterns

supported-<descriptor template name> indicates a support for the descriptor template as defined in its original specification. For example "supported-rfc5777classifier" is a Descriptor Template that conforms to the rfc5777-classifier (Figure 31) as defined in this document.

supported-<action template name> indicates a support for the action template as defined in its original specification.

"base-rule" is comprised of all base descriptors using an 'or' Descriptor-Match-Type and all Actions in no specific order.

"base-template" is comprised of the base rule.

"base-template" can be used to determine supported Action and Descriptor Templates. It can also be used to support an open template where any specific Descriptors and Actions can be applied, however, depending upon the Order of Actions it is likely to produce undesirable results.

One use case is supported via reservation of specific DPN-Keys:

Requested policies are those that the Client would like to be assigned to a DPN within a Mobility-Context. The naming convention is similar to those used for DPN Assignment via an Agent.

"Requested" is a Key that represents requested policies which have not been assigned to a specific DPN. No Role is assigned to the DPN.

"Requested-<Role>" represents requested policies that have not been assigned to a DPN and can only be assigned to DPNs that fulfill the specified Role.

It is possible to have policies in the "Requested" DPN that do not appear in other entries which reflects the inability to successfully assign the policy.

5.1.2. Monitor Messages

An Agent may reject a registration if it or the DPN has insufficient resources.

An Agent or DPN MAY temporarily suspend monitoring if insufficient resources exist. In such a case the Agent MUST notify the Client.

When a monitor has a reporting configuration of SCHEDULED it is automatically de-registered after the last Notify occurs.

If a SCHEDULED or PERIODIC configuration is provided during registration with the time related value (time or period respectively) of 0 a Notify is sent and the monitor is immediately de-registered. This method should, when a Monitor has not been installed, result in an immediate Notify sufficient for the Client's needs and lets the Agent realize the Client has no further need for the monitor to be registered.

Probe messages are used by a Client to retrieve information about a previously installed monitor. The Probe message SHOULD identify one or more monitors by means of including the associated monitor identifier. An Agent receiving a Probe message sends the requested information in a single or multiple Notify messages.

If the Monitor configuration associated with a Notify can be deferred, then the Notify MAY be bundled with other messages back to the Agent even if this results in a delay of the Notify.

The Monitor messages use the following data:

Monitor-Key: Monitor Key.

Monitor: A Monitor configuration (see Section 4.9.7).

Send-Data: An indicator that specifies that the final value MUST be sent as a notification from the Agent.

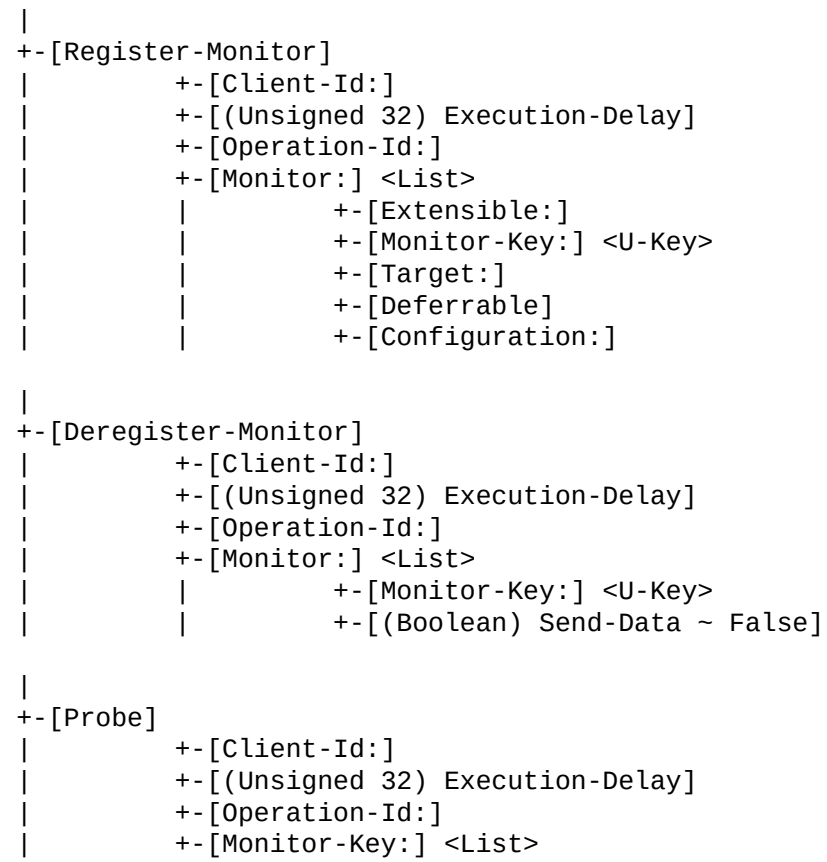


Figure 19: Monitor Messages

5.1.2.1. Asynchronous Notification

A Monitor Report can be sent as part of de-registration, a trigger based upon a Monitor Configuration or a Probe. A Report is comprised of the Monitor Key the report applies to, the Trigger for the report, a timestamp of when the report's associated event occurs and data, Report-Value, that is specific to the monitored value's type.

Triggers include but are not limited to

- Subscribed Event occurred
- Low Threshold Crossed
- High Threshold Crossed
- Periodic Report

- Scheduled Report
- Probe
- Deregistration Final Value
- Monitoring Suspended
- Monitoring Resumed
- DPN Available
- DPN Unavailable

Multiple Reports are sent in a Notify message. Each Notify is comprised of unique Notification Identifier from the Agent and timestamp indicating when the notification was created.

```
        |
        +-[ Notify ]
        |         +-[(Unsigned 32) Notification-Identifier:]
        |         +-[Timestamp:]
        |         +-[Report:] <List>
        |         |    +-[Trigger:]
        |         |    +-[Monitor-Key:]
        |         |    +-[Report-Value]
```

Figure 20: Monitor Messages

5.2. Protocol Operation

Please note that JSON is used to represent the information in Figures in this section but any over the wire representation that accurately reflects the information model MAY be used.

5.2.1. DPN Selection

In order to assign a DPN to a Mobility Context, the Client or Agent requires topology information. The Service-Group provides information, e.g. function, role, protocol, features and configuration, to determine suitable DPN interfaces.

Consider a Client attempting to select DPN interfaces that are served by a single Agent. In this example interfaces are present with different protocols, settings and features as shown in the following figure.

```
 "topology-information-model" : {
```

```
     "dpn" : [ {
      "dpn-key" : "dpn1",
      "interface" : [ {
        "interface-key" : "ifc1",
        "role" : "lma",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionA" : "OFF" ]
        } ]
      },{
        "interface-key" : "ifc2",
        "role" : "lma",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionC" : "OFF"  ]
         } ]
       },{
        "interface-key" : "ifc2-b",
        "role" : "mag",
        "protocol" : [ "pmip" ]
        } ] },
      {
      "dpn-key" : "dpn2",
      "interface" : [ {
        "interface-key" : "ifc1",
        "role" : "mag",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
          "index" : 0,
          "settings" : [ "optionA" : "OFF", "optionB" : "ON" ]
        } ]
      } ] }
     ],
     ...
     },
     "service-group" : [
     { "service-group-key" : "group1",
       "service-group-name" : "Anchors-OptionA-OFF",
       "role-key" : "lma",
       "protocol" : [ "pmip" ],
       "service-group-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionA" : "OFF" ]
       } ],
       "dpn" : [
       { "dpn-key" : "dpn1",
```

```
        "referenced-interface" : [ { "interface-key" : "ifc1" } ] }
      ]
    },{ "service-group-key" : "group2",
      "service-group-name" : "Anchors",
      "role-role" : "lma",
      "protocol" : [ "pmip" ],
      "dpn" : [
      { "dpn-key" : "dpn1",
        "referenced-interface" : [ { "interface-key" : "ifc2" } ] }
      ]
     },{ "service-group-key" : "group3",
      "service-group-name" : "MAGs",
      "role-role" : "mag",
      "protocol" : [ "pmip" ],
      "dpn" : [
      { "dpn-key" : "dpn2",
        "referenced-interface" : [ { "interface-key" : "ifc1" } ] },
      { "dpn-key" : "dpn1",
        "referenced-interface" : [ { "interface-key" : "ifc2-b" } ] }
      ]
     }
  ]
```

NOTE - A Setting is, in this example, a list of string attributes in a Configuration.

Figure 21: Monitor Messages

Two DPNs are present. The first, dpn1, has 3 interfaces. Two support the LMA role and both have settings. The third supports the MAG function. The second DPN, dpn2, provides a single interface with the MAG function.

Three ServiceGroups are presented. The first provides the PMIP protocol and LMA role. It also has a setting, OptionA, that is OFF and only contains ifc1 from dpn1.

The second group is comprised of interfaces that support the PMIP protocol and LMA function. It only contains ifc2 from dpn1. An interface that has setting(s) or feature(s) that must appear in a ServiceGroup SHOULD NOT appear in ServiceGroups that do not have those setting(s) or feature(s) present. Thus, ifc1 of dpn1 should not be present in this second Service-Group.

A third group is comprised of interfaces that support the MAG function of the LMA protocol. It contains the MAG interfaces form both dpn1 and dpn2.

Given the task to find a LMA that supports the PMIP protocol the Client can determine that dpn1 is its only option and, depending on its requirement of OptionA, can appropriately determine which interface to select.

5.2.2. Policy Creation and Installation

A Policy must be installed upon an Agent in order to install policies on the selected dpn(s). This requires construction of the Action(s), Descriptor(s) and Rule(s) used by the Policy.

The CONFIGURE message permits editing all information elements except monitors. The following figure shows use of a CONFIGURE message to install policy information on the Agent.

Figure 22: Example Policy Installation (focus on FPC reference point)

In this example a Descriptor "all-traffic" Template and an Action, "drop", Template are both empty Templates. The "deny-all" Rule Template is comprised of the action and descriptor. The Rule is included in "policy1". The policy's status is "Configured" as it is a complete policy ready for immediate use. The policy could be set as "Active" if the Client intends to use it upon immediate installation in a DPN.

Installation of the policy on dpn1 is shown in the following Figure. The Policy-Status is set to "Active" to make it immediately usable. Leaving the status as Configured would permit its installation on the DPN without an ability to use it in a Mobility Context. Such a use case is often referred to as policy pre-configuration.

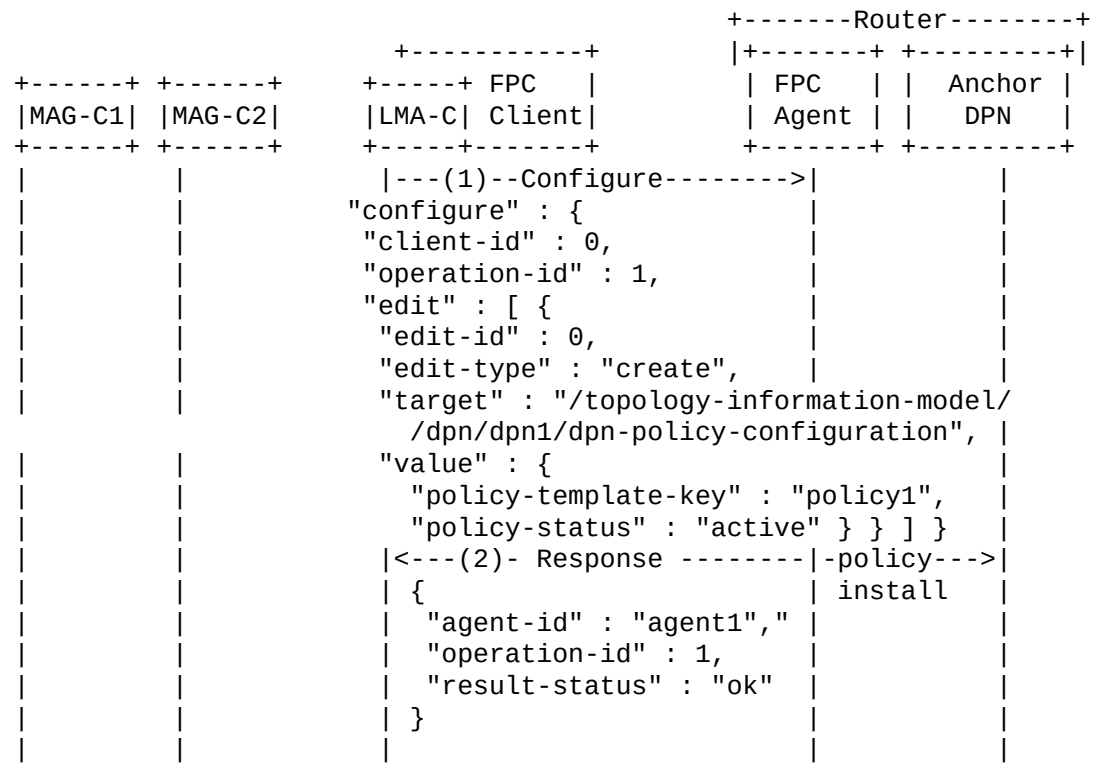


Figure 23: Example Policy Installation (focus on FPC reference point)

This message uses an edit type of "create" to add the policy template directly to the installed DPN policy set.

5.2.3. Simple RPC Operation

A Client and Agent MUST identify themselves using the Client Identifier and Agent Identifier respectively to ensure that, for all transactions, a recipient of a FPC message can unambiguously identify the sender of the FPC message.

A Client MAY direct the Agent to enforce a rule in a particular DPN by including a DPN Key value in a Mobility Context. Otherwise the Agent selects a suitable DPN to enforce one or more portions of a Mobility Context and notifies the Client about the selected DPN(s) using DPN Identifier(s).

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all edit status as well as subsequent edits, which indicates the result of processing the message, as part of the Configure response. In case the processing of the message results in a failure, the Agent sets the global

status, Error-Type and Error-Tag accordingly and MAY clear the entity, e.g. Mobility-Context, which caused the failure, in the response.

If based upon Agent configuration or the processing of the request possibly taking a significant amount of time the Agent MAY respond with a Notify-Follows indication with optional Subsequent-Edit(s) containing the partially completed entity modifications. When a Notify-Follows indication is sent in a response, the Agent will, upon completion or failure of the operation, respond with an asynchronous Configuration-Result-Notification to the Client.

A Client MAY add a property to a Mobility-Context without providing all required details of the attribute's value. In such case the Agent SHOULD determine the missing details and provide the completed property description, via Subsequent-Edit(s), back to the Client. If the processing will take too long or based upon Agent configuration, the Agent MAY respond with an OK for the Edit that indicates a Notify-Follows and also includes Subsequent-Edit(s) containing the partially completed entity edits.

In case the Agent cannot determine the missing value of an attribute's value per the Client's request, it leaves the attribute's value cleared, sets the Edit Result to Error and provides an Error-Type and Error-Tag. As example, the Control-Plane needs to setup a tunnel configuration in the Data-Plane but has to rely on the Agent to determine the tunnel endpoint which is associated with the DPN that supports the Mobility-Context. The Client adds the tunnel property attribute to the FPC message and clears the value of the attribute (e.g. IP address of the local tunnel endpoint). The Agent determines the tunnel endpoint and includes the completed tunnel property in its response to the Client in a Subsequent-Edit entry.

Figure 24 illustrates an exemplary session life-cycle based on Proxy Mobile IPv6 registration via MAG Control-Plane function 1 (MAG-C1) and handover to MAG Control-Plane function 2 (MAG-C2). Edge DPN1 represents the Proxy CoA after attachment, whereas Edge DPN2 serves as Proxy CoA after handover. As exemplary architecture, the FPC Agent and the network control function are assumed to be co-located with the Anchor-DPN, e.g. a Router.

The Target of the second request uses the Mobility-Context by name. Alternatively, the Target could have included the DPN-Key and Policy-Key to further reduce the amount of information exchanged. Setting the Target's value to the most specific node SHOULD be followed whenever practical.

+-------Router--------+


+-----------+ |+-------+ +---------+|
+------+ +------+ +-----+ FPC | | FPC | | Anchor |
|MAG-C1| |MAG-C2| |LMA-C| Client| | Agent | | DPN1 |
+------+ +------+ +-----+-------+ +-------+ +---------+
[MN attach] | | | |
-------------PBU----->		
		---(1)--Configure-------->	
	"configure" : {	
	"client-id" : 0,	
	"operation-id" : 3,	
	"edit" : [	
	"edit-id" : 0,	
	"edit-type" : "create",	
	"target" : "/mobility-context",
	"value" : {
	"mobility-context-key" : "ctxt1",
	"delegating-ip-prefix" : [<HNP>],
	"dpn" : [{
	"dpn-key" : "DPN1",
	"role" : "lma",
	"service-data-flow" : [{
	"identifier" : 0,
	"interface" : ["interface-key" : "ifc1"],
	"service-data-flow-policy-configuration" :[
	{"policy-template-key" :
"dl-tunnel-with-qos",
	"policy-status" : "active",
	"policy-configuration" : [
	{"index" : 0,
"qos-template" : <QOS Settings...>},
	{"index" : 1,
"tunnel" : <DL tunnel info...>},
	{"policy-template-key" : "ul-tunnel",
"policy-status" : "active",
	"policy-configuration" : [
	{"index" : 1,
"tunnel" : <UL tunnel info...>}] }]
	}] }] }] }] }	
		|--tun1 up->
		|--tc qos-->
		<---(2)- Response --------	-route add>
		{	
		"agent-id" : "agent1","	
		"operation-id" : 3,	
		"result-status" : "ok",	
		}	


|<------------PBA------|
| +----+ |
| |Edge| |
| |DPN1| |
| +----+ |
[MN handover]
|---PBU ---->|
|--(3)- CONFIG(MODIFY)---->|
"configure" : {
"client-id" : 0,
"operation-id" : 4,
"edit" : [
"edit-id" : 0,
"edit-type" : "merge",
"target" : "/mobility-context/ctxt1",
"value" : {
"mobility-context-key" : "ctxt1",
"dpn" : "[{
"dpn-key" : "DPN1",
"service-data-flow" : [{
"identifier" : 0,
"service-data-flow-policy-configuration":[
{"policy-template-key" :
"dl-tunnel-with-qos",
"policy-configuration" : [
{"index" : 1,
"tunnel" : <NEW tunnel info...>}]}]
}] }] }] }
|-tun1 mod->|
|<--PBA------|
|<---(4)- OK --------------|
| {
| "agent-id" : "agent1"," |
| "operation-id" : 4, |
| "result-status" : "ok", |
| }
| +----+ |
| |Edge| |
| |DPN2| |
| +----+ |

Figure 24: Single Agent with Handover (focus on FPC reference point)

After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node's (MN) traffic. The LMA-C adds a new logical Mobility-Context to the DPN to treat the MN's traffic (1) and includes a Mobility-Context-Key (ctxt1) in the Configure command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

The LMA-C adds policy template properties during the creation of the new Mobility-Context. One policy, "dl-tunnel-with-qos", is an example template that permits tunnel forwarding of traffic destined to the MN's HNP, i.e. downlink traffic, with optional QoS parameters. Another policy, "ul-tunnel", provides a simple uplink anchor termination template where uplink tunnel information is provided.

The downlink tunnel information specifies the destination endpoint (Edge DPN1).

Upon reception of the Mobility-Context, the FPC Agent utilizes local configuration commands to create the tunnel (tun1) as well as the traffic control (tc) to enable QoS differentiation. After configuration has been completed, the Agent applies a new route to forward all traffic destined to the MN's HNP specified as a property in the Mobility-Context and applied the configured tunnel interface (tun1).

During handover, the LMA-C receives an updating PBU from the handover target MAG-C2. The PBU refers to a new Data-Plane node (Edge DPN2) to represent the new tunnel endpoint in the downlink as required. The LMA-C sends a Configure message (3) to the Agent to modify the existing tunnel property of the existing Mobility-Context and to update the downlink tunnel endpoint from Edge DPN1 to Edge DPN2. Upon reception of the Configure message, the Agent applies updated tunnel property to the local configuration and responds to the Client (4).

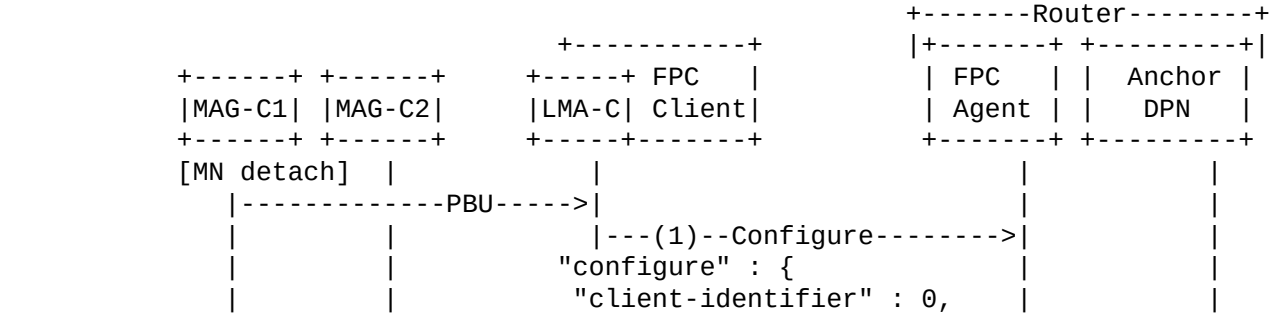


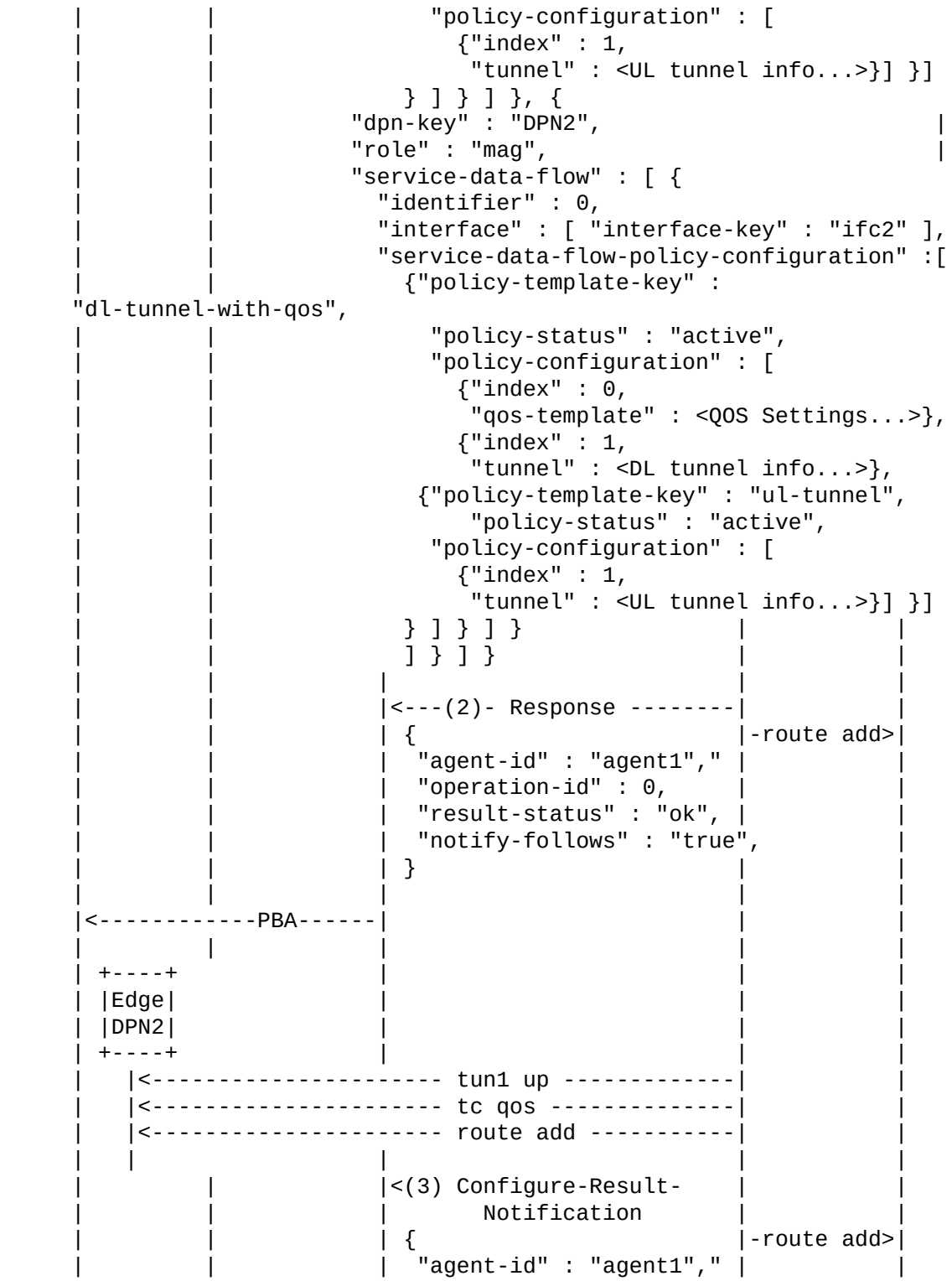
Figure 25: Single Agent with Deletion (focus on FPC reference point)

When a teardown of the session occurs, MAG-C1 will send a PBU with a lifetime value of zero. The LMA-C sends a Configure message (1) to the Agent to modify the existing tunnel property of the existing

Mobility-Context to delete the tunnel information. Upon reception of the Configure message, the Agent removes the tunnel configuration and responds to the Client (2). Per [RFC5213], the PBA is sent back immediately after the PBA is received.

If no valid PBA is received after the expiration of the MinDelayBeforeBCEDelete timer (see [RFC5213]), the LMA-C will send a Configure (3) message with a deletion request for the Context. Upon reception of the message, the Agent deletes the tunnel and route on the DPN and responds to the Client (4).

When a multi-DPN Agent is used the DPN list permits several DPNs to be provisioned in a single message for the single Mobility-Context.


"policy-configuration" : [
{"index" : 1,
"tunnel" : <UL tunnel info...>}] }]
}] }] }, {
"dpn-key" : "DPN2",
"role" : "mag",
"service-data-flow" : [{
"identifier" : 0,
"interface" : ["interface-key" : "ifc2"],
"service-data-flow-policy-configuration" :[
{"policy-template-key" :
"dl-tunnel-with-qos",
"policy-status" : "active",
"policy-configuration" : [
{"index" : 0,
"qos-template" : <QOS Settings...>},
{"index" : 1,
"tunnel" : <DL tunnel info...>},
{"policy-template-key" : "ul-tunnel",
"policy-status" : "active",
"policy-configuration" : [
{"index" : 1,
"tunnel" : <UL tunnel info...>}] }]
}] }] }
] }] }
(2)- Response
{
"agent-id" : "agent1","
"operation-id" : 0,
"result-status" : "ok",
"notify-follows" : "true",
}
-route add>
PBA
Edge
DPN2
tun1 up
tc qos
route add
(3) Configure-Result-
Notification
{
"agent-id" : "agent1","
-route add>

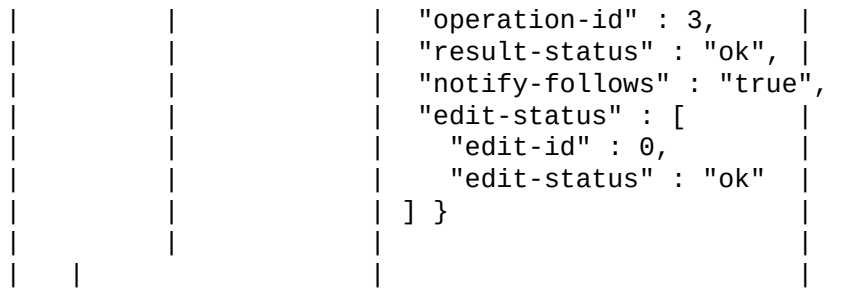


Figure 26: Exemplary Message Sequence for Multi-DPN Agent

Figure 26 shows how the first 2 messages in Figure 24 are supported when a multi-DPN Agent communicates with both Anchor DPN1 and Edge DPN2. In such a case, the FPC Client sends the downlink and uplink for both DPNs in the DPN Reference List of the same Mobility-Context. Message 1 shows the DPN Set with all entries. Each entry identifies the DPN.

The Agent responds with an OK and Notify-Follows indication while it simultaneously provisions both DPNs. Upon successful completion, the Agent responds to the Client with a Configuration-Result-Notification indicating the operation status.

5.2.4. Policy and Mobility on the Agent

A Client may build Policy and Topology using Configure messages.

The Client may add, modify or delete many DPN Policies as DPN Policy Configurations and Mobility-Contexts in a single FPC message. This includes linking Mobility-Contexts to DPN Policies as well as creating the Policy, Rules Actions and Descriptors. As example, a Rule which performs re-writing of an arriving packet’s destination IP address from IP_A to IP_B matching an associated Descriptor, can be enforced in the Data-Plane via an Agent to implicitly consider matching arriving packet’s source IP address against IP_B and re-write the source IP address to IP_A.

Figure 27 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.

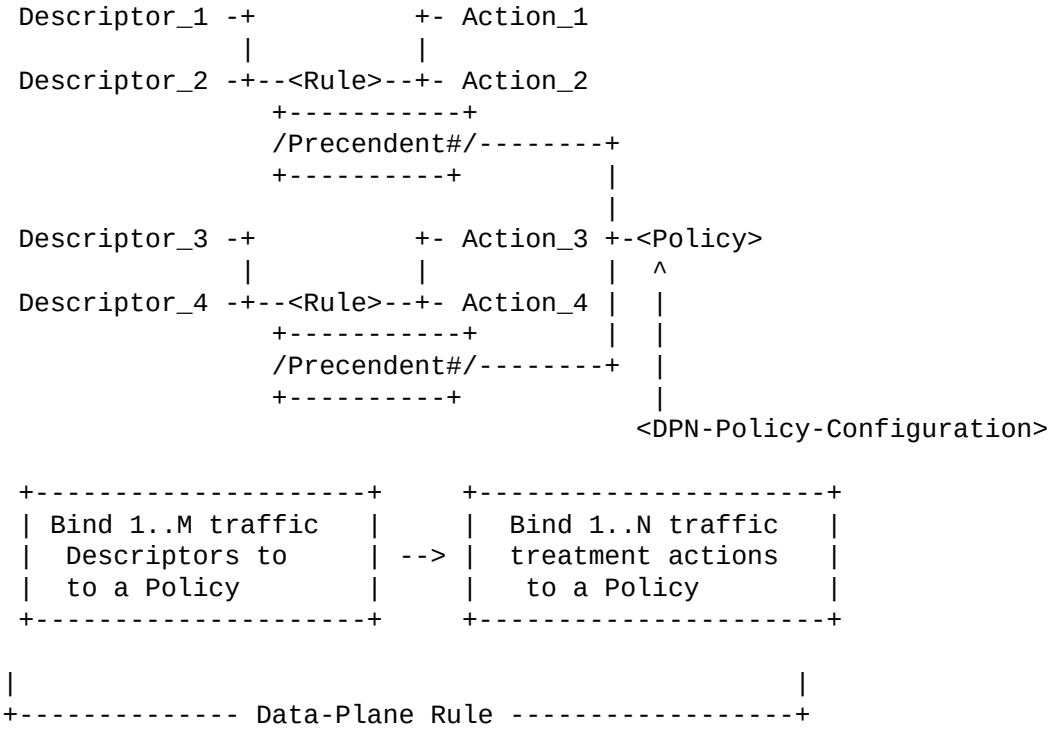


Figure 27: Structure of Configurable Policies

As depicted in Figure 27, the DPN Settings represents the anchor of Rules through the Policy / Rule hierarchy. A Client and Agent use the identifier of the associated Policy to directly access the Rule and perform modifications of traffic Descriptors or Action references. Arriving packets are matched against traffic according to Rule precedence and Descriptors. If a Rule is applicable the packet is treated according to the ordered Action values.

A Client associates a Precedence value for the Rule’s Descriptors, to allow unambiguous traffic matching on the Data-Plane.

Figure 28 illustrates the generic context configuration model as used between a Client and an Agent.

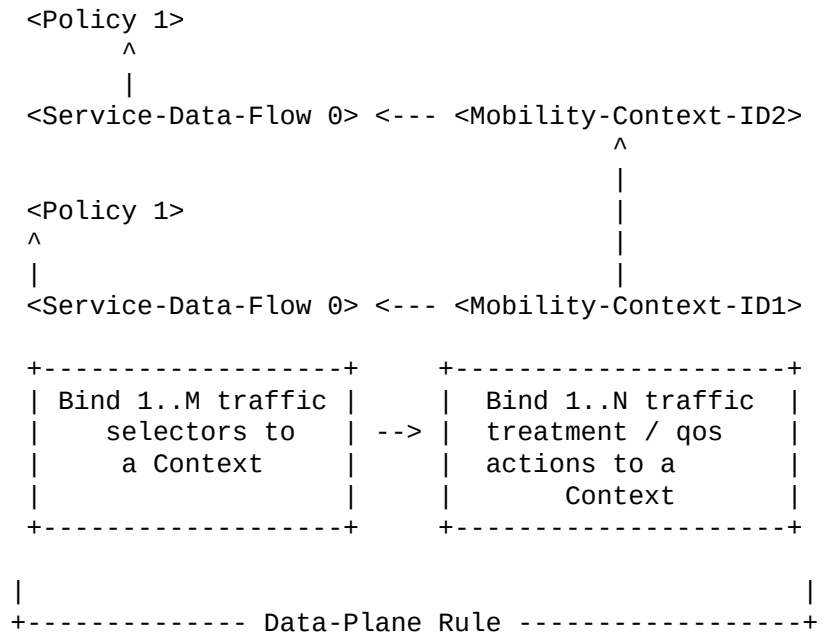


Figure 28: Mobility Context Hierarchy

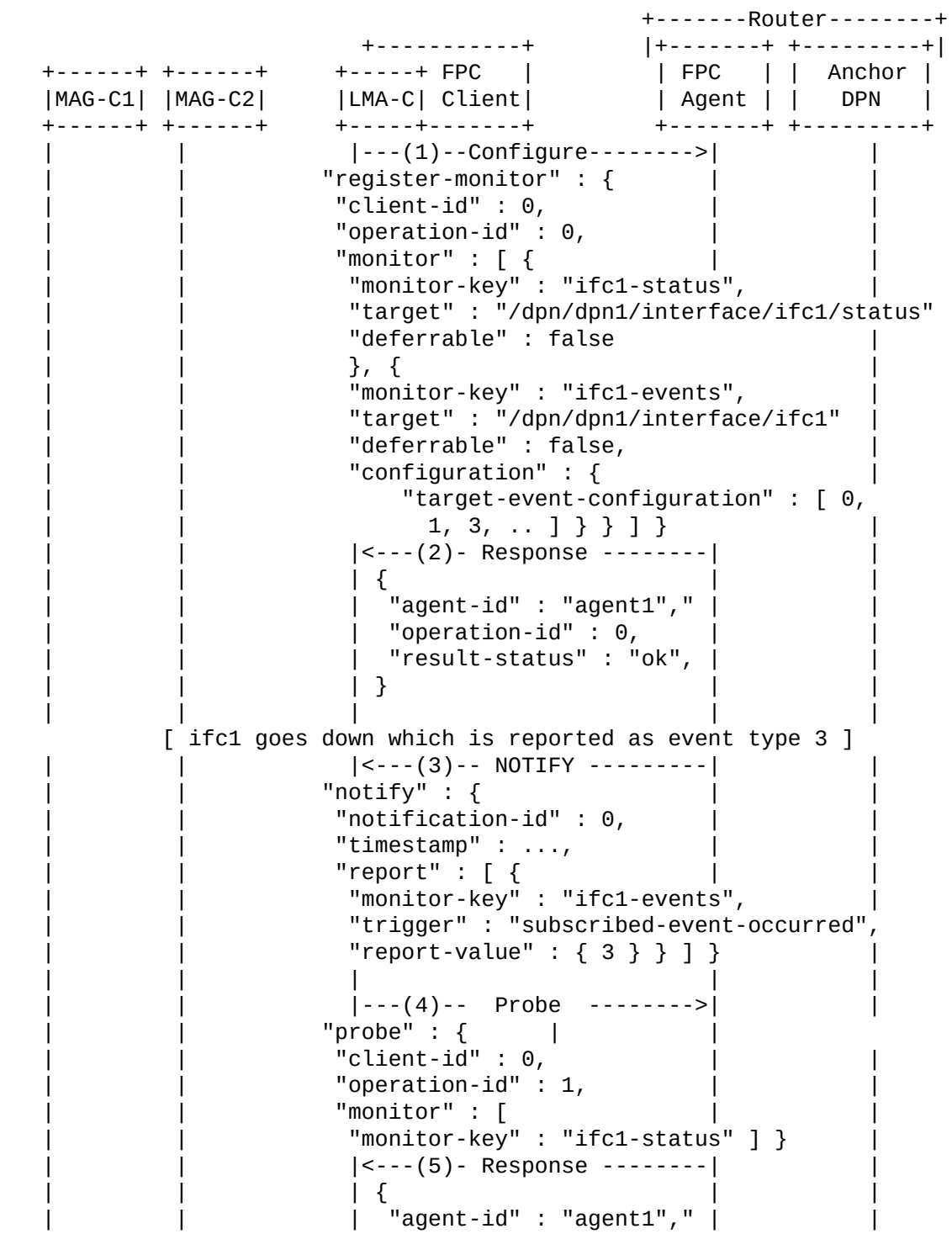
Figure 28 represents a mobility session hierarchy. A Client and Agent directly assigns values such as downlink traffic descriptors, QoS information, etc. A Client and Agent use the context identifiers to access the descriptors, qos information, etc. to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the qos or other mobility profile related Actions specified in the Mobility-Context's and Service-Data-Flow's' properties. If present, a Policy could contain tunnel information to encapsulate and forward the packet.

A second Mobility-Context also references Mobility-Context-ID1 in the figure. Based upon the technology a property in a parent context (parent mobility-context-id reference) MAY be inherited by its descendants. This permits concise over the wire representation. When a Client deletes a parent Context all children are also deleted.

5.2.5. Monitor Example

The following example shows the installation of a DPN level monitor (1) to observe ifc1 status, a property that is either "up" or "down", and another monitor to watch for interface events. The interface experiences an outage which is reported to the Client via a Notify (3) message. At a later time a Probe (4) and corresponding Notify (5) is sent. Finally, the monitors are de-registered (6).

Note, specific event identifiers and types are out of scope.



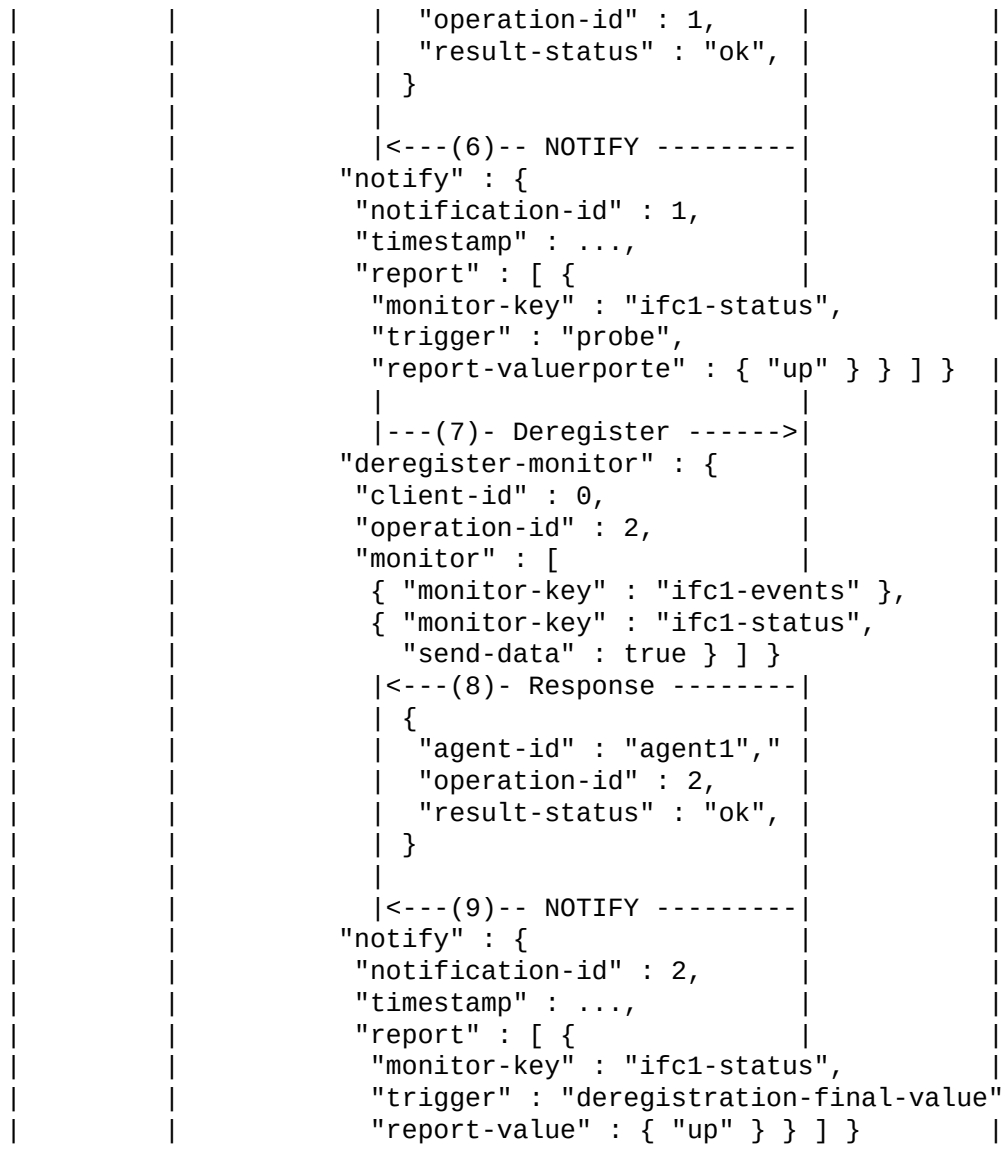


Figure 29: Monitor Example (focus on FPC reference point)

6. Templates and Command Sets

Configuration templates are shown below.

6.1. Monitor Configuration Templates

A periodic configuration specifies a time interval (ms) for reporting.

A scheduled configuration specifies a time for reporting.

A threshold configuration MUST have at least one hi or low threshold and MAY have both.

A Target-Events-Configuration is a list of Events that, when generated by the Target, results in a Monitor notification.

```
     |
     +-[Monitor] <List>
     ...
     |         +-[Configuration]
     |         |     +-[Periodic-Configuration]
     |         |     |       +-[(Unsigned32) Period:]
     ...
     |         +-[Configuration]
     |         |     +-[Schedule-Configuration]
     |         |     |       +-[(Unsigned32) Schedule:]
     ...
     |         +-[Configuration]
     |         |     +-[Threshold-Configuration]
     |         |     |       +-[(Unsigned32) Low]
     |         |     |       +-[(Unsigned32) Hi]
     ...
     |         +-[Configuration]
     |         |     +-[Target-Events-Configuration]
     |         |     |       +-[(Unsigned32) Event-Key:] <List>
```

Figure 30: Monitor Configuration Templates

6.2. Descriptor Templates

A IP-Prefix-Template MUST have at least the To or From IP Prefix / Length populated. The IP Prefix specifies and Address and Length.

The PMIP Traffic Selector template is mapped according to [RFC6088]

The RFC 5777 Classifier is a structured version of common filter rules and follows the format specified in [RFC5777]. The Flow-Label, Flow-Label range and ECN-IP-Codepoint specified in [RFC7660] are added to the Descriptor as well.

```
  |
  +-[ip-prefix-template]
  |       +-[(IP Prefix / Length) To-IP-Prefix]
  |       +-[(IP Prefix / Length) From-IP-Prefix]
  ...
  +-[pmip-traffic-selector]
```

```
|       +-[(Enumerated - IPv4 or IPv6) ts-format]
|       +-[ipsec-spi-range]
|       |   +-[ (ipsec-spi) start-spi: ]
|       |   +-[ (ipsec-spi) end-spi ]
|       +-[source-port-range]
|       |   +-[ (port-number) start-port: ]
|       |   +-[ (port-number) end-port ]
|       +-[destination-port-range]
|       |   +-[ (port-number) start-port: ]
|       |   +-[ (port-number) end-port ]
|       +-[source-address-range-v4]
|       |   +-[ (ipv4-address) start-address: ]
|       |   +-[ (ipv4-address) end-address ]
|       +-[destination-address-range-v4]
|       |   +-[ (ipv4-address) start-address: ]
|       |   +-[ (ipv4-address) end-address ]
|       +-[ds-range]
|       |   +-[ (dscp) start-ds: ]
|       |   +-[ (dscp) end-ds ]
|       +-[protocol-range]
|       |   +-[ (uint8) start-protocol: ]
|       |   +-[ (uint8) end-protocol ]
|       +-[source-address-range-v6]
|       |   +-[(ipv6-address) start-address: ]
|       |   +-[(ipv6-address) end-address ]
|       +-[destination-address-range-v6]
|       |   +-[(ipv6-address) start-address: ]
|       |   +-[(ipv6-address) end-address ]
|       +-[flow-label-range]
|       |   +-[(ipv6-flow-label) start-flow-label ]
|       |   +-[(ipv6-flow-label) end-flow-label ]
|       +-[traffic-class-range]
|       |   +-[ (dscp) start-traffic-class ]
|       |   +-[ (dscp) end-traffic-class ]
|       +-[next-header-range]
|       |   +-[ (uint8) start-next-header ]
|       |   +-[ (uint8) end-next-header ]
...
+-[rfc5777-classifier]
|       +-[Extensible: True]
|       +-[(uint8) protocol]
|       +-[(Enumerated - In/Out/Both) Direction]
|       +-[From-Spec] <List>
|       |    +-[(ip-address) IP-Address] <List>
|       |    +-[IP-Address-Range] <List>
|       |    |    +-[(ip-address) IP-Address-Start]
|       |    |    +-[(ip-address) IP-Address-End]
|       |    +-[IP-Address-Mask] <List>
```

```
|       |    |    +-[(ip-address) IP-Address:]
|       |    |    +-[(Unsigned 32) IP-Bit-Mask-Width:]
|       |    +-[(mac-address) MAC-Address] <List>
|       |    +-[MAC-Address-Mask] <List>
|       |    |    +-[(mac-address) MAC-Address:]
|       |    |    +-[(mac-address) MAC-Address-Mask-Pattern:]
|       |    +-[(eui64-address) EUI64-Address] <List>
|       |    +-[EUI64-Address-Mask] <List>
|       |    |    +-[(eui64-address) EUI64-Address:]
|       |    |    +-[(eui64-address) EUI64-Address-Mask-Pattern:]
|       |    +-[(Integer 32) Port] <List>
|       |    +-[Port-Range] <List>
|       |    |    +-[(Integer 32) Port-Start]
|       |    |    +-[(Integer 32) Port-End]
|       |    +-[(Boolean) Negated]
|       |    +-[(Boolean) Use-Assigned-Address]
|       +-[To-Spec] <List> (O)
|       |    +-[(ip-address) IP-Address] <List>
|       |    +-[IP-Address-Range] <List>
|       |    |    +-[(ip-address) IP-Address-Start]
|       |    |    +-[(ip-address) IP-Address-End]
|       |    +-[IP-Address-Mask] <List>
|       |    |    +-[(ip-address) IP-Address:]
|       |    |    +-[(Unsigned 32) IP-Bit-Mask-Width:]
|       |    +-[(mac-address) MAC-Address] <List>
|       |    +-[MAC-Address-Mask] <List>
|       |    |    +-[(mac-address) MAC-Address:]
|       |    |    +-[(mac-address) MAC-Address-Mask-Pattern:]
|       |    +-[(eui64-address) EUI64-Address] <List>
|       |    +-[EUI64-Address-Mask] <List>
|       |    |    +-[(eui64-address) EUI64-Address:]
|       |    |    +-[(eui64-address) EUI64-Address-Mask-Pattern:]
|       |    +-[(Integer 32) Port] <List>
|       |    +-[Port-Range] <List>
|       |    |    +-[(Integer 32) Port-Start]
|       |    |    +-[(Integer 32) Port-End]
|       |    +-[(Boolean) Negated]
|       |    +-[(Boolean) Use-Assigned-Address]
|       +-[(dscp) Diffserv-Code-Point] <List>
|       +-[(Boolean) Fragmentation-Flag ~ False]
|       +-[IP-Option] <List>
|       +-[TCP-Option] <List>
|       +-[TCP-Flags]
|       +-[ICMP-Type] <List>
|       +-[ETH-Option] <List>
|       +-[ecn-ip-codepoint] <List>
|       +-[(flowlabel) flow-label] <List>
|       +-[flow-label-range] <List>
```

```
|       |    +-[(flowlabel) flow-label-start]
|       |    +-[(flowlabel) flow-label-end]
```

Figure 31: Descriptor Templates

6.3. Tunnel Templates

The Network Service Header is specified in [RFC8300].

The MPLS SR Stack is specified in [I-D.ietf-spring-segment-routing-mpls].

The IPv6 SR Stack is specified in [I-D.ietf-6man-segment-routing-header].

A tunnel MUST have the local-address or remote-address (or both) populated.

For GRE, the gre-key MUST be present.

For GTP (GPRS Tunneling Protocol), the following attributes MAY be present

local tunnel endpoint identifier (teid) - MUST be present if local-address is nonempty

remote tunnel endpoint identifier (teid) - MUST be present if remote-address is nonempty

sequence-numbers-on - Indicates that sequence numbers will be used

Tunnels can be used as Next Hop and Descriptor values.

```
     |
     +-[next-hop-template]
     |     +-[Extensible: True]
     |     +-[(ip-address) address]
     |     +-[(mac-address) mac-address]
     |     +-[(service-path-id) service-path]
     |     +-[(mpls-label) mpls-path]
     |     +-[(network service header) nsh]
     |     +-[(Unsigned Integer) interface]
     |     +-[(Unsigned 128) segment-identifier]
     |     +-[(MPLS Stack) mpls-label-stack]
     |     +-[(MPLS SR Stack) mpls-sr-stack]
     |     +-[(IPv6 SR Stack) srv6-stack]
     |     +-[tunnel-template]
     ...
     |
     +-[tunnel-template]
     |     +-[Extensible: True]
     |     +-[(address) local-address]
     |     +-[(address) remote-address]
     |     +-[mtu]
     |     +-[(Enumeration - ipv4(0), ipv6(1), dual(2) payload_type:]
     |     +-[(Enumeration - ip-in-ip(0),
     |              udp(1), gre(2), gtpv1(3), gtpv2(4)) type:]
     |     +-[interface]
     |     +-[next-hop]
     |     +-[gre-key:] (type == gre)
     |     +-[gtp-info] (type == gtpv1 or type == gtpv2 )
     |     |    +-[(Unsigned 32) local-teid]
     |     |    +-[(Unsigned 32) remote-teid]
     |     |    +-[(Boolean) sequence-numbers-on] (type == gtpv1)
```

Figure 32: Tunnel Templates

6.4. Action Templates

The following figure shows common next-hop (set next-hop) and tunnel templates for Actions.

Drop action has no values.

Rewrite uses a Descriptor to set the values of the packet. Exactly one Descriptor MUST be present. Only the Destination and Source port fields, if present, are used from the Descriptor.

Copy-Forward creates a copy of the packet and then forwards it in accordance to the nexthop value.

```
                    |
                    +-[drop-template]
                    ...
                    |
                    +-[rewrite-template]
                    |     +-[Extensible: True]
                    |     +-[ip-prefix-template]
                    |     +-[pmip-traffic-selector]
                    |     +-[rfc5777-classifier]
                    ...
                    |
                    +-[copy-forward-template]
                    |     +-[Extensible: True]
                    |     +-[next-hop:]
```

Figure 33: Action Templates

6.5. Quality of Service Action Templates

PMIP QoS is specified in [RFC7222].

```
    |
    +-[qos-template]
    |     +-[Extensible: True]
    |     +-[(dscp) trafficclass]
    |     +-[pmip-qos]
    |     |     +-[(Unsigned 32) per-mn-agg-max-dl]
    |     |     +-[(Unsigned 32) per-mn-agg-max-ul]
    |     |     +-[per-session-agg-max-dl]
    |     |     |    +-[(Unsigned 32) max-rate:]
    |     |     |    +-[(Boolean) service-flag:]
    |     |     |    +-[(Boolean) exclude-flag:]
    |     |     +-[per-session-agg-max-ul]
    |     |     |    +-[(Unsigned 32) max-rate:]
    |     |     |    +-[(Boolean) service-flag:]
    |     |     |    +-[(Boolean) exclude-flag:]
    |     |     +-[allocation-retention-priority]
    |     |     |    +-[(Unsigned 8) priority-level:]
    |     |     |    +-[(Enumeration) preemption-capability:]
    |     |     |    +-[(Enumeration) preemption-vulnerability:]
    |     |     +-[(Unsigned 32) agg-max-dl]
    |     |     +-[(Unsigned 32) agg-max-ul]
    |     |     +-[(Unsigned 32) gbr-dl]
    |     |     +-[(Unsigned 32) gbr-ul]
```

Figure 34: QoS Templates

6.6. PMIP Command-Set

The following Command Set values are supported for IETF PMIP.

- assign-ip - Assign the IP Address for the mobile session.
- assign-dpn - Assign the Data-plane Node.
- session - Assign values for the Session Level.
- uplink - Command applies to uplink.
- downlink - Command applies to downlink.

6.7. 3GPP Specific Templates and Command-Set

3GPP support is optional and detailed in this section. The following acronyms are used:

APN-AMBR: Access Point Name Aggregate Maximum Bit Rate

UE-AMBR: User Equipment Aggregate Maximum Bit Rate

QCI: QoS Class Identifier

EBI: EPS Bearer Identity

LBI: Linked Bearer Identity

IMSI: International Mobile Subscriber Identity

TFT: Traffic Flow Template (TFT)

Generally, 3GPP QoS values should use the qos-template. Note: User Equipment Aggregate Maximum Bit Rate (UE-AMBR) maps to the per-mn-agg-max-dl and per-mn-agg-max-ul.

```
                    |
                    +-[ MN-Policy-Template ]
                    |     +-[(Unsigned 64) imsi:]
                    ...
                    +-[tunnel-template]
                    |     +-[Extensible: True]
                    |     +-[(unsigned 4) ebi:]
                    |     +-[(unsigned 4) lbi]
                    ...
                    +-[qos-template]
                    |     +-[Extensible: True]
                    |     +-[(unsigned 4) qos-class-identifier]
                    |     +-[(Unsigned 32) ue-agg-max-bitrate]
                    |     +-[(Unsigned 32) apn-agg-max-bitrate]
                    ...
```

Figure 35: 3GPP Mobility Templates

```
|
+-[ packet-filter ]
|     +-[Extensible: True]
|     +-[(Unsigned 8) identifier:]
|     +-[Contents:] <List>
|     |     +-[(ip-address) ipv4-ipv6-local]
|     |     +-[(ipv6-prefix) ipv6-prefix-local]
|     |     +-[(ip-address) ipv4-ipv6-remote]
|     |     +-[(ipv6-prefix) ipv6-prefix-remote]
|     |     +-[(Unsigned 8) protocol-next-header]
|     |     +-[(Unsigned 16) local-port]
|     |     +-[local-port-range]
|     |     |      +-[(Unsigned 16) local-port-lo]
|     |     |      +-[(Unsigned 16) local-port-hi]
|     |     +-[(Unsigned 16) remote-port]
|     |     +-[remote-port-range]
|     |     |      +-[(Unsigned 16) remote-port-lo]
|     |     |      +-[(Unsigned 16) remote-port-hi]
|     |     +-[(Unsigned 32) sec-parameter-index]
|     |     +-[(dscp) traffic-class]
|     |     +-[traffic-class-range]
|     |     |      +-[(dscp) traffic-class-lo]
|     |     |      +-[(dscp) traffic-class-hi]
|     |     +-[(dscp) flow-label]
...
```

Figure 36: 3GPP Packet Filter Template (Descriptor)

The following Command Set values are supported for 3GPP.

- assign-ip - Assign the IP Address for the mobile session.
- assign-fteid-ip - Assign the Fully Qualified TEID (F-TEID) LOCAL IP address.
- assign-fteid-teid - Assign the Fully Qualified TEID (F-TEID) LOCAL TEID.
- session - Assign values for the Session Level. When this involves 'assign-fteid-ip' and 'assign-fteid-teid', the values are part of the default bearer.
- uplink - Command applies to uplink.
- downlink - Command applies to downlink.
- assign-dpn - Assign the Data-plane Node.

7. Implementation Status

Three FPC Agent implementations have been made to date. The first was based upon Version 03 of the draft and followed Model 1. The second follows Version 04 of the document. Both implementations were OpenDaylight plug-ins developed in Java by Sprint. Version 04 is now primarily enhanced by GS Labs. Version 03 was known as fpcagent and version 04's implementation is simply referred to as 'fpc'. A third has been developed on an ONOS Controller for use in MCORD projects.

fpcagent's intent was to provide a proof of concept for FPC Version 03 Model 1 in January 2016 and research various errors, corrections and optimizations that the Agent could make when supporting multiple DPNs.

As the code developed to support OpenFlow and a proprietary DPN from a 3rd party, several of the advantages of a multi-DPN Agent became obvious including the use of machine learning to reduce the number of Flows and Policy entities placed on the DPN. This work has driven new efforts in the DIME WG, namely Diameter Policy Groups [I-D.bertz-dime-policygroups].

A throughput performance of tens per second using various NetConf based solutions in OpenDaylight made fpcagent, based on version 03, undesirable for call processing. The RPC implementation improved throughput by an order of magnitude but was not useful based upon FPC's Version 03 design using two information models. During this time the features of version 04 and its converged model became attractive and the fpcagent project was closed in August 2016.

fpcagent will no longer be developed and will remain a proprietary implementation.

The learnings of fpcagent has influenced the second project, fpc. Fpc is also an OpenDaylight project but is an open source release as the Opendaylight FpcAgent plugin (https://wiki.opendaylight.org/view/Project_Proposals:FpcAgent). This project is scoped to be a fully compliant FPC Agent that supports multiple DPNs including those that communicate via OpenFlow. The following features present in this draft and others developed by the FPC development team have already led to an order of magnitude improvement.

Migration of non-realtime provisioning of entities such as topology and policy allowed the implementation to focus only on the rpc.

Using only 5 messages and 2 notifications has also reduced implementation time.

Command Sets, an optional feature in this specification, have eliminated 80% of the time spent determining what needs to be done with a Context during a Create or Update operation.

Op Reference is an optional feature modeled after video delivery. It has reduced unnecessary cache lookups. It also has the additional benefit of allowing an Agent to become cacheless and effectively act as a FPC protocol adapter remotely with multi-DPN support or co-located on the DPN in a single-DPN support model.

Multi-tenant support allows for Cache searches to be partitioned for clustering and performance improvements. This has not been capitalized upon by the current implementation but is part of the development roadmap.

Use of Contexts to pre-provision policy has also eliminated any processing of Ports for DPNs which permitted the code for CONFIGURE and CONF_BUNDLE to be implemented as a simple nested FOR loops (see below).

Initial v04 performance results without code optimizations or tuning allow reliable provisioning of 1K FPC Mobility-Contexts processed per second on a 12 core server. This results in 2x the number of transactions on the southbound interface to a proprietary DPN API on the same machine.

fpc currently supports the following:

1 proprietary DPN API

Policy and Topology as defined in this specification using OpenDaylight North Bound Interfaces such as NetConf and RestConf

CONFIG and CONF_BUNDLE (all operations)

DPN assignment, Tunnel allocations and IPv4 address assignment by the Agent or Client.

Immediate Response is always an OK_NOTIFY_FOLLOWS.

```
assignment system (receives rpc call):
  perform basic operation integrity check
  if CONFIG then
    goto assignments
    if assignments was ok then
      send request to activation system
      respond back to client with assignment data
    else
      send back error
    end if
  else if CONF_BUNDLE then
    for each operation in bundles
    goto assignments
    if assignments was ok then
      hold onto data
    else
      return error with the assignments that occurred in
        prior operations (best effort)
    end if
    end for
    send bundles to activation systems
  end if

assignments:
  assign DPN, IPv4 Address and/or tunnel info as required
  if an error occurs undo all assignments in this operation
  return result

activation system:
  build cache according to op-ref and operation type
  for each operation
    for each Context
      for each DPN / direction in Context
        perform actions on DPN according to Command Set
      end for
    end for
  end for
  commit changes to in memory cache
  log transaction for tracking and notification
                                  (CONFIG_RESULT_NOTIFY)
```

Figure 37: fpc pseudo code

For further information please contact Lyle Bertz who is also a co-author of this document.

NOTE: Tenant support requires binding a Client ID to a Tenant ID (it is a one to many relation) but that is outside of the scope of this

specification. Otherwise, the specification is complete in terms of providing sufficient information to implement an Agent.

8. Security Considerations

Detailed protocol implementations for DMM Forwarding Policy Configuration must ensure integrity of the information exchanged between a FPC Client and a FPC Agent. Required Security Associations may be derived from co-located functions, which utilize the FPC Client and FPC Agent respectively.

The YANG modules defined in this memo are designed to be accessed via the NETCONF [RFC6241] or RESTCONF [RFC8040] protocol. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

The information model defined in the memo is designed to be access by protocols specified in extensions to this document or, if using the YANG modules, as described above.

There are a number of data nodes defined which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., a NETCONF edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/ vulnerability:

Nodes under the Policy tree provide generic policy enforcement and traffic classification. They can be used to block or permit traffic. If this portion of the model was to be compromised it may be used to block, identify or permit traffic that was not intended by the Tenant or FPC Client.

Nodes under the Topology tree provide definition of the Tenant’s forwarding topology. Any compromise of this information will provide topology information that could be used for subsequent attack vectors. Removal of topology can limit services.

Mobility-Context provides runtime only information and manipulated by remote procedure calls. The unwanted deletion or removal of such information would deny users service or provide services to unauthorized parties.

Some of the readable data nodes defined may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to

these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

IP address assignments in the Mobility-Context along with their associated tunnel configurations/identifiers (from the FPC base module)

Internaitonal Mobile Subscriber Identity (IMSI) and bearer identifiers in the Context when using the FPC base model

Some of the RPC operations defined may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

Configure sends Mobility-Context information which can include information of a sensitive or vulnerable nature in some network environments as described above.

Monitor related RPC operations do not specifically provide sensitive or vulnerable information but care must be taken by users to avoid identifier values that expose sensitive or vulnerable information.

Notifications MUST be treated with same level of protection and scrutiny as the operations they correspond to. For example, a Configure-Result-Notification provides the same information that is sent as part of the input and output of the Configure RPC operation.

General usage of FPC MUST consider the following:

FPC Naming Section 4.5 permits arbitrary string values but a user MUST avoid placing sensitive or vulnerable information in those values.

Policies that are very narrow and permit the identification of specific traffic, e.g. that of a single user, SHOULD be avoided.

9. IANA Considerations

This document registers six URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc
Registrant Contact: The DMM WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

```
      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-traffic-selector-types
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-settingsext
      Registrant Contact: The DMM WG of the IETF.
      XML: N/A, the requested URI is an XML namespace.

     URI: urn:ietf:params:xml:ns:yang:ietf-diam-trafficclassifier
     Registrant Contact: The DMM WG of the IETF.
     XML: N/A, the requested URI is an XML namespace.
```

This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020].

```
      name:         ietf-dmm-fpc
      namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-fpc
      prefix:       fpc
      reference:    TBD1

      name:         ietf-dmm-pmip-qos
      namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
      prefix:       qos-pmip
      reference:    TBD2

      name:         ietf-dmm-traffic-selector-types
      namespace:    urn:ietf:params:xml:ns:yang:
        ietf-dmm-traffic-selector-types
      prefix:       traffic-selectors
      reference:    TBD3

      name:         ietf-dmm-fpc-settingsext
      namespace:    urn:ietf:params:xml:ns:yang:
        ietf-dmm-fpc-settingsext
      prefix:       fpcbase
      reference:    TBD4

        name:         ietf-diam-trafficclassifier
        namespace:    urn:ietf:params:xml:ns:yang:
        ietf-diam-trafficclassifier
        prefix:       diamclassifier
        reference:    TBD5
```

10. Work Team Participants

Participants in the FPSM work team discussion include Satoru Matsushima, Danny Moses, Sri Gundavelli, Marco Liebsch, Pierrick Seite, Alper Yegin, Carlos Bernardos, Charles Perkins and Fred Templin.

11. References

11.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-14 (work in progress), June 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., Ed., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, DOI 10.17487/RFC5777, February 2010, <https://www.rfc-editor.org/info/rfc5777>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <https://www.rfc-editor.org/info/rfc6088>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <https://www.rfc-editor.org/info/rfc8072>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

11.2. Informative References

[I-D.bertz-dime-policygroups]
Bertz, L. and M. Bales, "Diameter Policy Groups and Sets", draft-bertz-dime-policygroups-05 (work in progress), December 2017.

[I-D.ietf-dmm-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-04 (work in progress), May 2018.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958, January 2005, <https://www.rfc-editor.org/info/rfc3958>.

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <https://www.rfc-editor.org/info/rfc5213>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7222] Liebsch, M., Seite, P., Yokota, H., Korhonen, J., and S. Gundavelli, "Quality-of-Service Option for Proxy Mobile IPv6", RFC 7222, DOI 10.17487/RFC7222, May 2014, <https://www.rfc-editor.org/info/rfc7222>.

[RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <https://www.rfc-editor.org/info/rfc7333>.

[RFC7660] Bertz, L., Manning, S., and B. Hirschman, "Diameter Congestion and Filter Attributes", RFC 7660, DOI 10.17487/RFC7660, October 2015, <https://www.rfc-editor.org/info/rfc7660>.

Appendix A. YANG Data Model for the FPC protocol

This section provides a type mapping for FPC structures in YANG. When being mapped to a specific information such as YANG the data type MAY change.

Keys for Actions, Descriptors, Rules, Policies, DPNs, Domains and Mobility-Contexts are specified as FPC-Identity which follows rules according to Section 4.5.

Action and Descriptor Templates are mapped as choices. This was done to ensure no duplication of Types and avoid use of identityref for typing.

Policy Expressions are provided as default values. NOTE that a static value CANNOT be supported in YANG.

Mapping of templates to YANG are performed as follows:

Value is defined as a choice statement for extensibility and therefore a type value is not necessary to discriminated types

Generic attributes are distinguished by the "Settings" type and holds ANY value. It is an any data node under configurations.

The CONFIGURE and CONFIGURE-RESULT-NOTIFICATION use the yang-patch-status which is a container for edits. This was done to maximize YANG reuse.

In the configure rpc, operation-id is mapped to patch-id and in an edit the edit-type is mapped to operation.

The Result-Status attribute is mapped to the 'ok' (empty leaf) or errors structure.

The Policy-Status is mapped to entity-state to reduce YANG size.

Five modules are defined:

- ietf-dmm-fpc (fpc) - Defines the base model and messages for FPC that are meant to be static in FPC.
- ietf-dmm-fpc-settingsext - A FPC module that defines the information model elements that are likely to be extended in FPC.
- ietf-pmip-qos (pmip-qos) - Defines proxy mobile IPv6 QoS parameters per RFC 7222
- ietf-trafficselectors-types (traffic-selectors) - Defines Traffic Selectors per [RFC6088]
- ietf-diam-trafficclassifier (diamclassifier) - Defines the Classifier per [RFC5777]

All modules defined in this specification make use of (import) ietf-inet-types as defined in [RFC6991].

ietf-dmm-fpc-settingsext and ietf-diam-trafficclassifier make use of (imports) ietf-yang-types as defined in [RFC6991].

ietf-dmm-fpc imports the restconf (ietf-restconf) [RFC8040] and yang patch (ietf-yang-patch) [RFC8072] modules.

ietf-pmip-qos and ietf-dmm-fpc-settings import the trafficselector from the ietf-traffic-selector-types module.

ietf-dmm-fpc-settings also imports the qosattribute (ietf-pmip-qos) and classifier (ietf-diam-trafficclassifier).

ietf-dmm-fpc-settingsext groups various settings, actions and descriptors and is used by the fpc module (ietf-dmm-fpc).

The following groupings are intended for reuse (import) by other modules.

- qosoption (ietf-qos-pmip module)

- qosattribute (ietf-qos-pmip module)
- qosoption (ietf-qos-pmip module)
- Allocation-Retention-Priority-Value (ietf-qos-pmip module)
- trafficselector (ietf-traffic-selector-types)
- classifier (ietf-diam-trafficclassifier)
- packet-filter (ietf-dmm-fpc-settingsext)
- instructions (ietf-dmm-fpc-settingsext)
- fpc-descriptor-value (ietf-dmm-fpc-settingsext)
- fpc-action-value (ietf-dmm-fpc-settingsext)

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

DPNs conformant to NMDA MAY only have policies, installed policies, topology, domains and mobility session information that has been assigned to it in its intended and operational datastores. What is housed in the operational datastore MAY be determined on a per DPN basis and using the Entity-Status as a guideline based upon tradeoffs described in Section 4.6.

ServiceGroups are not expected to appear in operational datastores of DPNs as they remain in and are used by FPC Agents and Clients. They MAY be operationally present in DNS when using the Dynamic Delegation and Discovery System (DDDS) as defined in [RFC3958] or the operational datastore of systems that provide equivalent functionality.

A.1. FPC YANG Model

This module defines the information model and protocol elements specified in this document.

This module references [RFC6991], [RFC8040] and the fpc-settingsext module defined in this document.

```
<CODE BEGINS> file "ietf-dmm-fpc@2018-05-17.yang"
module ietf-dmm-fpc {
  yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc";
    prefix fpc;
```

```
       import ietf-inet-types { prefix inet;
           revision-date 2013-07-15; }
       import ietf-dmm-fpc-settingsext { prefix fpcbase;
           revision-date 2018-05-17; }
       import ietf-diam-trafficclassifier { prefix rfc5777;
           revision-date 2018-05-17; }
       import ietf-restconf { prefix rc;
           revision-date 2017-01-26; }
       import ietf-yang-patch { prefix ypatch;
           revision-date 2017-02-22; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Jouni Korhonen
                     <mailto:jouni.nospam@gmail.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for
        Forwarding Policy Configuration Protocol (FPCP).

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

        This document is subject to BCP 78 and the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info) in effect on the date of
        publication of this document. Please review these documents
        carefully, as they describe your rights and restrictions with
        respect to this document. Code Components extracted from this
        document must include Simplified BSD License text as described
        in Section 4.e of the Trust Legal Provisions and are provided
        without warranty as described in the Simplified BSD License.";

       revision 2018-05-17 {
```

```
      description "Initial Revision.";
      reference "draft-ietf-dmm-fpc-cpdp-10";
    }

      //General Structures
      grouping templatedef {
          leaf extensible {
              type boolean;
              description "Indicates if the template is extensible";
          }
          leaf-list static-attributes {
              type string;
              description "Attribute (Name) whose value cannot
                  change";
          }
          leaf-list mandatory-attributes {
              type string;
              description "Attribute (Name) of optional attributes
                that MUST be present in instances of this tempplate.";
          }
          leaf entity-state {
              type enumeration {
                  enum initial {
                      description "Inital Configuration";
                  }
                  enum partially-configured {
                      description "Partial Configuration";
                  }
                  enum configured {
                      description "Confgured";
                  }
                  enum active {
                      description "Active";
                  }
              }
              default initial;
              description "Entity State";
          }
          leaf version {
              type uint32;
              description "Template Version";
          }
          description "Teamplate Definition";
      }
      typedef fpc-identity {
          type union {
              type uint32;
              type instance-identifier;
```

```
            type string;
        }
        description "FPC Identity";
    }
    grouping index {
        leaf index {
            type uint16;
            description "Index";
        }
        description "Index Value";
    }

    // Policy Structures
    grouping descriptor-template-key {
        leaf descriptor-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Descriptor Key";
        }
        description "Descriptor-Template Key";
    }
    grouping action-template-key {
        leaf action-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Action Key";
        }
        description "Action-Template Key";
    }
    grouping rule-template-key {
        leaf rule-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Rule Identifier";
        }
        description "Rule Key";
    }
    grouping policy-template-key {
        leaf policy-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Rule Identifier";
        }
        description "Rule Key";
    }

    grouping fpc-setting-value {
        anydata setting;
```

```
            description "FPC Setting Value";
        }
        // Configuration / Settings
        grouping policy-configuration-choice {
               choice policy-configuration-value {
                    case descriptor-value {
                        uses fpcbase:fpc-descriptor-value;
                        description "Descriptor Value";
                    }
                    case action-value {
                        uses fpcbase:fpc-action-value;
                        description "Action Value";
                    }
                    case setting-value {
                        uses fpc:fpc-setting-value;
                        description "Setting";
                    }
                    description "Policy Attributes";
                }
                description "Policy Configuration Value Choice";
        }
        grouping policy-configuration {
            list policy-configuration {
                key index;
                uses fpc:index;
                uses fpc:policy-configuration-choice;
                description "Policy Configuration";
            }
            description "Policy Configuration Value";
        }
        grouping ref-configuration {
            uses fpc:policy-template-key;
            uses fpc:policy-configuration;
            uses fpc:templatedef;
            description "Policy-Configuration Entry";
        }

        // FPC Policy
        grouping policy-information-model {
          list action-template {
            key action-template-key;
            uses fpc:action-template-key;
            uses fpcbase:fpc-action-value;
            uses fpc:templatedef;
            description "Action Template";
          }
          list descriptor-template {
            key descriptor-template-key;
```

```
            uses fpc:descriptor-template-key;
            uses fpcbase:fpc-descriptor-value;
            uses fpc:templatedef;
            description "Descriptor Template";
          }
          list rule-template {
            key rule-template-key;
            uses fpc:rule-template-key;
            leaf descriptor-match-type {
                type enumeration {
                    enum or {
                        value 0;
                        description "OR logic";
                    }
                    enum and {
                        value 1;
                        description "AND logic";
                    }
                }
                mandatory true;
                description "Type of Match (OR or AND) applied
                    to the descriptor-configurations";
            }
            list descriptor-configuration {
                key "descriptor-template-key";
                uses fpc:descriptor-template-key;
                leaf direction {
                    type rfc5777:direction-type;
                    description "Direction";
                }
                list attribute-expression {
                    key index;
                    uses fpc:index;
                    uses fpcbase:fpc-descriptor-value;
                    description "Descriptor Attributes";
                }
                uses fpc:fpc-setting-value;
                description "A set of Descriptor references";
            }
            list action-configuration {
                key "action-order";
                leaf action-order {
                    type uint32;
                    mandatory true;
                    description "Action Execution Order";
                }
                uses fpc:action-template-key;
                list attribute-expression {
```

```
                key index;
                uses fpc:index;
                uses fpcbase:fpc-action-value;
                description "Action Attributes";
            }
            uses fpc:fpc-setting-value;
            description "A set of Action references";
        }
        uses fpc:templatedef;
        list rule-configuration {
            key index;
            uses fpc:index;
            uses fpc:policy-configuration-choice;
            description "Rule Configuration";
        }
        description "Rule Template";
      }
      list policy-template {
        key policy-template-key;
        uses fpc:policy-template-key;
        list rule-template {
            key "precedence";
            unique "rule-template-key";
            leaf precedence {
                type uint32;
                mandatory true;
                description "Rule Precedence";
            }
            uses fpc:rule-template-key;
            description "Rule Entry";
        }
        uses fpc:templatedef;
        uses fpc:policy-configuration;
        description "Policy Template";
      }
      description "FPC Policy Structures";
    }

    // Topology Information Model
    identity role {
        description "Role";
    }
    grouping dpn-key {
        leaf dpn-key {
            type fpc:fpc-identity;
            description "DPN Key";
        }
        description "DPN Key";
```

```
    }
    grouping role-key {
        leaf role-key {
            type identityref {
                base "fpc:role";
            }
            mandatory true;
            description "Access Technology Role";
        }
        description "Access Technology Role key";
    }
    grouping interface-key {
        leaf interface-key{
            type fpc:fpc-identity;
            mandatory true;
            description "interface identifier";
        }
        description "Interface Identifier key";
    }
    identity interface-protocols {
        description "Protocol supported by the interface";
    }
    identity features {
        description "Protocol features";
    }

  // Mobility Context
  grouping mobility-context {
    leaf mobility-context-key {
      type fpc:fpc-identity;
      mandatory true;
      description "Mobility Context Key";
    }
    leaf-list delegating-ip-prefix {
        type inet:ip-prefix;
        description "IP Prefix";
    }
    leaf parent-context {
      type fpc:fpc-identity;
      description "Parent Mobility Context";
    }
    leaf-list child-context {
        type fpc:fpc-identity;
        description "Child Mobility Context";
    }
    container mobile-node {
        leaf-list ip-address {
            type inet:ip-address;
```

```
                description "IP Address";
            }
            leaf imsi {
                type fpcbase:imsi-type;
                description "IMSI";
            }
            list mn-policy-configuration {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "MN Policy Configuration";
            }
            description "Mobile Node";
        }
        container domain {
            leaf domain-key {
                type fpc:fpc-identity;
                description "Domain Key";
            }
            list domain-policy-settings {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "MN Policy Configuration";
            }
            description "Domain";
        }
        list dpn {
            key dpn-key;
            uses fpc:dpn-key;
            list dpn-policy-configuration {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "DPN Policy Configuration";
            }
            leaf role {
                type identityref {
                    base "fpc:role";
                }
                description "Role";
            }
            list service-data-flow {
                key identifier;
                leaf identifier {
                    type uint32;
                    description "Generic Identifier";
                }
                leaf service-group-key {
                    type fpc:fpc-identity;
                    description "Service Group Key";
```

```
                }
                list interface {
                    key interface-key;
                    uses fpc:interface-key;
                    description "interface assigned";
                }
                list service-data-flow-policy-configuration {
                    key policy-template-key;
                    uses fpc:ref-configuration;
                    description "Flow Policy Configuration";
                }
                description "Service Dataflow";
            }
            description "DPN";
        }
        description "Mobility Context";
    }

    // Events, Probes & Notifications
    identity event-type {
      description "Base Event Type";
    }
    typedef event-type-id {
      type uint32;
      description "Event ID Type";
    }
    grouping monitor-key {
      leaf monitor-key {
        type fpc:fpc-identity;
        mandatory true;
        description "Monitor Key";
      }
      description "Monitor Id";
    }
    grouping monitor-config {
      uses fpc:templatedef;
      uses fpc:monitor-key;
      leaf target {
        type string;
        description "target";
      }
      leaf deferrable {
        type boolean;
        description "Indicates reports related to this
          config can be delayed.";
      }
      choice configuration {
        mandatory true;
```

```
        leaf period {
            type uint32;
            description "Period";
        }
        case threshold-config {
          leaf low {
            type uint32;
            description "low threshold";
          }
          leaf hi {
            type uint32;
            description "high threshold";
          }
          description "Threshold Config Case";
        }
        leaf schedule {
            type uint32;
            description "Reporting Time";
        }
        leaf-list event-identities {
            type identityref {
              base "fpc:event-type";
            }
            description "Event Identities";
        }
        leaf-list event-ids {
            type uint32;
              description "Event IDs";
        }
        description "Event Config Value";
      }
      description "Monitor Configuration";
    }

    // Top Level Structures
    list tenant {
     key "tenant-key";
     leaf tenant-key {
          type fpc:fpc-identity;
          description "Tenant Key";
     }
     container topology-information-model {
       config false;
       list service-group {
          key "service-group-key role-key";
          leaf service-group-key {
              type fpc:fpc-identity;
              mandatory true;
```

```
            description "Service Group Key";
        }
        leaf service-group-name {
            type string;
            description "Service Group Name";
        }
        uses fpc:role-key;
        leaf role-name {
            type string;
            mandatory true;
            description "Role Name";
        }
        leaf-list protocol {
          type identityref {
            base "interface-protocols";
          }
          min-elements 1;
          description "Supported protocols";
        }
        leaf-list feature {
            type identityref {
                base "interface-protocols";
            }
            description "Supported features";
        }
        list service-group-configuration {
            key index;
            uses fpc:index;
            uses fpc:policy-configuration-choice;
            description "Settings";
        }
        list dpn {
            key dpn-key;
            uses fpc:dpn-key;
            min-elements 1;
            list referenced-interface {
                key interface-key;
                uses fpc:interface-key;
                leaf-list peer-service-group-key {
                    type fpc:fpc-identity;
                    description "Peer Service Group";
                }
                description "Referenced Interface";
            }
            description "DPN";
        }
        description "Service Group";
    }
```

```
      list dpn {
        key dpn-key;
        uses fpc:dpn-key;
        leaf dpn-name {
          type string;
          description "DPN name";
        }
        leaf dpn-resource-mapping-reference {
          type string;
          description "Reference to underlying DPN resource(s)";
        }
        leaf domain-key {
          type fpc:fpc-identity;
          description "Domains";
        }
        leaf-list service-group-key {
          type fpc:fpc-identity;
          description "Service Group";
        }
        list interface {
          key "interface-key";
          uses fpc:interface-key;
          leaf interface-name {
              type string;
              description "Service Endpoint Interface Name";
          }
          leaf role {
              type identityref {
                  base "fpc:role";
              }
              description "Roles supported";
          }
          leaf-list protocol {
            type identityref {
              base "interface-protocols";
            }
            description "Supported protocols";
          }
          list interface-configuration {
           key index;
           uses fpc:index;
           uses fpc:policy-configuration-choice;
           description "Interface settings";
         }
         description "DPN interfaces";
        }
        list dpn-policy-configuration {
          key policy-template-key;
```

```
          uses fpc:ref-configuration;
          description "DPN Policy Configuration";
        }
        description "Set of DPNs";
      }
      list domain {
        key domain-key;
        leaf domain-key {
          type fpc:fpc-identity;
          mandatory true;
          description "Domain Key";
        }
        leaf domain-name {
          type string;
          description "Domain displayname";
        }
        list domain-policy-configuration {
          key policy-template-key;
          uses fpc:ref-configuration;
          description "Domain Configuration";
        }
        description "List of Domains";
      }
      container dpn-checkpoint {
          uses fpc:basename-info;
          description "DPN Checkpoint information";
      }
      container service-group-checkpoint {
          uses fpc:basename-info;
          description "Service Group Checkpoint information";
      }
      container domain-checkpoint {
          uses fpc:basename-info;
          description "Domain Checkpoint information";
      }
      description "FPC Topology grouping";
    }
    container policy-information-model {
         config false;
         uses fpc:policy-information-model;
         uses fpc:basename-info;
         description "Policy";
    }
    list mobility-context {
         key "mobility-context-key";
         config false;
         uses fpc:mobility-context;
         description "Mobility Context";
```

```
      }
      list monitor {
           key monitor-key;
           config false;
           uses fpc:monitor-config;
           description "Monitor";
      }
      description "Tenant";
     }

       typedef agent-identifier {
           type fpc:fpc-identity;
           description "Agent Identifier";
       }
       typedef client-identifier {
           type fpc:fpc-identity;
           description "Client Identifier";
       }
       grouping basename-info {
             leaf basename {
               type fpc:fpc-identity;
               description "Rules Basename";
             }
             leaf base-checkpoint {
               type string;
               description "Checkpoint";
             }
             description "Basename Information";
       }

       // RPCs
       grouping client-id {
           leaf client-id {
               type fpc:client-identifier;
               mandatory true;
               description "Client Id";
           }
           description "Client Identifier";
       }
       grouping execution-delay {
           leaf execution-delay {
               type uint32;
               description "Execution Delay (ms)";
           }
           description "Execution Delay";
       }
       typedef ref-scope {
         type enumeration {
```

```
            enum none {
              value 0;
              description "no references";
            }
            enum op {
              value 1;
              description "All references are intra-operation";
            }
            enum bundle {
              value 2;
              description "All references in exist in bundle";
            }
            enum storage {
              value 3;
              description "One or more references exist in storage.";
            }
            enum unknown {
              value 4;
              description "The location of the references are unknown.";
            }
          }
          description "Search scope for references in the operation.";
        }
        rpc configure {
            description "Configure RPC";
            input {
                uses client-id;
                uses execution-delay;
                uses ypatch:yang-patch;
            }
            output {
                uses ypatch:yang-patch-status;
            }
        }
        augment "/configure/input/yang-patch/edit" {
            leaf reference-scope {
                type fpc:ref-scope;
                description "Reference Scope";
            }
            uses fpcbase:instructions;
            description "yang-patch edit augments for configure rpc";
        }
        grouping subsequent-edits {
             list subsequent-edit {
               key edit-id;
               ordered-by user;

               description "Edit list";
```

```
            leaf edit-id {
              type string;
              description "Arbitrary string index for the edit.";
            }

            leaf operation {
              type enumeration {
                enum create {
                  description "Create";
                }
                enum delete {
                  description "Delete";
                }
                enum insert {
                  description "Insert";
                }
                enum merge {
                  description "Merge";
                }
                enum move {
                  description "Move";
                }
                enum replace {
                  description "Replace";
                }
                enum remove {
                  description
                    "Delete the target node if it currently exists.";
                }
              }
              mandatory true;
              description
                "The datastore operation requested";
            }

            leaf target {
              type ypatch:target-resource-offset;
              mandatory true;
              description
                "Identifies the target data node";
            }

            leaf point {
             when "(../operation = 'insert' or ../operation = 'move')"
              + "and (../where = 'before' or ../where = 'after')" {
                description
                  "This leaf only applies for 'insert' or 'move'
                   operations, before or after an existing entry.";
```

```
        }
        type ypatch:target-resource-offset;
        description
          "The absolute URL path for the data node";
      }

      leaf where {
       when "../operation = 'insert' or ../operation = 'move'" {
          description
            "This leaf only applies for 'insert' or 'move'
             operations.";
        }
        type enumeration {
          enum before {
            description
              "Insert or move a data node before.";
          }
          enum after {
            description
              "Insert or move a data node after.";
          }
         enum first {
            description
              "Insert or move a data node so it becomes ordered
               as the first entry.";
          }
          enum last {
            description
              "Insert or move a data node so it becomes ordered
               as the last entry.";
          }
        }
        default last;
        description
          "Identifies where a data resource will be inserted
           or moved.";
      }

      anydata value {
        when "../operation = 'create' "
           + "or ../operation = 'merge' "
           + "or ../operation = 'replace' "
           + "or ../operation = 'insert'" {
          description
            "The anydata 'value' is only used for 'create',
             'merge', 'replace', and 'insert' operations.";
        }
        description
```

```
                "Value used for this edit operation.";
              }
            }
            description "Subsequent Edits";
        }
        augment "/configure/output/yang-patch-status/edit-status/edit/"
            + "edit-status-choice/ok" {
            leaf notify-follows {
                type boolean;
                description "Notify Follows Indication";
            }
            uses fpc:subsequent-edits;
            description "Configure output augments";
        }

        grouping op-header {
            uses client-id;
            uses execution-delay;
            leaf operation-id {
                type uint64;
                mandatory true;
                description "Operation Identifier";
            }
            description "Common Operation header";
        }
        grouping monitor-response {
            leaf operation-id {
                type uint64;
                mandatory true;
                description "Operation Identifier";
            }
            choice edit-status-choice {
                   description
                     "A choice between different types of status
                      responses for each 'edit' entry.";
                   leaf ok {
                     type empty;
                     description
                       "This 'edit' entry was invoked without any
                        errors detected by the server associated
                        with this edit.";
                   }
                   case errors {
                     uses rc:errors;
                     description
                       "The server detected errors associated with the
                        edit identified by the same 'edit-id' value.";
                   }
```

```
          }
          description "Monitor Response";
      }

    // Common RPCs
    rpc register_monitor {
      description "Used to register monitoring of parameters/events";
      input {
        uses fpc:op-header;
        list monitor {
          key monitor-key;
          uses fpc:monitor-config;
          description "Monitor Configuration";
        }
      }
      output {
          uses fpc:monitor-response;
      }
    }
    rpc deregister_monitor {
      description "Used to de-register monitoring of
        parameters/events";
      input {
        uses fpc:op-header;
        list monitor {
          key monitor-key;
          uses fpc:monitor-key;
          min-elements 1;
          leaf send_data {
              type boolean;
              description "Indicates if NOTIFY with final data
                  is desired upon deregistration";
          }
          description "Monitor Identifier";
        }
      }
      output {
          uses fpc:monitor-response;
      }
    }
    rpc probe {
      description "Probe the status of a registered monitor";
      input {
        uses fpc:op-header;
        list monitor {
          key monitor-key;
          uses fpc:monitor-key;
          min-elements 1;
```

```
          description "Monitor";
        }
      }
      output {
          uses fpc:monitor-response;
      }
    }

    // Notification Messages & Structures
    notification config-result-notification {
      uses ypatch:yang-patch-status;
      description "Configuration Result Notification";
    }
    augment "/config-result-notification" {
          uses fpc:subsequent-edits;
          description "config-result-notificatio augment";
    }

    identity notification-cause {
      description "Notification Cause";
    }
    identity subscribed-event-occurred {
      base "notification-cause";
      description "Subscribed Event Occurence";
    }
    identity low-threshold-crossed {
      base "notification-cause";
      description "Subscribed Event Occurence";
    }
    identity high-threshold-crossed {
      base "notification-cause";
      description "Subscribed Event Occurence";
    }
    identity periodic-report {
      base "notification-cause";
      description "Periodic Report";
    }
    identity scheduled-report {
      base "notification-cause";
      description "Scheduled Report";
    }
    identity probe {
      base "notification-cause";
      description "Probe";
    }
    identity deregistration-final-value {
      base "notification-cause";
      description "Probe";
```

```
    }
    identity monitoring-suspension {
      base "notification-cause";
      description "Indicates monitoring suspension";
    }
    identity monitoring-resumption {
      base "notification-cause";
      description "Indicates that monitoring has resumed";
    }
    identity dpn-available {
      base "notification-cause";
      description "DPN Candidate Available";
    }
    identity dpn-unavailable {
      base "notification-cause";
      description "DPN Unavailable";
    }
    notification notify {
      leaf notification-id {
        type uint32;
        description "Notification Identifier";
      }
      leaf timestamp {
        type uint32;
        description "timestamp";
      }
      list report {
          key monitor-key;
          uses fpc:monitor-key;
          min-elements 1;
          leaf trigger {
              type identityref {
                  base "notification-cause";
              }
              description "Notification Cause";
          }
          choice value {
              case dpn-candidate-available {
                  leaf node-id {
                      type inet:uri;
                      description "Topology URI";
                  }
                  list supported-interface-list {
                      key role-key;
                      uses fpc:role-key;
                      description "Support Intefaces";
                  }
                  description "DPN Candidate Information";
```

```
                }
                case dpn-unavailable {
                    leaf dpn-id {
                      type fpc:fpc-identity;
                      description "DPN Identifier for DPN Unavailable";
                    }
                    description "DPN Unavailable";
                }
                anydata report-value {
                    description "Any non integer report";
                }
                description "Report Value";
            }
            description "Report";
        }
        description "Notify Message";
      }
    }
    <CODE ENDS>
```

A.2. FPC YANG Settings and Extensions Model

This module defines the base data elements in FPC that are likely to be extended.

This module references [RFC6991], ietf-trafficselector-types and ietf-pmip-qos modules.

```
<CODE BEGINS> file "ietf-dmm-fpc-settingsext@2018-05-17.yang"
module ietf-dmm-fpc-settingsext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-settingsext";
  prefix fpcbase;

    import ietf-inet-types { prefix inet;
        revision-date 2013-07-15; }
    import ietf-trafficselector-types { prefix traffic-selectors;
        revision-date 2018-05-17; }
    import ietf-yang-types { prefix ytypes;
        revision-date 2013-07-15; }
    import ietf-pmip-qos { prefix pmipqos;
        revision-date 2018-05-17; }
    import ietf-diam-trafficclassifier { prefix rfc5777;
        revision-date 2018-05-17; }

    organization "IETF Distributed Mobility Management (DMM)
      Working Group";
```

```
       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Sri Gundavelli
                     <mailto:sgundave@cisco.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
       "This module contains YANG definition for
        Forwarding Policy Configuration Protocol(FPCP).

         It contains Settings defintions as well as Descriptor and
         Action extensions.

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

        This document is subject to BCP 78 and the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info) in effect on the date of
        publication of this document. Please review these documents
        carefully, as they describe your rights and restrictions with
        respect to this document. Code Components extracted from this
        document must include Simplified BSD License text as described
        in Section 4.e of the Trust Legal Provisions and are provided
        without warranty as described in the Simplified BSD License.";

   revision 2018-05-17 {
       description "Initial Revision.";
       reference "draft-ietf-dmm-fpc-cpdp-10";
   }

       //Tunnel Information
        identity tunnel-type {
             description "Tunnel Type";
         }
         identity grev1 {
             base "fpcbase:tunnel-type";
             description "GRE v1";
```

```
        }
        identity grev2 {
            base "fpcbase:tunnel-type";
            description "GRE v2";
        }
        identity ipinip {
            base "fpcbase:tunnel-type";
            description "IP in IP";
        }
        identity gtpv1 {
            base "fpcbase:tunnel-type";
            description "GTP version 1 Tunnel";
        }
        identity gtpv2 {
            base "fpcbase:tunnel-type";
            description "GTP version 2 Tunnel";
        }

      grouping tunnel-value {
        container tunnel-info {
          leaf tunnel-local-address {
              type inet:ip-address;
              description "local tunnel address";
          }
          leaf tunnel-remote-address {
              type inet:ip-address;
              description "remote tunnel address";
          }
          leaf mtu-size {
              type uint32;
              description "MTU size";
          }
          leaf tunnel {
              type identityref {
                  base "fpcbase:tunnel-type";
              }
            description "tunnel type";
            }
            leaf payload-type {
                type enumeration {
                    enum ipv4 {
                        value 0;
                        description "IPv4";
                    }
                    enum ipv6 {
                        value 1;
                        description "IPv6";
                    }
```

```
                    enum dual {
                        value 2;
                        description "IPv4 and IPv6";
                    }
                }
                description "Payload Type";
            }
            leaf gre-key {
                type uint32;
                description "GRE_KEY";
            }
            container gtp-tunnel-info {
                leaf local-tunnel-identifier {
                    type uint32;
                    description "Tunnel Endpoint IDentifier (TEID)";
                }
                leaf remote-tunnel-identifier {
                    type uint32;
                    description "Tunnel Endpoint IDentifier (TEID)";
                }
                leaf sequence-numbers-enabled {
                    type boolean;
                    description "Sequence No. Enabled";
                }
                description "GTP Tunnel Information";
            }
            leaf ebi {
                type fpcbase:ebi-type;
                description "EPS Bearier Identifier";
            }
            leaf lbi {
                type fpcbase:ebi-type;
                description "Linked Bearier Identifier";
            }
            description "Tunnel Information";
          }
          description "Tunnel Value";
        }

      ////////////////////////////
      // DESCRIPTOR DEFINITIONS

        // From 3GPP TS 24.008 version 13.5.0 Release 13
          typedef packet-filter-direction {
              type enumeration {
                enum preRel7Tft {
                  value 0;
                  description "Pre-Release 7 TFT";
```

```
            }
            enum uplink {
              value 1;
              description "uplink";
            }
            enum downlink {
              value 2;
              description "downlink";
            }
            enum bidirectional {
              value 3;
              description "bi-direcitonal";
            }
          }
          description "Packet Filter Direction";
        }
        typedef component-type-id {
            type uint8 {
              range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 |"
              + " 80 | 81 | 96 | 112 | 128";
            }
            description "Specifies the Component Type";
        }
        grouping packet-filter {
          leaf direction {
              type fpcbase:packet-filter-direction;
              description "Filter Direction";
          }
          leaf identifier {
              type uint8 {
                range "1..15";
              }
              description "Filter Identifier";
          }
          leaf evaluation-precedence {
              type uint8;
              description "Evaluation Precedence";
          }
          list contents {
            key component-type-identifier;
            description "Filter Contents";
            leaf component-type-identifier {
                type fpcbase:component-type-id;
                description "Component Type";
            }
            choice value {
              leaf ipv4-local {
                  type inet:ipv4-address;
```

```
                    description "IPv4 Local Address";
                }
                leaf ipv6-prefix-local {
                    type inet:ipv6-prefix;
                    description "IPv6 Local Prefix";
                }
                leaf ipv4-ipv6-remote {
                    type inet:ip-address;
                    description "Ipv4 Ipv6 remote address";
                }
                leaf ipv6-prefix-remote {
                    type inet:ipv6-prefix;
                    description "IPv6 Remote Prefix";
                }
                leaf next-header {
                    type uint8;
                    description "Next Header";
                }
                leaf local-port {
                    type inet:port-number;
                    description "Local Port";
                }
                case local-port-range {
                  leaf local-port-lo {
                    type inet:port-number;
                    description "Local Port Min Value";
                  }
                  leaf local-port-hi {
                    type inet:port-number;
                    description "Local Port Max Value";
                  }
                }
                leaf remote-port {
                    type inet:port-number;
                    description "Remote Port";
                }
                case remote-port-range {
                  leaf remote-port-lo {
                    type inet:port-number;
                    description "Remote Por Min Value";
                  }
                  leaf remote-port-hi {
                    type inet:port-number;
                    description "Remote Port Max Value";
                  }
                }
                leaf ipsec-index {
                    type traffic-selectors:ipsec-spi;
```

```
                description "IPSec Index";
              }
              leaf traffic-class {
                  type inet:dscp;
                  description "Traffic Class";
              }
              case traffic-class-range {
                  leaf traffic-class-lo {
                    type inet:dscp;
                    description "Traffic Class Min Value";
                  }
                  leaf traffic-class-hi {
                    type inet:dscp;
                    description "Traffic Class Max Value";
                  }
              }
              leaf-list flow-label {
                  type inet:ipv6-flow-label;
                  description "Flow Label";
              }
              description "Component Value";
            }
          }
          description "Packet Filter";
        }

      grouping prefix-descriptor {
          leaf destination-ip {
            type inet:ip-prefix;
            description "Rule of destination IP";
          }
          leaf source-ip {
            type inet:ip-prefix;
            description "Rule of source IP";
          }
          description "Traffic descriptor based upon source/
            destination as IP prefixes";
      }

    grouping fpc-descriptor-value {
      choice descriptor-value {
        mandatory true;
        leaf all-traffic {
            type empty;
            description "admit any";
        }
        leaf no-traffic {
            type empty;
```

```
            description "deny any";
          }
          case prefix-descriptor {
            uses fpcbase:prefix-descriptor;
            description "IP Prefix descriptor";
          }
          case pmip-selector {
            uses traffic-selectors:traffic-selector;
            description "PMIP Selector";
          }
          container rfc5777-classifier-template {
                uses rfc5777:classifier;
                description "RFC 5777 Classifier";
          }
          container packet-filter {
                uses fpcbase:packet-filter;
                description "Packet Filter";
          }
          case tunnel-info {
            uses fpcbase:tunnel-value;
            description "Tunnel Descriptor (only
                considers source info)";
          }
          description "Descriptor Value";
        }
        description "FPC Descriptor Values";
      }

        // Next Hop Structures
        typedef fpc-service-path-id {
            type uint32 {
                range "0..33554431";
            }
            description "SERVICE_PATH_ID";
        }
        typedef fpc-mpls-label {
            type uint32 {
              range "0..1048575";
            }
            description "MPLS label";
        }
        typedef segment-id {
            type string {
                length "16";
            }
            description "SR Segement Identifier";
        }
        grouping fpc-nexthop {
```

```
            choice next-hop-value {
                leaf ip-address {
                    type inet:ip-address;
                    description "IP Value";
                }
                leaf mac-address {
                    type ytypes:mac-address;
                    description "MAC Address Value";
                }
                leaf service-path {
                    type fpcbase:fpc-service-path-id;
                    description "Service Path Value";
                }
                leaf mpls-path {
                    type fpcbase:fpc-mpls-label;
                    description "MPLS Value";
                }
                leaf nsh {
                    type string {
                        length "16";
                    }
                    description "Network Service Header";
                }
                leaf interface {
                    type uint16;
                    description "If (interface) Value";
                }
                leaf segment-identifier {
                    type fpcbase:segment-id;
                    description "Segment Id";
                }
                leaf-list mpls-label-stack {
                    type fpcbase:fpc-mpls-label;
                    description "MPLS Stack";
                }
                leaf-list mpls-sr-stack {
                    type fpcbase:fpc-mpls-label;
                    description "MPLS SR Stack";
                }
                leaf-list srv6-stack {
                    type fpcbase:segment-id;
                    description "Segment Id";
                }
                case tunnel-info {
                    uses fpcbase:tunnel-value;
                    description "Tunnel Descriptor (only
                    considers source info)";
                }
```

```
            description "Value";
          }
          description "Nexthop Value";
        }

        ////////////////////////////
        // PMIP Integration         //
          typedef pmip-commandset {
                type bits {
                    bit assign-ip {
                      position 0;
                      description "Assign IP";
                    }
                    bit assign-dpn {
                      position 1;
                      description "Assign DPN";
                    }
                    bit session {
                      position 2;
                      description "Session Level";
                    }
                    bit uplink {
                      position 3;
                      description "Uplink";
                    }
                    bit downlink {
                      position 4;
                      description "Downlink";
                    }
                }
                description "PMIP Instructions";
          }
        ////////////////////////////
        // 3GPP Integration         //

          // Type Defs
          typedef fpc-qos-class-identifier {
              type uint8 {
                  range "1..9";
              }
              description "QoS Class Identifier (QCI)";
          }
          typedef ebi-type {
            type uint8 {
              range "0..15";
            }
            description "EUTRAN Bearere Identifier (EBI) Type";
          }
```

```
        typedef imsi-type {
            type uint64;
            description
                "International Mobile Subscriber Identity (IMSI)
                  Value Type";
        }
        // Instructions
        typedef threegpp-instr {
          type bits {
            bit assign-ip {
              position 0;
              description "Assign IP Address/Prefix";
            }
            bit assign-fteid-ip {
              position 1;
              description "Assign FTEID-IP";
            }
            bit assign-fteid-teid {
              position 2;
              description "Assign FTEID-TEID";
            }
            bit session {
              position 3;
              description "Commands apply to the Session Level";
            }
            bit uplink {
              position 4;
              description "Commands apply to the Uplink";
            }
            bit downlink {
              position 5;
              description "Commands apply to the Downlink";
            }
            bit assign-dpn {
              position 6;
              description "Assign DPN";
            }
          }
          description "Instruction Set for 3GPP R11";
        }

      ////////////////////////////
      // ACTION VALUE AUGMENTS
      grouping fpc-action-value {
          choice action-value {
              mandatory true;
              leaf drop {
                  type empty;
```

```
                description "Drop Traffic";
            }
            container rewrite {
                choice rewrite-value {
                    case prefix-descriptor {
                        uses fpcbase:prefix-descriptor;
                        description "IP Prefix descriptor";
                    }
                    case pmip-selector {
                        uses traffic-selectors:traffic-selector;
                        description "PMIP Selector";
                    }
                    container rfc5777-classifier-template {
                        uses rfc5777:classifier;
                        description "RFC 5777 Classifier";
                    }
                    description "Rewrite Choice";
                }
                description "Rewrite/NAT value";
            }
            container copy-forward-nexthop {
                    uses fpcbase:fpc-nexthop;
                    description "Copy Forward Value";
            }
            container nexthop {
                    uses fpcbase:fpc-nexthop;
                    description "NextHop Value";
            }
            case qos {
                leaf trafficclass {
                    type inet:dscp;
                    description "Traffic Class";
                }
                uses pmipqos:qosattribute;
                leaf qci {
                    type fpcbase:fpc-qos-class-identifier;
                    description "QCI";
                }
                leaf ue-agg-max-bitrate {
                    type uint32;
                    description "UE Aggregate Max Bitrate";
                }
                leaf apn-ambr {
                    type uint32;
                    description
                     "Access Point Name Aggregate Max Bit Rate";
                }
                description "QoS Attributes";
```

```
                }
                description "Action Value";
            }
            description "FPC Action Value";
        }

      // Instructions
      grouping instructions {
        container command-set {
          choice instr-type {
            leaf instr-3gpp-mob {
                type fpcbase:threegpp-instr;
                description "3GPP GTP Mobility Instructions";
            }
            leaf instr-pmip {
                type pmip-commandset;
                description "PMIP Instructions";
            }
            description "Instruction Value Choice";
          }
          description "Instructions";
        }
        description "Instructions Value";
      }
    }
    <CODE ENDS>
```

A.3. PMIP QoS Model

This module defines the base protocol elements specified in this document.

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-pmip-qos@2018-05-17.yang"
module ietf-pmip-qos {
    yang-version 1.1;

    namespace
      "urn:ietf:params:xml:ns:yang:ietf-pmip-qos";

    prefix "qos-pmip";

    import ietf-inet-types {
      prefix inet;
      revision-date 2013-07-15;
    }
    import ietf-trafficselector-types { prefix traffic-selectors;
```

```
           revision-date 2018-05-17; }

       organization "IETF Distributed Mobility Management (DMM)
         Working Group";

       contact
          "WG Web:   <http://tools.ietf.org/wg/netmod/>
           WG List:  <mailto:netmod@ietf.org>

           WG Chair: Dapeng Liu
                     <mailto:maxpassion@gmail.com>

           WG Chair: Sri Gundavelli
                     <mailto:sgundave@cisco.com>

           Editor:   Satoru Matsushima
                     <mailto:satoru.matsushima@g.softbank.co.jp>

           Editor:   Lyle Bertz
                     <mailto:lylebe551144@gmail.com>";

       description
         "This module contains a collection of YANG definitions for
        quality of service paramaters used in Proxy Mobile IPv6.

        Copyright (c) 2016 IETF Trust and the persons identified as the
        document authors. All rights reserved.

        This document is subject to BCP 78 and the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info) in effect on the date of
        publication of this document. Please review these documents
        carefully, as they describe your rights and restrictions with
        respect to this document. Code Components extracted from this
        document must include Simplified BSD License text as described
        in Section 4.e of the Trust Legal Provisions and are provided
        without warranty as described in the Simplified BSD License.";

     revision 2018-05-17 {
       description "Initial Revision.";
       reference "RFC 6088: Traffic Selectors for Flow Bindings";
     }

       // Type Definitions

       // QoS Option Field Type Definitions
     typedef sr-id {
       type uint8;
```

```
        description
         "An 8-bit unsigned integer used for identifying the QoS
          Service Request.";
      }

      typedef traffic-class {
        type inet:dscp;
        description
          "Traffic Class consists of a 6-bit DSCP field followed by a
           2-bit reserved field.";
       reference
           "RFC 3289: Management Information Base for the
              Differentiated Services Architecture
            RFC 2474: Definition of the Differentiated Services Field
                      (DS Field) in the IPv4 and IPv6 Headers
            RFC 2780: IANA Allocation Guidelines For Values In
                      the Internet Protocol and Related Headers";
      }

      typedef operational-code {
        type enumeration {
          enum RESPONSE {
            value 0;
            description "Response to a QoS request";
          }
          enum ALLOCATE {
            value 1;
            description "Request to allocate QoS resources";
          }
          enum DE-ALLOCATE {
            value 2;
            description "Request to de-Allocate QoS resources";
          }
          enum MODIFY {
            value 3;
            description "Request to modify QoS parameters for a
                previously negotiated QoS Service Request";
          }
          enum QUERY {
            value 4;
            description "Query to list the previously negotiated QoS
                Service Requests that are still active";
          }
          enum NEGOTIATE {
            value 5;
            description "Response to a QoS Service Request with a
              counter QoS proposal";
          }
```

```
        }
        description
         "The type of QoS request. Reserved values:   (6) to (255)
                  Currently not used.  Receiver MUST ignore the option
                  received with any value in this range.";
       }

     //Value definitions
     typedef Per-MN-Agg-Max-DL-Bit-Rate-Value {
        type uint32;
        description
            "The aggregate maximum downlink bit rate that is
            requested/allocated for all the mobile node's IP flows.
            The measurement units are bits per second.";
     }

      typedef Per-MN-Agg-Max-UL-Bit-Rate-Value {
        type uint32;
        description
          "The aggregate maximum uplink bit rate that is
              requested/allocated for the mobile node's IP flows.  The
              measurement units are bits per second.";
      }

      // Generic Structure for the uplink and downlink
      grouping Per-Session-Agg-Max-Bit-Rate-Value {
        leaf max-rate {
          type uint32;
          mandatory true;
          description
          "The aggregate maximum bit rate that is requested/allocated
       for all the IP flows associated with that mobility session.
       The measurement units are bits per second.";
       }
       leaf service-flag {
        type boolean;
        mandatory true;
        description
         "This flag is used for extending the scope of the
          target flows for Per-Session-Agg-Max-UL/DL-Bit-Rate
          from(UL)/to(DL) the mobile node's other mobility sessions
          sharing the same Service Identifier.";
        reference
          "RFC 5149 - Service Selection mobility option";
       }
       leaf exclude-flag {
         type boolean;
         mandatory true;
```

```
          description
           "This flag is used to request that the uplink/downlink
          flows for which the network is providing
               Guaranteed-Bit-Rate service be excluded from the
               target IP flows for which
               Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.";
        }
       description "Per-Session-Agg-Max-Bit-Rate Value";
      }

      grouping Allocation-Retention-Priority-Value {
        leaf priority-level {
          type uint8 {
            range "0..15";
          }
          mandatory true;
          description
          "This is a 4-bit unsigned integer value. It is used to decide
           whether a mobility session establishment or modification
           request can be accepted; this is typically used for
           admission control of Guaranteed Bit Rate traffic in case of
           resource limitations.";
        }
        leaf preemption-capability {
          type enumeration {
           enum enabled {
             value 0;
             description "enabled";
           }
           enum disabled {
             value 1;
             description "disabled";
           }
           enum reserved1 {
             value 2;
             description "reserved1";
           }
           enum reserved2 {
             value 3;
             description "reserved2";
           }
          }
          mandatory true;
          description
          "This is a 2-bit unsigned integer value.  It defines whether a
           service data flow can get resources tha were already
           assigned to another service data flow with a lower priority
           level.";
```

```
        }
        leaf preemption-vulnerability {
          type enumeration {
           enum enabled {
             value 0;
             description "enabled";
           }
           enum disabled {
             value 1;
             description "disabled";
           }
           enum reserved1 {
             value 2;
             description "reserved1";
           }
           enum reserved2 {
             value 3;
             description "reserved2";
           }
          }
          mandatory true;
          description
          "This is a 2-bit unsigned integer value.  It defines whether a
            service data flow can lose the resources assigned to it in
            order to admit a service data flow with a higher priority
            level.";
        }
       description "Allocation-Retention-Priority Value";
      }

      typedef Aggregate-Max-DL-Bit-Rate-Value {
         type uint32;
         description
           "The aggregate maximum downlink bit rate that is
            requested/allocated for downlink IP flows.  The measurement
            units are bits per second.";
      }

       typedef Aggregate-Max-UL-Bit-Rate-Value {
         type uint32;
         description
           "The aggregate maximum downlink bit rate that is
            requested/allocated for downlink IP flows.  The measurement
            units are bits per second.";
       }

       typedef Guaranteed-DL-Bit-Rate-Value {
         type uint32;
```

```
        description
        "The guaranteed bandwidth in bits per second for downlink
          IP flows.  The measurement units are bits per second.";
      }

      typedef Guaranteed-UL-Bit-Rate-Value {
        type uint32;
        description
          "The guaranteed bandwidth in bits per second for uplink
           IP flows.  The measurement units are bits per second.";
      }

      grouping QoS-Vendor-Specific-Attribute-Value-Base {
        leaf vendorid {
          type uint32;
          mandatory true;
          description
           "The Vendor ID is the SMI (Structure of Management
            Information) Network Management Private Enterprise Code of
            the IANA-maintained 'Private Enterprise Numbers'
            registry.";
          reference
            "'PRIVATE ENTERPRISE NUMBERS', SMI Network Management
              Private Enterprise Codes, April 2014,
               <http://www.iana.org/assignments/enterprise-numbers>";
        }
        leaf subtype {
          type uint8;
          mandatory true;
          description
            "An 8-bit field indicating the type of vendor-specific
             information carried in the option.  The namespace for this
             sub-type is managed by the vendor identified by the
             Vendor ID field.";
        }
        description
          "QoS Vendor-Specific Attribute.";
      }

      //Primary Structures (groupings)
      grouping qosattribute {
          leaf per-mn-agg-max-dl {
              type qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value;
              description "Per-MN-Agg-Max-DL-Bit-Rate Value";
          }
          leaf per-mn-agg-max-ul {
              type qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value;
              description "Per-MN-Agg-Max-UL-Bit-Rate Value";
```

```
            }
            container per-session-agg-max-dl {
                uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                description "Per-Session-Agg-Max-Bit-Rate Value";
            }
            container per-session-agg-max-ul {
                uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                description "Per-Session-Agg-Max-Bit-Rate Value";
            }
            uses qos-pmip:Allocation-Retention-Priority-Value;
            leaf agg-max-dl {
                type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
                description "Aggregate-Max-DL-Bit-Rate Value";
            }
            leaf agg-max-ul {
                type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
                description "Aggregate-Max-UL-Bit-Rate Value";
            }
            leaf gbr-dl {
                type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
                description "Guaranteed-DL-Bit-Rate Value";
            }
            leaf gbr-ul {
                type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
                description "Guaranteed-UL-Bit-Rate Value";
            }
            description "PMIP QoS Attributes. Note Vendor option
            is not a part of this grouping";
        }

        grouping qosoption {
            leaf srid {
                type sr-id;
                mandatory true;
                description "Service Request Identifier";
            }
            leaf trafficclass {
                type traffic-class;
                mandatory true;
                description "Traffic Class";
            }
            leaf operationcode {
                type operational-code;
                mandatory true;
                description "Operation Code";
            }
            uses qos-pmip:qosattribute;
            uses qos-pmip:QoS-Vendor-Specific-Attribute-Value-Base;
```

```
            container traffic-selector {
                uses traffic-selectors:traffic-selector;
                description "traffic selector";
            }
            description "PMIP QoS Option";
        }
    }
    <CODE ENDS>
```

A.4. Traffic Selectors YANG Model

This module defines traffic selector types commonly used in Proxy Mobile IP (PMIP).

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-trafficselector-types@2018-05-17.yang"
module ietf-trafficselector-types {
 yang-version 1.1;

 namespace
 "urn:ietf:params:xml:ns:yang:ietf-trafficselector-types";

 prefix "traffic-selectors";

 import ietf-inet-types {
   prefix inet;
   revision-date 2013-07-15;
 }

 organization "IETF Distributed Mobility Management (DMM)
 Working Group";

 contact
 "WG Web: <http://tools.ietf.org/wg/netmod/>
 WG List: <mailto:netmod@ietf.org>

 WG Chair: Dapeng Liu
 <mailto:maxpassion@gmail.com>

 WG Chair: Sri Gundavelli
 <mailto:sgundave@cisco.com>

 Editor: Satoru Matsushima
 <mailto:satoru.matsushima@g.softbank.co.jp>

 Editor: Lyle Bertz
 <mailto:lylebe551144@gmail.com>";
```

```
    description
    "This module contains a collection of YANG definitions for
    traffic selectors for flow bindings.

    Copyright (c) 2016 IETF Trust and the persons identified as the
    document authors. All rights reserved.

    This document is subject to BCP 78 and the IETF Trust's Legal
    Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info) in effect on the date of
    publication of this document. Please review these documents
    carefully, as they describe your rights and restrictions with
    respect to this document. Code Components extracted from this
    document must include Simplified BSD License text as described
    in Section 4.e of the Trust Legal Provisions and are provided
    without warranty as described in the Simplified BSD License.";

     revision 2018-05-17 {
       description
        "Initial Revision.";
       reference
        "RFC 6088: Traffic Selectors for Flow Bindings";
     }

    // Identities
      identity traffic-selector-format {
        description
        "The base type for Traffic-Selector Formats";
      }

      identity ipv4-binary-selector-format {
        base traffic-selector-format;
        description
          "IPv4 Binary Traffic Selector Format";
      }

      identity ipv6-binary-selector-format {
        base traffic-selector-format;
        description
          "IPv6 Binary Traffic Selector Format";
      }

      // Type definitions and groupings
      typedef ipsec-spi {
        type uint32;
        description
         "The first 32-bit IPsec Security Parameter Index (SPI)
         value on data. This field is defined in [RFC4303].";
```

```
          reference
          "RFC 4303: IP Encapsulating Security
          Payload (ESP)";
        }

        grouping traffic-selector-base {
          description "A grouping of the commen leaves between the
            v4 and v6 Traffic Selectors";
          container ipsec-spi-range {
            presence "Enables setting ipsec spi range";
            description
            "Inclusive range representing IPSec Security Parameter
            Indices to be used. When only start-spi is present, it
            represents a single spi.";
        leaf start-spi {
            type ipsec-spi;
            mandatory true;
            description
              "The first 32-bit IPsec SPI value on data.";
            }
        leaf end-spi {
              type ipsec-spi;
              must ". >= ../start-spi" {
                error-message
                  "The end-spi must be greater than or equal
                   to start-spi";
            }
           description
             "If more than one contiguous SPI value needs to be matched,
             then this field indicates the end value of a range.";
            }
         }
         container source-port-range {
           presence "Enables setting source port range";
           description
            "Inclusive range representing source ports to be used.
             When only start-port is present, it represents a single
          port. These value(s) are from the range of port numbers
             defined by IANA (http://www.iana.org).";
           leaf start-port {
              type inet:port-number;
              mandatory true;
              description
              "The first 16-bit source port number to be matched";
           }
           leaf end-port {
              type inet:port-number;
              must ". >= ../start-port" {
```

```
            error-message
             "The end-port must be greater than or equal to start-port";
           }
           description
            "The last 16-bit source port number to be matched";
          }
        }
        container destination-port-range {
          presence "Enables setting destination port range";
          description
           "Inclusive range representing destination ports to be used.
           When only start-port is present, it represents a single
           port.";
            leaf start-port {
              type inet:port-number;
              mandatory true;
              description
              "The first 16-bit destination port number to be matched";
           }
           leaf end-port {
             type inet:port-number;
             must ". >= ../start-port" {
             error-message
               "The end-port must be greater than or equal to
              start-port";
           }
           description
       "The last 16-bit destination port number to be matched";
        }
      }
    }

    grouping ipv4-binary-traffic-selector {
      container source-address-range-v4 {
         presence "Enables setting source IPv4 address range";
         description
          "Inclusive range representing IPv4 addresses to be used. When
          only start-address is present, it represents a single
          address.";
         leaf start-address {
           type inet:ipv4-address;
           mandatory true;
          description
           "The first source address to be matched";
         }
         leaf end-address {
           type inet:ipv4-address;
           description
```

```
            "The last source address to be matched";
          }
        }
        container destination-address-range-v4 {
           presence "Enables setting destination IPv4 address range";
           description
             "Inclusive range representing IPv4 addresses to be used.
             When only start-address is present, it represents a
             single address.";
           leaf start-address {
             type inet:ipv4-address;
             mandatory true;
             description
              "The first destination address to be matched";
           }
           leaf end-address {
             type inet:ipv4-address;
             description
              "The last destination address to be matched";
           }
        }
        container ds-range {
           presence "Enables setting dscp range";
           description
            "Inclusive range representing DiffServ Codepoints to be used.
            When only start-ds is present, it represents a single
            Codepoint.";
           leaf start-ds {
             type inet:dscp;
             mandatory true;
             description
              "The first differential service value to be matched";
         }
         leaf end-ds {
           type inet:dscp;
           must ". >= ../start-ds" {
             error-message
               "The end-ds must be greater than or equal to start-ds";
           }
           description
             "The last differential service value to be matched";
         }
        }
        container protocol-range {
          presence "Enables setting protocol range";
          description
            "Inclusive range representing IP protocol(s) to be used. When
             only start-protocol is present, it represents a single
```

```
          protocol.";
        leaf start-protocol {
          type uint8;
          mandatory true;
          description
            "The first 8-bit protocol value to be matched.";
        }
        leaf end-protocol {
          type uint8;
          must ". >= ../start-protocol" {
            error-message
              "The end-protocol must be greater than or equal to
             start-protocol";
          }
        description
          "The last 8-bit protocol value to be matched.";
        }
      }
      description "ipv4 binary traffic selector";
    }
     grouping ipv6-binary-traffic-selector {
      container source-address-range-v6 {
        presence "Enables setting source IPv6 address range";
         description
          "Inclusive range representing IPv6 addresses to be used.
          When only start-address is present, it represents a
          single address.";
         leaf start-address {
           type inet:ipv6-address;
           mandatory true;
           description
           "The first source address, from the
           range of 128-bit IPv6 addresses to be matched";
         }
         leaf end-address {
           type inet:ipv6-address;
           description
               "The last source address, from the
               range of 128-bit IPv6 addresses to be matched";
         }
      }
      container destination-address-range-v6 {
        presence "Enables setting destination IPv6 address range";
        description
          "Inclusive range representing IPv6 addresses to be used.
           When only start-address is present, it represents a
           single address.";
        leaf start-address {
```

```
          type inet:ipv6-address;
          mandatory true;
          description
              "The first destination address, from the
              range of 128-bit IPv6 addresses to be matched";
        }
        leaf end-address {
          type inet:ipv6-address;
          description
              "The last destination address, from the
              range of 128-bit IPv6 addresses to be matched";
      }
    }
    container flow-label-range {
      presence "Enables setting Flow Label range";
      description
        "Inclusive range representing IPv4 addresses to be used. When
         only start-flow-label is present, it represents a single
         flow label.";
      leaf start-flow-label {
        type inet:ipv6-flow-label;
        description
          "The first flow label value to be matched";
      }
      leaf end-flow-label {
        type inet:ipv6-flow-label;
        must ". >= ../start-flow-label" {
          error-message
            "The end-flow-lable must be greater than or equal to
             start-flow-label";
        }
        description
           "The first flow label value to be matched";
      }
     }
    container traffic-class-range {
      presence "Enables setting the traffic class range";
      description
       "Inclusive range representing IPv4 addresses to be used. When
        only start-traffic-class is present, it represents a single
        traffic class.";
      leaf start-traffic-class {
        type inet:dscp;
        description
         "The first traffic class value to be matched";
        reference
         "RFC 3260: New Terminology and Clarifications for Diffserv
          RFC 3168: The Addition of Explicit Congestion Notification
```

```
          (ECN) to IP";
      }
      leaf end-traffic-class {
        type inet:dscp;
        must ". >= ../start-traffic-class" {
          error-message
            "The end-traffic-class must be greater than or equal to
             start-traffic-class";
        }
        description
          "The last traffic class value to be matched";
      }
    }
    container next-header-range {
      presence "Enables setting Next Header range";
      description
       "Inclusive range representing Next Headers to be used. When
        only start-next-header is present, it represents a
        single Next Header.";
      leaf start-next-header {
        type uint8;
        description
         "The first 8-bit next header value to be matched.";
      }
      leaf end-next-header {
        type uint8;
        must ". >= ../start-next-header" {
          error-message
            "The end-next-header must be greater than or equal to
            start-next-header";
        }
        description
          "The last 8-bit next header value to be matched.";
      }
    }
    description "ipv6 binary traffic selector";
  }

    grouping traffic-selector {
      leaf ts-format {
         type identityref {
           base traffic-selector-format;
         }
         description "Traffic Selector Format";
      }
      uses traffic-selectors:traffic-selector-base;
      uses traffic-selectors:ipv4-binary-traffic-selector;
      uses traffic-selectors:ipv6-binary-traffic-selector;
```

```
       description
        "The traffic selector includes the parameters used to match
          packets for a specific flow binding.";
       reference
        "RFC 6089: Flow Bindings in Mobile IPv6 and Network
          Mobility (NEMO) Basic Support";
     }
   }
   <CODE ENDS>
```

A.5. RFC 5777 Classifier YANG Model

This module defines the RFC 5777 Classifer.

This module references [RFC5777].

```
<CODE BEGINS> file "ietf-diam-trafficclassifier@2018-05-17.yang"
module ietf-diam-trafficclassifier {
 yang-version 1.1;

 namespace
 "urn:ietf:params:xml:ns:yang:ietf-diam-trafficclassifier";

 prefix "diamclassifier";

 import ietf-inet-types {
   prefix inet;
   revision-date 2013-07-15;
 }
 import ietf-yang-types { prefix yang-types; }

 organization "IETF Distributed Mobility Management (DMM)
 Working Group";

 contact
 "WG Web: <http://tools.ietf.org/wg/netmod/>
 WG List: <mailto:netmod@ietf.org>

 WG Chair: Dapeng Liu
 <mailto:maxpassion@gmail.com>

 WG Chair: Sri Gundavelli
 <mailto:sgundave@cisco.com>

 Editor: Satoru Matsushima
 <mailto:satoru.matsushima@g.softbank.co.jp>

 Editor: Lyle Bertz
```

```
    <mailto:lylebe551144@gmail.com>";

    description
    "This module contains a collection of YANG definitions for
    traffic classification and QoS Attributes for Diameter.

    Copyright (c) 2018 IETF Trust and the persons identified as the
    document authors. All rights reserved.

    This document is subject to BCP 78 and the IETF Trust's Legal
    Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info) in effect on the date of
    publication of this document. Please review these documents
    carefully, as they describe your rights and restrictions with
    respect to this document. Code Components extracted from this
    document must include Simplified BSD License text as described
    in Section 4.e of the Trust Legal Provisions and are provided
    without warranty as described in the Simplified BSD License.";

     revision 2018-05-17 {
       description
           "Initial";
       reference
       "RFC 5777: Traffic Classification and Quality of Service (QoS)
           Attributes for Diameter";
     }

       typedef eui64-address-type {
           type string {
               length "6";
           }
           description
               "specifies a single layer 2 address in EUI-64 format.
                The value is an 8-octet encoding of the address as
                it would appear in the frame header.";
       }
       typedef direction-type {
           type enumeration {
               enum IN {
                   value 0;
                   description
                       "Applies to flows from the managed terminal.";
               }
               enum OUT {
                   value 1;
                   description
                    "Applies to flows to the managed terminal.";
               }
```

```
                enum BOTH {
                    value 2;
                    description
                        "Applies to flows both to and from the managed
                            terminal.";
                }
            }
            description
                "Specifies in which direction to apply the classifier.";
        }
        typedef negated-flag-type {
            type enumeration {
                enum False { value 0;
                    description "false"; }
                enum True { value 1;
                    description "True"; }
            }
            description
                "When set to True, the meaning of the match is
                 inverted and the classifier will match addresses
                 other than those specified by the From-Spec or
                 To-Spec AVP.

                 Note that the negation does not impact the port
                    comparisons.";
        }
        grouping index {
            leaf index {
                type uint16;
                mandatory true;
                description "Identifier used for referencing";
            }
            description "Index Value";
        }
        grouping to-from-spec-value {
            leaf-list ip-address {
                type inet:ip-address;
                description "IP address";
            }
            list ip-address-range {
                key index;
                uses diamclassifier:index;
                leaf ip-address-start {
                    type inet:ip-address;
                    description "IP Address Start";
                }
                leaf ip-address-end {
                    type inet:ip-address;
```

```
                description "IP Address End";
            }
            description "IP Address Range";
        }
        leaf-list ip-address-mask {
            type inet:ip-prefix;
            description "IP Address Mask";
        }
        leaf-list mac-address {
            type yang-types:mac-address;
            description "MAC address";
        }
        list mac-address-mask {
            key mac-address;
            leaf mac-address {
                type yang-types:mac-address;
                mandatory true;
                description "MAC address";
            }
            leaf macaddress-mask-pattern {
                type yang-types:mac-address;
                mandatory true;
                description
                 "The value specifies the bit positions of a
                  MAC address that are taken for matching.";
            }
            description "MAC Address Mask";
        }
        leaf-list eui64-address {
            type diamclassifier:eui64-address-type;
            description "EUI64 Address";
        }
        list eui64-address-mask {
            key eui64-address;
            leaf eui64-address {
                type diamclassifier:eui64-address-type;
                mandatory true;
                description "eui64 address";
            }
            leaf eui64-address-mask-pattern {
                type diamclassifier:eui64-address-type;
                mandatory true;
                description
                 "The value is 8 octets specifying the bit
                  positions of a EUI64 address that are taken
                 for matching.";
            }
            description "EUI64 Address Mask";
```

```
            }
            leaf-list port {
                type inet:port-number;
                description "Port Number";
            }
            list port-range {
                key index;
                uses diamclassifier:index;
                leaf ip-address-start {
                    type inet:port-number;
                    description "Port Start";
                }
                leaf ip-address-end {
                    type inet:port-number;
                    description "Port End";
                }
                description "Port Range";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            leaf use-assigned-address {
                type boolean;
                description "Use Assigned Address";
            }
            description
                "Basic traffic description value";
        }

        grouping option-type-group {
            leaf option-type {
                type uint8;
                mandatory true;
                description "Option Type";
            }
            leaf-list ip-option-value {
                type string;
                description "Option Value";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            description "Common X Option Pattern";
        }
        typedef vlan-id {
            type uint32 {
```

```
                range "0..4095";
            }
            description "VLAN ID";
        }

      grouping classifier {
        leaf protocol {
            type uint8;
            description "Protocol";
        }
        leaf direction {
            type diamclassifier:direction-type;
            description "Direction";
        }
        list from-spec {
            key index;
            uses diamclassifier:index;
            uses diamclassifier:to-from-spec-value;
            description "from specification";
        }
        list to-spec {
            key index;
            uses diamclassifier:index;
            uses diamclassifier:to-from-spec-value;
            description "to specification";
        }
        leaf-list disffserv-code-point {
            type inet:dscp;
            description "DSCP";
        }
        leaf fragmentation-flag {
        type enumeration {
                enum DF {
                    value 0;
                    description "Don’t Fragment";
                }
                enum MF {
                    value 1;
                    description "More Fragments";
                }
            }
            description "Fragmenttation Flag";
        }
        list ip-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "IP Option Value";
        }
```

```
        list tcp-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "TCP Option Value";
        }
        list tcp-flag {
            key tcp-flag-type;
            leaf tcp-flag-type {
                type uint32;
                mandatory true;
                description "TCP Flag Type";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            description "TCP Flags";
        }
        list icmp-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "ICMP Option Value";
        }
        list eth-option {
            key index;
            uses diamclassifier:index;
            container eth-proto-type {
                leaf-list eth-ether-type {
                    type string {
                        length "2";
                    }
                    description "value of ethertype field";
                }
                leaf-list eth-sap {
                    type string {
                        length "2";
                    }
                    description "802.2 SAP";
                }
                description "Ether Proto Type";
            }
            list vlan-id-range {
                key index;
                uses diamclassifier:index;
                leaf-list s-vlan-id-start {
                    type diamclassifier:vlan-id;
                    description "S-VID  VLAN ID Start";
                }
```

```
                leaf-list s-vlan-id-end {
                    type diamclassifier:vlan-id;
                    description "S-VID VLAN ID End";
                }
                leaf-list c-vlan-id-start {
                    type diamclassifier:vlan-id;
                    description "C-VID  VLAN ID Start";
                }
                leaf-list c-vlan-id-end {
                    type diamclassifier:vlan-id;
                    description "C-VID  VLAN ID End";
                }
                description "VLAN ID Range";
            }
            list user-priority-range {
                key index;
                uses diamclassifier:index;
                leaf-list low-user-priority {
                    type uint32 {
                        range "0..7";
                    }
                    description "Low User Priority";
                }
                leaf-list high-user-priority {
                    type uint32 {
                        range "0..7";
                    }
                    description "High User Priority";
                }
                description "User priority range";
            }
            description "Ether Option";
        }
        description "RFC 5777 Classifier";
      }
    }
    <CODE ENDS>
```

Appendix B. FPC YANG Tree Structure

This section only shows the structure for FPC YANG model. NOTE, it does NOT show the settings, Action values or Descriptor Value.

```
descriptor_value:
+--rw (descriptor-value)
  +--:(all-traffic)
  |  +--rw all-traffic?                    empty
  +--:(no-traffic)
```

```
|  +--rw no-traffic?                    empty
+--:(prefix-descriptor)
|  +--rw destination-ip?                inet:ip-prefix
|  +--rw source-ip?                     inet:ip-prefix
+--:(pmip-selector)
|  +--rw ts-format?                     identityref
|  +--rw ipsec-spi-range!
|  |  +--rw start-spi    ipsec-spi
|  |  +--rw end-spi?     ipsec-spi
|  +--rw source-port-range!
|  |  +--rw start-port    inet:port-number
|  |  +--rw end-port?     inet:port-number
|  +--rw destination-port-range!
|  |  +--rw start-port    inet:port-number
|  |  +--rw end-port?     inet:port-number
|  +--rw source-address-range-v4!
|  |  +--rw start-address    inet:ipv4-address
|  |  +--rw end-address?     inet:ipv4-address
|  +--rw destination-address-range-v4!
|  |  +--rw start-address    inet:ipv4-address
|  |  +--rw end-address?     inet:ipv4-address
|  +--rw ds-range!
|  |  +--rw start-ds    inet:dscp
|  |  +--rw end-ds?     inet:dscp
|  +--rw protocol-range!
|  |  +--rw start-protocol    uint8
|  |  +--rw end-protocol?     uint8
|  +--rw source-address-range-v6!
|  |  +--rw start-address    inet:ipv6-address
|  |  +--rw end-address?     inet:ipv6-address
|  +--rw destination-address-range-v6!
|  |  +--rw start-address    inet:ipv6-address
|  |  +--rw end-address?     inet:ipv6-address
|  +--rw flow-label-range!
|  |  +--rw start-flow-label?   inet:ipv6-flow-label
|  |  +--rw end-flow-label?     inet:ipv6-flow-label
|  +--rw traffic-class-range!
|  |  +--rw start-traffic-class?   inet:dscp
|  |  +--rw end-traffic-class?     inet:dscp
|  +--rw next-header-range!
|     +--rw start-next-header?   uint8
|     +--rw end-next-header?     uint8
+--:(rfc5777-classifier-template)
|  +--rw rfc5777-classifier-template
|     +--rw protocol?               uint8
|     +--rw direction?              diamclassifier:direction-type
|     +--rw from-spec* [index]
|     |  +--rw index                   uint16
```

```
|     |  +--rw ip-address*             inet:ip-address
|     |  +--rw ip-address-range* [index]
|     |  |  +--rw index               uint16
|     |  |  +--rw ip-address-start?   inet:ip-address
|     |  |  +--rw ip-address-end?     inet:ip-address
|     |  +--rw ip-address-mask*        inet:ip-prefix
|     |  +--rw mac-address*            yang-types:mac-address
|     |  +--rw mac-address-mask* [mac-address]
|     |  |  +--rw mac-address             yang-types:mac-address
|     |  |  +--rw macaddress-mask-pattern  yang-types:mac-address
|     |  +--rw eui64-address*
|                    diamclassifier:eui64-address-type
|     |  +--rw eui64-address-mask* [eui64-address]
|     |  |  +--rw eui64-address
|                    diamclassifier:eui64-address-type
|     |  |  +--rw eui64-address-mask-pattern
|                    diamclassifier:eui64-address-type
|     |  +--rw port*                   inet:port-number
|     |  +--rw port-range* [index]
|     |  |  +--rw index               uint16
|     |  |  +--rw ip-address-start?   inet:port-number
|     |  |  +--rw ip-address-end?     inet:port-number
|     |  +--rw negated?
|                    diamclassifier:negated-flag-type
|     |  +--rw use-assigned-address?   boolean
|     +--rw to-spec* [index]
|     |  +--rw index                   uint16
|     |  +--rw ip-address*             inet:ip-address
|     |  +--rw ip-address-range* [index]
|     |  |  +--rw index               uint16
|     |  |  +--rw ip-address-start?   inet:ip-address
|     |  |  +--rw ip-address-end?     inet:ip-address
|     |  +--rw ip-address-mask*        inet:ip-prefix
|     |  +--rw mac-address*            yang-types:mac-address
|     |  +--rw mac-address-mask* [mac-address]
|     |  |  +--rw mac-address               yang-types:mac-address
|     |  |  +--rw macaddress-mask-pattern   yang-types:mac-address
|     |  +--rw eui64-address*
|                    diamclassifier:eui64-address-type
|     |  +--rw eui64-address-mask* [eui64-address]
|     |  |  +--rw eui64-address
|                    diamclassifier:eui64-address-type
|     |  |  +--rw eui64-address-mask-pattern
|                    diamclassifier:eui64-address-type
|     |  +--rw port*                   inet:port-number
|     |  +--rw port-range* [index]
|     |  |  +--rw index               uint16
|     |  |  +--rw ip-address-start?   inet:port-number
```

```
|     |  |  +--rw ip-address-end?     inet:port-number
|     |  +--rw negated?
                  diamclassifier:negated-flag-type
|     |  +--rw use-assigned-address?   boolean
|     +--rw disffserv-code-point*   inet:dscp
|     +--rw fragmentation-flag?     enumeration
|     +--rw ip-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw tcp-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw tcp-flag* [tcp-flag-type]
|     |  +--rw tcp-flag-type    uint32
|     |  +--rw negated?         diamclassifier:negated-flag-type
|     +--rw icmp-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw eth-option* [index]
|        +--rw index                  uint16
|        +--rw eth-proto-type
|        |  +--rw eth-ether-type*   string
|        |  +--rw eth-sap*          string
|        +--rw vlan-id-range* [index]
|        |  +--rw index              uint16
|        |  +--rw s-vlan-id-start*   diamclassifier:vlan-id
|        |  +--rw s-vlan-id-end*     diamclassifier:vlan-id
|        |  +--rw c-vlan-id-start*   diamclassifier:vlan-id
|        |  +--rw c-vlan-id-end*     diamclassifier:vlan-id
|        +--rw user-priority-range* [index]
|           +--rw index                 uint16
|           +--rw low-user-priority*    uint32
|           +--rw high-user-priority*   uint32
+--:(packet-filter)
|  +--rw packet-filter
|     +--rw direction?             fpcbase:packet-filter-direction
|     +--rw identifier?              uint8
|     +--rw evaluation-precedence?   uint8
|     +--rw contents* [component-type-identifier]
|        +--rw component-type-identifier fpcbase:component-type-id
|        +--rw (value)?
|           +--:(ipv4-local)
|           |  +--rw ipv4-local?                 inet:ipv4-address
|           +--:(ipv6-prefix-local)
|           |  +--rw ipv6-prefix-local?          inet:ipv6-prefix
```

```
     |           +--:(ipv4-ipv6-remote)
     |           |  +--rw ipv4-ipv6-remote?           inet:ip-address
     |           +--:(ipv6-prefix-remote)
     |           |  +--rw ipv6-prefix-remote?         inet:ipv6-prefix
     |           +--:(next-header)
     |           |  +--rw next-header?                uint8
     |           +--:(local-port)
     |           |  +--rw local-port?                 inet:port-number
     |           +--:(local-port-range)
     |           |  +--rw local-port-lo?              inet:port-number
     |           |  +--rw local-port-hi?              inet:port-number
     |           +--:(remote-port)
     |           |  +--rw remote-port?                inet:port-number
     |           +--:(remote-port-range)
     |           |  +--rw remote-port-lo?             inet:port-number
     |           |  +--rw remote-port-hi?             inet:port-number
     |           +--:(ipsec-index)
     |           |  +--rw ipsec-index?     traffic-selectors:ipsec-spi
     |           +--:(traffic-class)
     |           |  +--rw traffic-class?              inet:dscp
     |           +--:(traffic-class-range)
     |           |  +--rw traffic-class-lo?           inet:dscp
     |           |  +--rw traffic-class-hi?           inet:dscp
     |           +--:(flow-label)
     |              +--rw flow-label*    inet:ipv6-flow-label
     +--:(tunnel-info)
        +--rw tunnel-info
           +--rw tunnel-local-address?    inet:ip-address
           +--rw tunnel-remote-address?   inet:ip-address
           +--rw mtu-size?                uint32
           +--rw tunnel?                  identityref
           +--rw payload-type?            enumeration
           +--rw gre-key?                 uint32
           +--rw gtp-tunnel-info
           |  +--rw local-tunnel-identifier?    uint32
           |  +--rw remote-tunnel-identifier?   uint32
           |  +--rw sequence-numbers-enabled?   boolean
           +--rw ebi?                     fpcbase:ebi-type
           +--rw lbi?                     fpcbase:ebi-type

   action_value:
   +--:(action-value)
   |  +--rw (action-value)
   |     +--:(drop)
   |     |  +--rw drop?                          empty
   |     +--:(rewrite)
   |     |  +--rw rewrite
   |     |     +--rw (rewrite-value)?
```

```
   |     |        +--:(prefix-descriptor)
   |     |        |  +--rw destination-ip?              inet:ip-prefix
   |     |        |  +--rw source-ip?                   inet:ip-prefix
   |     |        +--:(pmip-selector)
   |     |        |  +--rw ts-format?                     identityref
   |     |        |  +--rw ipsec-spi-range!
   |     |        |  |  +--rw start-spi    ipsec-spi
   |     |        |  |  +--rw end-spi?     ipsec-spi
   |     |        |  +--rw source-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw destination-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw source-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw destination-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw ds-range!
   |     |        |  |  +--rw start-ds    inet:dscp
   |     |        |  |  +--rw end-ds?     inet:dscp
   |     |        |  +--rw protocol-range!
   |     |        |  |  +--rw start-protocol    uint8
   |     |        |  |  +--rw end-protocol?     uint8
   |     |        |  +--rw source-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw destination-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw flow-label-range!
   |     |        |  |  +--rw start-flow-label?   inet:ipv6-flow-label
   |     |        |  |  +--rw end-flow-label?     inet:ipv6-flow-label
   |     |        |  +--rw traffic-class-range!
   |     |        |  |  +--rw start-traffic-class?   inet:dscp
   |     |        |  |  +--rw end-traffic-class?     inet:dscp
   |     |        |  +--rw next-header-range!
   |     |        |     +--rw start-next-header?   uint8
   |     |        |     +--rw end-next-header?     uint8
   |     |        +--:(rfc5777-classifier-template)
   |     |           +--rw rfc5777-classifier-template
   |     |              +--rw protocol?               uint8
   |     |              +--rw direction?
                         diamclassifier:direction-type
   |     |              +--rw from-spec* [index]
   |     |              |  +--rw index                  uint16
```

```
   |     |             |  +--rw ip-address*             inet:ip-address
   |     |             |  +--rw ip-address-range* [index]
   |     |             |  |  +--rw index               uint16
   |     |             |  |  +--rw ip-address-start?   inet:ip-address
   |     |             |  |  +--rw ip-address-end?     inet:ip-address
   |     |             |  +--rw ip-address-mask*        inet:ip-prefix
   |     |             |  +--rw mac-address*    yang-types:mac-address
   |     |             |  +--rw mac-address-mask* [mac-address]
   |     |             |  |  +--rw mac-address
   |     |             |  |  |        yang-types:mac-address
   |     |             |  |  +--rw macaddress-mask-pattern
   |     |             |  |  |        yang-types:mac-address
   |     |             |  +--rw eui64-address*
   |     |             |  |        diamclassifier:eui64-address-type
   |     |             |  +--rw eui64-address-mask* [eui64-address]
   |     |             |  |  +--rw eui64-address
   |     |             |  |  |        diamclassifier:eui64-address-type
   |     |             |  |  +--rw eui64-address-mask-pattern
   |     |             |  |  |        diamclassifier:eui64-address-type
   |     |             |  +--rw port*                inet:port-number
   |     |             |  +--rw port-range* [index]
   |     |             |  |  +--rw index               uint16
   |     |             |  |  +--rw ip-address-start?   inet:port-number
   |     |             |  |  +--rw ip-address-end?     inet:port-number
   |     |             |  +--rw negated?
   |     |             |  |        diamclassifier:negated-flag-type
   |     |             |  +--rw use-assigned-address?   boolean
   |     |             +--rw to-spec* [index]
   |     |             |  +--rw index                   uint16
   |     |             |  +--rw ip-address*             inet:ip-address
   |     |             |  +--rw ip-address-range* [index]
   |     |             |  |  +--rw index               uint16
   |     |             |  |  +--rw ip-address-start?   inet:ip-address
   |     |             |  |  +--rw ip-address-end?     inet:ip-address
   |     |             |  +--rw ip-address-mask*        inet:ip-prefix
   |     |             |  +--rw mac-address*
   |     |             |  |        yang-types:mac-address
   |     |             |  +--rw mac-address-mask* [mac-address]
   |     |             |  |  +--rw mac-address
   |     |             |  |  |        yang-types:mac-address
   |     |             |  |  +--rw macaddress-mask-pattern
   |     |             |  |  |        yang-types:mac-address
   |     |             |  +--rw eui64-address*
   |     |             |  |        diamclassifier:eui64-address-type
   |     |             |  +--rw eui64-address-mask* [eui64-address]
   |     |             |  |  +--rw eui64-address
   |     |             |  |  |        diamclassifier:eui64-address-type
   |     |             |  |  +--rw eui64-address-mask-pattern
```

```
                                    diamclassifier:eui64-address-type
   |     |              |  +--rw port*                  inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index                uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw disffserv-code-point*   inet:dscp
   |     |              +--rw fragmentation-flag?     enumeration
   |     |              +--rw ip-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw tcp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw tcp-flag* [tcp-flag-type]
   |     |              |  +--rw tcp-flag-type    uint32
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw icmp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw eth-option* [index]
   |     |                 +--rw index                 uint16
   |     |                 +--rw eth-proto-type
   |     |                 |  +--rw eth-ether-type*   string
   |     |                 |  +--rw eth-sap*          string
   |     |                 +--rw vlan-id-range* [index]
   |     |                 |  +--rw index              uint16
   |     |                 |  +--rw s-vlan-id-start*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw s-vlan-id-end*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-start*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-end*
                                    diamclassifier:vlan-id
   |     |                 +--rw user-priority-range* [index]
   |     |                    +--rw index                uint16
   |     |                    +--rw low-user-priority*   uint32
```

```
   |     |                    +--rw high-user-priority*   uint32
   |     +--:(copy-forward-nexthop)
   |     |  +--rw copy-forward-nexthop
   |     |     +--rw (next-hop-value)?
   |     |        +--:(ip-address)
   |     |        |  +--rw ip-address?           inet:ip-address
   |     |        +--:(mac-address)
   |     |        |  +--rw mac-address?          ytypes:mac-address
   |     |        +--:(service-path)
   |     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
   |     |        +--:(mpls-path)
   |     |        |  +--rw mpls-path?            fpcbase:fpc-mpls-label
   |     |        +--:(nsh)
   |     |        |  +--rw nsh?                  string
   |     |        +--:(interface)
   |     |        |  +--rw interface?            uint16
   |     |        +--:(segment-identifier)
   |     |        |  +--rw segment-identifier?   fpcbase:segment-id
   |     |        +--:(mpls-label-stack)
   |     |        |  +--rw mpls-label-stack*     fpcbase:fpc-mpls-label
   |     |        +--:(mpls-sr-stack)
   |     |        |  +--rw mpls-sr-stack*        fpcbase:fpc-mpls-label
   |     |        +--:(srv6-stack)
   |     |        |  +--rw srv6-stack*           fpcbase:segment-id
   |     |        +--:(tunnel-info)
   |     |           +--rw tunnel-info
   |     |              +--rw tunnel-local-address?    inet:ip-address
   |     |              +--rw tunnel-remote-address?   inet:ip-address
   |     |              +--rw mtu-size?                uint32
   |     |              +--rw tunnel?                  identityref
   |     |              +--rw payload-type?            enumeration
   |     |              +--rw gre-key?                 uint32
   |     |              +--rw gtp-tunnel-info
   |     |              |  +--rw local-tunnel-identifier?    uint32
   |     |              |  +--rw remote-tunnel-identifier?   uint32
   |     |              |  +--rw sequence-numbers-enabled?   boolean
   |     |              +--rw ebi?                     fpcbase:ebi-type
   |     |              +--rw lbi?                     fpcbase:ebi-type
   |     +--:(nexthop)
   |     |  +--rw nexthop
   |     |     +--rw (next-hop-value)?
   |     |        +--:(ip-address)
   |     |        |  +--rw ip-address?           inet:ip-address
   |     |        +--:(mac-address)
   |     |        |  +--rw mac-address?          ytypes:mac-address
   |     |        +--:(service-path)
   |     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
   |     |        +--:(mpls-path)
```

```
   |     |        |  +--rw mpls-path?            fpcbase:fpc-mpls-label
   |     |        +--:(nsh)
   |     |        |  +--rw nsh?                 string
   |     |        +--:(interface)
   |     |        |  +--rw interface?           uint16
   |     |        +--:(segment-identifier)
   |     |        |  +--rw segment-identifier?  fpcbase:segment-id
   |     |        +--:(mpls-label-stack)
   |     |        |  +--rw mpls-label-stack*    fpcbase:fpc-mpls-label
   |     |        +--:(mpls-sr-stack)
   |     |        |  +--rw mpls-sr-stack*       fpcbase:fpc-mpls-label
   |     |        +--:(srv6-stack)
   |     |        |  +--rw srv6-stack*          fpcbase:segment-id
   |     |        +--:(tunnel-info)
   |     |           +--rw tunnel-info
   |     |              +--rw tunnel-local-address?    inet:ip-address
   |     |              +--rw tunnel-remote-address?   inet:ip-address
   |     |              +--rw mtu-size?                uint32
   |     |              +--rw tunnel?                  identityref
   |     |              +--rw payload-type?            enumeration
   |     |              +--rw gre-key?                 uint32
   |     |              +--rw gtp-tunnel-info
   |     |              |  +--rw local-tunnel-identifier?    uint32
   |     |              |  +--rw remote-tunnel-identifier?   uint32
   |     |              |  +--rw sequence-numbers-enabled?   boolean
   |     |              +--rw ebi?                     fpcbase:ebi-type
   |     |              +--rw lbi?                     fpcbase:ebi-type
   |     +--:(qos)
   |        +--rw trafficclass?                  inet:dscp
   |        +--rw per-mn-agg-max-dl?
                    qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value
   |        +--rw per-mn-agg-max-ul?
                    qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value
   |        +--rw per-session-agg-max-dl
   |        |  +--rw max-rate        uint32
   |        |  +--rw service-flag    boolean
   |        |  +--rw exclude-flag    boolean
   |        +--rw per-session-agg-max-ul
   |        |  +--rw max-rate        uint32
   |        |  +--rw service-flag    boolean
   |        |  +--rw exclude-flag    boolean
   |        +--rw priority-level                 uint8
   |        +--rw preemption-capability           enumeration
   |        +--rw preemption-vulnerability        enumeration
   |        +--rw agg-max-dl?
                    qos-pmip:Aggregate-Max-DL-Bit-Rate-Value
   |        +--rw agg-max-ul?
                    qos-pmip:Aggregate-Max-UL-Bit-Rate-Value
```

```
|        +--rw gbr-dl?
                  qos-pmip:Guaranteed-DL-Bit-Rate-Value
|        +--rw gbr-ul?
                  qos-pmip:Guaranteed-UL-Bit-Rate-Value
|        +--rw qci?
                  fpcbase:fpc-qos-class-identifier
|        +--rw ue-agg-max-bitrate?             uint32
|        +--rw apn-ambr?                       uint32

policy-configuration-value:
|  |  |     +--rw (policy-configuration-value)?
|  |  |        +--:(descriptor-value)
|  |  |        |  ...
|  |  |        +--:(action-value)
|  |  |        |  ...
|  |  |        +--:(setting-value)
|  |  |           +--rw setting?                        <anydata>

policy-configuration:
|  |  |  +--rw policy-configuration* [index]
|  |  |     +--rw index                            uint16
|  |  |     +--rw extensible?             boolean
|  |  |     +--rw static-attributes*      string
|  |  |     +--rw mandatory-attributes*   string
|  |  |     +--rw entity-state?           enumeration
|  |  |     +--rw version?                uint32
|  |  |     +--rw (policy-configuration-value)?
|  |  |        ...

module: ietf-dmm-fpc
+--rw tenant* [tenant-key]
  +--rw tenant-key                    fpc:fpc-identity
  +--rw topology-information-model
  |  +--rw service-group* [service-group-key role-key]
  |  |  +--rw service-group-key     fpc:fpc-identity
  |  |  +--rw service-group-name?   string
  |  |  +--rw role-key              identityref
  |  |  +--rw role-name?            string
  |  |  +--rw protocol*             identityref
  |  |  +--rw feature*              identityref
  |  |  +--rw service-group-configuration* [index]
  |  |  |  +--rw index                             uint16
  |  |  |  +--rw (policy-configuration-value)?
  |  |  |        |  ...
  |  |  +--rw dpn* [dpn-key]
  |  |     +--rw dpn-key                fpc:fpc-identity
  |  |     +--rw referenced-interface* [interface-key]
  |  |        +--rw interface-key             fpc:fpc-identity
```

```
|  |        +--rw peer-service-group-key*   fpc:fpc-identity
|  +--rw dpn* [dpn-key]
|  |  +--rw dpn-key                          fpc:fpc-identity
|  |  +--rw dpn-name?                        string
|  |  +--rw dpn-resource-mapping-reference?  string
|  |  +--rw domain-key                       fpc:fpc-identity
|  |  +--rw service-group-key*               fpc:fpc-identity
|  |  +--rw interface* [interface-key]
|  |  |  +--rw interface-key        fpc:fpc-identity
|  |  |  +--rw interface-name?      string
|  |  |  +--rw role?                identityref
|  |  |  +--rw protocol*            identityref
|  |  |  +--rw interface-configuration* [index]
|  |  |     +--rw (policy-configuration-value)?
|  |  |           |  ...
|  |  +--rw dpn-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        +--rw index    uint16
|  |          +--rw (policy-configuration-value)?
|  |               |  ...
|  +--rw domain* [domain-key]
|  |  +--rw domain-key        fpc:fpc-identity
|  |  +--rw domain-name?      string
|  |  +--rw domain-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |              |         ...
|  +--rw dpn-checkpoint
|  |  +--rw basename?          fpc:fpc-identity
|  |  +--rw base-checkpoint?   string
|  +--rw service-group-checkpoint
|  |  +--rw basename?          fpc:fpc-identity
|  |  +--rw base-checkpoint?   string
|  +--rw dpn-checkpoint
|  |  +--rw basename?                   fpc:fpc-identity
|  |  +--rw base-checkpoint?            string
+--rw policy-information-model
|  +--rw action-template* [action-template-key]
|  |  +--rw action-template-key        fpc:fpc-identity
|  |  +--rw (action-value)
|  |  |                   ...
|  |  +--rw extensible?                boolean
|  |  +--rw static-attributes*         string
|  |  +--rw mandatory-attributes*      string
|  |  +--rw entity-state?              enumeration
|  |  +--rw version?                   uint32
|  +--rw descriptor-template* [descriptor-template-key]
```

```
|  |  +--rw descriptor-template-key         fpc:fpc-identity
|  |  +--rw (descriptor-value)
|  |  |                    ...
|  |  +--rw extensible?                     boolean
|  |  +--rw static-attributes*              string
|  |  +--rw mandatory-attributes*           string
|  |  +--rw entity-state?                   enumeration
|  |  +--rw version?                        uint32
|  +--rw rule-template* [rule-template-key]
|  |  +--rw rule-template-key           fpc:fpc-identity
|  |  +--rw descriptor-match-type       enumeration
|  |  +--rw descriptor-configuration* [descriptor-template-key]
|  |  |  +--rw descriptor-template-key    fpc:fpc-identity
|  |  |  +--rw direction?                 rfc5777:direction-type
|  |  |  +--rw setting?                        <anydata>
|  |  |  +--rw attribute-expression* [index]
|  |  |     +--rw index                           uint16
|  |  |     +--rw (descriptor-value)
|  |  |     |  ...
|  |  +--rw action-configuration* [action-order]
|  |  |  +--rw action-order            uint32
|  |  |  +--rw action-template-key     fpc:fpc-identity
|  |  |  +--rw setting?                        <anydata>
|  |  |  +--rw attribute-expression* [index]
|  |  |     +--rw index                     uint16
|  |  |     +--rw (action-value)
|  |  |     |  ...
|  |  +--rw extensible?                 boolean
|  |  +--rw static-attributes*          string
|  |  +--rw mandatory-attributes*       string
|  |  +--rw entity-state?               enumeration
|  |  +--rw version?                    uint32
|  |  +--rw rule-configuration* [index]
|  |     +--rw index    uint16
|  |         +--rw (policy-configuration-value)?
|  |           |  ...
|  +--rw policy-template* [policy-template-key]
|  |  +--rw policy-template-key     fpc:fpc-identity
|  |  +--rw rule-template* [precedence]
|  |  |  +--rw precedence           uint32
|  |  |  +--rw rule-template-key    fpc:fpc-identity
|  |  +--rw extensible?             boolean
|  |  +--rw static-attributes*      string
|  |  +--rw mandatory-attributes*   string
|  |  +--rw entity-state?           enumeration
|  |  +--rw version?                uint32
|  |  +--rw policy-configuration* [index]
|  |     ...
```

```
|  +--rw basename?              fpc:fpc-identity
|  +--rw base-checkpoint?       string
+--rw mobility-context* [mobility-context-key]
|  +--rw mobility-context-key    fpc:fpc-identity
|  +--rw delegating-ip-prefix*   inet:ip-prefix
|  +--rw parent-context?         fpc:fpc-identity
|  +--rw child-context*          fpc:fpc-identity
|  +--rw mobile-node
|  |  +--rw ip-address*    inet:ip-address
|  |  +--rw imsi?          fpcbase:imsi-type
|  |  +--rw mn-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        ...
|  +--rw domain
|  |  +--rw domain-key?        fpc:fpc-identity
|  |  +--rw domain-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        ...
|   +--rw dpn* [dpn-key]
|      +--rw dpn-key              fpc:fpc-identity
|      +--rw dpn-policy-configuration* [policy-template-key]
|      |  +--rw policy-template-key     fpc:fpc-identity
|      |  +--rw policy-configuration* [index]
|      |     ...
|      +--rw role?                identityref
|      +--rw service-data-flow* [identifier]
|         +--rw identifier           uint32
|         +--rw service-group-key?   fpc:fpc-identity
|         +--rw interface* [interface-key]
|         |  +--rw interface-key    fpc:fpc-identity
|         +--rw service-data-flow-policy-
                 configuration* [policy-template-key]
|            +--rw policy-template-key     fpc:fpc-identity
|            +--rw policy-configuration* [index]
|               ...
+--rw monitor* [monitor-key]
  +--rw extensible?             boolean
  +--rw static-attributes*      string
  +--rw mandatory-attributes*   string
  +--rw entity-state?           enumeration
  +--rw version?                uint32
  +--rw monitor-key             fpc:fpc-identity
  +--rw target?                 string
  +--rw deferrable?             boolean
  +--rw (configuration)
     +--:(period)
```

```
          |  +--rw period?           uint32
          +--:(threshold-config)
          |  +--rw low?              uint32
          |  +--rw hi?               uint32
          +--:(schedule)
          |  +--rw schedule?         uint32
          +--:(event-identities)
          |  +--rw event-identities*       identityref
          +--:(event-ids)
             +--rw event-ids*        uint32

     rpcs:
       +---x configure
       |  +---w input
       |  |  +---w client-id          fpc:client-identifier
       |  |  +---w execution-delay?   uint32
       |  |  +---w yang-patch
       |  |     +---w patch-id    string
       |  |     +---w comment?    string
       |  |     +---w edit* [edit-id]
       |  |        +---w edit-id         string
       |  |        +---w operation       enumeration
       |  |        +---w target          target-resource-offset
       |  |        +---w point?          target-resource-offset
       |  |        +---w where?          enumeration
       |  |        +---w value?          <anydata>
       |  |        +---w reference-scope?   fpc:ref-scope
       |  |        +---w command-set
       |  |           +---w (instr-type)?
       |  |              +--:(instr-3gpp-mob)
       |  |              |  +---w instr-3gpp-mob? fpcbase:threegpp-instr
       |  |              +--:(instr-pmip)
       |  |                 +---w instr-pmip?       pmip-commandset
       |  +--ro output
       |     +--ro yang-patch-status
       |        +--ro patch-id       string
       |        +--ro (global-status)?
       |        |  +--:(global-errors)
       |        |  |  +--ro errors
       |        |  |     +--ro error*
       |        |  |        +--ro error-type       enumeration
       |        |  |        +--ro error-tag        string
       |        |  |        +--ro error-app-tag?   string
       |        |  |        +--ro error-path?      instance-identifier
       |        |  |        +--ro error-message?   string
       |        |  |        +--ro error-info?      <anydata>
       |        |  +--:(ok)
       |        |     +--ro ok?            empty
```

```
|        +--ro edit-status
|           +--ro edit* [edit-id]
|              +--ro edit-id            string
|              +--ro (edit-status-choice)?
|                 +--:(ok)
|                 |  +--ro ok?                empty
|                 |  +--ro notify-follows?    boolean
|                 |  +--ro subsequent-edit* [edit-id]
|                 |     +--ro edit-id      string
|                 |     +--ro operation    enumeration
|                 |     +--ro target
                              ypatch:target-resource-offset
|                 |     +--ro point?
                                  ypatch:target-resource-offset
|                 |     +--ro where?       enumeration
|                 |     +--ro value?       <anydata>
|                 +--:(errors)
|                    +--ro errors
|                       +--ro error*
|                          +--ro error-type       enumeration
|                          +--ro error-tag        string
|                          +--ro error-app-tag?   string
|                          +--ro error-path?
                                    instance-identifier
|                          +--ro error-message?   string
|                          +--ro error-info?      <anydata>
+---x register_monitor
|  +---w input
|  |  +---w client-id          fpc:client-identifier
|  |  +---w execution-delay?   uint32
|  |  +---w operation-id              uint64
|  |  +---w monitor* [monitor-key]
|  |     +---w extensible?             boolean
|  |     +---w static-attributes*      string
|  |     +---w mandatory-attributes*   string
|  |     +---w entity-state?           enumeration
|  |     +---w version?                uint32
|  |     +---w monitor-key             fpc:fpc-identity
|  |     +---w target?                 string
|  |     +---w deferrable?             boolean
|  |     +---w (configuration)
|  |        +--:(period)
|  |        |  +---w period?                uint32
|  |        +--:(threshold-config)
|  |        |  +---w low?                   uint32
|  |        |  +---w hi?                    uint32
|  |        +--:(schedule)
|  |        |  +---w schedule?              uint32
```

```
|  |        +--:(event-identities)
|  |        |  +---w event-identities*       identityref
|  |        +--:(event-ids)
|  |           +---w event-ids*             uint32
|  +--ro output
|     +--ro operation-id     uint64
|     +--ro (edit-status-choice)?
|        +--:(ok)
|        |  +--ro ok?       empty
|        +--:(errors)
|           +--ro errors
|              +--ro error*
|                 +--ro error-type       enumeration
|                 +--ro error-tag        string
|                 +--ro error-app-tag?   string
|                 +--ro error-path?      instance-identifier
|                 +--ro error-message?   string
|                 +--ro error-info?      <anydata>
+---x deregister_monitor
|  +---w input
|  |  +---w client-id          fpc:client-identifier
|  |  +---w execution-delay?   uint32
|  |  +---w operation-id              uint64
|  |  +---w monitor* [monitor-key]
|  |     +---w monitor-key    fpc:fpc-identity
|  |     +---w send_data?     boolean
|  +--ro output
|     +--ro operation-id     uint64
|     +--ro (edit-status-choice)?
|        +--:(ok)
|        |  +--ro ok?       empty
|        +--:(errors)
|           +--ro errors
|              +--ro error*
|                 +--ro error-type       enumeration
|                 +--ro error-tag        string
|                 +--ro error-app-tag?   string
|                 +--ro error-path?      instance-identifier
|                 +--ro error-message?   string
|                 +--ro error-info?      <anydata>
+---x probe
   +---w input
   |  +---w client-id          fpc:client-identifier
   |  +---w execution-delay?   uint32
   |  +---w operation-id              uint64
   |  +---w monitor* [monitor-key]
   |     +---w monitor-key    fpc:fpc-identity
   +--ro output
```

```
            +--ro operation-id     uint64
            +--ro (edit-status-choice)?
               +--:(ok)
               |  +--ro ok?       empty
               +--:(errors)
                  +--ro errors
                     +--ro error*
                        +--ro error-type       enumeration
                        +--ro error-tag        string
                        +--ro error-app-tag?   string
                        +--ro error-path?      instance-identifier
                        +--ro error-message?   string
                        +--ro error-info?      <anydata>

    notifications:
      +---n config-result-notification
      |  +--ro yang-patch-status
      |  |  +--ro patch-id       string
      |  |  +--ro (global-status)?
      |  |  |  +--:(global-errors)
      |  |  |  |  +--ro errors
      |  |  |  |     +--ro error*
      |  |  |  |        +--ro error-type       enumeration
      |  |  |  |        +--ro error-tag        string
      |  |  |  |        +--ro error-app-tag?   string
      |  |  |  |        +--ro error-path?      instance-identifier
      |  |  |  |        +--ro error-message?   string
      |  |  |  |        +--ro error-info?      <anydata>
      |  |  |  +--:(ok)
      |  |  |     +--ro ok?            empty
      |  |  +--ro edit-status
      |  |     +--ro edit* [edit-id]
      |  |        +--ro edit-id    string
      |  |        +--ro (edit-status-choice)?
      |  |           +--:(ok)
      |  |           |  +--ro ok?        empty
      |  |           +--:(errors)
      |  |              +--ro errors
      |  |                 +--ro error*
      |  |                    +--ro error-type       enumeration
      |  |                    +--ro error-tag        string
      |  |                    +--ro error-app-tag?   string
      |  |                    +--ro error-path?
      |  |                            instance-identifier
      |  |                    +--ro error-message?   string
      |  |                    +--ro error-info?      <anydata>
      |  +--ro subsequent-edit* [edit-id]
      |     +--ro edit-id      string
```

```
|     +--ro operation    enumeration
|     +--ro target       ypatch:target-resource-offset
|     +--ro point?       ypatch:target-resource-offset
|     +--ro where?       enumeration
|     +--ro value?       <anydata>
+---n notify
   +--ro notification-id?   uint32
   +--ro timestamp?         uint32
   +--ro report* [monitor-key]
      +--ro monitor-key                fpc:fpc-identity
      +--ro trigger?                   identityref
      +--ro (value)?
         +--:(dpn-candidate-available)
         |  +--ro node-id?                     inet:uri
         |  +--ro supported-interface-list* [role-key]
         |     +--ro role-key    identityref
         +--:(dpn-unavailable)
         |  +--ro dpn-id?                    fpc:fpc-identity
         +--:(report-value)
            +--ro report-value?              <anydata>
```

Figure 38: YANG FPC Agent Tree

Appendix C. Change Log

C.1. Changes since Version 09

The following changes have been made since version 09

Migration to a Template based framework. This affects all elements. The framework has a template definition language.

Basename is split into two aspects. The first is version which applies to Templates. The second is checkpointing which applies to specific sections only.

Rule was inside Policy and now is Rule-Template and stands as a peer structure to Policy.

Types, e.g. Descriptor Types, Action Types, etc., are now templates that have no values filled in.

The embedded rule has been replaced by a template that has no predefined variables. All rules, pre-configured or embedded, are realized as Policy instantiations.

The Unassigned DPN is used to track requests vs. those that are installed, i.e. Agent assignment of Policy is supported.

The Topology system supports selection information by ServiceGroup or ServiceEndpoint.

DPN Peer Groups and DPN Groups are now PeerServiceGroup and ServiceGroup.

Bulk Configuration and Configuration now follow a style similar to YANG Patch. Agents MAY response back with edits it made to complete the Client edit request.

RFC 5777 Classifiers have been added.

All operations have a common error format.

C.2. Changes since Version 10

The following changes have been made since version 10

Sevice-Endpoints eliminated. Service-Group and DPN interfaces changed to hold information previously held by Service-Endpoint as noted in ML during IETF 101.

Service-Group resides under the Topology-Information-Mode

The Domain now has a checkpoint and the Topology Information Model checkpoint was removed to avoid any overlaps in checkpoints.

Scrubbed YANG for NMDA compliance and Guidelines (RFC 6087bis).

Monitor lifecycle, policy and policy installation examples added.

Authors' Addresses

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

Email: satoru.matsushima@g.softbank.co.jp

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: lylebe551144@gmail.com

Marco Liebsch
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Phone: +49 6221 4342146
Email: liebsch@neclab.eu

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org