DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

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

October 30, 2017

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-09

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. An 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 that data-plane nodes. The data-plane abstractions presented in this document is extensible, in order to support many different types of mobility management systems and data-plane functions.

Status of This Memo

Table of Contents

1. Introduction

This document describes Forwarding Policy Configuration (FPC), a system for managing the separation of data-plane and control-plane. FPC enables flexible mobility management using FPC agent and FPC client functions. An FPC agent exports an abstract interface to 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 other applications which require data-plane control, can utilize the FPC client at various granularities of operation. The operations are capable of configuring a single Data-Plane Node (DPN) directly, as well as multiple DPNs as determined by abstracted data-plane models on the FPC agent.

A FPC agent provides data-plane abstraction in the following three areas:

Topology: DPNs are grouped and abstracted according to well-known concepts of mobility management such as access networks, anchors and domains. A FPC agent provides an interface to the abstract DPN-groups that enables definition of a topology for the forwarding plane. For example, access nodes may be assigned to a DPN-group which peers to a DPN-group of anchor nodes.

Policy: A Policy embodies the mechanisms for processing specific traffic flows or packets. This is needed for QoS, for packet processing to rewrite headers, etc. A Policy consists of one or more rules. Each rule is composed of Descriptors and Actions. Descriptors in a rule identify traffic flows, and Actions apply

treatments to packets that match the Descriptors in the rule. An arbitrary set of policies can applied to a particular collection of flows throgh the use of a Configurable-Policy.

Mobility: A mobility session which is active on a mobile node is abstracted as a Mobility-Context with associated runtime concrete attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Mobility-Contexts are attached to DPNs via DPN-References. The References assign pre-defined Policies that were requested to be enforced as part of the mobility signaling request. Policy may also be realized by Embedded-Rules in the DPN-Reference. Such policies are typically highly specialized (e.g. negotiated as a part of signaling), were not pre-provisioned or designed as a template. Hence, they are not reusable across Mobility-Contexts.

Monitors provide a mechanism to produce reports when events regarding Configurable Policies, Mobility Contexts, DPNs or the Agent occur.

The Agent assembles applicable sets of forwarding policies for the mobility sessions from the data model, and then renders those policies into specific configurations for each DPN to which the sessions attached. The specific protocols and configurations to configure DPN from a FPC Agent are outside the scope of this document.

The data-plane abstractions may be extended 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].

DPN:	A data-plane node (DPN) is capable of deploying data-plane features. DPNs may be switches or routers regardless of their realiziation, i.e. whether they are hardware or software based.
FPC Agent:	A functional entity in FPC that manages DPNs and provides abstracted data-plane networks to mobility management systems and/or applications through FPC Clients.

FPC Client:	A functional entity in FPC that is integrated with mobility management systems and/or applications to control forwarding policy, mobility sessions and DPNs.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions.
Domain:	One or more DPNs that form a data-plane network. A mobility management system or an application in a tenant may utilize a single or multiple domains.
Configurable Policy:	A set of Rules (forwarding policies) installed upon a DPN.
Mobility Context:	An abstracted endpoint of a mobility session associated with runtime attributes.

3. FPC Design Objectives and Deployment

To fulfill the requirements described in [RFC7333], FPC enables mobility control-planes and applications to configure DPNs with various roles of the mobility management as described in [I-D.ietf-dmm-deployment-models].

FPC defines building blocks of FPC Agent and FPC Client, as well as data models for the necessary data-plane abstractions. The attributes defining those data models serve as protocol elements for the interface between the FPC Agent and the FPC Client.

Mobility control-planes and applications integrate the FPC Client function. The FPC Client connects to FPC Agent functions. The Client and the Agent communicate based on information models for the data-plane abstractions described in Section 4. The data models allow the control-plane and the applications to support forwarding policies on the Agent for their mobility sessions.

The FPC Agent carries out the required configuration and management of the DPN(s). The Agent determines DPN configurations according to the forwarding policies requested by the FPC Client. The DPN configurations could be specific to each DPN implementation such that how FPC Agent determines implementation specific configuration for a DPN is outside of the scope of this document. Along with the models, the control-plane and the applications put Policies to the Agent prior to creating their mobility sessions.

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

This architecture is illustrated in Figure 1. An FPC Agent may be implemented in a network controller that handles multiple DPNs, or there is a simple case where another FPC Agent may itself be integrated into a DPN.

This document does not adopt a specific protocol for the FPC interface protocol and it is out of scope. However it must be capable of supporting FPC protocol messages and transactions described in Section 5.

```
                           +-------------------------+
                           | Mobility Control-Plane  |
                           |          and            |
                           |      Applications       |
                           |+-----------------------+|
                           ||      FPC Client       ||
                           |+----------^------------+|
                           +-----------|-------------+
               FPC interface protocol  |
                       +---------------+-----------------+
                       |                                 |
         Network       |                                 |
         Controller    |                      DPN        |
           +-----------|-------------+        +----------|---------+
           |+----------v------------+|        |+---------v--------+|
           ||   [Data-plane model]  ||        ||[Data-plane model]||
           ||       FPC Agent       ||        ||    FPC Agent     ||
           |+-----------------------+|        |+------------------+|
           |+------------+----------+|        |                    |
           ||SB Protocols|FPC Client||        |  DPN Configuration |
           ||   Modules  |  Module  ||        +--------------------+
           |+------^-----+----^-----+|
           +-------|----------|------+
                   |          |
         Other     |          | FPC interface
         Southband |          | Protocol
         Protocols |          |
                   |          +-----------------+
                   |                            |
       DPN         |                 DPN        |
        +----------|---------+       +----------|---------+
        |+---------v--------+|       |+---------v--------+|
        ||  Configuration   ||       ||[Data-plane model]||
        || Protocol module  ||       ||     FPC Agent    ||
        |+------------------+|       |+------------------+|
        |                    |       |                    |
        | DPN Configuration  |       |  DPN Configuration |
        +--------------------+       +--------------------+
```

Figure 1: Reference Forwarding Policy Configuration (FPC)
Architecture

The FPC architecture supports multi-tenancy; an 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.

Note that all FPC models SHOULD be configurable. The FPC interface protocol in Figure 1 is only required to handle runtime data in the Mobility model. The rest of the FPC models, namely Topology and Policy, may be pre-configured, and in that case real-time protocol exchanges would not be required for them. Operators that are tenants in the FPC data-plane could configure Topology and Policy on the Agent through other means, such as Restconf [I-D.ietf-netconf-restconf] or Netconf [RFC6241].

4. FPC Mobility Information Model

4.1. Model Notation and Conventions

In order to clarify the description of the information model, this draft uses the convention describe in the below.

Information model entities, e.g. DPNs, Rules, etc., are listed in a hierarchical notation where all entities with the same hierarchical level are located on the same left-justified position one after the other. When entities are composed of sub-entities, they will appear shifted to the right.

```
                 |
                 +-[entity2]
                 |         +-[entity2.1]
                 |         +-[entity2.2]
```

Figure 2: Model Notation - An Example

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

List: A collection of entities (some could be of the same kind)

Set: A collection of entities without duplications

Name: a human-readable string

Key: a unique value. There are 3 types of keys:

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

G-Key: a globally unique key - unique within a tenant

L-Key: a unique key within the set of values in local namespace. For example, there exist interfaces with teh same name, e.g. "interface0", in two different DPNs within the same tenant but there can only be one "interface0" within each DPN, i.e. its local Interface-Id L-Key space.

The Settings pattern, i.e. attributes whose names contain the word 'Settings', is a genera set that holds unique properties. If multiple values of a property are present they are held in a List for the property within the settings container. The container acts as an extensible placeholder for properties (attribute/value pairs).

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

```
          Example: The following example shows 3 entities:
                   - Entity1 is a globally unique key and has an
                           optional, 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 3

When expanding entity1 into an information model such as YANG it would result in two values: entity1-GKey and entity1-Name.

4.2. Core Structure - Information Model Components

The substructures that comprise the information model are:

Topology: defines the different DPNs and links between them.

Policy-Definitions: describes how to handle flows: how to identify traffic and what actions are performed on data packets

Configurable-Policy: policies that are relatively static. These policies are usually pre-provisioned and are modified only when the network configuration is updated or when new policy rules are configured

Mobility-Context: policies that are created or modified during the network's operation. In most cases, on a per-flow or per session basis

Monitor: a list of events that trigger notification messages from FPC Agents to FPC Clients

```
                 :
                 |
                 +-[Mobility]
                 |          +
                 |          |
                 :          +-[Topology]
                            |
                            +-[Policy]
                            |
                            +-[Configurable-Policy] <Set>
                            |
                            +-[Mobility-Context] <Set>
                            |
                            +-[Monitor] <Set>
```

Figure 4: Mobility Information Model - Core Structure

4.3. Topology

The Topology sub-structure specifies virtual DPNs and the relations between them. It is used by the network management entity to select the most appropriate DPN resources for handling specific session flows.

A virtual DPN is a logical entity that performs DPN functionality (packet movement and management). It may represent a physical DPN unit, a sub-function of a physical DPN or a collection of physical DPNs. The motivation to refer to virtual DPNs in the information model rather than physical DPNs is to provide flexibility for network architects to define which DPN-selection capabilities are performed by the FPC Agent (distributed) and which by the FPC client (centralized).

When a virtual DPN is mapped to a physical DPN, the FPC client has maximum knowledge of the DPN architecture and uses it to perform DPN selection for specific sessions. 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 and only the FPC Agent can address the specific associated physical DPNs.

The Topology sub-structure is comprised of the following sub-structures:

DPN-Set: the set of virtual DPNs in a network configuration

DPN-Type-Set: a set of DPN-Type entities

DPN-Group-Set: a set of virtual DPNs that supports a specific administrative purpose (in/out bound, roaming, subnetwork with common specific configuration etc.)

Domain: a set of Domains tha represent a logical partition of network resources

```
                          |
                          +-[Topology]
                          |          +-[DPN] <Set>
                          |          +-[DPN-Type] <Set>
                          |          +-[DPN-Group] <Set>
                          |          +-[Domain] <Set>
```

Figure 5: Topology Substructure

4.3.1. DPN

The DPN set sub-structure specifies a subset of virtual DPNs in the network . Some of the DPNs may be identical in functionality and only differ by their key.

```
            |
            +-[DPN] <Set>
            |     +-[DPN-Id] <G-Key>, <Name> (O)
            |     +-[DPN-Resource-Mapping-Reference] (O)
            |     +-[Interface] <Set>
            |                 +-[Interface-Reference]
            |                 |           +-[Access-Technology] <U-Key>
            |                 |           +-[Role] <U-Key>
            |                 |           +-[Interface-Id] <L-Key>
            |                 +-[Interface-Settings] <Set> (O)
```

Figure 6: DPN Substructure

Each DPN entry contains the following information:

DPN-Id (Key): A unique Identifier of the virtual DPN

DPN-Id (Name): the human-readable display string

DPN-Resource-Mapping-Reference (O): A reference to the underlying implementation, e.g. physical node, virtual element, etc. that supports this DPN. This value MUST be non-empty prior to Dynamic-Policies being installed upon the DPN.

Interface-Set: the set of interfaces (through which data packets are received and transmitted) of that Virtual DPN. The virtual DPN abstracts one or multiple physical DPNs each having one or more interfaces. The Interface-set sub-structure is a collection of all interface types available by this virtual DPN. The purpose of this information is not to describe each interface of each DPN, but rather, to indicate the entire set of interface types for the sake of the DPN selection process, when a DPN with specific interface capabilities is required. Each interface entry has a reference to a defined DPN-Type entry in the DPN-Type-Set sub-structure defined below and additional information that is specific to that interface:

Interface-Reference - a 3-tuple key that uniquely references an entry in the DPN-Type-Set sub-structure: Access-Technology, Role and Interface-Id.

Interface-Settings - An optional set of settings of this interface (that do not affect the DPN selection of active or enabled interfaces). Examples: MTU size, display name, etc.

4.3.2. DPN-Type

DPN-Type is the collection of all possible types of interfaces defined for DPNs in the network. The interfaces are grouped according to their access technology, e.g. 3GPP, WiMAX, 802.x and Role, e.g. LMA, MAG, PGW, AMF etc. Within a group, interfaces may have additional properties that are more specific, a list of features and optionally settings relevant to DPN selection. This information is used when searching for resources in a network to carry out required operations on data traffic.

```
                  |
                  +-[DPN-Type] <Set>
                  |     +-[Access-Technology] <U-Key>,<Name> (O)
                  |     +-[Role] <U-Key>, <Name> (O)
                  |     +-[Interface] <Set>
                  |             +-[Interface-Id] <L-Key>, <Name> (O)
                  |             +-[Interface-Protocol] <Set>
                  |             +-[Features] <Set> (O)
                  |             +-[Interface-Settings] <Set> (O)
```

Figure 7: DPN Type

Each DPN-Type entry contains the following information:

Access-technology: the technology used in the access network that originated the signaling (WiMAX, 3GPP, 802.x etc.)

Role: the role (MAG, LMA, PGW, AMF etc.) of the device sharing the interfaces specified below.

Interface: A set of interfaces possible for the group above. Each interface carries the following information:

> Interface-Id: a key that is used together with the 2-tuple Access-Technology and Role, to create a 3-tuple key that is referred to be the interface definition of virtual DPNs
>
> Interface-Protocols: rotocols supported by this interface - PMIP, S5-GTP, S5-PMIP etc.
>
> Features: an optional set of supported features which further determine the suitability of the interface to the desired operation
>
> Interface-Settings: an optional set of settings that MUST be known when determining the suitability of an interface for the specific request. The difference between 'Features' and 'Settings' is that 'Features' are static while 'Settings' may be configured. Examples: SequenceNumber=ON/OFF

The entries Access-Technology and Role represent a tuple key that uniquely identifies the set of interfaces that may be available for DPNs of the specific type.

4.3.3. DPN-Group

A DPN-Group is a list of a group of DPNs serving some administrative purpose. Each group contains a list of DPNs (referenced by DPN-Id) and selected interfaces (referenced by the interface's 3-tuple key). The interfaces are listed rather than referred implicitly by each specific DPN to enable to define a subset of a DPN interfaces to be part of the group.

```
       |
       +-[DPN-Group] <Set>
       |           +-[DPN-Group-Id] <G-Key>, <Name> (O)
                   +-[Referenced-Interface] <Set>
                   |           +-[Interface-Id] <L-Key>
                   |           +-[Role] <U-Key>
                   |           +-[Access-Technology] <U-Key>
                   |           +-[Supporting-DPN-Id] <Set>
                   |           +-[DPN-Group-Peer-Reference] <Set> (O)
                   +-[DPN-Peer-Group] <Set>
                   |           +-[Remote-DPN-Group-Id] <L-key>
                   |           +-[Interface-Settings] <Set> (O)
                   +-[Domain-Id-Reference]
```

Figure 8: DPN Group

Each DPN-Group entry contains the following information:

DPN-Group (Key): A unique Identifer of the DPN-Group

DPN-Group (Name): the human-readable display string

Referenced-Interfaces: A set of interfaces and the DPNs / associated DPN-Peer-Groups that support them. Each entry contains

Interface-Id: a key that is used together with the 2-tuple Access-Technology and Role, to create a 3-tuple key that is referred to be the interface definition of virtual DPNs

Role: the role (MAG, LMA, PGW, AMF etc.) of the device sharing the interfaces specified below.

Access-technology: the technology used in the access network that originated the signaling (WiMAX, 3GPP, 802.x etc.)

Role: the role (MAG, LMA, PGW, AMF etc.) of the device sharing the interfaces specified below.

Supporting-DPN-Id (Set):   A set of DPN-Id-References that support the specific interface for this DPN-Group.

Interface-Settings: an optional set of settings that MUST be known when determining the suitability of an interface for the specific request.F

DPN-Peer-Group: A set of Remote (from the DPN-Group's point of view) DPN-Groups. When communication occurs between the DPN-Group and

DPN-Peer-Group the Interface-Settings MUST be used. Each entry contains the following information: entry contains

Remote-DPN-Group-Id: A unique Identifier of the DPN-Peer-Group

Interface-Settings: an optional set of settings that MUST be known when determining the suitability of an interface for the specific request.

4.3.4. Domain

A Domain represents a logica abstraction of a group of heterogeneous Topology resources. Other models, outside of the scope of this specificaiton, provide the details for the Domain.

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

Figure 9: Domainn

Each Domain entry contains the following information:

Domain (Key): A unique Identifier of the Domain

Domain (Name): the human-readable display string

Domain-Reference: A link to the underlying resource / informaiton that provides further details regarding the domain.

4.4. Policy

The Policy substructure defines and identifies grouped Rules for enforcement on the data plane. Rules comprise traffic descriptors and actions on how to treat traffic in case the traffic descriptor matches a data packet. The Policy substructure is independent of a policy context, whether it's an administratively configurable policy which applies to all or a defined aggregate of data flows, or a mobility context-related policy, which is associated with a mobility session and may apply only to data traffic of an associated mobile node while being registered.

In addition to the Policy substructure, the Core Structure per Section 4.2 holds accordingly separate entries for the Configurable-Policy as well as for Mobility-Context, which do not only define their own policies by embedded rules, but comprise references to policy definitions in the Policy substructure, to which additional

settings can be applies on a per-Configurable-Policy basis and on a per-Mobility-Context bases respectively.

Traffic descriptions and traffic treatment actions are defined separately in Descriptor-Definition and Action-Definition respectively. Binding between traffic descriptors and associated traffic treatment action is defined in a set of rule definition entries (Rule-Definition) which comprise references to entries in the set of traffic descriptors (Descriptor-Reference) and traffic treatment actions (Action-Reference). Accordingly, a single rule or a group of rules are bound to a policy in the set of policy definitions (Policy-Definition) by reference to entries in the set of rule definitions (Rule-Id).

```
     |
     +-[Policy]
     |      +-[Policy-Definition] <Set>
     |      |             +-[Policy-Id] <G-Key> (M)
     |      |             +-[Rule-Reference] Set (M)
     |      |             +-[Precedence] <L-Key> (M)
     |      |             +-[Rule-Id-Reference] (M)
     |      +-[Rule-Definition] <Set>
     |      |             +-[Rule-Id] <L-Key> (M)
     |      |             +-[Descriptor-Match-Type] (M)
     |      |             +-[Descriptor-Reference] <Set>
     |      |             |                +-[Descriptor-Id-Reference]
     |      |             |                +-[Direction] (O)
     |      |             +-[Action-Reference] <Set>
     |      |                              +-[Action-Id-Reference]
     |      |                              +-[Action-Order]
     |      +-[Descriptor-Definition] <Set>
     |      |             +-[Descriptor -Id] <L-Key> (M)
     |      |             +-[Descriptor-Type]
     |      |             +-[Descriptor-Value]
     |      +-[Action-Definition] <Set>
     |                    +-[Action-Id] <L-Key> (M)
     |                    +-[Action-Type]
     |                    +-[Action-Value]
     |
```

Figure 10: Policy Substructure

The Policy Substructure contains the following entries:

Policy-Definition: A set of policy definitions which bind a single or multiple rules to a policy.

Policy-Id: Identifies a policy definition.

Rule-Reference: Assigns a set of rule definitions, which are bound to a policy definition, by reference to the associated Rule in the Rule-Definition Substructure.

Precedence: Defines the order with which the rules must be applied.

Rule-Id-Reference: Identifies a rule.

Rule-Definition: A set of rule definitions which bind a single or multiple traffic descriptors (by reference to Descriptor-Definition) to a single or multiple traffic treatment actions (by reference to Action-Definition).

Rule-Id: Identifies a rule definition.

Descriptor-Match-Type: Conjuction of Descriptor-Values to apply as match to DPN traffic, which can be either OR or AND. The identified conjunction applies to all Descriptor-Definitions in the given Rule-Definition.

Descriptor-Reference: Assigns a set of descriptors to the rule by reference to the associated Descriptor-Definition.

Descriptor-Id-Reference: Identifies the referred Descriptor-Definition

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, allows omitting a separate entry for each direction. When not present, the direction is implied by the Descriptor's values.

Action-Reference: Assigns a set of actions to the rule by reference to the associated Action-Definition.

Action-Id-Reference: Identifies the referred Action-Definition.

Action-Order: Defines the order how actions are executed in case the rule applies per match of the associated traffic descriptor.

Descriptor-Definition: A set of descriptor definitions, each being identified by a key (Descriptor-Id)

Descriptor-Id: Identifies a descriptor definition.

Descriptor-Type: Identifies the type of descriptor, e.g. an IPv6 traffic selector per [RFC 6088], for unambiguous 2 interpretation of a Descriptor-Value.

Descriptor-Value: Defines all required attribute-value pairs per the traffic descriptor type identified in Descriptor-Type.

Action-Definition: A set of action definitions.

Action-Id: Identifies an action definition.

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

Action-Value: Defines all required attribute-value pairs per the action type identified in Action-Type.

4.5. Configurable Policy

```
         |
         +-[Configurable-Policy] <Set>
         |               +-[DPN-Id-Reference] <U-Key>
         |               +-[Installed-Policy] <List>
         |               |           +-[Installed-Policy-Id] <G-Key>
         |               |           +-[Policy-Id-Reference]
         |               |           +-[Policy-Settings] <Set> (O)
         |               +-[Settings] <Set> (O)
         |
```

Figure 11: Configurable Policy

The Configurable-Policy Substructure contains the following entries:

DPN-Id-Reference: Refers to the DPN to which the policy applies.

Installed-Policy: A list of policies that apply to an identified DPN.

Installed-Policy-Id: Holds the identifier of the DPN at which the policy has been installed.

Policy-Id-Reference: Assigns a policy by reference to the associated Policy-Definition.

Policy-Settings: Settings that apply to the previously referenced policy to complement rules or make them concrete, e.g. in case the descriptor representing a packet match has not or unambiguously been defined in the poliicy sub-structure.

Settings: Settings that apply to multiple policies at the
   identified DPN.

4.6. Mobility-Context

The Mobility-Context Substructure holds entries associated with a mobile node's mobility sessions. At least one instance is created at the mobile node's registration to serve as parent context. Additional instances holding child context with reference to parent context can be created. Child context holds, for example, descriptors of mobile node data traffic which needs to be treated different from traffic that matches descriptor of the parent context, e.g. in terms of QoS. Child context can inherit some attributes/ values from parent context, such as traffic encapsulation and forwarding information, but can hold different or additional attributes/values that apply to traffic that matches the descriptor of the child context.

Termination a parent context implies termination of all dependent child context, e.g. at deregistration of a mobile node.

```
   |
   +-[Mobility-Context] <Set>
   |             +-[Mobility-Context-Id] <G-Key>
   |             +-[DPN-Group-Id-Reference] (O)
   |             +-[Parent-Mobility-Context-Id-Reference] (O)
   |             +-[DPN-References] <List>
   |             |          +-[DPN-Id-Reference]
   |             |          +-[Direction] (O)
   |             |          +-[DPN-Settings-Complementary]
   |             |          +-[Interface-Id-Reference]
   |             |          +-[Embedded-Rule] <Set> (O)
   |             |          +-[Assigned-Policy-Reference] <Set> (O)
   |             +-[Requested-Policy-Reference] <Set> (O)
   |             +-[Context-Settings-Complementary] <Set> (O)
```

Figure 12: Mobility Context

The Mobility-Context Substructure holds the following entries:

Mobility-Context-Id: Identifies a Mobility-Context

DPN-Group-Id-Reference: Assigns a DPN-Group, which groups DPN that
   are used during the DPN selection procedure, by reference.

Parent-Mobility-Context-Id-Reference: Assigns a parent Mobility-
   Context to aquire settings as required.

DPN-References: Holds a list of DPNs with the associated concrete policies that apply to the DPN. An entry MUST have at least one value present in its Assigned-Policy-Refence or Embedded-Rule sets in order to be valid.

DPN-Id-Reference: Assigns a DPN, to which the policy applies, by reference.

Direction: Indicates if a rule 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.

DPN-Settings-Complementary: Complementary seeings that apply to the DPN for the given Mobility-Context.

Interface-Id-Reference: Assigns the selected interface of the DPN my reference.

Embedded-Rule: Rule that applies to the DPN for this Mobility-Context. This rule is embedded and not refereced in the Policy substructure.

Assigned-Policy-Reference: A set of references to the list of Requested-Policy-References per this Mobility-Context.

Requested-Policy-Reference: Assigns a list of policies in Policy-Definitions that can apply to any DPN per this Mobility-Context. DPN-Reference entries can select from this list of policy references to apply to the associated DPN.

Context-Settings-Complementary: Complementary settings that apply to the referenced policies per this Mobility-Context.

The Rules Template format is aligned with the Rule Substructure. It can represent an Embedded-Rule per the Mobility-Context Substructure.

```
             |
             +-[Embedded-Rule] <List>
             |         +-[Rule-Id] (M)
             |         +-[Precedence] <L-Key>
             |         +-[Descriptor-Match-Type] (M)
             |         +-[Descriptor-Definition] <Set>
             |         |             +-[Descriptor-Id] <L-Key> (M)
             |         |             +-[Descriptor-Type]
             |         |             +-[Descriptor-Value]
             |         +-[Action-Definition] <Set>
             |                       +-[Action-Order] <L-Key> (M)
             |                       +-[Action-Id] (O)
             |                       +-[Action-Type]
             |                       +-[Action-Value]
```

Figure 13: Rule Template

The Embedded-Rule template holds the following entries:

Rule-Id: Identifies a rule definition. It is provided for convenience.

Precedence: Defines the order with which the rules must be applied and is used as the key.

Descriptor-Match-Type: Conjuction of Descriptor-Values to apply as match to DPN traffic, which can be either OR or AND. The identified conjunction applies to all Descriptor-Definitions in the given Rule-Definition

Descriptor-Definition: A set of descriptor definitions, each being identified by a key (Descriptor-Id)

Descriptor-Id: Identifies a Descriptor-Definition

Descriptor-Type: Identifies the type of descriptor, e.g. an IPv6 traffic selector per [RFC 6088], for unambiguous 2 interpretation of a Descriptor-Value

Descriptor-Value: Defines all required attribute-value pairs per the traffic descriptor type identified in Descriptor-Type.

Action-Definition: A set of action definitions.

Action-Order: Defines the order how actions are executed in case the rule applies per match of the associated traffic descriptor.

Action-Id: Identifies an action definition.

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

Action-Value: Defines all required attribute-value pairs per the action type identified in Action-Type.

4.7. Monitors

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

When a Monitor is specified, the configuration MUST be applicable to the attribute/entity monitored. For example, a Monitor using a Threshold configuration cannot be applied to a Context, because Contexts do not have thresholds. But such a monitor could be applied to a numeric threshold property of a Context.

```
                     |
                     +-[Monitor] <List>
                     |         +-[Monitor-Id] <U-Key>
                     |         +-[Target]
                     |         +-[Binding-Information] (O)
                     |         +-[Deterrable] (O)
                     |         +-[Configuration]
```

Figure 14: Monitor Substructure

Monitor-Id: Name of the Monitor. The Id format MUST conform to Section 4.8.

Target: Description of what is to be monitored. This can be an Event, a Dynamic Policy, an installed DPN Policy, or values of a Dynamic-Policy attribute. When the type is an attribute of Mobility-Context, the target name is a concatenation of the Context-Id and the relative path (separated by '/') to the attribute(s) to be monitored. Target must provide unambiguously identify the monitored attribute and the location (DPN).

Binding-Information: Complements (ambigous) Target information to define the Monitor in an unambigous way.

Deterrable: Indicates that a monitoring report can be delayed in a defined delay budget for possible bundling with other reports

Configuration: Determined by the Monitor subtype. The recipient of a notification as monitor report is specified in the Configuration. Four reporting types are defined:

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

* Event 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.

4.8. Namespace and Format

The identifiers and names in FPC models which reside in the same namespace must be unique. That uniqueness must be kept in agent or data-plane tenant namespace on an Agent. 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 Contexts in all tenants on an Agent, the Agent SHOULD define that policy to be visible from all the tenants. In this case, the Agent assigns an unique identifier in the agent namespace and effectively creates a U-Key although only a G-Key is required.

The format of 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 types of format that dictate the choice of FPC protocol. However the choice of identifiers which are used in Mobility model need to be considered to handle runtime parameters in real-time.

4.9. Attribute Application

When search for attributes in Settings to apply for Embeded Rules or Assigned-Policy-References the following search order is applied by the Agent / DPN for Mobilty-Contexts:

1. DPN-Settings-Complementary of the DPN-Reference entry

2. Context-Settings-Complementary of the Mobility-Context

3. If Parent-Mobility-Context-Id-Reference is not empty the following are also searched:

    1. DPN-Settings-Complementary of the DPN-Reference entry for the same interface, if present.

    2. Context-Settings-Complementary of this parent Mobility-Context

    3. Optionally, if this Mobility-Context has a non-empty Parent-Mobility-Context-Id-Reference this step MAY be repeated but it is NOT reccommended as it could slow system performance.

4. Interface-Settings of the DPN interface

Only looking at settings of the first Parent-Mobility-Context-Id-Reference is recommended for performance reasons. If further depth of search is supported the Agent MUST make this known to the Client.

For Configurable Policy the following search order is applied:

1. Policy-Settings of the Installed-Policy

2. Settings of the Configurable-Policy

5. Protocol

NOTE - The terms Context and Mobility-Context are used interchangeable throughout the rest of this document.

5.1. Protocol Messages and Semantics

Five message types are supported:

Message	Type	Description
CONF	HEADER OP_TYPE BODY	Configure processes a single operation.
CONF_BUNDLE	HEADER OP_TYPE REF_SCOPE TRANS_STRATEGY 1*[OP_ID BODY]	A Conf-bundle takes multiple operations that are to be executed as a group with partial failures allowed. They are executed according to the OP_ID value coupled with the BODY in ascending order. If a CONF_BUNDLE fails, any entities provisioned in the CURRENT operation are removed. However, any successful operations completed prior to the current operation are preserved in order to reduce system load.
REG_MONITOR	HEADER *[MONITOR]	Register a monitor at an Agent. The message includes information about the attribute to monitor and the reporting method. Note that a MONITOR_CONFIG is required for this operation.
DEREG_MONITOR	HEADER 1*[MONITOR_ID] [SEND_DATA]	Deregister monitors from an Agent. Monitor IDs are provided. SEND_DATA is an optional boolen that indicates if a successful DEREG triggers a NOTIFY with final data.
PROBE	HEADER MONITOR_ID	Probe the status of a registered monitor.

Table 1: Client to Agent Messages

Each message contains a header with the Client Identifier, an execution delay timer and an operation identifier. The delay, in ms, is processed as the delay for operation execution from the time the operation is received by the Agent.

The Client Identifier is used by the Agent to associate specific configuration characteristics, e.g. options used by the Client when

communicating with the Agent, as well as the association of the Client and tenant in the information model.

CONF_BUNDLE also has the Transaction Strategy (TRANS_STRATEGY) attribute. This value specifies the behavior of the Agent when an operation fails while processing a CONF_BUNDLE message. The value of 'default' uses the default strategy defined for the message. The value 'all_or_nothing' will roll back all successfully executed operations within the bundle as well as the operation that failed.

An FPC interface protocol used to support this specification may not need to support CONF_BUNDLE messages or specific TRANS_STRATEGY types beyond 'default' when the protocol provides similar semantics. However, this MUST be clearly defined in the specification that defines the interface protocol.

An Agent will respond with an ERROR, OK, or an OK WITH INDICATION that remaining data will be sent via a notify from the Agent to the Client Section 5.1.1.4.2 for CONFIG and CONF_BUNDLE requests. When returning an 'ok' of any kind, optional data MAY be present.

Two Agent notifications are supported:

Message	Type	Description
CONFIG_RESULT_NOTIFY	See Table 13	An asynchronous notification from Agent to Client based upon a previous CONFIG or CONF_BUNDLE request.
NOTIFY	See Table 14	An asynchronous notification from Agent to Client based upon a registered MONITOR.

Table 2: Agent to Client Messages (notifications)

The HEADER is a part of all messages and is comprised of the following information:

CLT_ID: The Client Identifier

DELAY (OPTIONAL): The time (in ms) to delay the execution of ther operaiton on the DPN once it is received by the Agent.

OP_ID: Operation Identifier

Results will be supplied per operation input. Each result contains the RESULT_STATUS and OP_ID that it corresponds to. RESULT_STATUS values are:

OK - Success

ERR - An Error has occurred

If an error occurs, information MUST be returned in the response. Error informaiton is comprised of the following:

ERROR_TYPE_ID (Unsigned 32, REQUIRED) - The identifier of a specific error type The values are TRANSPORT (0), RPC (1), PROTOCOL(2) or APPLICATION (3).

ERROR_TAG - (String, REQUIRED) - enumerated error tag.

ERROR_APP_TAG - (String, OPTIONAL) - Application specific error tag.

ERROR_MESSAGE - (String, OPTIONAL) - A message describing the error.

ERROR_INFO - (Any Data, OPTIONAL) - Any data required for the response.

5.1.1. CONFIG and CONF_BUNDLE Messages

CONFIG and CONF_BUNDLE include OP_TYPE as part of the header information:

OP_TYPE: specifies the type of operation. Valid values are 'create' (0), 'update' (1), 'query' (2) or 'delete' (3).

The BODY is comprised of the following information:

COMMAND_SET: Specifies the Command Set (see Section 5.1.1.2).

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

BODY INTERNALS: A list of entities under Policy, e.g. Policy-Definitions, Action-Definitions, etc. as well as Installed Policies and Contexts when the OP_TYPE is 'create' or 'update'. Otherwise it is a list of Targets for 'query' or 'deletion'. See Section 6.1.3.2 for details.

CONF_BUNDLES includes an OP_ID with each body for tracking of the bundle's subtransactions.

5.1.1.1. Agent Operation Processing

The Agent will process entities provided in an operation in the following order:

1. Action-Definitions
2. Descriptor Defintions
3. Rule Definitions
4. Policy Definitions
5. Installed Policies
6. Mobility Contexts according to COMMAND_SET

5.1.1.2. Command Bitsets

The COMMAND_SET is a technology specific bitset that allows for a single entity to be sent in an operation with multiple requested 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 is reserved for Agent based IP assignment.

5.1.1.3. Reference Scope

The Reference Scope is an optional feature that provides the scope of references used in a configuration command, i.e. CONFIG or CONF_BUNDLE. These scopes are defined as

- none - All entities have no references to other entities.
- op - All references are contained in the operation body, i.e. only intra-operation references exist.
- bundle - All references exist in bundle (inter-operation/intra-bundle). NOTE - If this value is present in a CONFIG message it is equivalent to 'op'.
- storage - One or more references exist outside of the operation and bundle. A lookup to cache / storage is required.

o unknown - the location of the references are unknown. This is treated as a 'storage' type.

An agent that only accepts 'op' or 'bundle' 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. Even when an Agent supports all message types an 'op' or 'bundle' scoped message can be processed quickly by the Agent as it does not require storage access.

5.1.1.4. Operation Response

5.1.1.4.1. Immediate Response

For CONF and CONF_BUNDLE the Response MAY include the the following:

NOTIFY_FOLLOWS - A boolean indicator that the Operation has been accepted by the Agent but further processing is required. A CONFIG_RESULT_NOTIFY will be sent once the processing has succeeded or failed.

ENTITIES - Optionally, entities created or partially fulfilled as part of the operation as specified in Table 12 For Clients that need attributes back quickly for call processing, the AGENT MUST respond back with an OK_NOTIFY_FOLLOWS and minimally the attributes assigned by the Agent in the response. These situations MUST be determined through the use of Command Sets (see Section 5.1.1.2).

5.1.1.4.2. Asynchronous Notification

A CONFIG_RESULT_NOTIFY occurs after the Agent has completed processing related to a CONFIG or CONF_BUNDLE request. It is an asynchronous communication from the Agent to the Client.

The values of the CONFIG_RESULT_NOTIFY are detailed in Table 13.

5.1.2. Monitors

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 also 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 is deterrable then the NOTIFY MAY be bundled with other messages back to the Agent even if this results in a delay of the NOTIFY.

5.1.2.1. Asynchronous Notification

A NOTIFY can be sent as part of de-registraiton, a trigger based upon a Monitor Configuration or a PROBE. A NOTIFY is comprised of unique Notification Identifier from the Agent, the Monitor ID the notification applies to, the Trigger for the notification, a timestamp of when the notification's associated event occurs and data that is specific to the monitored value's type.

5.2. Protocol Operation

5.2.1. Simple RPC Operation

An FPC Client and Agent MUST identify themselves using the CLI_ID and AGT_ID respectively to ensure that for all transactions a recipient of an 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_ID value in a Context. Otherwise the Agent selects a suitable DPN to enforce a Context and notifies the Client about the selected DPN using the DPN_ID.

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all entities as well as status information, which indicates the result of processing the message, using the RESPONSE_BODY property. In case the processing of the message results in a failure, the Agent sets the ERROR_TYPE and ERROR_TAG accordingly and MAY clear the entity, e.g. Context or Configurable-Policy, 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 indicaiton with an optional RESPONSE_BODY containing the partially completed entities. When a NOTIFY_FOLLOWS indication is indicated, the Agent will, upon completion or failure of the operation, respond with an asynchronous CONFIG_RESULT_NOTIFY to the Client.

A Client MAY add a property to a 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 back to the Client. If the processing will take too long or based upon Agent configuration, the Agent MAY respond with an OK that indicates a NOTIFY_FOLLOWS and also includes a RESPONSE_BODY containing the partially completed entities.

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 in the RESPONSE_BODY and sets the RESULT to Error, 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 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.

Figure 15 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.

```
                                                 +-------Router--------+
                           +-----------+         |+-------+ +---------+|
   +------+ +------+     +-----+ FPC   |          | FPC   | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|          | Agent | |   DPN   |
   +------+ +------+     +-----+-------+          +-------+ +---------+
   [MN attach]  |            |                          |           |
      |-------------PBU----->|                          |           |
      |         |            |---(1)--CONFIG(CREATE)--->|           |
      |         |            |   [ MOBILTY_CONTEXT_ID,  |--tun1 up->|
      |         |            | DPNREFLIST:[[DPN1, BOTH  |           |
      |         |            |  DPN_SETTINGS_COMPL:[    |           |
      |         |            |   DOWNLINK(QOS/TUN),     |           |
      |         |            |   UPLINK(QOS/TUN)] ]]    |--tc qos-->|
      |         |            |   CTXT_SETTINGS_COMPL:[  |           |
      |         |            |     IP_PREFIX(HNP) ] ]   |           |
      |         |            |<---(2)- OK --------------|-route add>|
      |         |            |                          |           |
      |<------------PBA------|                          |           |
      |         |            |                          |           |
      | +----+  |            |                          |           |
      | |Edge|  |            |                          |           |
      | |DPN1|  |            |                          |           |
      | +----+  |            |                          |           |
      |   |                                                         |
      |   |-=======================================================-|
      |                      |                          |           |
      |   [MN handover]      |                          |           |
      |         |---PBU ---->|                          |           |
      |         |            |--(3)- CONFIG(MODIFY)---->|           |
      |         |<--PBA------|    [ MOBILTY_CONTEXT_ID  |-tun1 mod->|
      |         |            | DPNREFLIST:[[DPN1, BOTH  |           |
      |         |            |    DPN_SETTINGS_COMPL:[  |           |
      |         |            |   DOWNLINK(TUN),         |           |
      |         |  +----+    |   UPLINK(TUN)] ]] ]      |           |
      |         |  |Edge|    |<---(4)- OK --------------|           |
      |         |  |DPN2|    |                          |           |
      |         |  +----+    |                          |           |
      |         |    |       |                          |           |
      |         |    |-============================================-|
      |         |            |                          |           |
```

Figure 15: Exemplary Message Sequence (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 Context to the DPN to treat the MN's traffic

(1) and includes a Context Identifier (CONTEXT_ID) to the CONFIG command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier and the Direction the entry applies to, BOTH.

The LMA-C adds properties during the creation of the new Context. One property is added to the DPN Settings Complimentary, to specify the forwarding tunnel type and endpoints (Anchor DPN, Edge DPN1) in each direction (as required). Another property is added to specify the QoS differentiation, which the MN's traffic should experience. At reception of the 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 Context's Complementary Settings 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 endpoints in the downlink and uplink, as required. The LMA-C sends a CONFIG message (3) to the Agent to modify the existing tunnel property of the existing Context and to update the tunnel endpoint from Edge DPN1 to Edge DPN2. Upon reception of the CONFIG message, the Agent applies updated tunnel property to the local configuration and responds to the Client (4).

```
                                                 +-------Router--------+
                           +-----------+         |+-------+ +---------+|
   +------+ +------+     +-----+ FPC   |          | FPC   | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|          | Agent | |   DPN   |
   +------+ +------+     +-----+-------+          +-------+ +---------+
   [MN attach]  |            |                          |           |
      |-------------PBU----->|                          |           |
      |         |            |---(1)--CONFIG(MODIFY)--->|           |
      |<------------PBA------|   [ CONTEXT_ID,          |--tun1   ->|
      |         |            | DPNREFLIST:[[DPN1, BOTH |           |
      |         |            |    DPN_SETTINGS_COMPL:[  |           |
      |         |            |   DOWNLINK(TUN delete),  |    down   |
      |         |            | UPLINK(TUN delete)] ]] ] |           |
      |         |            |                          |           |
      |         |            |<-(2)- OK ----------------|           |
      |         |            |                          |           |
      |         |  [ MinDelayBeforeBCEDelete expires ]  |           |
      |         |            |                          |           |
      |         |            |---(3)--CONFIG(DELETE)--->|-- tun1 -->|
      |         |            |                          |  delete   |
      |         |            |<-(4)- OK ----------------|           |
      |         |            |                          |-- route ->|
      |         |            |                          |   remove  |
      |         |            |                          |           |
```

Figure 16: Exemplary Message Sequence (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 CONFIG message (1) to the Agent to modify the existing tunnel property of the existing Context to delete the tunnel information.) Upon reception of the CONFIG 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 CONFIG (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 Conext.

```
                           +-----------+           +-------+ +---------+
   +------+ +------+     +-----+ FPC   |           | FPC   | |  Anchor |
   |MAG-C1| |MAG-C2|     |LMA-C| Client|           | Agent | |   DPN1  |
   +------+ +------+     +-----+-------+           +-------+ +---------+
   [MN attach]  |            |                          |           |
      |-------------PBU----->|                          |           |
      |         |            |---(1)--CONFIG(CREATE)--->|           |
      |         |            |   [ MOBILTY_CONTEXT_ID,  |--tun1 up->|
      |         |            | DPNREFLIST:[[DPN1, BOTH  |           |
      |         |            |    DPN_SETTINGS_COMPL:[  |           |
      |         |            |   DOWNLINK(QOS/TUN),     |           |
      |         |            |   UPLINK(QOS/TUN) ] ],   |--tc qos-->|
      |         |            |    [DPN2, BOTH           |           |
      |         |            |    DPN_SETTINGS_COMPL:[  |           |
      |         |            |   DOWNLINK(QOS/TUN),     |           |
      |         |            |   UPLINK(QOS/TUN) ] ] ], |           |
      |         |            |   CTXT_SETTINGS_COMPL [  |           |
      |         |            |     IP_PREFIX(HNP) ] ]   |           |
      |         |            |<-(2)- OK_NOTIFY_FOLLOWS -|-route add>|
      |         |            |                          |           |
      |<------------PBA------|                          |           |
      |         |            |                          |           |
      | +----+               |                          |           |
      | |Edge|               |                          |           |
      | |DPN2|               |                          |           |
      | +----+               |                          |           |
      |   |<---------------------- tun1 up -------------|           |
      |   |<---------------------- tc qos --------------|           |
      |   |<---------------------- route add -----------|           |
      |   |                  |                          |           |
      |   |                  |<(3) CONFIG_RESULT_NOTIFY |           |
      |   |                  |   [ Response Data ]      |           |
      |   |                  |                          |           |
```

Figure 17: Exemplary Message Sequence for Multi-DPN Agent

Figure 17 shows how the first 2 messages in Figure 15 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 Context. Message 1 shows the DPN Reference List with all entries. Each entry identifies the DPN and direction (one of 'uplink', 'downlink' or 'both').

The Agent responds with an OK and NOTIFY_FOLLOWS indication while it simultaneoulsy provisions both DPNs. Upon successful completion, the Agent responds to the Client with a CONFIG_RESULT_NOTIFY indicating the operation status.

5.2.2. Policy And Mobility on the Agent

A Client may build Policy and Topology using any mechanism on the Agent. Such entities are not always required to be constructed in realtime and, therefore, there are no specific messages defined for them in this specification.

The Client may add, modify or delete many Installed Policies and Contexts in a single FPC message. This includes linking Contexts to Actions and Descriptors, i.e. a Rule. 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 18 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.

```
          Descriptor_1 -+          +- Action_1
                        |          |
          Descriptor_2 -+--<Rule>--+- Action_2
                          +------+
                          /Order#/-------------+
                          +------+             |
                                               |
          Descriptor_3 -+          +- Action_3 +-<Policy>
                        |          |           |  ^
          Descriptor_4 -+--<Rule>--+- Action_4 |  |
                          +------+             |  |
                          /Order#/-------------+  |
                          +------+                |
                                                 <Intsalled-Policy>

          +---------------------+     +----------------------+
          | Bind 1..M traffic   |     |  Bind 1..N traffic   |
          |  Descriptors to     | --> |  treatment actions   |
          |  to a Policy and    |     |   to a Policy and    |
          | Configurable-Policy |     |  Configurable-Policy |
          +---------------------+     +----------------------+

         |                                                 |
         +-------------- Data-Plane Rule ------------------+
```

Figure 18: Structure of Configurable Policies

As depicted in Figure 18, the Configurable-Policy 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. A Client and Agent use the identifiers to access the Descriptors or Actions to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the treatment Actions specified in the list of properties associated with the Configurable-Policy.

A Client complements a rule's Descriptors with a Rule's Order (priority) value to allow unambiguous traffic matching on the Data-Plane.

Figure 19 illustrates the generic context configuration model as used between a FPC Client and a FPC Agent.

```
             TrafficSelector_1
                      |
             profile-parameters
                      |
             mobility-profile-- dl ------+
                                ^        |
                                |      qos-profile
                           <ContextID1>       |
                                ^        per-mn-agg-max-dl_2
                                |
                           <ContextID2>

             +-------------------+     +---------------------+
             | Bind 1..M traffic |     |  Bind 1..N traffic  |
             |    selectors to   | --> |  treatment / qos    |
             |     a Context     |     |  actions to a       |
             |                   |     |       Context       |
             +-------------------+     +---------------------+

            |                                                 |
            +-------------- Data-Plane Rule ------------------+
```

Figure 19: Structure of Contexts

As depicted in Figure 19, the Context 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 Context's properties. If present, the final action is to use a Context's tunnel information to encapsulate and forward the packet.

A second Context also references context1 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.

5.2.3. Optimization for Current and Subsequent Messages

5.2.3.1. Configuration Bundles

Bundles provide transaction boundaries around work in a single message. Operations in a bundle MUST be successfully executed in the order specified. This allows references created in one operation to be used in a subsequent operation in the bundle.

The example bundle shows in Operation 1 (OP 1) the creation of a Context 1 which is then referenced in Operation 2 (OP 2) by CONTEXT_ID 2. If OP 1 fails then OP 2 will not be executed. The advantage of the CONF_BUNDLE is preservation of dependency orders in a single message as opposed to sending multiple CONFIG messages and awaiting results from the Agent.

When a CONF_BUNDLE fails, any entities provisioned in the CURRENT operation are removed, however, any successful operations completed prior to the current operation are preserved in order to reduce system load.

```
                                +-------Router--------+
              +-----------+     |+-------+ +---------+|
              |   FPC     |      | FPC   | |  Anchor |
              |  Client   |      | Agent | |   DPN   |
              +-----------+      +-------+ +---------+
                   |                          |           |
                   |--CONF_BUNDLE(CREATE)---->|           |
                   | [ OP 1,                  |           |
                   |   [ MOBILTY_CONTEXT_ID 1,  |--tun1 up->|
                   | DPNREFLIST:[[DPN1, BOTH  |           |
                   |  DPN_SETTINGS_COMPL:[    |           |
                   |   DOWNLINK(QOS/TUN),     |           |
                   |   UPLINK(QOS/TUN)] ]]    |--tc qos-->|
                   |   CTXT_SETTINGS_COMPL:[  |           |
                   |     IP_PREFIX(HNP) ] ] ],|           |
                   | [ OP 2,                  |           |
                   |   [ MOBILTY_CONTEXT_ID,  |--tun1 up->|
                   | DPNREFLIST:[[DPN1, BOTH  |           |
                   |  DPN_SETTINGS_COMPL:[    |           |
                   |   DOWNLINK(QOS/TUN),     |           |
                   |   UPLINK(QOS/TUN)] ]]    |--tc qos-->|
                   | PARENT_CONTEXT_ID_REF=1  |           |
                   |  ] ]                     |           |
                   |                          |           |
```

Figure 20: Exemplary Bundle Message (focus on FPC reference point)

5.2.3.2. **Command Bitsets (Optional)**

Command Sets permit the ability to provide a single, unified data structure, e.g. Mobility-Context, and specify which activities are expected to be performed on the DPN. This has some advantages

o Rather than sending N messages with a single operation performed on the DPN a single message can be used with a Command Set that specifies the N DPN operations to be executed.

o Errors become more obvious. For example, if the HNP is NOT provided but the Client did not specify that the HNP should be assigned by the Agent this error is easily detected. Without the Command Set the default behavior of the Agent would be to assign the HNP and then respond back to the Client where the error would be detected and subsequent messaging would be required to remedy the error. Such situations can increase the time to error detection and overall system load without the Command Set present.

o Unambiguous provisioning specification. The Agent is exactly in sync with the expectations of the Client as opposed to guessing

what DPN work could be done based upon data present at the Agent. This greatly increases the speed by which the Agent can complete work.

- Permits different technologies with different instructions to be supported in FPC.

As Command Bitsets are technology specific, e.g. PMIP or 3GPP Mobility, the type of work varies on the DPN and the amount of data present in a Context or Port will vary. Using the technology specific instructions allows the Client to serve multiple technologies and MAY result in a more stateless Client as the instructions are transferred the Agent which will match the desired, technology specific instructions with the capabilities and over the wire protocol of the DPN more efficiently.

5.2.3.3. Reference Scope(Optional)

Although entities MAY refer to any other entity of an appropriate type, e.g. Contexts can refer to Policies or other Contexts, the Reference Scope gives the Agent an idea of where those references reside. They may be in the same operation, an operation in the same CONF_BUNDLE message or in storage. There may also be no references. This permits the Agent to understand when it can stop searching for reference it cannot find. For example, if a CONF_BUNDLE message uses a Reference Scope of type 'op' then it merely needs to keep an operation level cache and consume no memory or resources searching across the many operations in the CONF_BUNDLE message or the data store.

Agents can also be stateless by only supporting the 'none', 'op' and 'bundle' reference scopes. This does not imply they lack storage but merely the search space they use when looking up references for an entity. The figure below shows the caching hierarchy provided by the Reference Scope

Caches are temporarily created at each level and as the scope includes more caches the amount of entities that are searched increases. Figure 21 shows an example containment hierarchy provided for all caches.

```
                             +---------------+
                             | Global Cache  |
                             |  (storage)    |
                             +------+--------+
                                    |
                                    +----------------------+
                                    |                      |
                             +------+--------+      +------+--------+
                             | Bundle Cache  |      | Bundle Cache  |
                             |   (bundle)    | .... |   (bundle)    |
                             +------+--------+      +------+--------+
                                    |
               +--------------------+--------------------+
               |                    |                    |
      +--------+---------+ +--------+---------+ +--------+---------+
      | Operation Cache  | | Operation Cache  | | Operation Cache  |
      |       (op)       | |       (op)       | |       (op)       |
      +------------------+ +------------------+ +------------------+

                                (no cache)
```

Figure 21: Exemplary Hierarchical Cache

5.2.3.4. Basename Registry Feature (Optional)

The Optional BaseName Registry support feature is provided to permit Clients and tenants with common scopes, referred to in this specification as BaseNames, to track the state of provisioned policy information on an Agent. The registry records the BaseName and Checkpoint set by a Client. If a new Client attaches to the Agent it can query the Registry to determine the amount of work that must be executed to configure the Agent to a BaseName / checkpoint revision. A State value is also provided in the registry to help Clients coordinate work on common BaseNames.

6. Protocol Message Details

6.1. Data Structures And Type Assignment

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

6.1.1. Policy Structures

Structure	Field	Type
ACTION	ACTION_ID	FPC-Identity (Section 4.8)
ACTION	ACTION_TYPE	[32, unsigned integer]
ACTION	ACTION_VALUE	Type specific
DESCRIPTOR	DESCRIPTOR_ID	FPC-Identity (Section 4.8)
DESCRIPTOR	DESCRIPTOR_TYPE	[32, unsigned integer]
DESCRIPTOR	DESCRIPTOR_VALUE	Type specific
POLICY	POLICY_ID	FPC-Identity (Section 4.8)
POLICY	RULES	*[PRECEDENCE RULE_ID] PRECENDENCE is [32, unsigned integer]. For Rule see Table 4

Table 3: Action Fields

Policies contain a list of Rules by their order value. Each Rule contains Descriptors with optional directionality and Actions with order values that specifies action execution ordering if the Rule has multiple actions.

Rules consist of the following fields.

Field	Type	Sub-Fields
RULE_ID	FPC-Identity (Section 4.8)	
MATCH_TYPE	ENUMERATION [2, unsigned bits] ('AND' or 'OR')	
RULE_DESCRIPTORS	*[DESCRIPTOR_ID DIRECTION]	DIRECTION [2, unsigned bits] is an ENUMERATION (uplink, downlink or both).
RULE_ACTIONS	*[ACTION_ID ACTION_ORDER]	ACTION-ORDER [8, unsigned integer] specifies action execution order.

Table 4: Rule Fields

6.1.2. Mobility Structures

Field	Type
DPN_ID	FPC-Identity (Section 4.8)
1*[INSTALLED_POLICY_ID POLICY_ID_REFERENCE POLICY_SETTINGS] DPN_POLICY_SETTINGS	
INSTALLED_POLICY_ID	FPC-Identity (Section 4.8)
POLICY_ID_REFERENCE	POLICY_ID
POLICY_SETTINGS	A collection of policy specific setings (properties)
DPN_POLICY_SETTINGS	A collection of setings (properties) that affect multiple installed policies.

Table 5: Configurable-Policy Fields

Field	Type
MOBILITY_CONTEXT_ID	FPC-Identity (Section 4.8)
DPN_GROUP_ID_REFERENCE	FPC-Identity (Section 4.8)
PARENT_MOBILITY_CONTEXT_ID_REFERNCE	FPC-Identity (Section 4.8)
DPNS [NOTE 2]	*[DPN_REFERENCE]
REQUEST_POLICY_REFERENCES	* [POLICY_ID]
CONTEXT_SETTINGS_COMPLEMENTARY	A Collection of Settings (properties).

Table 6: Mobility Context Fields

Field	Type
DPN_ID	FPC-Identity (Section 4.8)
DIRECTION	See Table 4
INTERFACE_ID_REF	FPC-Identity (Section 4.8)
EMBEDDED_RULES	*[EMBEDDED_RULE]
DPN_SETTINGS_COMPLEMENTARY	A Collection of Settings (properties).
ASSIGNED_POLICY_REFERENCES	* [POLICY_ID]

Table 7: DPN_REFERENCE Fields

Field	Type
RULE_ID	FPC-Identity (Section 4.8)
MATCH_TYPE	See Table 4
PRECEDENCE	See Table 3
ACTION_DEFINITION_SET	*[ACTION_ORDER ACTION_ID ACTION_TYPE ACTION_VALUE]
DESCRIPTOR_DEFINITION_SET	*[DESCRIPTOR_ID DESCRIPTOR_TYPE DESCRIPTOR_VALUE]

Table 8: EMBEDDED_RULE Fields

6.1.2.1. Monitors

Field	Type	Description
MONITOR	MONITOR_ID DETERRABLE TARGET BINDING_INFORMATION [REPORT_CONFIG]	
DETERRABLE	boolean	Deterrability indicator.
BINDING_INFORMATION	String	
MONITOR_ID	FPC-Identity. See Section 4.8	
EVENT_TYPE_ID	[8, Event Type ID]	Event Type (unsigned integer).
TARGET	OCTET STRING (See Section 4.7)	
REPORT_CONFIG	[8, REPORT-TYPE] [TYPE_SPECIFIC_INFO]	
PERIODIC_CONFIG	[32, period]	report interval (ms).
THRESHOLD_CONFIG	[32, low] [32, hi]	thresholds (at least one value must be present)
SCHEDULED_CONFIG	[32, time]	
EVENTS_CONFIG	*[EVENT_TYPE_ID]	

Table 9: Monitor Structures and Attributes

TRIGGERS include but are not limited to the following values:

- o Events specified in the Event List of an EVENTS CONFIG
- o LOW_THRESHOLD_CROSSED
- o HIGH_THRESHOLD_CROSSED

- PERIODIC_REPORT
- SCHEDULED_REPORT
- PROBED
- DEREG_FINAL_VALUE

6.1.3. Message Attributes

6.1.3.1. Header

Each operation contains a header with the following fields:

Field	Type	Messages
CLIENT_ID	FPC-Identity (Section 4.8)	All
DELAY	[32, unsigned integer]	All
OP_ID	[64, unsigned integer]	All
OP_REF_SCOPE	[4, ENUMERATION]	Values are none(0), op(1), bundle(2), storage(3) or unknown(4)

Table 10: Message Header Fields

6.1.3.2. CONFIG and CONF_BUNDLE Attributes and Notifications

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
OP_TYPE	[8, op type]	CONFIG and CONF_BUNDLE
COMMAND_SET	FPC Command Bitset. See Section 5.1.1.2.	C,U
INSTALLED_POLICIES	*[INSTALLED_POLICY]	C,U
MOBILITY-CONTEXTS	*[MOBILITY-CONTEXT [COMMAND_SET [NOTE 1]]]	C,U
TARGETS	FPC-Identity (Section 4.8) *[DPN_ID]	Q,D
POLICIES	*[POLICY]	C,U
RULES	*[RULE]	C,U
DESCRIPTORS	*[DESCRIPTOR]	C,U
ACTIONS	*[ACTION]	C,U

Table 11: CONFIG and CONF_BUNDLE OP_BODY Fields

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
OP_ID	[64, unsigned integer]	All
STATUS	[1, Enumerated]	OK(0) or Error(1)
NOTIFY_FOLLOWS	boolean	
POLICIES	*[POLICY]	C,U
RULES	*[RULE]	C,U
DESCRIPTORS	*[DESCRIPTOR]	C,U
ACTIONS	*[ACTION]	C,U
INSTALLED_POLICIES	*[INSTALLED_POLICY]	C,U [NOTE 1]
CONTEXTS	*[CONTEXT [COMMAND_SET [NOTE 1]]]	C,U [NOTE 1]
TARGETS	*[FPC-Identity (Section 4.8)]	Q,D [NOTE 1]
ERROR_TYPE_ID	[32, unsigned integer]	All [NOTE 2]
ERROR_TAG	[1024, octet string]	All [NOTE 2, 3]

Table 12: Immediate Response RESPONSE_BODY Fields

Notes:

NOTE 1 - Present in OK and OK with NOTIFY_FOLLOWS for both CONFIG and CONF_BUNDLE. MAY also be present in an CONF_BUNDLE Error response (ERR) if one of the operations completed successfully.

NOTE 2 - Present only for Error (ERR) responses.

NOTE 3 - Other Error Info (Strings) MAY also be present.

Field	Type	Description
AGENT_ID	FPC-Identity (Section 4.8)	
OP_ID	[64, unsigned integer]	All
STATUS	[1, Enumerated]	OK(0) or Error(1)
NOTIFICATION_ID	[32, unsigned integer]	A Notification Identifier used to determine notification order.
TIMESTAMP	[32, unsigned integer]	The time that the notification occurred.
DATA	*[[OP_ID (if CONF_BUNDLE)] RESPONSE_BODY (Table 12)]	

Table 13: CONFIG_RESULT_NOTIFY Asynchronous Notification Fields

6.1.3.3. Monitors

Field	Type	Description
NOTIFICATION_ID	[32, unsiged integer]	
TIMESTAMP	[32, unsigned integer]	
CAUSE	[32, unsigned integer]	
NOTIFY	MONITOR [NOTIFICATION_DATA]	NOTIFICATION_DATA is the value of the monitored target if this is not ean error.

Table 14: Monitor Notifications

7. Derived and Subtyped Attributes

This section notes settings and derived attributes.

Field	Type	Detail
TUN_LOCAL_ADDRESS	IP Address	[NOTE 1]
TUN_REMOTE_ADDRESS	IP Address	[NOTE 1]
TUN_MTU	[32, unsigned integer]	
TUN_PAYLOAD_TYPE	[2, bits]	Enumeration: payload_ipv4(0), payload_ipv6(1) or payload_dual (2).
TUN_TYPE	[8, unsigned integer]	Enumeration: IP-in-IP(0), UDP(1), GRE(2) and GTP(3).
TUN_IF	[16, unsigned integer]	Input interface index.
MOBILITY_SPECIFIC_TUN_PARAMS	[IETF_PMIP_MOB_PROFILE | 3GPP_MOB_PROFILE]	[NOTE 1]
NEXTHOP	Varies	[NOTE 1] See Table 18.
QOS_PROFILE_PARAMS	[3GPP_QOS | PMIP_QOS]	[NOTE 1]

NOTE 1 - These parameters are extensible. The Types may be extended for Field value by future specifications or in the case of Vendor Specific Attributes by enterprises.

Table 15: Context Tunnel And QoS Settings

Field (Type Value)	Type Value	Type	Description
TO_PREFIX (0)	[IP Address] [Prefix Len]	Aggregated or per-host destination IP address/prefix descriptor.	FROM_PREFIX (1)
[IP Address] [Prefix Len]	Aggregated or per-host source IP address/prefix descriptor.	TRAFFIC_SELECTOR (2)	Format per specification [RFC6088].
Traffic Selector.			

Table 16: Descriptor Subtypes

Field (Type Value)	Type	Description
DROP (0)	Empty	Drop the associated packets.
REWRITE (1)	[in_src_ip] [out_src_ip] [in_dst_ip] [out_dst_ip] [in_src_port] [out_src_port] [in_dst_port] [out_dst_port]	Rewrite IP Address (NAT) or IP Address / Port (NAPT).
COPY_FORWARD (2)	FPC-Identity. See Section 4.8.	Copy all packets and forward them to the provided identity. The value of the identity MUST be a port or context.

Table 17: Action Subtypes

Field (Type Value)	Type	Description
IP_ADDR (0)	IP Address	An IP Address.
MAC_ADDR (1)	MAC Address	A MAC Address.
SERVICE_PATH_ID (2)	[24, unsigned integer]	Service Path Identifier (SPI)
MPLS_LABEL (3)	[20, unsigned integer]	MPLS Label
NSH (4)	[SERVICE_PATH_ID] [8, unsigned integer]	See [I-D.ietf-sfc-nsh]
INTERFACE_INDEX (5)	[16, unsigned integer]	Interface Index (an unsigned integer).
SEGMENT_ID (6)	[128, unsigned integer]	Segement Identifier.
MPLS_LABEL_STACK (7)	7	1*[20 bit labels]
MPLS SR Stack. See [I-D.ietf-spring-segment-routing-mpls].	SRV6_STACK (8)	32+ Bytes
See [I-D.ietf-6man-segment-routing-header].		

Table 18: Next Hop Subtypes

Field	Type Value	Type	Description
QOS	0	[qos index type] [index] [DSCP]	Refers to a single index and DSCP to write to the packet.
GBR	1	[32, unsigned integer]	Guaranteed bit rate.
MBR	2	[32, unsigned integer]	Maximum bit rate.
PMIP_QOS	3	Varies by Type	A non-traffic selector PMIP QoS Attribute per [RFC7222]

Table 19: QoS Subtypes

Field	Type Value	Type	Description
IPIP_TUN	0		IP in IP Configuration
UDP_TUN	1	[src_port] [dst_port]	UDP Tunnel - source and/or destination port
GRE_TUN	2	[32, GRE Key]	GRE Tunnel.

Table 20: Tunnel Subtypes

The following COMMAND_SET values are supported for IETF_PMIP.

o assign-ip - Assign the IP Address for the mobile session.

o assign-dpn - Assign the Dataplane Node.

o session - Assign values for the Session Level.

o uplink - Command applies to uplink.

o downlink - Command applies to downlink.

7.1. 3GPP Specific Extenstions

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

APN-AMBR: Access Point Name Aggregate Maximum Bit Rate

ARP: Allocation of Retention Priority

EBI: EPS Bearer Identity

GBR: Guaranteed Bit Rate

GTP: GPRS (General Packet Radio Service) Tunneling Protocol

IMSI: International Mobile Subscriber Identity

MBR: Maximum Bit Rate

QCI: QoS Class Identifier

TEID: Tunnel Endpoint Identifier.

TFT: Traffic Flow Template (TFT)

UE-AMBR: User Equipment Aggregate Maximum Bit Rate

NOTE: GTP Sequence Number (SEQ_NUMBER) is used in failover and handover.

Field	Type Value	Namespace / Entity Extended	Type
GTPV1	3	Tunnel Subtypes namespace.	LOCAL_TEID REMOTE_TEID SEQ_NUMBER
GTPV2	4	Tunnel Subtypes namespace.	LOCAL_TEID REMOTE_TEID SEQ_NUMBER
LOCAL_TEID	N/A	N/A	[32, unisgned integer]
REMOTE_TEID	N/A	N/A	[32, unisgned integer]
SEQ_NUMBER	N/A	N/A	[32, unisgned integer]
TFT	3	Descriptors Subtypes namespace.	Format per TS 24.008 Section 10.5.6.12.
IMSI	N/A	Context (new attribute)	[64, unsigned integer]
EBI	N/A	Context (new attribute)	[4, unsigned integer]
3GPP_QOS	4	QoS Subtypes namespace.	[8, qci] [32, gbr] [32, mbr] [32, apn_ambr] [32, ue_ambr] ARP
ARP	N/A	N/A	See Allocation-Retention-Priority from [RFC7222]

Table 21: 3GPP Attributes and Structures

The following COMMAND_SET values are supported for 3GPP.

o assign-ip - Assign the IP Address for the mobile session.

o assign-dpn - Assign the Dataplane Node.

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

8. 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 03 was known as fpcagent and version 04's implementation is simply referred to as 'fpc'. A third has been devloped 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 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 lead 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 colocated 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 2-5K FPC Contexts processed per second on a 2013 Mac laptop. This results in 2x the number of transactions on the southbound interface to a proprietary DPN API on the same machine.

Current v04 performance results without code optimizations or tuning allow 1-2K FPC Contexts processed per second on a 2013 Mac laptop. 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 22: 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.

9. Security Considerations

Detailed protocol implementations for DMM Forwarding Policy Configuration must ensure integrity of the information exchanged between an FPC Client and an 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 is 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 defintion 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.

Nodes under the Mobility Tree are runtime only 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 Context along with their associated tunnel configurations/identifiers (from the FPC base module)

Internaional Mobile Subscriber Identity (IMSI) and bearer identifiers in the Context when using the optional 3GPP module

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:

CONFIG and CONF_BUNDLE send 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 specicially provide sensitive or vulnerable informaiton but care must be taken by users to avoid identifier values that expose sensitive or vulnerable information.

Notications MUST be treated with same level of protection and scrutiny as the operations they correspond to. For example, a CONFIG_RESULT_NOTIFY notification provides the same information that is sent as part of the input and output of the CONFIG and CONF_BUNDLE RPC operations.

General usage of FPC MUST consider the following:

FPC Naming Section 4.8 permits arbirtrary string values but a users 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.

10. 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.
```

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
```

11. 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.

12. References

12.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-07 (work in progress), July 2017.

[I-D.ietf-sfc-nsh]
Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", draft-ietf-sfc-nsh-27 (work in progress), October 2017.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-10 (work in progress), June 2017.

[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>.

[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>.

[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>.

12.2. Informative References

[I-D.bertz-dime-policygroups]
Bertz, L. and M. Bales, "Diameter Policy Groups and Sets", draft-bertz-dime-policygroups-04 (work in progress), June 2017.

[I-D.ietf-dmm-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-02 (work in progress), August 2017.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

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

[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>.

[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>.

Appendix A. YANG Data Model for the FPC protocol

These modules define YANG definitions. When mapping from the FPC model was performed U-Key values, ACTION_TYPES and DESCRIPTOR_TYPES as well as many enumerations were mapped to identity types. ACTION_TYPES and DESCRIPTOR_TYPES are also optional as the type can be inferred from the value. Four modules are defined:

o ietf-dmm-fpc (fpc) - Defines the base model and messages for FPC that are meant to be static in FPC.

- ietf-dmm-fpc-settingsext An 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 RFC 6088

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@2017-09-27.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-restconf { prefix restconf;
        revision-date 2017-01-26; }
    import ietf-dmm-fpc-settingsext { prefix fpcbase;
    revision-date 2017-09-27;
  }

    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 2017-09-27 {
       description "Version 10 updates.";
       reference "draft-ietf-dmm-fpc-cpdp-10";
     }
   revision 2017-07-22 {
       description "Version 08 updates.";
       reference "draft-ietf-dmm-fpc-cpdp-08";
   }
   revision 2017-03-08 {
       description "Version 06 updates.";
       reference "draft-ietf-dmm-fpc-cpdp-06";
   }
   revision 2016-08-03 {
       description "Initial Revision.";
       reference "draft-ietf-dmm-fpc-cpdp-05";
   }
       feature fpc-basename-registry {
         description "Ability to track Base Names already provisioned
           on the Agent";
       }
       feature fpc-bundles {
         description "Ability for Client to send multiple bundles of
           actions to an Agent";
       }
       feature fpc-auto-binding {
         description "Allows a FPC Agent to advertise Topology Objects
           that could be DPNs";
       }
       feature operation-ref-scope {
```

```
         description "Provides the scope of refeneces in an operation.
           Used to optmize the Agent processing.";
       }

     //General Structures
       typedef fpc-identity {
           type union {
               type uint32;
               type string;
               type instance-identifier;
           }
           description "FPC Identity";
       }
       grouping target-value {
         leaf target {
             type fpc-identity;
         mandatory true;
         description "Target Identity";
         }
         description "FPC Target Value";
       }
     // Topology
     typedef fpc-interface-id {
           type fpc:fpc-identity;
           description "DPN interface Identifier";
     }
       identity interface-protocols {
           description "Protocol supported by the interface";
       }
       identity features {
           description "Protocol features";
       }
     // Settings
     grouping settings {
       container settings-set {
         uses fpcbase:fpc-settings;
         description "Settings";
       }
       description "Settings container";
     }
     //Topology - Groupings
     grouping interface-settings {
       container interface-settings-set {
         description "Interface settings";
       }
       description "Generic interface settings container";
     }
     grouping access-technology-key {
```

```
         leaf access-technology {
           type identityref {
             base "fpcbase:access-technology";
           }
           mandatory true;
           description "Access Technology";
         }
         description "Access Technology key";
       }
       grouping role-key {
         leaf role {
           type identityref {
             base "fpcbase:role";
           }
           mandatory true;
           description "Access Technology Role";
         }
         description "Access Technology Role key";
       }
       grouping interface-id-key {
         leaf interface-id {
           type fpc:fpc-interface-id;
           mandatory true;
           description "interface identifier";
         }
         description "Interface Identifier key";
       }
       grouping dpn-identifier-key {
         leaf dpn-id {
           type fpc:fpc-identity;
           mandatory true;
           description "DPN Identifier Type";
         }
         description "DPN Identifier key";
       }
       grouping dpn-interface-reference {
         uses fpc:access-technology-key;
         uses fpc:role-key;
         uses fpc:interface-id-key;
         description "A reference to a DPN interface";
       }
       // Topology Grouping
       grouping fpc-topology {
         list dpn-set {
           key dpn-id;
           uses fpc:dpn-identifier-key;
           leaf dpn-name {
             type string;
```

```
            description "DPN name";
          }
          leaf dpn-resource-mapping-reference {
            type string;
            description "Reference to underlying DPN resource(s)";
          }
          list interface-set {
            key "access-technology role interface-id";
            uses fpc:dpn-interface-reference;
            uses fpc:interface-settings;
            description "DPN interfaces";
          }
          description "Set of DPNs";
        }
        list dpn-type-set {
          key "access-technology role";
          uses fpc:access-technology-key;
          leaf access-technology-name {
            type string;
            description "Access Technology Name";
          }
          uses fpc:role-key;
          leaf role-name {
            type string;
            description "Access Technology Role Name";
          }
          list interface-set {
            key interface-id;
            uses fpc:interface-id-key;
            leaf interface-name {
              type string;
              description "DPN-Type Interface Name";
            }
            leaf-list interface-protocol-set {
              type identityref {
                base "interface-protocols";
              }
              description "Supported protocols";
            }
            leaf-list feature-set {
              type identityref {
                base "interface-protocols";
              }
              description "Supported features";
            }
            uses fpc:interface-settings;
            description "A DPN interface types";
          }
```

```
         description "Set of DPN types";
       }
       list dpn-group-set {
         key "dpn-group-id";
         leaf dpn-group-id {
           type fpc:fpc-identity;
           mandatory true;
           description "DPN Group Identifier";
         }
         list referenced-dpns-set {
           key "access-technology role interface-id";
           uses fpc:dpn-interface-reference;
           leaf-list supporting-dpn-id-set {
             type fpc:fpc-identity;
             description "DPNs that suppport this group";
           }
           leaf-list dpn-group-peer-id-set {
             type fpc:fpc-identity;
             description "DPN Peer Groups reference";
           }
           description "A list of DPNs supporting a group
             by DPN Type";
         }
         list dpn-group-peer-set {
           key remote-dpn-group-id;
           leaf remote-dpn-group-id {
             type fpc:fpc-identity;
             mandatory true;
             description "Remote DPN Group identifier";
           }
           uses fpc:interface-settings;
           description "Locally applied settings used for
             the referenced DPN-Group (peer group).";
         }
         leaf domain-id {
           type fpc:fpc-identity;
           description "Domain Identiifer";
         }
         description "List of DPN groups";
       }
       list domain-set {
         key domain-id;
         leaf domain-id {
           type fpc:fpc-identity;
           mandatory true;
           description "Domain Identifier";
         }
         leaf domain-name {
```

```
            type string;
            description "Domain displayname";
          }
          leaf domain-reference {
            type string;
            description "Reference to domain resources";
          }
          description "List of Domains";
        }
        description "FPC Topology grouping";
      }
      // Policy Structures
        // Descriptor Structure
        identity fpc-descriptor-type {
            description "A traffic descriptor";
        }
        grouping descriptor-definition {
            leaf descriptor-id {
                type fpc:fpc-identity;
                mandatory true;
                description "Descriptor Id";
            }
            leaf descriptor-type {
                type identityref {
                  base "fpc-descriptor-type";
                }
                description "Descriptor Type Value";
            }
        uses fpcbase:fpc-descriptor-value;
            description "FPC Descriptor Definition";
        }
        // Action Structure
        identity fpc-action-type {
            description "Action Type";
        }
        grouping action-definition {
            leaf action-id {
                type fpc:fpc-identity;
                description "Action Identifier";
            }
            leaf action-type {
                type identityref {
                  base "fpc-action-type";
                }
                description "Action Type";
            }
            uses fpcbase:fpc-action-value;
            description "FPC Action Definition";
```

```
       }
       // Rule Structure
       typedef fpc-direction-type {
          type enumeration {
            enum uplink {
              description "uplink";
            }
            enum downlink {
              description "Downlink";
            }
            enum both {
              description "Both";
           }
          }
          description "FPC Direction";
       }
       grouping fpc-rule-id {
       leaf rule-id {
         type fpc:fpc-identity;
         mandatory true;
         description "Rule Identifier";
       }
       description "FPC Rule-Id key";
       }
       grouping match-type {
           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-reference-set.";
         }
           description "Map Type Grouping";
       }
       grouping fpc-action-order {
           leaf action-order {
               type uint32;
               mandatory true;
               description "Action Execution Order";
           }
```

```
          description "Action Order Leaf";
      }
      grouping rule-definition {
          uses fpc:fpc-rule-id;
      uses fpc:match-type;
          list descriptor-reference-set {
            key "descriptor-id-reference";
            leaf descriptor-id-reference {
                type fpc:fpc-identity;
                mandatory true;
                description "Descriptor Id Reference";
            }
            leaf direction {
                type fpc:fpc-direction-type;
                description "Direction";
            }
            description "A set of Descriptor references";
          }
          list action-reference-set {
            key "action-order";
        uses fpc:fpc-action-order;
            leaf action-id-reference {
        type fpc:fpc-identity;
        mandatory true;
        description "Action Identifier Reference";
            }
            description "A set of Action references";
          }
          description
            "Rule.Definition";
      }
      // Policy Structures
      grouping fpc-precedence {
      leaf precedence {
        type uint32;
        mandatory true;
        description "Rule Precedence";
      }
      description "FPC Rule Precedence";
      }
      grouping policy {
          leaf policy-id {
              type fpc:fpc-identity;
              description "Policy Identifier";
          }
          list rule-set {
              key "precedence";
              unique "rule-id-reference";
```

```
               uses fpc:fpc-precedence;
               leaf rule-id-reference {
                   type fpc:fpc-identity;
                   mandatory true;
                   description "Rule Identifier";
               }
               description "Rule Entry";
           }
           description "FPC Policy";
       }
       // FPC Policy
     grouping fpc-policy {
       list action-definition-set {
         key action-id;
         uses fpc:action-definition;
         description "List of Actions";
       }
       list descriptor-definition-set {
         key descriptor-id;
         uses fpc:descriptor-definition;
         description "List of Descriptors";
       }
       list rule-definition-set {
         key rule-id;
         uses fpc:rule-definition;
         description "List of Rules";
       }
       list policy-definition-set {
         key policy-id;
         uses fpc:policy;
         description "List of Policies";
       }
       description "FPC Policy Structures";
     }
       // Mobility Structures
     grouping configurable-policy-set {
       list installed-policy-list {
         key dpn-id-reference;
         leaf dpn-id-reference {
           type fpc:fpc-identity;
           description "Installed Policy identifier";
         }
         list installed-policy-set {
           key installed-policy-id;
           leaf installed-policy-id {
             type fpc:fpc-identity;
             description "Installed Policy Identifier";
           }
```

```
           leaf policy-id-reference {
             type fpc:fpc-identity;
             description "Installed Policy Identifier";
           }
           container policy-settings {
             uses fpcbase:fpc-settings;
             description "Policy Settings";
           }
           description "Policy installed upon a DPN";
         }
         description "Configurable Policy";
         uses fpc:settings;
       }
       description "List of installed DPN policies and settings";
     }
       // Dynamic Policy
       grouping mobility-context {
       leaf mobility-context-id {
         type fpc:fpc-identity;
         mandatory true;
         description "Mobility Context Identifier";
       }
       leaf dpn-group-id-reference {
         type fpc:fpc-identity;
         description "Group ID used when DPN selecitons were
           made";
       }
       leaf parent-mobility-context-id-reference {
         type fpc:fpc-identity;
         description "Parent Mobility Context";
       }
       list dpn-reference-list {
         key "dpn-id-reference direction";
         leaf dpn-id-reference {
           type fpc:fpc-identity;
           mandatory true;
           description "DPN Id reference";
         }
         leaf direction {
           type fpc:fpc-direction-type;
           mandatory true;
           description "Direction of DPN assignment";
         }
         container dpn-settings-complementary {
           uses fpcbase:fpc-settings;
           description "Complentary Settings";
         }
         leaf interface-id-reference {
```

```
           type fpc:fpc-interface-id;
           mandatory true;
           description "referenced interface";
         }
         list embedded-rule-set {
           key "precedence";
           unique "rule-id";
           uses fpc:fpc-rule-id;
           uses fpc:match-type;
           uses fpc:fpc-precedence;
           list action-definition-set {
             key "action-order";
             uses fpc:fpc-action-order;
             uses fpc:action-definition;
             description "List of Actions";
           }
           list descriptor-definition-set {
             key descriptor-id;
             uses fpc:descriptor-definition;
             description "List of Descriptors";
           }
           description "List of FPC Embedded Rule Definitions";
         }
         leaf-list assigned-policy-reference-set {
           type fpc:fpc-identity;
           description "List of Policies request to be enforced for
             this Mobility Context";
         }
         description "DPN List";
       }

       leaf-list requested-policy-reference-set {
         type fpc:fpc-identity;
         description "List of Policies request to be enforced for
           this Mobility Context";
       }
       container context-settings-complementary {
         uses fpcbase:fpc-settings;
         description "Context Settings";
       }
       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-id {
       leaf monitor-id {
         type fpc:fpc-identity;
         mandatory true;
         description "Monitor Identifier";
       }
       description "Monitor Id";
     }
       grouping monitor-config {
           uses fpc:monitor-id;
       leaf deterrable {
         type boolean;
         description "Indicates reports related to this
           config can be delayed.";
       }
       container binding-information {
         description "Placeholder for information helpful
           to binding the monitor ot the correct target";
       }
       uses fpc:target-value;
       choice configuration {
         mandatory true;
         case periodic-config {
           leaf period {
             type uint32;
             description "Period";
           }
           description "Periodic Config Case";
         }
         case threshold-config {
           leaf lo-thresh {
             type uint32;
             description "lo threshold";
           }
           leaf hi-thresh {
             type uint32;
             description "hi threshold";
           }
           description "Threshold Config Case";
         }
         case scheduled-config {
           leaf report-time {
             type uint32;
             description "Reporting Time";
           }
           description "Scheduled Config Case";
```

```
        }
        case events-config-ident {
          leaf-list event-identities {
            type identityref {
              base "fpc:event-type";
            }
            description "Event Identities";
          }
          description "Events Config Identities Case";
        }
        case events-config {
          leaf-list event-ids {
            type uint32;
              description "Event IDs";
          }
          description "Events Config Case";
        }
        description "Event Config Value";
      }
      description "Monitor Configuration";
    }
      grouping report {
        uses fpc:monitor-config;
        choice report-value {
          leaf trigger {
            type fpc:event-type-id;
            description "Trigger Identifier";
          }
          case simple-empty {
            leaf nothing {
              type empty;
              description "Empty Value";
            }
            description "Empty Case";
          }
          case simple-val32 {
            leaf val32 {
              type uint32;
              description "Unsigned 32 bit value";
            }
            description "Simple Value Case";
          }
          case list-simple-val32 {
              leaf-list  val32-list {
                type uint32;
                description "Unsigned 32 bit value";
              }
              description "Simple Value Case";
```

```
             }
           description "Report Value";
         }
         description "Monitor Report";
       }
       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-state {
               type string;
               description "Current State";
             }
             leaf base-checkpoint {
               type string;
               description "Checkpoint";
             }
             description "Basename Information";
       }
       // Top Level Structures
     container tenants {
       list tenant {
         key "tenant-id";
         leaf tenant-id {
             type fpc:fpc-identity;
             description "Tenant ID";
         }
             container mobility {
              container topology {
                  uses fpc:fpc-topology;
                uses fpc:basename-info {
                  if-feature fpc:fpc-basename-registry;
                }
                 description "Topology";
               }
              container policy {
                uses fpc:fpc-policy;
                uses fpc:basename-info {
                  if-feature fpc:fpc-basename-registry;
```

```
                }
                description "Policy";
              }
               uses fpc:configurable-policy-set;
               list mobility-context-set {
                   key "mobility-context-id";
                   config false;
                   uses fpc:mobility-context;
                   description "Mobility Context Set";
               }
               list monitor-set {
                 key monitor-id;
                 uses fpc:monitor-config;
                 description "Monitor Configurations";
               }
               description "Mobility Elements";
             }
         description "Tenant";
       }
       description "Tenant List";
     }
       // RPC
       // RPC Specific Structures
       typedef op-identifier {
           type uint64;
           description "Operation Identifier";
       }
       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.";
        }
        grouping context-operation {
          uses fpc:mobility-context;
          uses fpcbase:instructions;
          description "Context Operation";
        }
        grouping payload {
            uses fpc:configurable-policy-set;
        list mobility-context-set {
            key "mobility-context-id";
            uses fpc:mobility-context;
            description "Mobility Context Set";
        }
        uses fpc:fpc-policy;
        description "Payload";
      }
        grouping op-header {
        leaf client-id {
          type fpc:client-identifier;
          mandatory true;
          description "Client ID";
        }
        leaf delay {
          type uint32;
          description "Operation Delay (ms)";
        }
        leaf op-id {
          type op-identifier;
          mandatory true;
          description "Operation Identifier";
        }
        description "Common Operation header";
        }
      grouping fpc-op-type {
        leaf op-type {
          type enumeration {
            enum create {
              value 0;
              description "create";
            }
            enum update {
              value 1;
              description "update";
            }
            enum query {
```

```
            value 2;
            description "query";
          }
          enum delete {
            value 3;
            description "delete";
          }
        }
        mandatory true;
        description "Type";
      }
      description "FPC Operation Type";
    }
    grouping fpc-op-ref-scope {
      leaf op-ref-scope {
        if-feature operation-ref-scope;
        type fpc:ref-scope;
        description "Reference Scope";
      }
      description "FPC OP-REF Scope";
    }
    grouping op-input {
      uses fpc:fpc-op-ref-scope;
      uses fpcbase:instructions;
      choice op_body {
        case create_or_update {
          uses fpc:payload;
          description "Create/Update input";
        }
        case delete_or_query {
          uses fpc:target-value;
          description "Delete/Query input";
        }
        description "Opeartion Input value";
      }
      description "Operation Input";
    }
    typedef result-status {
      type enumeration {
        enum ok {
          value 0;
          description "OK";
        }
        enum err {
          value 1;
          description "Error";
        }
      }
```

```
       description "Result Status";
     }
     grouping status-value {
       leaf op-id {
         type op-identifier;
         mandatory true;
         description "Operation Identifier";
       }
       leaf status {
         type result-status;
         mandatory true;
         description "Status";
       }
       description "Status value for all messages";
     }
     grouping result {
       uses fpc:status-value;
       uses fpc:result-body;
       description "General Result grouping";
     }
     grouping result-body {
       leaf notify-follows {
         type boolean;
         description "Indicates that a notification will
           follow regarding this result";
       }
       choice result-type {
         case err {
           uses restconf:errors;
           description "Error Information";
         }
         case create-or-update-success {
           uses fpc:payload;
           description "Create/Update Success";
         }
         case delete_or_query-success {
           uses fpc:target-value;
           description "Delete/Query Success";
         }
         case empty-case {
           description "Empty Case";
         }
         description "Result Value";
       }
       description "Common Result member body";
     }
     // Common RPCs
     rpc configure {
```

```
         description "CONF message";
         input {
           uses fpc:op-header;
           uses fpc:fpc-op-type;
           uses fpc:op-input;
         }
         output {
           uses fpc:result;
         }
       }
       rpc configure-bundles {
         if-feature fpc:fpc-bundles;
         description "CONF_BUNDLES message";
         input {
           uses fpc:op-header;
           uses fpc:fpc-op-type;
           uses fpc:fpc-op-ref-scope;
           list bundles {
             key op-id;
             leaf op-id {
               type op-identifier;
               mandatory true;
               description "Operation Identifier";
             }
             uses fpc:op-input;
             description "List of operations";
           }
         }
         output {
           uses fpc:status-value;
           list bundles {
             key op-id;
             uses fpc:result;
             description "Operation Identifier";
           }
         }
       }
       rpc reg_monitor {
         description "Used to register monitoring of parameters/events";
         input {
           uses fpc:op-header;
           uses fpc:monitor-config;
         }
         output {
           uses fpc:status-value;
           uses restconf:errors;
         }
       }
```

```
     rpc dereg_monitor {
       description "Used to de-register monitoring of
         parameters/events";
       input {
         uses fpc:op-header;
         leaf-list monitor-set {
           type fpc:fpc-identity;
           min-elements 1;
           description "Monitor Identifier";
         }
         leaf send_data {
           type boolean;
           description "Indicates if NOTIFY with final data
             is desired upon deregistration";
         }
       }
       output {
         uses fpc:status-value;
         uses restconf:errors;
       }
     }
     rpc probe {
       description "Probe the status of a registered monitor";
       input {
         uses fpc:op-header;
         uses fpc:monitor-id;
       }
       output {
         uses fpc:status-value;
         uses restconf:errors;
       }
     }
     // Notification Messages & Structures
     grouping notification-header {
       leaf notification-id {
         type uint32;
         description "Notification Identifier";
       }
       leaf timestamp {
         type uint32;
         description "timestamp";
       }
       description "Notification Header";
     }
     notification config-result-notification {
       uses fpc:notification-header;
       uses fpc:status-value;
       choice value {
```

```
          case config-result {
            uses result-body;
            description "CONF Result";
          }
          case config-bundle-result {
            list bundles {
              key op-id;
              uses fpc:result;
              description "Operation Identifier";
            }
            description "CONF_BUNDLES Result";
          }
          description "Config Result value";
        }
        description "CONF/CONF_BUNDLES Async Result";
      }
      identity notification-cause {
        description "Notification Cause";
      }
      identity dpn-availabilty-change {
        base "notification-cause";
        description "DPN Candidate/Exisitng DPN Availablity Change";
      }
      identity monitoring-suspension {
        base "notification-cause";
        description "Indicates monitoring suspension";
      }
      identity monitoring-resumption {
        base "notification-cause";
        description "Indicates that monitoring has resumed";
      }
      identity monitor-notification {
        base "notification-cause";
        description "Indicates 1+ monitor reports";
      }
      notification notify {
        uses fpc:notification-header;
        leaf cause {
          type identityref {
            base "notification-cause";
          }
          description "Notification Cause";
        }
        choice value {
          case dpn-candidate-available {
            if-feature fpc:fpc-auto-binding;
            leaf node-id {
              type inet:uri;
```

```
            description "Topology URI";
          }
          list supported-interface-list {
            key "access-technology role";
            uses fpc:access-technology-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";
          }
          description "DPN Unavailable";
        }
        case monitor-notification {
          list reports {
            uses fpc:report;
            description "Reports";
          }
          description "Monitor Notification";
        }
        description "Notify Value";
      }
      description "Notify Message";
    }
  }
  <CODE ENDS>
```

A.2. YANG Models

A.2.1. 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@2017-09-27.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 2017-10-29; }
       import ietf-yang-types { prefix ytypes;
           revision-date 2013-07-15; }
       import ietf-pmip-qos { prefix pmipqos;
           revision-date 2016-02-10; }

       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).

         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 2017-09-27 {
```

```
        description "Version 10 updates.";
        reference "draft-ietf-dmm-fpc-cpdp-10";
    }
    revision 2017-07-22 {
        description "Version 08 updates.";
        reference "draft-ietf-dmm-fpc-cpdp-08";
    }
    revision 2017-03-08 {
        description "Version 06 updates.";
        reference "draft-ietf-dmm-fpc-cpdp-06";
    }
    revision 2016-08-03 {
        description "Initial Revision.";
        reference "draft-ietf-dmm-fpc-cpdp-05";
    }

        // Next Hop Structures - SETTING
        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";
        }
        identity fpc-nexthop-type {
            description "NAT Service";
        }

        grouping fpc-nexthop {
            leaf nexthop-type {
                type identityref {
                  base "fpcbase:fpc-nexthop-type";
                }
                mandatory true;
                description "Nexthop Type";
            }
            choice nexthop-value {
                  mandatory true;
                case ip-nexthop {
                    leaf ip {
                      type inet:ip-address;
                      description "IP Value";
                    }
```

```
                   description "IP Case";
               }
               case macaddress-nexthop {
                   leaf macaddress {
                     type ytypes:mac-address;
                     description "MAC Address Value";
                   }
               }
               case servicepath-nexthop {
                   leaf servicepath {
                       type fpcbase:fpc-service-path-id;
                       description "Service Path Value";
                   }
                   description "Service Path Case";
               }
               case mplslabel-nexthop {
                   leaf lsp {
                       type fpcbase:fpc-mpls-label;
                       description "MPLS Value";
                   }
                   description "Service Path Case";
               }
               case if-nexthop {
                   leaf if-index {
                       type uint16;
                       description "If (interface) Value";
                   }
                   description "Service Path Case";
               }
               description "Value";
           }
           description "Nexthop Value";
       }

       // Address Translation - ACTION
       grouping simple-nat {
         leaf outbound-nat-address {
           type inet:ip-address;
           description "Outbound NAT Address";
         }
         description "Simple NAT value";
       }
       grouping simple-napt {
         leaf source-port {
           type inet:port-number;
           description "Source Port";
         }
         leaf outbound-napt-address {
```

```
            type inet:ip-address;
            description "Outbound NAPT Address";
          }
          leaf destination-port {
            type inet:port-number;
            description "Destination Port";
          }
          description "Simple NAPT Configuration";
        }

        // COPY FORWARD - ACTION
        grouping copy-forward {
          container destination {
            choice value {
              case nexthop-case {
          container nexthop {
                  uses fpcbase:fpc-nexthop;
                  description "Next Hop";
              }
                description "Port Forward Case";
              }
              description "Copy Forward Value";
            }
            description "destination";
          }
          description "Copy Then Forward to Port/Context Action";
        }

        //////////////////////////////
        // PMIP Integration         //
          identity pmip-tunnel-type {
              description "PMIP Tunnel Type";
          }
          identity grev1 {
              base "pmip-tunnel-type";
              description "GRE v1";
          }
          identity grev2 {
              base "pmip-tunnel-type";
              description "GRE v2";
          }
          identity ipinip {
              base "pmip-tunnel-type";
              description "IP in IP";
          }
          grouping pmip-tunnel-info {
              leaf pmip-tunnel-type {
                  type identityref {
```

```
                     base "pmip-tunnel-type";
                 }
                 description "PMIP Mobility";
             }
             choice pmip-tunnel-value {
                 case gre {
                     leaf gre-key {
                         type uint32;
                         description "GRE_KEY";
                     }
                     description "GRE Value";
                 }
                 description "PMIP Mobility value";
             }
             uses traffic-selectors:traffic-selector;
             description "PMIP Tunnel Information";
         }
         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         //
     //    Tunnel Types
         identity threeGPP-tunnel-type {
             description "3GPP Base Tunnel Type";
         }
```

```
         identity gtpv1 {
             base "fpcbase:threeGPP-tunnel-type";
             description "GTP version 1 Tunnel";
         }
         identity gtpv2 {
             base "fpcbase:threeGPP-tunnel-type";
             description "GTP version 2 Tunnel";
         }
         // QoS Profile
         typedef fpc-qos-class-identifier {
             type uint8 {
                 range "1..9";
             }
             description "QoS Class Identifier (QCI)";
         }
         grouping threeGPP-QoS {
             description "3GPP QoS Attributes";
             leaf qci {
                 type fpc-qos-class-identifier;
                 description "QCI";
             }
             leaf gbr {
                 type uint32;
                 description "Guaranteed Bit Rate";
             }
             leaf mbr {
                 type uint32;
                 description "Maximum Bit Rate";
             }
             leaf apn-ambr {
                 type uint32;
                 description "Access Point Name Aggregate Max Bit Rate";
             }
             leaf ue-ambr {
                 type uint32;
                 description "User Equipment Aggregate Max Bit Rate";
             }
             container arp {
                 uses pmipqos:Allocation-Retention-Priority-Value;
                 description "Allocation Retention Priority";
             }
         }
         typedef ebi-type {
           type uint8 {
             range "0..15";
           }
           description "EUTRAN Bearere Identifier (EBI) Type";
         }
```

```
        // From 3GPP TS 24.008 version 13.5.0 Release 13
        typedef component-type-enum {
             type enumeration {
                 enum ipv4RemoteAddress {
                   value 16;
                   description "IPv4 Remote Address";
                 }
                 enum ipv4LocalAddress  {
                   value 17;
                   description "IPv4 Local Address";
                 }
                 enum ipv6RemoteAddress {
                   value 32;
                   description "IPv6 Remote Address";
                 }
                 enum ipv6RemoteAddressPrefix {
                   value 33;
                   description "IPv6 Remote Address Prefix";
                 }
                 enum ipv6LocalAddressPrefix {
                   value 35;
                   description "IPv6 Local Address Prefix";
                 }
                 enum protocolNextHeader {
                   value 48;
                   description "Protocol (IPv4) or NextHeader (IPv6)
                     value";
                 }
                 enum localPort {
                   value 64;
                   description "Local Port";
                 }
                 enum localPortRange {
                   value 65;
                   description "Local Port Range";
                 }
                 enum reomotePort {
                   value 80;
                   description "Remote Port";
                 }
                 enum remotePortRange {
                   value 81;
                   description "Remote Port Range";
                 }
                 enum secParamIndex {
                   value 96;
                   description "Security Parameter Index (SPI)";
                 }
```

```
                 enum tosTraffClass {
                   value 112;
                   description "TOS Traffic Class";
                 }
                 enum flowLabel {
                   value 128;
                   description "Flow Label";
                 }
             }
             description "TFT Component Type";
         }
         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 {
                case ipv4-local {
                  leaf ipv4-local {
                    type inet:ipv4-address;
                    description "IPv4 Local Address";
                  }
                }
                case ipv6-prefix-local {
                  leaf ipv6-prefix-local {
                    type inet:ipv6-prefix;
                    description "IPv6 Local Prefix";
                  }
                }
                case ipv4-ipv6-remote {
                  leaf ipv4-ipv6-remote {
                    type inet:ip-address;
                    description "Ipv4 Ipv6 remote address";
                  }
                }
                case ipv6-prefix-remote {
                  leaf ipv6-prefix-remote {
                    type inet:ipv6-prefix;
                    description "IPv6 Remote Prefix";
                  }
                }
                case next-header {
                  leaf next-header {
                    type uint8;
                    description "Next Header";
                  }
                }
                case local-port {
                  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";
                }
              }
              case remote-port {
                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";
                }
              }
              case ipsec-index {
                leaf ipsec-index {
                  type traffic-selectors:ipsec-spi;
                  description "IPSec Index";
                }
              }
              case traffic-class {
                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";
```

```
                  }
                }
                case flow-label-type {
                  leaf-list flow-label {
                    type inet:ipv6-flow-label;
                    description "Flow Label";
                  }
                }
                description "Component Value";
              }
            }
            description "Packet Filter";
          }
          grouping tft {
            list packet-filters {
                key identifier;
                uses fpcbase:packet-filter;
                description "List of Packet Filters";
            }
            description "Packet Filter List";
          }
          typedef imsi-type {
              type uint64;
              description
                  "International Mobile Subscriber Identity (IMSI)
                    Value Type";
          }
          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";
         }

         grouping threegpp-tunnel-info {
             leaf tunnel-type {
                 type identityref  {
                   base "fpcbase:threeGPP-tunnel-type";
                 }
                 description "3GPP Tunnel Subtype";
             }
             leaf tunnel-identifier {
                 type uint32;
                 description "Tunnel Endpoint IDentifier (TEID)";
             }
             choice tft-or-ref {
               case defined-tft {
                 uses fpcbase:tft;
               }
               description "TFT Value";
             }
             description "3GPP TFT and Tunnel Information";
           }

         grouping threegpp-properties {
             leaf imsi {
               type fpcbase:imsi-type;
               description "IMSI";
             }
             leaf ebi {
               type fpcbase:ebi-type;
               description "EUTRAN Bearere Identifier (EBI)";
             }
             leaf lbi {
               type fpcbase:ebi-type;
               description "Linked Bearer Identifier (LBI)";
             }
             description "3GPP Mobility Session Properties";
           }
```

```
       ///////////////////////////////
       // ACTION VALUE AUGMENTS
       grouping fpc-action-value {
           choice action-value {
               mandatory true;
               case drop {
                 leaf drop {
                   type empty;
                   description "Drop Traffic";
                 }
               }
               case simple-nat {
                   uses fpcbase:simple-nat;
                   description "Simple NAT value";
               }
               case simple-napt {
                   uses fpcbase:simple-napt;
                   description "Simple NAPT Value";
               }
               case copy-forward {
                   uses fpcbase:copy-forward;
                   description "Copy Forward Value";
               }
               case pmip-selector {
                   uses traffic-selectors:traffic-selector;
                   description "PMIP Selector";
               }
               description "Action Value";
           }
           description "FPC Action Value";
       }

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

```
         }
         case 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";
         }
         case pmip-selector {
           uses traffic-selectors:traffic-selector;
           description "PMIP Selector";
         }
         case threegpp-tft {
             uses fpcbase:tft;
             description "3GPP TFT";
         }
         description "Descriptor Value";
       }
       description "FPC Descriptor Values";
     }

       //SETTINGS DEFINITIONS
       grouping fpc-settings {
           leaf-list delegated-ip-prefixes {
               type inet:ip-prefix;
               description "Delegated Prefix(es)";
           }
           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";
           }
           container mobility-tunnel-parameters {
               choice profile-parameters {
                   case nothing {
                     leaf none {
```

```
                      type empty;
                      description "Empty Value";
                    }
                    description "No Parameters Case";
                  }
          case pmip {
            uses pmip-tunnel-info;
          }
          case threegpp {
            uses threegpp-tunnel-info;
            uses threegpp-properties;
          }
                  description "Mobility Profile Parameters";
              }
              description
              "Profile specific tunnel parameters";
          }
          container nexthop {
              uses fpcbase:fpc-nexthop;
              description "Next Hop";
          }
          container qos-profile-parameters {
              choice value {
                  description "QoS Value";
              }
              description "QoS Parameters";
          }
          description "A collection of settings";
      }
      identity access-technology {
      description "The technology used in the access network";
    }
      identity role {
      description "The access-technology function of the DPN";
  }
    identity ietf-pmip-access-type {
      base "fpcbase:access-technology";
      description "PMIP Access";
    }
    identity threeGPP-access-type {
      base "fpcbase:access-technology";
      description "3GPP Access Type";
    }
     // Instructions
    grouping instructions {
      container instructions {
        choice instr-type {
          case threegpp-instr {
```

```
            leaf instr-3gpp-mob {
              type fpcbase:threegpp-instr;
              description "3GPP GTP Mobility Instructions";
            }
          }
          case pmip-instr {
            leaf instr-pmip {
              type pmip-commandset;
              description "PMIP Instructions";
            }
          }
          description "Instruction Value Choice";
        }
        description "Instructions";
      }
      description "Instructions Value";
    }
  }
  <CODE ENDS>
```

A.2.2. PMIP QoS Model

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

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-pmip-qos@2017-10-29.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; }

    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 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 2017-10-29 {
        description "Base Version";
         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.";
       }

       // QoS Attribute Types
```

```
       //The enumeration value for mapping - don't confuse with the
       //  identities
       typedef qos-attrubite-type-enum {
         type enumeration {
           enum Reserved {
             value 0;
             description "This value is reserved and cannot be used";
           }
           enum Per-MN-Agg-Max-DL-Bit-Rate {
             value 1;
             description "Per-Mobile-Node Aggregate Maximum Downlink
                 Bit Rate.";
           }
           enum Per-MN-Agg-Max-UL-Bit-Rate {
             value 2;
             description "Per-Mobile-Node Aggregate Maximum Uplink Bit
               Rate.";
           }
           enum Per-Session-Agg-Max-DL-Bit-Rate {
             value 3;
             description "Per-Mobility-Session Aggregate Maximum
               Downlink Bit Rate.";
           }
           enum Per-Session-Agg-Max-UL-Bit-Rate {
             value 4;
             description "Per-Mobility-Session Aggregate Maximum
                Uplink Bit Rate.";
           }
           enum Allocation-Retention-Priority {
             value 5;
             description "Allocation and Retention Priority.";
           }
           enum Aggregate-Max-DL-Bit-Rate {
             value 6;
             description "Aggregate Maximum Downlink Bit Rate.";
           }
           enum Aggregate-Max-UL-Bit-Rate {
             value 7;
             description "Aggregate Maximum Uplink Bit Rate.";
           }
           enum Guaranteed-DL-Bit-Rate {
             value 8;
             description "Guaranteed Downlink Bit Rate.";
           }
           enum Guaranteed-UL-Bit-Rate {
             value 9;
             description "Guaranteed Uplink Bit Rate.";
           }
```

```
           enum QoS-Traffic-Selector {
             value 10;
             description "QoS Traffic Selector.";
           }
           enum QoS-Vendor-Specific-Attribute {
             value 11;
             description "QoS Vendor-Specific Attribute.";
           }
         }
        description
       "The type of the QoS attribute.  This specification reserves
          the following reserved values.
            (12) to (254) -  Reserved
               These values are reserved for future allocation.

            (255)  Reserved
               This value is reserved and cannot be used.";
       }

       // Attribute Type as Identities
       // Added for convenience of inclusion and extension in
       //    other YANG modules.
       identity qos-attribute-type {
         description
           "Base type for Quality of Service Attributes";
       }

       identity Per-MN-Agg-Max-DL-Bit-Rate-type {
         base qos-attribute-type;
         description
           "Per-Mobile-Node Aggregate Maximum Downlink Bit Rate.";
       }

      identity Per-MN-Agg-Max-UL-Bit-Rate-type {
         base qos-attribute-type;
         description
           "Per-Mobile-Node Aggregate Maximum Uplink Bit Rate";
      }

      identity Per-Session-Agg-Max-DL-Bit-Rate-type {
         base qos-attribute-type;
         description
          "Per-Mobility-Session Aggregate Maximum Downlink Bit Rate.";
     }

      identity Per-Session-Agg-Max-UL-Bit-Rate-type {
         base qos-attribute-type;
         description
```

```
          "Per-Mobility-Session Aggregate Maximum Uplink Bit Rate.";
      }

      identity Allocation-Retention-Priority-type {
         base qos-attribute-type;
         description
           "Allocation and Retention Priority.";
      }

      identity Aggregate-Max-DL-Bit-Rate-type {
         base qos-attribute-type;
         description "Aggregate Maximum Downlink Bit Rate.";
      }

     identity Aggregate-Max-UL-Bit-Rate-type {
         base qos-attribute-type;
         description "Aggregate Maximum Uplink Bit Rate.";
     }

     identity Guaranteed-DL-Bit-Rate-type {
         base qos-attribute-type;
         description "Guaranteed Downlink Bit Rate.";
     }

     identity Guaranteed-UL-Bit-Rate-type {
         base qos-attribute-type;
         description "Guaranteed Uplink Bit Rate.";
     }

     identity QoS-Traffic-Selector-type {
         base qos-attribute-type;
         description "QoS Traffic Selector.";
     }

     identity QoS-Vendor-Specific-Attribute-type {
         base qos-attribute-type;
         description "QoS Vendor-Specific Attribute.";
     }

     //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 prioirty-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 premption-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 that were already
           assigned to another service data flow with a lower priority
           level.";
        }
        leaf premption-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.";
       }

       //NOTE - We do NOT add the Status Codes or other changes in
       // PMIP in this module

       //Primary Structures (groupings)
       grouping qosattribute {
           leaf attributetype {
               type identityref {
                   base qos-attribute-type;
               }
               mandatory true;
               description "the attribute type";
           }

           //All of the sub-types by constraint
           choice attribute-choice {
               case per-mn-agg-max-dl-case {
                   when "./attributetype = "
                      + "'Per-MN-Agg-Max-DL-Bit-Rate-type'";
                   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";
                   }
                   description "Per-MN-Agg-Max-DL-Bit-Rate Case";
               }
               case per-mn-agg-max-ul-case {
```

```
                   when "./attributetype = "
                     + "'Per-MN-Agg-Max-UL-Bit-Rate-type'";
                   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";
                   }
                   description "Per-MN-Agg-Max-UL-Bit-Rate Case";
               }
               case per-session-agg-max-dl-case {
                   when "./attributetype = "
                   + "'Per-Session-Agg-Max-DL-Bit-Rate-type'";
                   container per-session-agg-max-dl {
                       uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                       description "Per-Session-Agg-Max-Bit-Rate Value";
                   }
                   description "Per-Session-Agg-Max-Bit-Rate Case";
               }
               case per-session-agg-max-ul-case {
                   when "./attributetype = "
                   + "'Per-Session-Agg-Max-UL-Bit-Rate-type'";
                   container per-session-agg-max-ul {
                       uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                       description "Per-Session-Agg-Max-Bit-Rate Value";
                   }
                   description "Per-Session-Agg-Max-Bit-Rate Case";
               }
               case allocation-retention-priority-case {
                   when "./attributetype = "
                      + "'Allocation-Retention-Priority-type'";
                   uses qos-pmip:Allocation-Retention-Priority-Value;
                   description "Allocation-Retention-Priority Case";
               }
               case agg-max-dl-case {
                   when "./attributetype = "
                     + "'Aggregate-Max-DL-Bit-Rate-type'";
                   leaf agg-max-dl {
                       type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
                       description "Aggregate-Max-DL-Bit-Rate Value";
                   }
                   description "Aggregate-Max-DL-Bit-Rate Case";
               }
               case agg-max-ul-case {
                   when "./attributetype = "
                     + "'Aggregate-Max-UL-Bit-Rate-type'";
                   leaf agg-max-ul {
                       type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
                       description "Aggregate-Max-UL-Bit-Rate Value";
                   }
```

```
                   description "Aggregate-Max-UL-Bit-Rate Case";
               }
               case gbr-dl-case {
                   when "./attributetype =
                      'Guaranteed-DL-Bit-Rate-type'";
                   leaf gbr-dl {
                       type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
                       description "Guaranteed-DL-Bit-Rate Value";
                   }
                   description "Guaranteed-DL-Bit-Rate Case";
               }
               case gbr-ul-case {
                   when "./attributetype =
                    'Guaranteed-UL-Bit-Rate-type'";
                   leaf gbr-ul {
                       type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
                       description "Guaranteed-UL-Bit-Rate Value";
                   }
                   description "Guaranteed-UL-Bit-Rate Case";
               }
               case traffic-selector-case {
                   when "./attributetype = 'QoS-Traffic-Selector-type'";
                   container traffic-selector {
                       uses traffic-selectors:traffic-selector;
                       description "traffic selector";
                   }
                   description "traffic selector Case";
               }
               description "Attribute Value";
           }
           description "PMIP QoS Attribute";
       }

       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";
```

```
           }
           list attributes {
               unique "attributetype";
               uses qosattribute;
               min-elements 1;
               description "Attributes";
           }
           description "PMIP QoS Option";
       }
   }
   <CODE ENDS>
```

A.2.3. 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@2017-10-29.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: 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 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 2017-10-29 {
        description "Base Version";
         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-selector-base;
```

```
       uses ipv4-binary-traffic-selector;
       uses 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";
     }

     grouping ts-list {
       list selectors {
         key index;
         leaf index {
           type uint64;
           description "index";
         }
         uses traffic-selector;
         description "traffic selectors";
       }
       description "traffic selector list";
     }
   }

   <CODE ENDS>
```

A.3. FPC YANG Data Model Structure

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

```
   module: ietf-dmm-fpc
   +--rw mobility
      +--rw topology
      |  +--rw dpn-set* [dpn-id]
      |  |  +--rw dpn-id                               fpc:fpc-identity
      |  |  +--rw dpn-name?                            string
      |  |  +--rw dpn-resource-mapping-reference?      string
      |  |  +--rw interface-set* [access-technology role interface-id]
      |  |     +--rw access-technology         identityref
      |  |     +--rw role                      identityref
      |  |     +--rw interface-id              fpc:fpc-interface-id
      |  |     +--rw interface-settings-set
      |  +--rw dpn-type-set* [access-technology role]
      |  |  +--rw access-technology         identityref
      |  |  +--rw access-technology-name?   string
      |  |  +--rw role                      identityref
```

```
      |  |  +--rw role-name?                string
      |  |  +--rw interface-set* [interface-id]
      |  |     +--rw interface-id              fpc:fpc-interface-id
      |  |     +--rw interface-name?           string
      |  |     +--rw interface-protocol-set*   identityref
      |  |     +--rw feature-set*              identityref
      |  |     +--rw interface-settings-set
      |  +--rw dpn-group-set* [dpn-group-id]
      |  |  +--rw dpn-group-id           fpc:fpc-identity
      |  |  +--rw referenced-dpns-set*
                        [access-technology role interface-id]
      |  |  |  +--rw access-technology        identityref
      |  |  |  +--rw role                     identityref
      |  |  |  +--rw interface-id             fpc:fpc-interface-id
      |  |  |  +--rw supporting-dpn-id-set*   fpc:fpc-identity
      |  |  |  +--rw dpn-group-peer-id-set*   fpc:fpc-identity
      |  |  +--rw dpn-group-peer-set* [remote-dpn-group-id]
      |  |  |  +--rw remote-dpn-group-id       fpc:fpc-identity
      |  |  |  +--rw interface-settings-set
      |  |  +--rw domain-id?             fpc:fpc-identity
      |  +--rw domain-set* [domain-id]
      |  |  +--rw domain-id           fpc:fpc-identity
      |  |  +--rw domain-name?        string
      |  |  +--rw domain-reference?   string
      |  +--rw basename?          fpc:fpc-identity
      |  +--rw base-state?        string
      |  +--rw base-checkpoint?   string
      +--rw policy
      |  +--rw action-definition-set* [action-id]
      |  |  +--rw action-id      fpc:fpc-identity
      |  |  +--rw action-type?   identityref
      |  |  +--rw (action-value)?
      |  +--rw descriptor-definition-set* [descriptor-id]
      |  |  +--rw descriptor-id      fpc:fpc-identity
      |  |  +--rw descriptor-type?   identityref
      |  |  +--rw (descriptor-value)?
      |  +--rw rule-definition-set* [rule-id]
      |  |  +--rw rule-id                     fpc:fpc-identity
      |  |  +--rw descriptor-match-type       enumeration
      |  |  +--rw descriptor-reference-set* [descriptor-id-reference]
      |  |  |  +--rw descriptor-id-reference    fpc:fpc-identity
      |  |  |  +--rw direction?                 fpc:fpc-direction-type
      |  |  +--rw action-reference-set* [action-order]
      |  |     +--rw action-order           uint32
      |  |     +--rw action-id-reference    fpc:fpc-identity
      |  +--rw policy-definition-set* [policy-id]
      |  |  +--rw policy-id    fpc:fpc-identity
      |  |  +--rw rule-set* [precedence]
```

```
      |  |     +--rw precedence           uint32
      |  |     +--rw rule-id-reference    fpc:fpc-identity
      |  +--rw basename?                    fpc:fpc-identity
      |  +--rw base-state?                  string
      |  +--rw base-checkpoint?             string
      +--rw installed-policy-list* [dpn-id-reference]
      |  +--rw dpn-id-reference        fpc:fpc-identity
      |  +--rw installed-policy-set* [installed-policy-id]
      |  |  +--rw installed-policy-id    fpc:fpc-identity
      |  |  +--rw policy-id-reference?   fpc:fpc-identity
      |  |  +--rw policy-settings
      |  +--rw settings-set
      +--ro mobility-context-set* [mobility-context-id]
      |  +--ro mobility-context-id                     fpc:fpc-identity
      |  +--ro dpn-group-id-reference?                 fpc:fpc-identity
      |  +--ro parent-mobility-context-id-reference?   fpc:fpc-identity
      |  +--ro dpn-reference-list* [dpn-id-reference direction]
      |  |  +--ro dpn-id-reference                 fpc:fpc-identity
      |  |  +--ro direction
                               fpc:fpc-direction-type
      |  |  +--ro dpn-settings-complementary
      |  |  +--ro interface-id-reference           fpc:fpc-interface-id
      |  |  +--ro embedded-rule-set* [precedence]
      |  |  |  +--ro rule-id                     fpc:fpc-identity
      |  |  |  +--ro descriptor-match-type       enumeration
      |  |  |  +--ro precedence                  uint32
      |  |  |  +--ro action-definition-set* [action-order]
      |  |  |  |  +--ro action-order    uint32
      |  |  |  |  +--ro action-id?      fpc:fpc-identity
      |  |  |  |  +--ro action-type?    identityref
      |  |  |  |  +--ro (action-value)?
      |  |  |  +--ro descriptor-definition-set* [descriptor-id]
      |  |  |     +--ro descriptor-id      fpc:fpc-identity
      |  |  |     +--ro descriptor-type?   identityref
      |  |  |     +--ro (descriptor-value)?
      |  |  +--ro assigned-policy-reference-set*   fpc:fpc-identity
      |  +--ro requested-policy-reference-set*         fpc:fpc-identity
      |  +--ro context-settings-complementary
      +--rw monitor-set* [monitor-id]
         +--rw monitor-id             fpc:fpc-identity
         +--rw deterrable?            boolean
         +--rw binding-information
         +--rw target                 fpc-identity
         +--rw (configuration)
            +--:(periodic-config)
            |  +--rw period?                uint32
            +--:(threshold-config)
            |  +--rw lo-thresh?             uint32
```

```
            |  +--rw hi-thresh?             uint32
            +--:(scheduled-config)
            |  +--rw report-time?           uint32
            +--:(events-config-ident)
            |  +--rw event-identities*      identityref
            +--:(events-config)
               +--rw event-ids*             uint32
```

Figure 23: YANG FPC Agent Tree

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