DMM Working Group   S. Matsushima
Internet-Draft   SoftBank
Intended status: Standards Track   L. Bertz
Expires: September 14, 2017   Sprint
                                                        M. Liebsch
                                                               NEC
                                                     S. Gundavelli
                                                             Cisco
                                                          D. Moses
                                                 Intel Corporation
                                                        C. Perkins
                                                         Futurewei
                                                    March 13, 2017

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-06.txt

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 be abstracted as a Policy-group to
   be applied to a particular collection of flows, which is called
   the Virtual Port (Vport).

Mobility: A mobility session which is active on a mobile node is
   abstracted as a Context with associated runtime concrete
   attributes, such as tunnel endpoints, tunnel identifiers,
   delegated prefix(es), routing information, etc. Contexts are
   attached to DPN-groups along with consequence of the control
   plane. One or multiple Contexts which have same sets of policies
   are assigned Vports which abstract those policy sets. A Context
   can belong to multiple Vports which serve various kinds of purpose
   and policy. Monitors provide a mechanism to produce reports when
   events regarding Vports, Sessions, 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.

Virtual Port (Vport):   A set of forwarding policies.

Context: An abstracted endpoint of a mobility session
associated with runtime attributes. Vports
may apply to Context which instantiates those
forwarding policies on a DPN.

3. FPC Architecture

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. Information Model for FPC

This section presents an information model representing the abstract
concepts of FPC, which are language and protocol neutral. Figure 2
shows an overview of the FPC data-plane information model.


```
     (Mobile operator tenant that abstracted data-plane is used)
             |
         +---FPC-Topology
         |     |
         |     +---DPNs
         |     |
         |     +---DPN-groups
         |     |
         |     +---Domains
         |
         +---FPC-Policy
         |     |
         |     +---Descriptors
         |     |
         |     +---Actions
         |     |
         |     +---Policies
         |     |
         |     +---Policy-groups
         |
         +---FPC-Mobility
               |
               +---Vports
               |
               +---Contexts
```

Figure 2: FPC Data-plane Information Model

4.1. FPC-Topology

   Topology abstraction enables a physical data-plane network to support
   multiple overlay topologies. An FPC-Topology consists of DPNs, DPN-
   groups and Domains which abstract data-plane topologies for the
   Client's mobility control-planes and applications.

   Utilizing a FPC Agent, a mobile operator can create virtual DPNs in
   an overlay network. Those such virtual DPNs are treated the same as
   physical forwarding DPNs in this document.

4.1.1. DPNs

   The DPNs define all available nodes to a tenant of the FPC data-plane
   network. FPC Agent defines DPN binding to actual nodes. The role of
   a DPN in the data-plane is determined at the time the DPN is assigned
   to a DPN-group.


```
                     (FPC-Topology)
                         |
                     +---DPNs
                            |
                            +---DPN-id
                            |
                            +---DPN-name
                            |
                            +---DPN-groups
                            |
                            +---Node-reference
```
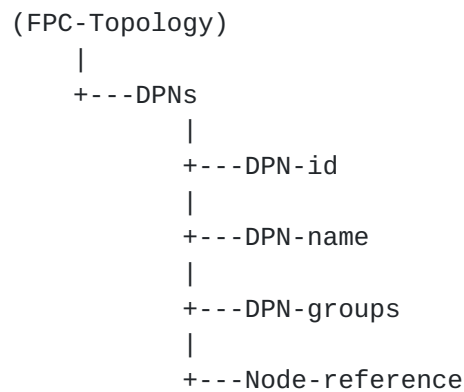

                   Figure 3: DPNs Model Structure

   DPN-id: The identifier for the DPN. The ID format MUST conform to
      Section 4.4.

   DPN-name: The name of the DPN.

   DPN-groups: The list of DPN-groups to which the DPN belongs.

   Node-reference: Indicates a physical node, or a platform of
      virtualization, to which the DPN is bound by the Agent. The
      Agent SHOULD maintain that node's information, including IP
      address of management and control protocol to connect them. In
      the case of a node as a virtualization platform, FPC Agent
      directs the platform to instantiate a DPN to which a DPN-group
      attributes.

4.1.2. DPN-groups

   A DPN-group is a set of DPNs which share certain specified data-plane
   attributes. DPN-groups define the data-plane topology consisting of
   a DPN-group of access nodes connecting to an anchor node's DPN-group.

   A DPN-group has attributes such as its data-plane role, supported
   access technologies, mobility profiles, connected peer groups and
   domain. A DPN may be assigned to multiple DPN-groups in different
   data-plane roles or different domains.


```
                  (FPC-Topology)
                       |
                   +---DPN-groups
                           |
                           +---DPN-group-id
                           |
                           +---Data-plane-role
                           |
                           +---Domains
                           |
                           +---Access-type
                           |
                           +---Mobility-profile
                           |
                           +---DPN-group-peers
```
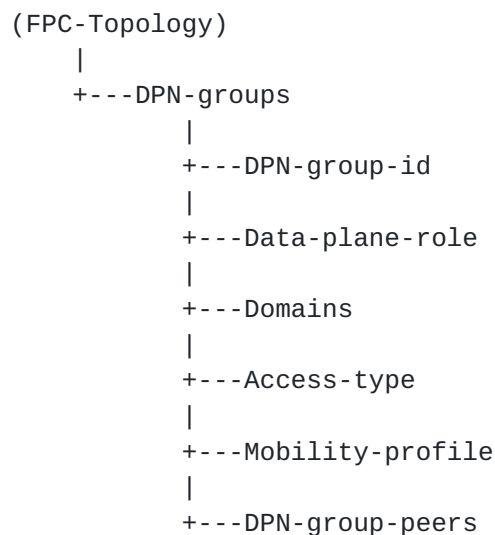

                   Figure 4: DPN-groups Model Structure

   DPN-group-id: The identifier of the DPN-group. The ID format MUST
      conform to Section 4.4.

   Data-plane-role: The data-plane role of the DPN-group, such as
      access-dpn, anchor-dpn.

   Domains: The domains to which the DPN-group belongs.

   Access-type: The access type supported by the DPN-group such as
      ethernet(802.3/11), 3gpp cellular(S1, RAB), if any.

   Mobility-profile: Identifies a supported mobility profile, such as
      ietf-pmip, or 3gpp. New profiles may be defined as extensions of
      this specification. Mobility profiles are defined so that some
      or all data-plane parameters of the mobility contexts that are
      part of the profile can be automatically determined by the FPC
      Agent.

   DPN-group-peers: The remote peers of the DPN-group with parameters
      described in Section 4.1.2.1.

4.1.2.1. DPN-group Peers

   DPN-group-peers lists relevant parameters of remote peer DPNs as
   illustrated in Figure 5.

                     (DPN-groups)
                         |
                        +---DPN-group-peers
                              |
                             +---Remote-DPN-group-id
                              |
                             +---Remote-mobility-profile
                              |
                             +---Remote-data-plane-role
                              |
                             +---Remote-endpoint-address
                              |
                             +---Local-endpoint-address
                              |
                             +---MTU-size
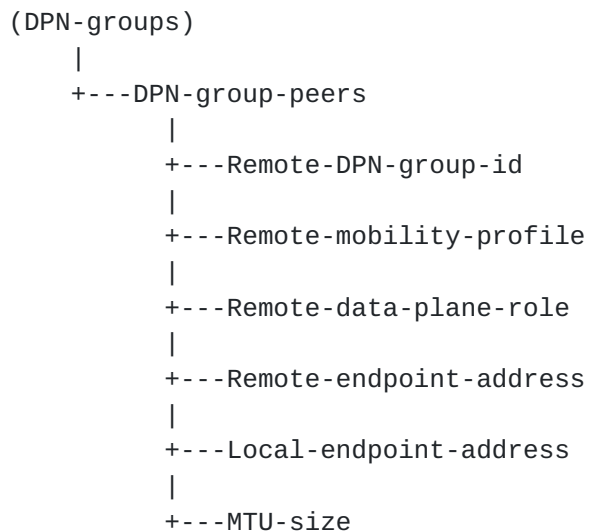
                  Figure 5: DPN-groups Peer Model Structure

   Remote-DPN-group-id: The ID of the peering DPN-Group. The ID format
      MUST conform to Section 4.4.

   Remote-mobility-profile: The mobility-profile for the peering DPN-
      group. Currently defined profiles are ietf-pmip, or 3gpp. New
      profiles may be defined as extensions of this specification.

   Remote-data-plane-role: The data-plane role of the peering DPN-
      group.

   Remote-endpoint-address: Defines Endpoint address of the peering
      DPN-group.

   Local-endpoint-address: Defines Endpoint address of its own DPN-
      group to peer the remote DPN-group.

   MTU-size: Defines MTU size of traffic between the DPN-Group and this
      DPN-group-peer.

4.1.3. Domains

   A domain is defined by an operator to refer to a particular network,
   considered as a system of cooperating DPN-groups. Domains may
   represent services or applications that are resident within an
   operator's network.

                    (FPC-Topology)
                        |
                    +---Domains
                            |
                            +---Domain-id
                            |
                            +---Domain-name
                            |
                            +---Domain-type
                            |
                            +---Domain-reference
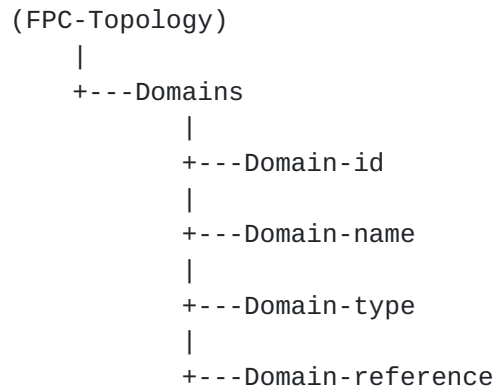
                   Figure 6: Domain Model Structure

   Domain-id: Identifier of Domain. The ID format MUST conform to
      Section 4.4.

   Domain-name: The name of the Domain.

   Domain-type: Specifies which address families are supported within
      the domain.

   Domain-reference: Indicates a set of resources for the domain which
      consists a topology of physical nodes, platforms of
      virtualization and physical/virtual links with certain bandwidth,
      etc,.

4.2. FPC-Policy

   The FPC-Policy consists of Descriptors, Actions, Policies and Policy-
   groups. These can be viewed as configuration data, in contrast to
   Contexts and Vports, which are structures that are instantiated on
   the Agent. The Descriptors and Actions in a Policy referenced by a
   Vport are active when the Vport is in an active Context, i.e. they
   can be applied to traffic on a DPN.

4.2.1. Descriptors

   Descriptors defines classifiers of specific traffic flows, such as
   those based on source and destination addresses, protocols, port
   numbers of TCP/UDP/SCTP/DCCP, or any way of classifying packets.
   Descriptors are defined by specific profiles that may be produced by
   3gpp, ietf or other SDOs. Many specifications also use the terms
   Filter, Traffic Descriptor or Traffic Selector [RFC6088]. A packet
   that meets the criteria of a Descriptor is said to satisfy, pass or
   be consumed by the Descriptor. Descriptors are assigned an
   identifier and contain a type and value.

```
                   (FPC-Policy)
                        |
                   +---Descriptors
                           |
                           +---Descriptor-id
                           |
                           +---Descriptor-type
                           |
                           +---Descriptor-value
```
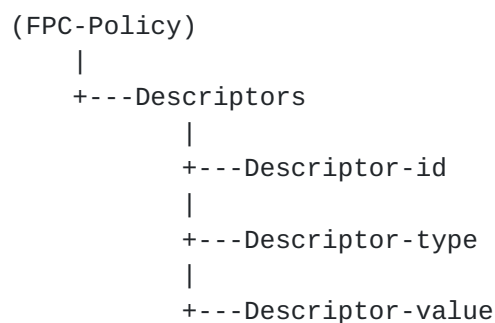
                 Figure 7: Descriptor Model Structure

   Descriptor-id: Identifier of Descriptor. The ID format MUST conform
        to Section 4.4.

   Descriptor-type: The descriptor type, which determines the
        classification of a specific traffic flows, such as source and
        destination addresses, protocols, port numbers of TCP/UDP/SCTP/
        DCCP, or any other way of selecting packets.

   Descriptor-value: The value of Descriptor such as IP prefix/address,
        protocol number, port number, etc.

4.2.2. Actions

   A Policy defines a list of Actions that are to be applied to traffic
   meeting the criteria defined by the Descriptors. Actions include
   traffic management such as shaping, policing based on given
   bandwidth, and connectivity actions such as pass, drop, forward to
   given nexthop. Actions may be defined as part of specific profiles
   which are produced by 3gpp, ietf or other SDOs.

```
                         (FPC-Policy)
                              |
                         +---Actions
                                 |
                                 +---Action-id
                                 |
                                 +---Action-type
                                 |
                                 +---Action-value
```
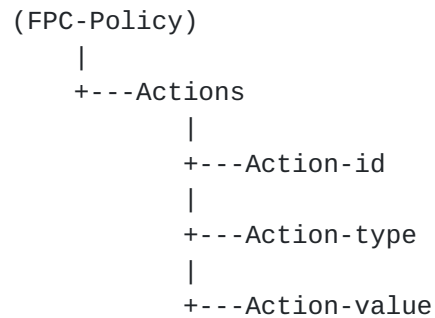

                    Figure 8: Action Model Structure

   Action-id: Identifier for the Action. The ID format MUST conform to
      Section 4.4.

   Action-type: The type of the action -- i.e. how to treat the
      specified traffic flows. Examples include pass, drop, forward to
      a given nexthop value, shape or police based on given bandwidth
      value, etc.

   Action-value: Specifies a value for the Action-type, such as
      bandwidth, nexthop address or drop, etc.

4.2.3. Policies

   Policies are collections of Rules. Each Policy has a Policy
   Identifier and a list of Rule/Order pairs. The Order and Rule values
   MUST be unique in the Policy. Unlike the AND filter matching of each
   Rule the Policy uses an OR matching to find the first Rule whose
   Descriptors are satisfied by the packet. The search for a Rule to
   apply to packet is executed according to the unique Order values of
   the Rules. This is an ascending order search, i.e. the Rule with the
   lowest Order value is tested first and if its Descriptors are not
   satisfied by the packet the Rule with the next lowest Order value is
   tested. If a Rule is not found then the Policy does not apply.
   Policies contain Rules (not references to Rules).

```
                 (FPC-Policy)
                     |
                 +---Policies
                       |
                       +---Policy-id
                       |
                       +---Rules
                             |
                             +---Order
                             |
                             +---Descriptors
                             |       |
                             |       +---Descriptor-id
                             |       |
                             |       +---Direction
                             |
                             +---Actions
                                   |
                                   +---Action-id
                                   |
                                   +---Action-Order
```
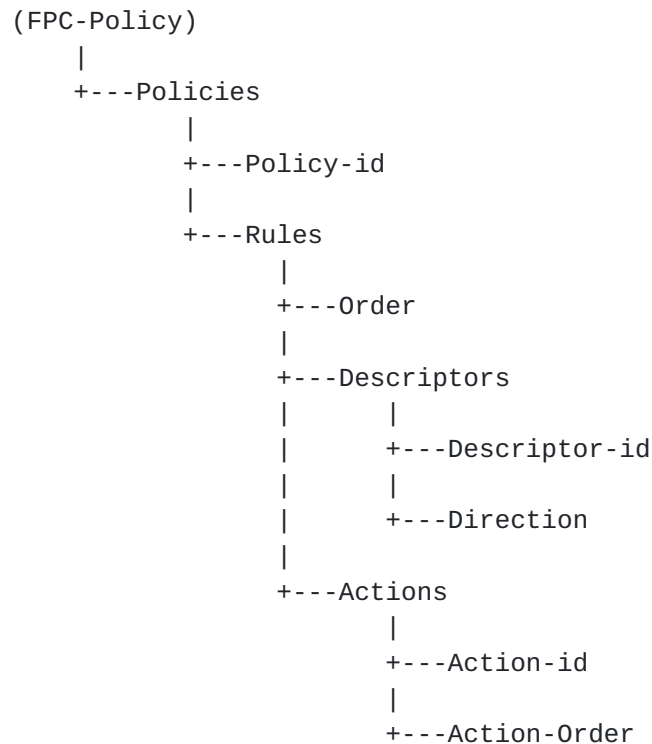
                     Figure 9: Model Structure for Policies

   Policy-id: Identifier of Policy. The ID format MUST conform to
      Section 4.4.

   Rules: List of Rules which are a collection of Descriptors and
      Actions. All Descriptors MUST be satisfied before the Actions
      are taken. This is known as an AND Descriptor list, i.e.
      Descriptor 1 AND Descriptor 2 AND ... Descriptor X all MUST be
      satisfied for the Rule to apply.

   Order: Specifies ordering if the Rule has multiple Descriptors and
      Action sets. Order values MUST be unique within the Rules list.

   Descriptors: The list of Descriptors.

   Descriptor-id: Identifies each Descriptor in the Rule.

   Direction: Specifies which direction applies, such as uplink,
      downlink or both.

   Actions: List of Actions.

   Action-id: Indicates each Action in the rule.

Action-Order: Specifies Action ordering if the Rule has multiple
    actions. Action-Order values MUST be unique within the Actions
    list.

4.2.4. Policy-groups

List of Policy-groups which are an aggregation of Policies. Common
applications include aggregating Policies that are defined by
different functions, e.g. Network Address Translation, Security,
etc. The structure has an Identifier and references the Policies via
their Identifiers.

```
                    (FPC-Policy)
                         |
                    +---Policy-groups
                            |
                            +---Policy-group-id
                            |
                            +---Policies
```
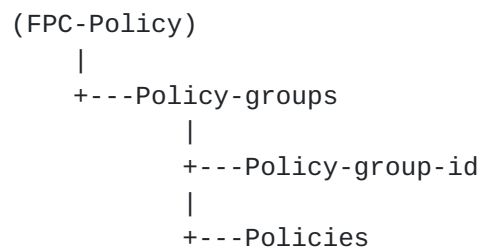
Figure 10: Policy-group Model Structure

Policy-group-id: The identifier of the Policy-group. The ID format
    MUST conform to Section 4.4.

Policies: List of Policies in the Policy-group.

4.3. FPC for Mobility Management

The FPC-Mobility consists of Vports and Contexts. A mobility session
is abstracted as a Context with its associated runtime concrete
attributes, such as tunnel endpoints, tunnel identifiers, delegated
prefix(es) and routing information, etc. A Vport abstracts a set of
policies applied to the Context.

4.3.1. Vport

A Vport represents a collection of policy groups, that is, a group of
rules that can exist independently of the mobility/session lifecycle.
Mobility control-plane applications create, modify and delete Vports
on FPC Agent through the FPC Client.

When a Vport is indicated in a Context, the set of Descriptors and
Actions in the Policies of the Vport are collected and applied to the
Context. They must be instantiated on the DPN as forwarding related

actions such as QoS differentiations, packet processing of encap/
decap, header rewrite, route selection, etc.


```
                     (FPC-Mobility)
                          |
                     +---Vports
                           |
                          +---Vport-id
                           |
                          +---Policy-groups
```
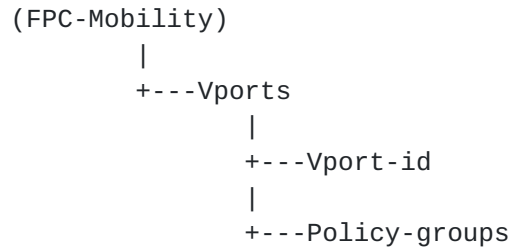

Figure 11: Vport Model Structure

Vport-id: The identifier of Vport. The ID format MUST conform to
      Section 4.4.

Policy-groups: List of references to Policy-groups which apply to
      the Vport.

4.3.2. Context

An endpoint of a mobility session is abstracted as a Context with its
associated runtime concrete attributes, such as tunnel endpoints,
tunnel identifiers, delegated prefix(es) and routing information,
etc. A mobility control-plane, or other applications, can create,
modify and delete contexts on an FPC Agent by using the FPC Client.

FPC Agent SHOULD determine runtime attributes of a Context from the
Vport's policies and the attached DPN's attributes. A mobility
control-plane, or other applications, MAY set some of the runtime
attributes directly when they create data-plane related attributes.
In the case of that a mobility control-plane assigns tunnel
identifiers, for instance.

```
                   (FPC-Mobility)
                        |
                      +---Contexts
                            |
                            +---Context-id
                            |
                            +---Vports
                            |
                            +---DPN-group
                            |
                            +---Delegated-ip-prefixes
                            |
                            +---Parent-context
```
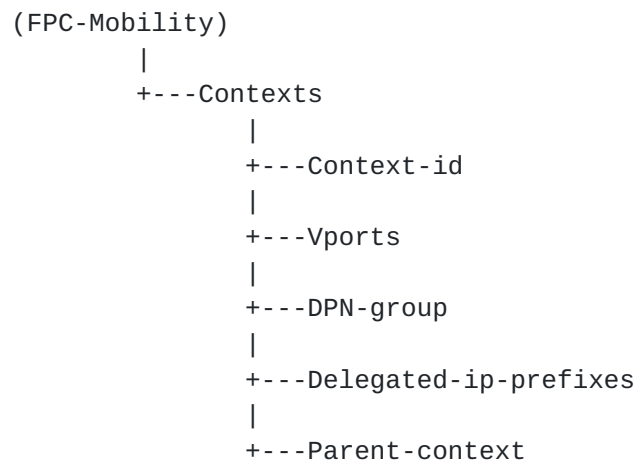
                  Figure 12: Common Context Model Structure

   Context-id: Identifier of the Context. The ID format MUST conform
      to Section 4.4.

   Vports: List of Vports. When a Context is applied to a Vport, the
      context is configured by policies at each such Vport. Vport-id
      references indicate Vports which apply to the Context. Context
      can be a spread over multiple Vports which have different
      policies.

   DPN-group: The DPN-group assigned to the Context.

   Delegated-ip-prefixes: List of IP prefixes to be delegated to the
      mobile node of the Context.

   Parent-context: Indicates a parent context from which this context
      inherits.

4.3.2.1. Single DPN Agent Case

   In the case where a FPC Agent supports only one DPN, the Agent MUST
   maintain Context data just for the DPN. The Agent does not need to
   maintain a Topology model. Contexts in single DPN case consists of
   following parameters for both direction of uplink and downlink.

```
            (Contexts)
                |
                +---UL-Tunnel-local-address
                |
                +---UL-Tunnel-remote-address
                |
                +---UL-MTU-size
                |
                +---UL-Mobility-specific-tunnel-parameters
                |
                +---UL-Nexthop
                |
                +---UL-QoS-profile-specific-parameters
                |
                +---UL-DPN-specific-parameters
                |
                +---UL-Vendor-specific-parameters
```

            Figure 13: Uplink Context Model of Single DPN Structure

   UL-Tunnel-local-address: Specifies uplink endpoint address of the
        DPN.

   UL-Tunnel-remote-address: Specifies uplink endpoint address of the
        remote DPN.

   UL-MTU-size: Specifies the uplink MTU size.

   UL-Mobility-specific-tunnel-parameters: Specifies profile specific
        uplink tunnel parameters to the DPN which the agent exists. This
        may, for example, include GTP/TEID for 3gpp profile, or GRE/Key
        for ietf-pmip profile.

   UL-Nexthop: Indicates next-hop information of uplink in external
        network such as IP address, MAC address, SPI of service function
        chain [I-D.ietf-sfc-nsh], SID of segment
        routing[I-D.ietf-6man-segment-routing-header]
        [I-D.ietf-spring-segment-routing-mpls], etc.

   UL-QoS-profile-specific-parameters: Specifies profile specific QoS
        parameters of uplink, such as QCI/TFT for 3gpp profile,
        [RFC6089]/[RFC7222] for ietf-pmip, or parameters of new profiles
        defined by extensions of this specification.

   UL-DPN-specific-parameters: Specifies optional node specific
        parameters needed by uplink such as if-index, tunnel-if-number
        that must be unique in the DPN.

   UL-Vendor-specific-parameters: Specifies a vendor specific parameter
      space for the uplink.

                (Contexts)
                    |
                    +---DL-Tunnel-local-address
                    |
                    +---DL-Tunnel-remote-address
                    |
                    +---DL-MTU-size
                    |
                    +---DL-Mobility-specific-tunnel-parameters
                    |
                    +---DL-Nexthop
                    |
                    +---DL-QoS-profile-specific-parameters
                    |
                    +---DL-DPN-specific-parameters
                    |
                    +---DL-Vendor-specific-parameters

          Figure 14: Downlink Context Model of Single DPN Structure

   DL-Tunnel-local-address: Specifies downlink endpoint address of the
      DPN.

   DL-Tunnel-remote-address: Specifies downlink endpoint address of the
      remote DPN.

   DL-MTU-size: Specifies the downlink MTU size of tunnel.

   DL-Mobility-specific-tunnel-parameters: Specifies profile specific
      downlink tunnel parameters to the DPN which the agent exists.
      This may, for example, include GTP/TEID for 3gpp profile, or GRE/
      Key for ietf-pmip profile.

   DL-Nexthop: Indicates next-hop information of downlink in external
      network such as IP address, MAC address, SPI of service function
      chain [I-D.ietf-sfc-nsh], SID of segment
      routing[I-D.ietf-6man-segment-routing-header]
      [I-D.ietf-spring-segment-routing-mpls], etc.

   DL-QoS-profile-specific-parameters: Specifies profile specific QoS
      parameters of downlink, such as QCI/TFT for 3gpp profile,
      [RFC6089]/[RFC7222] for ietf-pmip, or parameters of new profiles
      defined by extensions of this specification.

   DL-DPN-specific-parameters: Specifies optional node specific
      parameters needed by downlink such as if-index, tunnel-if-number
      that must be unique in the DPN.

   DL-Vendor-specific-parameters: Specifies a vendor specific parameter
      space for the downlink.

4.3.2.2. Multiple DPN Agent Case

   Alternatively, a FPC Agent may connect to multiple DPNs. The Agent
   MUST maintain a set of Context data for each DPN. The Context
   contains a list of DPNs, where each entry of the list consists of the
   parameters in Figure 15. A Context data for one DPN has two entries
   - one for uplink and another for downlink or, where applicable, a
   direction of 'both'.

```
            (Contexts)
                |
               +---DPNs
                     |
                     +---DPN-id
                     |
                     +---Direction
                     |
                     +---Tunnel-local-address
                     |
                     +---Tunnel-remote-address
                     |
                     +---MTU-size
                     |
                     +---Mobility-specific-tunnel-parameters
                     |
                     +---Nexthop
                     |
                     +---QoS-profile-specific-parameters
                     |
                     +---DPN-specific-parameters
                     |
                     +---Vendor-specific-parameters
```
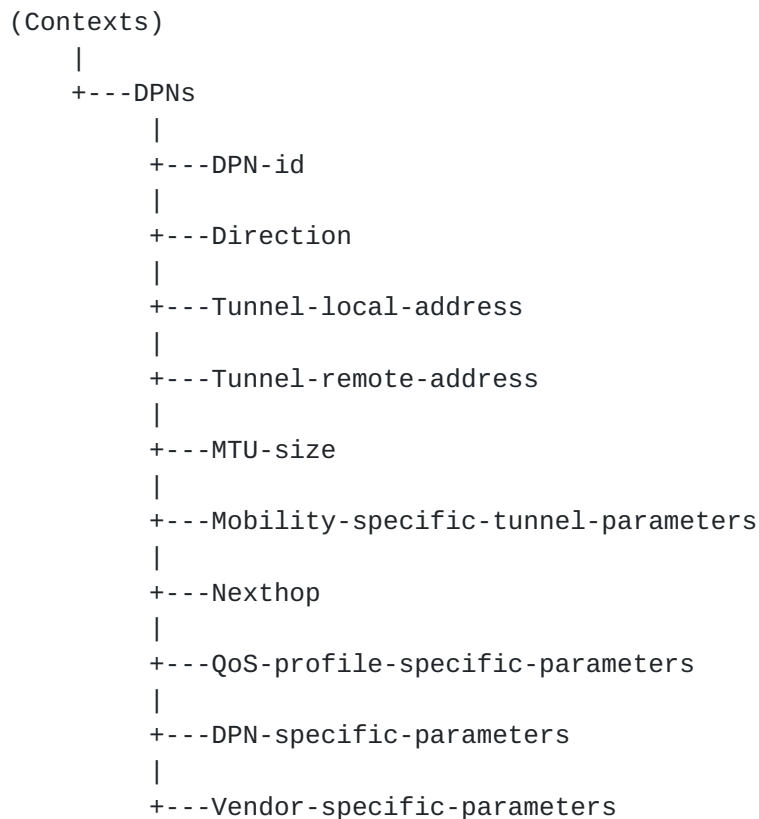
         Figure 15: Multiple-DPN Supported Context Model Structure

   DPN-id: Indicates DPN of which the runtime Context data installed.

   Direction: Specifies which side of connection at the DPN indicated -
      uplink, downlink or both.

   Tunnel-local-address: Specifies endpoint address of the DPN at the
      uplink or downlink.

   Tunnel-remote-address: Specifies endpoint address of remote DPN at
      the uplink or downlink.

   MTU-size: Specifies the packet MTU size on uplink or downlink.

   Mobility-specific-tunnel-parameters: Specifies profile specific
      tunnel parameters for uplink or downlink to the DPN. This may,
      for example, include GTP/TEID for 3gpp profile, or GRE/Key for
      ietf-pmip profile.

   Nexthop: Indicates next-hop information for uplink or downlink in
      external network such as IP address, MAC address, SPI of service
      function chain [I-D.ietf-sfc-nsh], SID of segment
      routing[I-D.ietf-6man-segment-routing-header]
      [I-D.ietf-spring-segment-routing-mpls], etc.

   QoS-profile-specific-parameters: Specifies profile specific QoS
      parameters for uplink or downlink to the DPN, such as QCI/TFT for
      3gpp profile, [RFC6089]/[RFC7222] for ietf-pmip, or parameters of
      new profiles defined by extensions of this specification.

   DPN-specific-parameters: Specifies optional node specific parameters
      needed by uplink or downlink to the DPN such like if-index,
      tunnel-if-number that must be unique in the DPN.

   Vendor-specific-parameters: Specifies a vendor specific parameter
      space for the DPN.

   Multi-DPN Agents will use only the DPNs list of a Context for
   processing as described in this section. A single-DPN Agent MAY use
   both the Single Agent DPN model Section 4.3.2.1 and the multi-DPN
   Agent Context described here.

4.3.3. 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.

```
                    (FPC-Mobility)
                          |
                      +---Monitors
                             |
                             +---Monitor-id
                             |
                             +---Target
                             |
                             +---Configuration
```
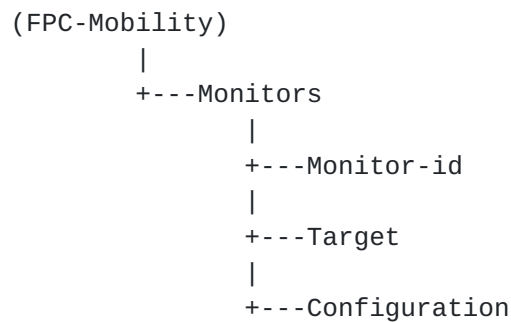
Figure 16: Common Monitor Model Structure

Monitor-id: Name of the Monitor. The ID format MUST conform to
     Section 4.4.

Target: Target to be monitored. This may be an event, a Context, a
     Vport or attribute(s) of Contexts. When the type is an
     attribute(s) of a Context, the target name is a concatenation of
     the Context-Id and the relative path (separated by '/') to the
     attribute(s) to be monitored.

Configuration: Determined by the Monitor subtype. Four report types
     are defined:

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

     * 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 to the Client.

     * Scheduled reporting specifies the time (in seconds since Jan
        1, 1970) when a notification for the monitor should be sent to
        the Client. 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 to the Client.

4.4. 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.

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. The Topology and Policy models are not
restricted to meet that requirement, as described in Section 3.

4.5. Attribute Application

Attributes in FPC Topology and Policy SHOULD be pre-configured in a
FPC Agent prior to Contexts and Vports. The FPC Agent requires those
pre-configured attributes to be able to derive a Context's detailed
runtime attributes.

When a FPC Client creates a Context, the FPC Client is then able to
indicate specific DPN-group(s) instead of all endpoint addresses of
the DPN(s) and MTU-size of the tunnels for example. This is because
that the FPC Agent can derive data for those details from the pre-
configured DPN-group information in the FPC Topology.

Similarly when a Vport is created for the Context, the FPC Agent can
derive detailed forwarding policies from the pre-configured Policy
information in the FPC Policy. The FPC Client thereby has no need to
indicate those specific policies to all of the Contexts which share
the same set of Policy-groups.

This is intentional as it provides FPC Clients the ability to reuse
pre-configured FPC Topology and FPC Policy attributes. It helps to
minimize over the wire exchanges and reduce system errors by
exchanging less information.

The Agent turns those derived data into runtime attributes of UL and
DL objects which are in the DPNs list of the Context (multiple-DPNs
Agent case) or directly under the Context (single-DPN Agent case).
The Agent consequently instantiates forwarding policies on DPN(s)
based on those attributes.

When a Context inherits another Context as its parent, missing
attributes in the child Context are provided by the Parent Context
(for example, IMSI defined in the 3GPP extension) .

It is noted that the Agent SHOULD update the Context's attributes
which are instantiated on DPN(s) when the applied attributes of
Topology and Policy are changed.

In the case of FPC Client modifying an existing runtime attribute of
a Context which the FPC Agent derived, the FPC Agent MUST overwrite
that attribute with the value which the Client brings to the Agent.
However risks exist, for example, the attributes could be outside of
allowable range of DPNs which the FPC Agent managed.

5. Protocol

5.1. Protocol Messages and Semantics

Five message types are supported:

```
+---------------+---------------+--------------------------------+
| Message       | Type          | Description                    |
+---------------+---------------+--------------------------------+
| CONF          | HEADER        | Configure processes a single   |
|               | ADMIN_STATE   | operation.                     |
|               | SESSION_STATE |                                |
|               | OP_TYPE BODY  |                                |
|               |               |                                |
| CONF_BUNDLE   | 1*[HEADER     | A Conf-bundle takes multiple   |
|               | ADMIN_STATE   | operations that are to be      |
|               | SESSION_STATE | executed as a group with partial |
|               | TRANS_STRATEGY | failures allowed. They are    |
|               | OP_TYPE BODY] | executed according to the OP_ID |
|               |               | value in the OP_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        | Register a monitor at an Agent. |
|               | ADMIN_STATE *[ | The message includes information |
|               | MONITOR ]     | about the attribute to monitor |
|               |               | and the reporting method.  Note |
|               |               | that a MONITOR_CONFIG is       |
|               |               | required for this operation.   |
|               |               |                                |
| DEREG_MONITOR | HEADER *[     | Deregister monitors from an    |
|               | MONITOR_ID ] [ | Agent. Monitor IDs are provided. |
|               | boolean ]     | Boolean (optional) indicates if |
|               |               | a successful DEREG triggers a  |
|               |               | NOTIFY with final data.        |
|               |               |                                |
| PROBE         | HEADER        | Probe the status of a registered |
|               | MONITOR_ID    | 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.

Messages that create or update Monitors and Entities, i.e. CONFIG,
CONF_BUNDLE and REG_MONITOR, specify an Administrative State which
specifies the Administrative state of the message subject(s) after
the successful completion of the operation. If the status is set to
virtual, any existing data on the DPN is removed. If the value is
set to disabled, and if that entity exists on the DPN, then an
operation to disable the associated entity will occur on the DPN . If
set to 'active' the DPN will be provisioned. Values are 'enabled',
'disabled', and 'virtual'.

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 5.1.1.6.2 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      | An asynchronous notification   |
|                      | Table 15 | from Agent to Client based upon |
|                      |          | a previous CONFIG or           |
|                      |          | CONF_BUNDLE request.           |
|                      |          |                                |
| NOTIFY               | See      | An asynchronous notification   |
|                      | Table 16 | from Agent to Client based upon |
|                      |          | a registered MONITOR.          |
+----------------------+----------+--------------------------------+
```

Table 2: Agent to Client Messages (notifications)

5.1.1. CONFIG and CONF_BUNDLE Messages

CONFIG and CONF_BUNDLE specify the following information for each
operation in addition to the header information:

SESSION_STATE: sets the expected state of the entities embedded in
    the operation body after successful completion of the operation.
    Values can be 'complete', 'incomplete' or 'outdated'. Any
    operation that is 'incomplete' MAY NOT result in communication
    between the Agent and DPN. If the result is 'outdated' any new
    operations on these entities or new references to these entities
    have unpredictable results.

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

COMMAND_SET: If the feature is supported, specifies the Command Set
    (see Section 5.1.1.4).

BODY: A list of Clones, if supported, Vports and Contexts when the
    OP_TYPE is 'create' or 'update'. Otherwise it is a list of
    Targets for 'query' or 'deletion'. See Section 6.2.2 for
    details.

5.1.1.1. Agent Operation Processing

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

1. Clone Instructions, if the feature is supported

2. Vports

3. Contexts according to COMMAND_SET order processing

The following Order Processing occurs when COMMAND Sets are present

1. The Entity-specific COMMAND_SET is processed according to its bit
    order unless otherwise specified by the technology specific
    COMMAND_SET definition.

2. Operation specific COMMAND_SET is processed upon all applicable
    entities (even if they had Entity-specific COMMAND_SET values
    present) according to its bit order unless otherwise specified by
    the technology specific COMMAND_SET definition.

3. Operation OP_TYPE is processed for all entities.

When deleting objects only their name needs to be provided. However,
attributes MAY be provided if the Client wishes to avoid requiring
the Agent cache lookups.

When deleting an attribute, a leaf reference should be provided.
This is a path to the attributes.

5.1.1.2. Policy RPC Support

This optional feature permits policy elements, (Policy-Group, Policy,
Action and Descriptor), values to be in CONFIG or CONF_BUNDLE
requests. It enables RPC based policy provisioning.

5.1.1.3. Cloning

Cloning is an optional feature that allows a Client to copy one
structure to another in an operation. Cloning is always done first
within the operation (see Operation Order of Execution for more
detail). If a Client wants to build an object then Clone it, use
CONF_BUNDLE with the first operation being the entities to be copied
and a second operation with the Cloning instructions. A CLONE
operation takes two arguments, the first is the name of the target to
clone and the second is the name of the newly created entity.
Individual attributes are not clonable; only Vports and Contexts can
be cloned.

5.1.1.4. Command Bitsets

The COMMAND_SET is a technology specific bitset that allows for a
single entity to be sent in an operation with requested sub-
transactions to be completed. For example, a 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.
Alternatively, an entity could be sent in an update operation that
would be considered incomplete, e.g. missing some required data in
for the entity, but has sufficient data to complete the instructions
provided in the COMMAND_SET.

5.1.1.5. 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

o none - all entities have no references to other entities. This
   implies only Contexts are present. Vports MUST have references to
   Policy-Groups.

o op - All references are contained in the operation body, i.e. only
   intra-operation references exist.

o 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'.

o storage - One or more references exist outside of the operation
   and bundle. A lookup to a cache / storage is required.

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

If supported by the Agent, when cloning instructions are present, the
scope MUST NOT be 'none'. When Vports are present the scope MUST be
'storage' or 'unknown'.

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.

5.1.1.6. Operation Response

5.1.1.6.1. Immediate Response

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

OK_NOTIFY_FOLLOWS - 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.

Any result MAY contain nothing or entities created or partially
fulfilled as part of the operation as specified in Table 14. 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 5.1.1.4).

If an error occurs the following information is returned.

ERROR_TYPE_ID (Unsigned 32) - The identifier of a specific error
type

ERROR_INFORMATION - An OPTIONAL string of no more than 1024
characters.

5.1.1.6.2. 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 15.

5.1.2. Monitors

When a monitor has a reporting configuration of SCHEDULED it is
automatically de-registered after the NOTIFY occurs. An Agent or DPN
may temporarily suspend monitoring if insufficient resources exist.
In such a case the Agent MUST notify the Client.

All monitored data can be requested by the Client at any time using
the PROBE message. Thus, reporting configuration is optional and
when not present only PROBE messages may be used for monitoring. If
a SCHEDULED or PERIODIC configuration is provided during registration
with the time related value (time or period respectively) of 0 a
NOTIFY is immediately 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. An Agent may reject a registration if
it or the DPN has insufficient resources.

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.

5.1.2.1. Operation Response

5.1.2.1.1. Immediate Response

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

Any OK result will contain no more information.

If an error occurs the following information is returned.

   ERROR_TYPE_ID (Unsigned 32) - The identifier of a specific error
   type

   ERROR_INFORMATION - An OPTIONAL string of no more than 1024
   characters.

5.1.2.1.2. 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_ID and
ERROR_INFORMATION accordingly and MAY clear the Context or Vport,
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 an OK_NOTIFY_FOLLOWS with an optional RESPONSE_BODY containing
the partially completed entities. When an OK_NOTIFY_FOLLOWS is sent,
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_NOTIFY_FOLLOWS with 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_ID and ERROR_INFORMATION. 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 17 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)--->|         |
     |          |          |   [ CONTEXT_ID,          |--tun1 up->|
     |          |          |   DOWNLINK(QOS/TUN),     |          |
     |          |          |   UPLINK(QOS/TUN),       |--tc qos-->|
     |          |          |    IP_PREFIX(HNP) ]      |          |
     |          |          |<---(2)- OK --------------|-route add>|
     |          |          |                          |          |
     |<------------PBA------|                          |          |
     |          |          |                          |          |
     | +----+   |          |                          |          |
     | |Edge|   |          |                          |          |
     | |DPN1|   |          |                          |          |
     | +----+   |          |                          |          |
     |    |     |          |                          |          |
     |    |-====================================================-|
     |          |          |                          |          |
     |   [MN handover]     |                          |          |
     |        |---PBU ---->|                          |          |
     |        |          |--(3)- CONFIG(MODIFY)---->|          |
     |        |<--PBA------|    [ CONTEXT_ID         |-tun1 mod->|
     |        |          |        DOWNLINK(TUN),     |          |
     |        | +----+   |        UPLINK(TUN) ]      |          |
     |        | |Edge|   |<---(4)- OK --------------|          |
     |        | |DPN2|   |                          |          |
     |        | +----+   |                          |          |
     |        |   |      |                          |          |
     |        |   |-========================================-|
     |        |          |                          |          |
```
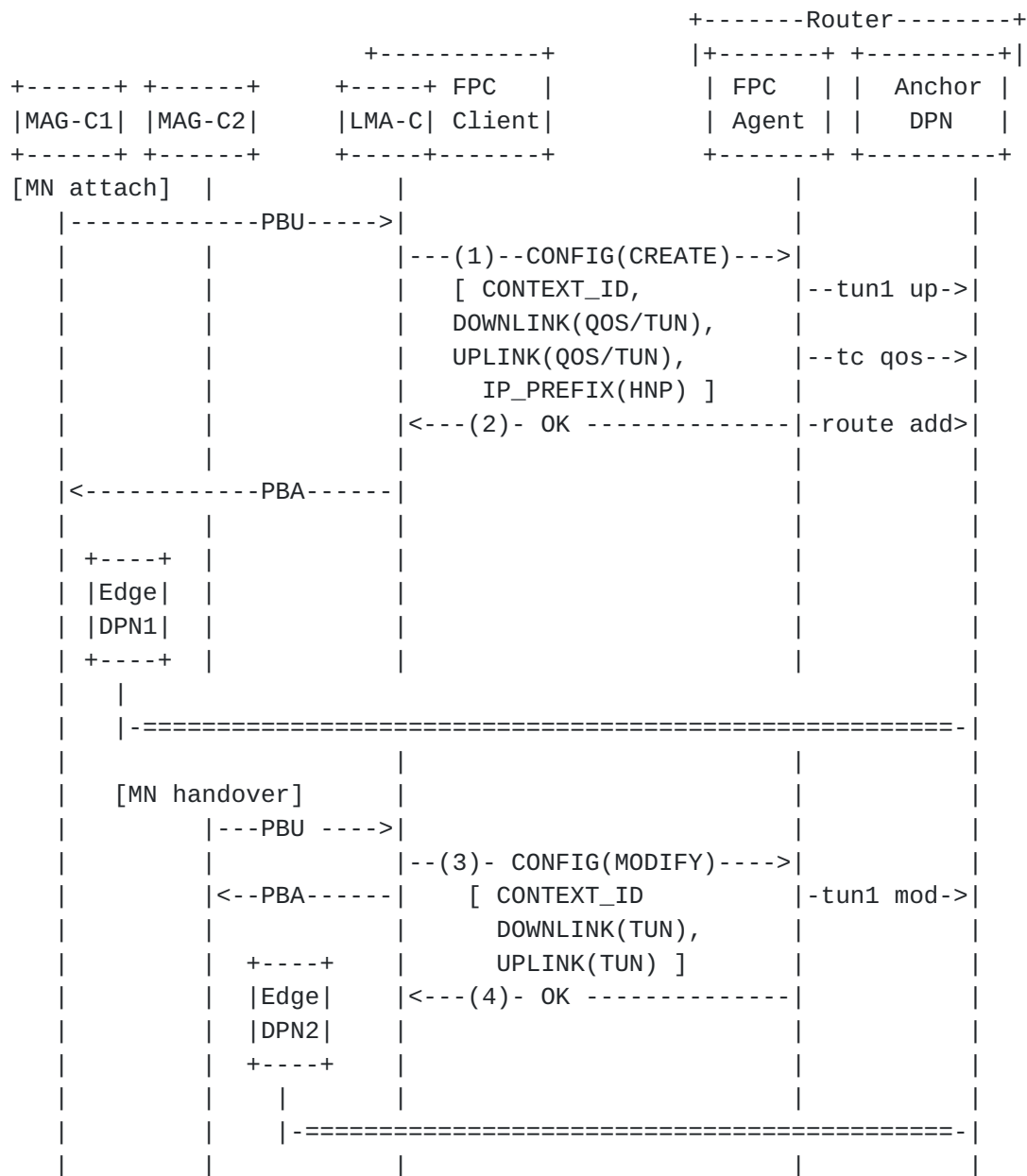
       Figure 17: Exemplary Message Sequence (focus on FPC reference point)

   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.

   The LMA-C adds properties during the creation of the new Context.
   One property is added 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 to 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   ->|
    |         |          |     DOWNLINK(TUN delete), |   down    |
    |         |          |     UPLINK(TUN delete) ]  |           |
    |         |          |                  |         |
    |         |          |<-(2)- OK ----------------|         |
    |         |          |                  |         |
    |         | [ MinDelayBeforeBCEDelete expires ]  |         |
    |         |          |                  |         |
    |         |          |---(3)--CONFIG(DELETE)--->|-- tun1 -->|
    |         |          |                  | delete    |
    |         |          |<-(4)- OK ----------------|         |
    |         |          |                  |-- route ->|
    |         |          |                  | remove    |
    |         |          |                  |         |
```
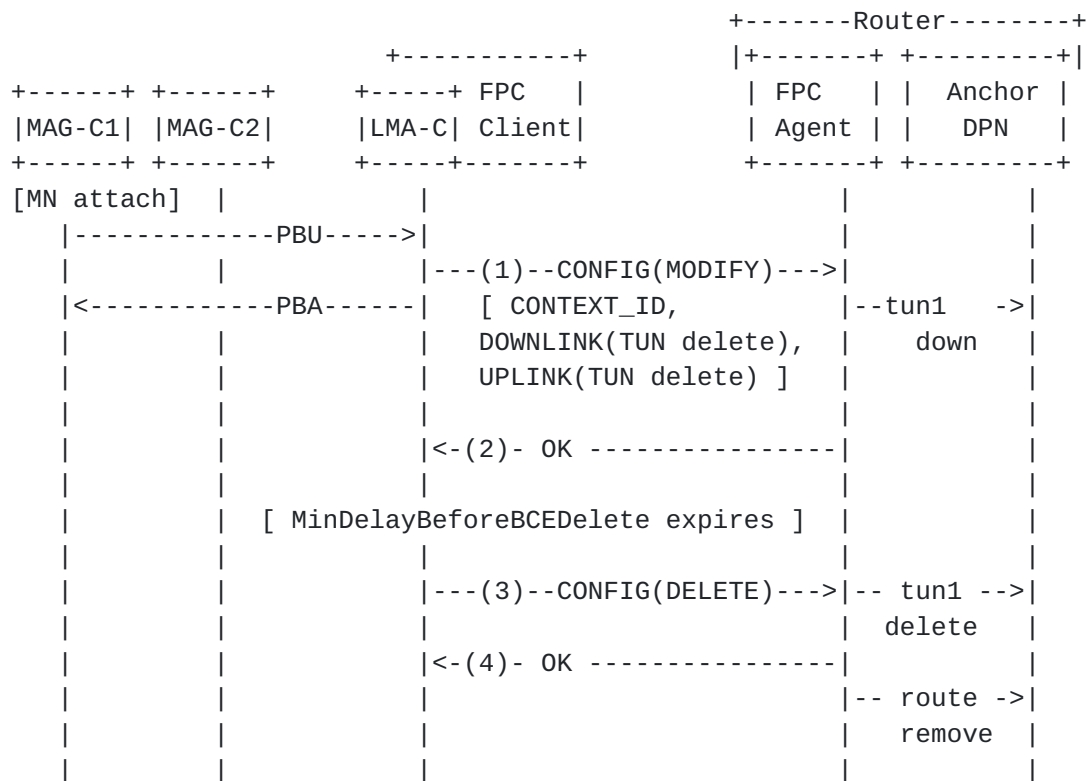
   Figure 18: Exemplary Message Sequence (focus on FPC reference point)

   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.


```
                          +-----------+        +-------+ +---------+
 +------+ +------+    +-----+ FPC   |        | FPC   | | Anchor  |
 |MAG-C1| |MAG-C2|    |LMA-C| Client|        | Agent | |   DPN1  |
 +------+ +------+    +-----+-------+        +-------+ +---------+
 [MN attach]  |           |                     |          |
    |------------PBU----->|                     |          |
    |          |          |---(1)--CONFIG(CREATE)--->|          |
    |          |          |    [ CONTEXT_ID, DPNS [   |--tun1 up->|
    |          |          |[DPN1,DOWNLINK(QOS/TUN)], |          |
    |          |          | [DPN1,UPLINK(QOS/TUN)],   |--tc qos-->|
    |          |          |[DPN2,DOWNLINK(QOS/TUN)], |          |
    |          |          | [DPN2,UPLINK(QOS/TUN)],   |          |
    |          |          |     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 19: Exemplary Message Sequence for Multi-DPN Agent

Figure 19 shows how the first 2 messages in Figure 17 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 "DPNS" list of the same Context. Message 1
shows the DPNS list with all entries. Each entry identifies the DPN
and direction (one of 'uplink', 'downlink' or 'both'). Generally,
the 'both' direction is not used for normal mobility session
processing. It is commonly used for the instantiation of Policies on
a specific DPN (see Section 5.2.4).

The Agent responds with an OK_NOTIFY_FOLLOWS 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 Vports 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 20 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 +-<PolicyID>
                     |           |        |  ^
       Descriptor_4 -+--<Rule>--+- Action_4 |  |
                    +------+             | <PolicyGroupID>
                    /Order#/-------------+  ^
                    +------+                |
                                        <VportID>


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


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

                 Figure 20: Structure of Policies and Vports

   As depicted in Figure 20, the Vport represents the anchor of Rules
   through the Policy-group, Policy, Rule hierarchy configured by any
   mechanism including RPC or N. 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 Vport.

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

   Figure 21 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 ----------------+
```
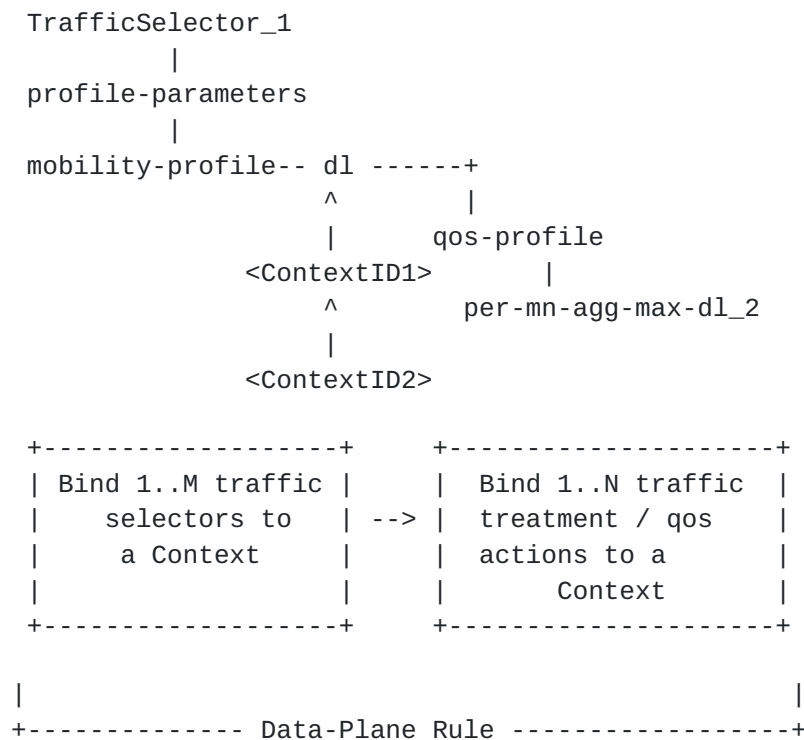
                   Figure 21: Structure of Contexts

   As depicted in Figure 21, 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 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. Bulk Data in a Single Operation

   A single operation MAY contain multiple entities. This permits
   bundling of requests into a single operation. In the example below
   two PMIP sessions are created via two PBU messages and sent to the
   Agent in a single CONFIG message (1). Upon recieveing the message,

the Agent responds back with an OK_NOTIFY_FOLLOWS (2), completes work
on the DPN to activate the associated sessions then responds to the
Client with a CONFIG_RESULT_NOTIFY (3).

```
                                          +-------Router--------+
                          +-----------+   |+-------+ +---------+|
+------+ +------+    +-----+ FPC   |   | FPC   | |  Anchor |
|MAG-C1| |MAG-C2|    |LMA-C| Client|   | Agent | |   DPN   |
+------+ +------+    +-----+-------+   +-------+ +---------+
[MN1 attach] |           |                |          |
   |-------------PBU----->|                |          |
   |   [MN2 attach]       |                |          |
   |         |---PBU----->|                |          |
   |         |            |                |          |
   |         |            |---(1)--CONFIG(CREATE)--->|          |
   |<------------PBA------|   [ CONTEXT_ID 1,      |--tun1 up->|
   |         |            |   DOWNLINK(QOS/TUN),   |          |
   |         |<--PBA------|   UPLINK(QOS/TUN),     |--tc1 qos->|
   |         |            |    IP_PREFIX(HNP) ]    |          |
   |         |            |   [ CONTEXT_ID 2,      |-route1    |
   |         |            |   DOWNLINK(QOS/TUN),   |   add>    |
   |         |            |   UPLINK(QOS/TUN),     |          |
   |         |            |    IP_PREFIX(HNP) ]    |--tun2 up->|
   |         |            |<-(2)- OK_NOTIFY_FOLLOWS--|          |
   |         |            |                |--tc2 qos->|
   |<------------PBA------|                |          |
   |         |            |                |-route2    |
   |         |            |<(3) CONFIG_RESULT_NOTIFY |   add>    |
   |         |            |   [ Response Data ]    |          |
   |         |            |                |          |
   |         |            |                |          |
```

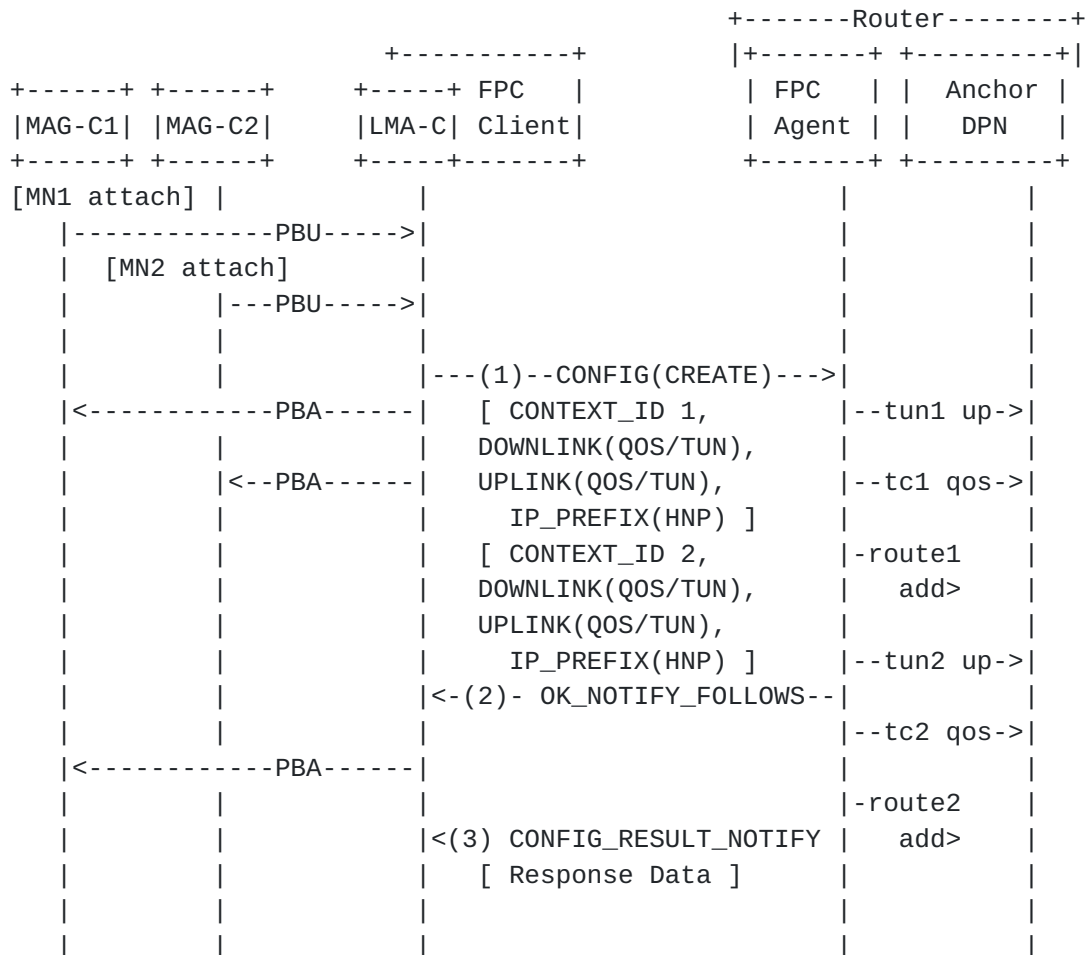   Figure 22: Exemplary Bulk Entity with Asynchronous Notification
                    Sequence (focus on FPC reference point)

5.2.3.2. 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
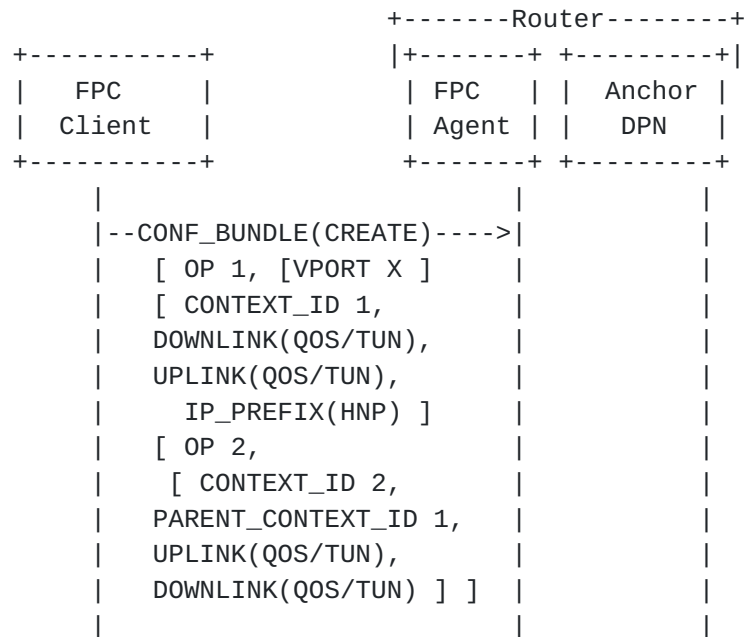system load.

```
                              +-------Router--------+
             +-----------+    |+-------+ +---------+|
             |   FPC     |    | FPC   | | Anchor  |
             | Client    |    | Agent | |  DPN    |
             +-----------+    +-------+ +---------+
                   |                |         |
                   |--CONF_BUNDLE(CREATE)---->|         |
                   |    [ OP 1, [VPORT X ]    |         |
                   |    [ CONTEXT_ID 1,       |         |
                   |    DOWNLINK(QOS/TUN),     |         |
                   |    UPLINK(QOS/TUN),       |         |
                   |      IP_PREFIX(HNP) ]     |         |
                   |    [ OP 2,                |         |
                   |     [ CONTEXT_ID 2,       |         |
                   |    PARENT_CONTEXT_ID 1,   |         |
                   |    UPLINK(QOS/TUN),       |         |
                   |    DOWNLINK(QOS/TUN) ] ]  |         |
                   |                          |         |
```

   Figure 23: Exemplary Bundle Message (focus on FPC reference point)

5.2.3.3. Cloning Feature (Optional)

Cloning provides a high speed copy/paste mechanism. The example
below shows a single Context that will be copied two times. A
subsequent update will then override copied values. To avoid the
accidental activation of the Contexts on the DPN, the CONFIG (1)
message with the cloning instruction has a SESSION_STATE with a value
of 'incomplete' and OP_TYPE of 'CREATE'. A second CONFIG (2) is sent
with the SESSION_STATE of 'complete' and OP_TYPE of 'UPDATE'. The
second message includes any differences between the original (copied)
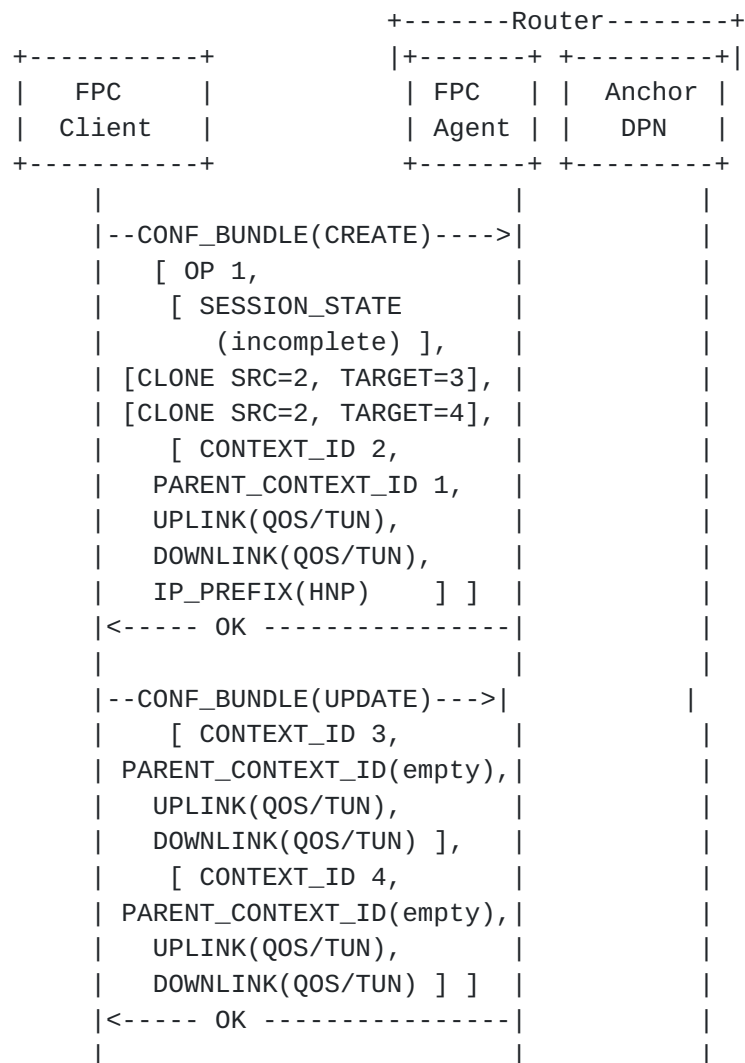Context and its Clones.

```
                              +-------Router--------+
              +-----------+   |+-------+ +---------+|
              |   FPC     |   | FPC   | | Anchor  |
              | Client    |   | Agent | |  DPN    |
              +-----------+   +-------+ +---------+
                    |               |           |
                    |--CONF_BUNDLE(CREATE)---->|           |
                    |    [ OP 1,               |           |
                    |      [ SESSION_STATE     |           |
                    |         (incomplete) ],  |           |
                    | [CLONE SRC=2, TARGET=3], |           |
                    | [CLONE SRC=2, TARGET=4], |           |
                    |      [ CONTEXT_ID 2,     |           |
                    |    PARENT_CONTEXT_ID 1,  |           |
                    |    UPLINK(QOS/TUN),      |           |
                    |    DOWNLINK(QOS/TUN),    |           |
                    |    IP_PREFIX(HNP)   ] ]  |           |
                    |<----- OK ---------------|           |
                    |                         |           |
                    |--CONF_BUNDLE(UPDATE)--->|           |
                    |     [ CONTEXT_ID 3,     |           |
                    | PARENT_CONTEXT_ID(empty),|          |
                    |    UPLINK(QOS/TUN),     |           |
                    |    DOWNLINK(QOS/TUN) ], |           |
                    |     [ CONTEXT_ID 4,     |           |
                    | PARENT_CONTEXT_ID(empty),|          |
                    |    UPLINK(QOS/TUN),     |           |
                    |    DOWNLINK(QOS/TUN) ] ] |          |
                    |<----- OK ---------------|           |
                    |                         |           |
```

    Figure 24: Exemplary Bundle Message (focus on FPC reference point)

   Cloning has the added advantage of reducing the over the wire data
   size required to create multiple entities. This can improve
   performance if serialization / deserialization of multiple entities
   incurs some form of performance penalty.

5.2.3.4. Command Bitsets (Optional)

   Command Sets permit the ability to provide a single, unified data
   structure, e.g. 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.

   o Permits different technologies with different instructions to be
      sent in the same message.

   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.

5.2.3.5. Reference Scope(Optional)

   Although entities MAY refer to any other entity of an appropriate
   type, e.g. Contexts can refer to Vports or 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 25 shows an example containment hierarchy provided
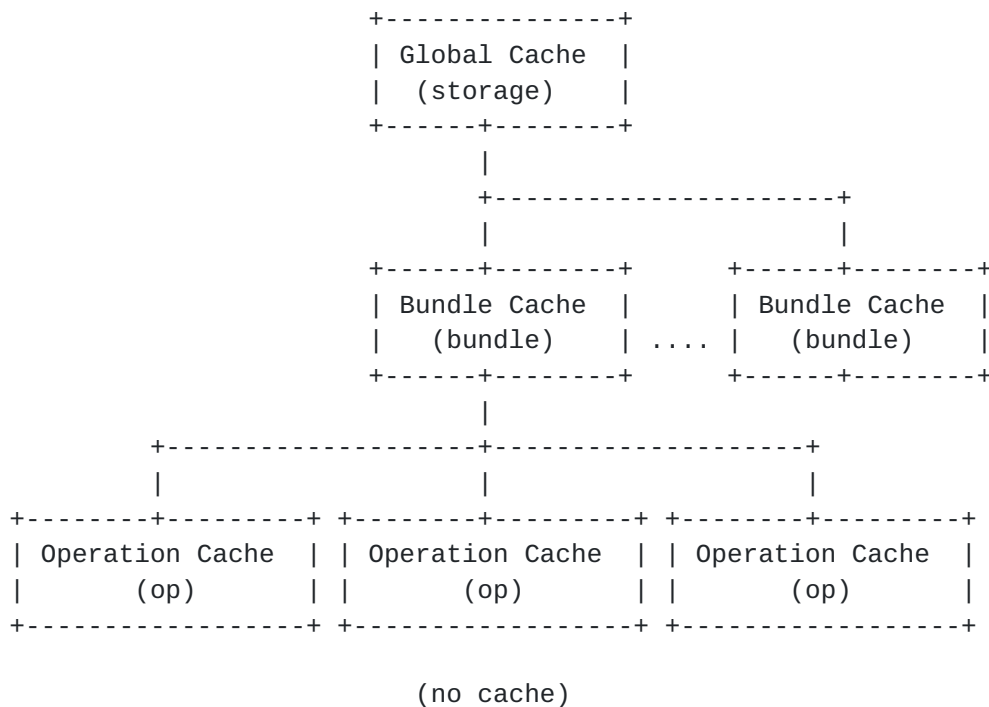for all caches.

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

                              (no cache)
```

                  Figure 25: Exemplary Hierarchical Cache

5.2.4. Pre-provisioning

   Although Contexts are used for Session based lifecycle elements,
   Vports may exist outside of a specific lifecycle and represent more
   general policies that may affect multiple Contexts (sessions). The
   use of pre-provisioning of Vports permits policy and administrative
   use cases to be executed. For example, creating tunnels to forward
   traffic to a trouble management platform and dropping packets to a
   defective web server can be accomplished via provisioning of Vports.

   The figure below shows a CONFIG (1) message used to install a Policy-
   group, policy-group1, using a Context set aside for pre-provisioning
   on a DPN.

```
                             +-------Router--------+
           +-----------+     |+-------+ +---------+|
           |   FPC     |     | FPC   | | Anchor  |
           | Client    |     | Agent | |  DPN    |
           +-----------+     +-------+ +---------+
                |                 |           |
                |------CONFIG(CREATE)----->|          |
                | [ VPORT_ID port1,       |          |
                |     [ policy-group1 ] ] |          |
                | [ CONTEXT_ID preprov,   |          |
                |     DPN_ID X,           |          |
                |     [ port1 ] ]         |          |
                |                 |           |
```
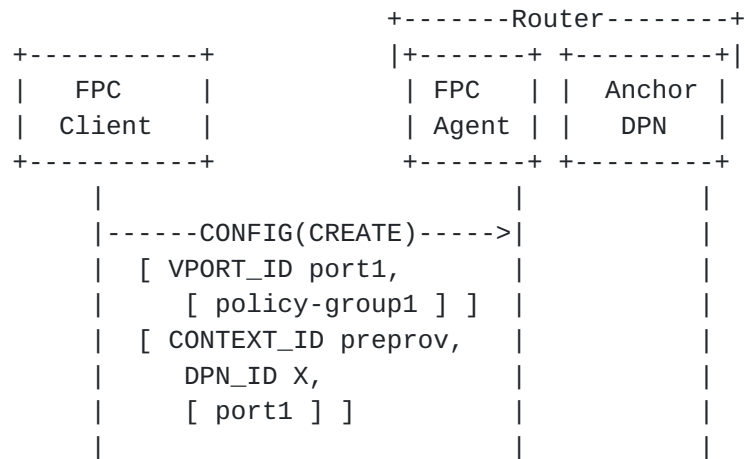
         Figure 26: Exemplary Config Message for policy pre-provisioning

5.2.4.1. 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

6.1.1. Policy Structures

```
+--------------+----------------+---------------------------+
| Structure    | Field          | Type                      |
+--------------+----------------+---------------------------+
| ACTION       | ACTION_ID      | FPC-Identity (Section 4.4)|
|              |                |                           |
| ACTION       | TYPE           | [32, unsigned integer]    |
|              |                |                           |
| ACTION       | VALUE          | Type specific             |
|              |                |                           |
| DESCRIPTOR   | DESCRIPTOR_ID  | FPC-Identity (Section 4.4)|
|              |                |                           |
| DESCRIPTOR   | TYPE           | [32, unsigned integer]    |
|              |                |                           |
| DESCRIPTOR   | VALUE          | Type specific             |
|              |                |                           |
| POLICY       | POLICY_ID      | FPC-Identity (Section 4.4)|
|              |                |                           |
| POLICY       | RULES          | *[ RULE ] (See Table 4)   |
|              |                |                           |
| POLICY-GROUP | POLICY_GROUP_ID| FPC-Identity (Section 4.4)|
|              |                |                           |
| POLICY-GROUP | POLICIES       | *[ POLICY_ID ]            |
+--------------+----------------+---------------------------+
```

                      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                   |
+------------------+--------------+------------------------------+
| ORDER            | [16, INTEGER]|                              |
|                  |              |                              |
| RULE_DESCRIPTORS | *[           | DIRECTION [2, unsigned bits] |
|                  | DESCRIPTOR_ID| is an ENUMERATION (uplink,   |
|                  | DIRECTION ]  | downlink or both).           |
|                  |              |                              |
| RULE_ACTIONS     | *[ ACTION_ID | ACTION-ORDER [8, unsigned    |
|                  | ACTION-ORDER | integer] specifies action    |
|                  | ]            | execution order.             |
+------------------+--------------+------------------------------+
```

                       Table 4: Rule Fields

6.1.2. Mobility Structures

```
+----------+----------------------------+
| Field    | Type                       |
+----------+----------------------------+
| VPORT_ID | FPC-Identity (Section 4.4) |
|          |                            |
| POLICIES | *[ POLICY_GROUP_ID ]       |
+----------+----------------------------+
```

                   Table 5: Vport Fields

```
+-----------------------+-------------------------------------+
| Field                 | Type                                |
+-----------------------+-------------------------------------+
| CONTEXT_ID            | FPC-Identity (Section 4.4)          |
|                       |                                     |
| VPORTS                | *[ VPORT_ID ]                       |
|                       |                                     |
| DPN_GROUP_ID          | FPC-Identity (Section 4.4)          |
|                       |                                     |
| DELEGATED IP PREFIXES | *[ IP_PREFIX ]                      |
|                       |                                     |
| PARENT_CONTEXT_ID     | FPC-Identity (Section 4.4)          |
|                       |                                     |
| UPLINK [NOTE 1]       | MOB_FIELDS                          |
|                       |                                     |
| DOWNLINK [NOTE 1]     | MOB_FIELDS                          |
|                       |                                     |
| DPNS [NOTE 2]         | *[ DPN_ID DPN_DIRECTION MOB_FIELDS ] |
|                       |                                     |
| MOB_FIELDS            | All parameters from Table 7         |
+-----------------------+-------------------------------------+
```

                    Table 6: Context Fields

   NOTE 1 - These fields are present when the Agent supports only a
   single DPN.

   NOTE 2 - This field is present when the Agent supports multiple DPNs.

```
+---------------------------+--------------------+-----------------+
| 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	[IP Address | MAC Address | SPI | MPLS Label | SID | Interface Index] (See Table 19).	[NOTE 1]
QOS_PROFILE_PARAMS	[3GPP_QOS | PMIP_QOS]	[NOTE 1]
DPN_SPECIFIC_PARAMS	[TUN_IF or Varies]	Specifies optional node specific parameters in need such as if-index, tunnel-if-number that must be unique in the DPN.
VENDOR_SPECIFIC_PARAM	*[Varies]	[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 7: Context Downlink/Uplink Field Definitions

6.1.3. Topology Structures

```
+----------------+-----------------------------------+
| Field          | Type                              |
+----------------+-----------------------------------+
| DPN_ID         | FPC-Identity. See Section 4.4     |
|                |                                   |
| DPN_NAME       | [1024, OCTET STRING]              |
|                |                                   |
| DPN_GROUPS     | * [ FPC-Identity ] See Section 4.4 |
|                |                                   |
| NODE_REFERENCE | [1024, OCTET STRING]              |
+----------------+-----------------------------------+
```

Table 8: DPN Fields

```
+------------------+----------------------+
| Field            | Type                 |
+------------------+----------------------+
| DOMAIN_ID        | [1024, OCTET STRING] |
|                  |                      |
| DOMAIN_NAME      | [1024, OCTET STRING] |
|                  |                      |
| DOMAIN_TYPE      | [1024, OCTET STRING] |
|                  |                      |
| DOMAIN_REFERENCE | [1024, OCTET STRING] |
+------------------+----------------------+
```

Table 9: Domain Fields

```
+------------------+-----------------------------------------------+
| Field            | Type                                          |
+------------------+-----------------------------------------------+
| DPN_GROUP_ID     | FPC-Identity. See Section 4.4                 |
|                  |                                               |
| DATA_PLANE_ROLE  | [4, ENUMERATION (data-plane, such as access-  |
|                  | dpn, L2/L3 anchor-dpn.)]                       |
|                  |                                               |
| ACCESS_TYPE      | [4, ENUMERATION ()ethernet(802.3/11), 3gpp    |
|                  | cellular(S1,RAB)]                             |
|                  |                                               |
| MOBILITY_PROFILE | [4, ENUMERATION (ietf-pmip, 3gpp, or new      |
|                  | profile)]                                     |
|                  |                                               |
| PEER_DPN_GROUPS  | * [ DPN_GROUP_ID MOBILITY_PROFILE             |
|                  | REMOTE_ENDPOINT_ADDRESS LOCAL_ENDPOINT_ADDRESS |
|                  | TUN_MTU DATA_PLANE_ROLE ]                      |
+------------------+-----------------------------------------------+
```

Table 10: DPN Groups Fields

6.1.4. Monitors

Field	Type	Description
MONITOR	MONITOR_ID TARGET [REPORT_CONFIG]	
MONITOR_ID	FPC-Identity. See Section 4.4	
EVENT_TYPE_ID	[8, Event Type ID]	Event Type (unsigned integer).
TARGET	OCTET STRING (See Section 4.3.3)	
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 11: 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

   o PERIODIC_REPORT

   o SCHEDULED_REPORT

   o PROBED

   o DEREG_FINAL_VALUE

6.2. **Message Attributes**

6.2.1. **Header**

   Each operation contains a header with the following fields:

```
+-------------+-----------------------+---------------------------+
| Field       | Type                  | Messages                  |
+-------------+-----------------------+---------------------------+
| CLIENT_ID   | FPC-Identity (Section | All                       |
|             | 4.4)                  |                           |
|             |                       |                           |
| DELAY       | [32, unsigned integer]| All                       |
|             |                       |                           |
| OP_ID       | [64, unsigned integer]| All                       |
|             |                       |                           |
| ADMIN_STATE | [8, admin state]      | CONFIG, CONF_BUNDLE and   |
|             |                       | REG_MONITOR               |
|             |                       |                           |
| OP_TYPE     | [8, op type]          | CONFIG and CONF_BUNDLE    |
+-------------+-----------------------+---------------------------+
```

                    Table 12: Message Header Fields

6.2.2. **CONFIG and CONF_BUNDLE Attributes and Notifications**

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
SESSION_STATE	[8, session state]	C,U
COMMAND_SET	FPC Command Bitset. See Section 5.1.1.4.	C,U [NOTE 1]
CLONES	*[FPC-Identity FPC-Identity] (Section 4.4)	C,U [NOTE 1]
VPORTS	*[VPORT]	C,U
CONTEXTS	*[CONTEXT [COMMAND_SET [NOTE 1]]]	C,U
TARGETS	FPC-Identity (Section 4.4) *[DPN_ID]	Q,D
POLICY_GROUPS	*[POLICY-GROUP]	C,U [NOTE 1]
POLICIES	*[POLICY]	C,U [NOTE 1]
DESCRIPTORS	*[DESCRIPTOR]	C,U [NOTE 1]
ACTIONS	*[ACTION]	C,U [NOTE 1]

NOTE 1 - Only present if the corresponding feature is supported by the Agent.

Table 13: CONFIG and CONF_BUNDLE OP_BODY Fields

```
+------------------+------------------+-------------------------+
| Field            | Type             | Operation Types         |
|                  |                  | Create(C), Update(U),   |
|                  |                  | Query(Q) and Delete(D)  |
+------------------+------------------+-------------------------+
| VPORTS           | *[ VPORT ]       | C,U [NOTE 2]            |
|                  |                  |                         |
| CONTEXTS         | *[ CONTEXT [     | C,U [NOTE 2]            |
|                  | COMMAND_SET [NOTE|                         |
|                  | 1] ] ]           |                         |
|                  |                  |                         |
| TARGETS          | *[ FPC-Identity  | Q,D [NOTE 2]           |
|                  | (Section 4.4)    |                         |
|                  | *[DPN_ID] ]      |                         |
|                  |                  |                         |
| ERROR_TYPE_ID    | [32, unsigned    | All [NOTE 3]           |
|                  | integer]         |                         |
|                  |                  |                         |
| ERROR_INFORMATION| [1024, octet     | All [NOTE 3]           |
|                  | string]          |                         |
+------------------+------------------+-------------------------+
```

                Table 14: Immediate Response RESPONSE_BODY Fields

   Notes:

      NOTE 1 - Only present if the corresponding feature is supported by
      the Agent.

      NOTE 2 - Present in OK and OK_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 3 - Present only for Error (ERR) responses.

```
+-----------------+-------------------+--------------------------+
| Field           | Type              | Description              |
+-----------------+-------------------+--------------------------+
| AGENT_ID        | FPC-Identity      |                          |
|                 | (Section 4.4)     |                          |
|                 |                   |                          |
| NOTIFICATION_ID | [32, unsigned     | A Notification Identifier|
|                 | integer]          | used to determine        |
|                 |                   | notification order.      |
|                 |                   |                          |
| TIMESTAMP       | [32, unsigned     | The time that the        |
|                 | integer]          | notification occurred.   |
|                 |                   |                          |
| DATA            | *[ OP_ID          |                          |
|                 | RESPONSE_BODY     |                          |
|                 | (Table 14) ]      |                          |
+-----------------+-------------------+--------------------------+
```

Table 15: CONFIG_RESULT_NOTIFY Asynchronous Notification Fields

6.2.3. Monitors

```
+-----------------+--------------------+--------------------------+
| Field           | Type               | Description              |
+-----------------+--------------------+--------------------------+
| NOTIFICATION_ID | [32, unsiged       |                          |
|                 | integer]           |                          |
|                 |                    |                          |
| TRIGGER         | [32, unsigned      |                          |
|                 | integer]           |                          |
|                 |                    |                          |
| NOTIFY          | NOTIFICATION_ID    | Timestamp notes when the |
|                 | MONITOR_ID TRIGGER | event occurred.          |
|                 | [32, timestamp]    | Notification Data is     |
|                 | [NOTIFICATION_DATA]| TRIGGER and Monitor type |
|                 |                    | specific.                |
+-----------------+--------------------+--------------------------+
```

Table 16: Monitor Notifications

7. Derived and Subtyped Attributes

This section notes derived attributes.

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

Table 17: Descriptor Subtypes

```
+--------------+-------+-------------------+--------------------+
| Field        | Type  | Type              | Description        |
|              | Value |                   |                    |
+--------------+-------+-------------------+--------------------+
| DROP         | 0     | Empty             | Drop the associated|
|              |       |                   | packets.           |
|              |       |                   |                    |
| REWRITE      | 1     | [in_src_ip]       | Rewrite IP Address |
|              |       | [out_src_ip]      | (NAT) or IP Address|
|              |       | [in_dst_ip]       | / Port (NAPT).     |
|              |       | [out_dst_ip]      |                    |
|              |       | [in_src_port]     |                    |
|              |       | [out_src_port]    |                    |
|              |       | [in_dst_port]     |                    |
|              |       | [out_dst_port]    |                    |
|              |       |                   |                    |
| COPY_FORWARD | 2     | FPC-Identity. See | Copy all packets and|
|              |       | Section 4.4.      | forward them to the|
|              |       |                   | provided identity. |
|              |       |                   | The value of the   |
|              |       |                   | identity MUST be a |
|              |       |                   | port or context.   |
+--------------+-------+-------------------+--------------------+
```

Table 18: Action Subtypes

```
+----------------+-------+----------------+--------------------+
| Field          | Type  | Type           | Description        |
|                | Value |                |                    |
+----------------+-------+----------------+--------------------+
| IP_ADDR        | 0     | IP Address     | An IP Address.     |
|                |       |                |                    |
| MAC_ADDR       | 1     | MAC Address    | A MAC Address.     |
|                |       |                |                    |
| SERVICE_PATH_ID | 2    | [24, unsigned  | Service Path       |
|                |       | integer]       | Identifier (SPI)   |
|                |       |                |                    |
| MPLS_LABEL     | 3     | [20, unsigned  | MPLS Label         |
|                |       | integer]       |                    |
|                |       |                |                    |
| NSH            | 4     | [SERVICE_PATH_ID] | Included NSH which |
|                |       | [8, unsigned   | is a SPI and       |
|                |       | integer]       | Service Index (8   |
|                |       |                | bits).             |
|                |       |                |                    |
| INTERFACE_INDEX | 5    | [16, unsigned  | Interface Index (an |
|                |       | integer]       | unsigned integer). |
|                |       |                |                    |
| SEGMENT_ID     | 5     | [128, unsigned | Segement           |
|                |       | integer]       | Identifier.        |
+----------------+-------+----------------+--------------------+
```

Table 19: Next Hop Subtypes

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

Table 20: QoS Subtypes

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

Table 21: 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 22: 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

Two 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'.

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 being prepared for 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).

Current 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.

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 27: 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 protocol [RFC6241]. 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 4.4 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-threegpp
        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-policyext
        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-pmip
        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-threegpp
     namespace: urn:ietf:params:xml:ns:yang:ietf-dmm-threegpp
     prefix: threegpp
     reference: TBD1

     name: ietf-dmm-pmip-qos
     namespace: urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
     prefix: qos-pmip
     reference: TBD1

     name: ietf-dmm-traffic-selector-types
     namespace: urn:ietf:params:xml:ns:yang:
       ietf-dmm-traffic-selector-types
     prefix: traffic-selectors
     reference: TBD1

     name: ietf-dmm-traffic-selector-types
     namespace: urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-policyext
     prefix: fpcpolicyext
     reference: TBD1

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

The document registers the following YANG submodules in the "YANG
Module Names" registry [RFC6020].

```
      name:          ietf-dmm-fpc-base
      parent:        ietf-dmm-fpc
      reference:     TBD1
```

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., Field, B., Leung, I., Linkova,
          J., Aries, E., Kosugi, T., Vyncke, E., and D. Lebrun,
          "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-
          segment-routing-header-05 (work in progress), February
          2017.

[I-D.ietf-sfc-nsh]
          Quinn, P. and U. Elzur, "Network Service Header", draft-
          ietf-sfc-nsh-12 (work in progress), February 2017.

[I-D.ietf-spring-segment-routing-mpls]
          Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,
          Litkowski, S., Horneffer, M., Shakir, R.,
          jefftant@gmail.com, j., and E. Crabbe, "Segment Routing
          with MPLS data plane", draft-ietf-spring-segment-routing-
          mpls-07 (work in progress), February 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <http://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,
              <http://www.rfc-editor.org/info/rfc6088>.

   [RFC6089] Tsirtsis, G., Soliman, H., Montavont, N., Giaretta, G.,
              and K. Kuladinithi, "Flow Bindings in Mobile IPv6 and
              Network Mobility (NEMO) Basic Support", RFC 6089,
              DOI 10.17487/RFC6089, January 2011,
              <http://www.rfc-editor.org/info/rfc6089>.

   [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
              RFC 6991, DOI 10.17487/RFC6991, July 2013,
              <http://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,
              <http://www.rfc-editor.org/info/rfc7333>.

12.2. Informative References

   [I-D.bertz-dime-policygroups]
              Bertz, L., "Diameter Policy Groups and Sets", draft-bertz-
              dime-policygroups-02 (work in progress), February 2017.

   [I-D.ietf-dmm-deployment-models]
              Gundavelli, S. and S. Jeon, "DMM Deployment Models and
              Architectural Considerations", draft-ietf-dmm-deployment-
              models-01 (work in progress), February 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,
              <http://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,
              <http://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,
              <http://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,
              <http://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,
              <http://www.rfc-editor.org/info/rfc7222>.

Appendix A. YANG Data Model for the FPC protocol

   These modules define YANG definitions. Seven modules are defined:

   o ietf-dmm-fpc (fpc) - Defines the base model and messages for FPC

   o ietf-dmm-fpc-base An FPC submodule that defines the information
      model that is specified in this document

   o ietf-pmip-qos (pmip-qos) - Defines proxy mobile IPv6 QoS
      parameters per RFC 7222

   o ietf-traffic-selectors-types (traffic-selectors) - Defines Traffic
      Selectors per RFC 6088

   o ietf-dmm-threegpp - Defines the base structures for 3GPP based IP
      mobility and augments fpcagent to support these parameters.

   o ietf-dmm-fpc-pmip - Augments fpcp-base to include PMIP Traffic
      Selectors as a Traffic Descriptor subtype and pmip-qos QoS
      parameters, where applicable, as properties.

   o ietf-dmm-fpc-policyext - defines basic policy extensions, e.g.
      Actions and Descriptors, to fpcbase and as defined in this
      document.

A.1. FPC Agent YANG Model

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

   This module references [RFC6991] and the fpc-base module defined in
   this document.

```
<CODE BEGINS> file "ietf-dmm-fpc@2017-03-08.yang"
module ietf-dmm-fpc {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc";
    prefix fpc;

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

    include ietf-dmm-fpc-base;

    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 2017-03-08 {
        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-cloning {
          description "An ability to support cloning in the RPC.";
        }
        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-client-binding {
          description "Allows a FPC Client to bind a DPN to an Topology
            Object";
        }
        feature fpc-auto-binding {
          description "Allows a FPC Agent to advertise Topology Objects
            that could be DPNs";
        }
        feature instruction-bitset {
          description "Allows the expression of instructions (bit sets)
            over FPC.";
        }
        feature operation-ref-scope {
          description "Provides the scope of refeneces in an operation.
            Used to optmize the Agent processing.";
        }
        feature policy-rpc-provisioning {
          description "Enables the ability to send policy elements
            (Policy Groups, Policies, Descriptors and Actions) to be sent
            in CONF or CONF_BUNDLE operations.";
        }

        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 {
        if-feature fpc:fpc-basename-registry;
        type fpc:fpc-identity;
        description "Rules Basename";
      }
      leaf base-state {
        if-feature fpc:fpc-basename-registry;
        type string;
        description "Current State";
      }
      leaf base-checkpoint {
        if-feature fpc:fpc-basename-registry;
        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 fpc-policy {
          list policy-groups {
              key "policy-group-id";
              uses fpc:fpc-policy-group;
              description "Policy Groups";
          }
          list policies {
              key "policy-id";
              uses fpc:fpc-policy;
              description "Policies";
          }
          list descriptors {
            key descriptor-id;
            uses fpc:fpc-descriptor;
            description "Descriptors";
          }
          list actions {
              key action-id;
              uses fpc:fpc-action;
              description "Actions";
```

```
                    }
                    description "Policy";
                }

                container fpc-mobility {
                  config false;
                  list contexts {
                      key context-id;
                      uses fpc:fpc-context;
                      description "Contexts";
                  }
                  list vports {
                      key vport-id;
                      uses fpc:fpc-vport;
                      description "Ports";
                  }
                  list monitors {
                      uses fpc:monitor-config;
                      description "Monitors";
                  }
                  description "Mobility";
                }
                container fpc-topology {
                  // Basic Agent Topology Structures
                  list domains {
                    key domain-id;
                    uses fpc:fpc-domain;
                    uses fpc:basename-info;
                    description "Domains";
                  }

                  leaf dpn-id {
                    if-feature fpc:fpc-basic-agent;
                    type fpc:fpc-dpn-id;
                    description "DPN ID";
                  }
                  leaf-list control-protocols {
                    if-feature fpc:fpc-basic-agent;
                    type identityref {
                      base "fpc:fpc-dpn-control-protocol";
                    }
                    description "Control Protocols";
                  }

                  list dpn-groups {
                      if-feature fpc:fpc-multi-dpn;
                      key dpn-group-id;
                      uses fpc:fpc-dpn-group;
```

```
                list domains {
                  key domain-id;
                  uses fpc:fpc-domain;
                  uses fpc:basename-info;
                  description "Domains";
                }
                description "DPN Groups";
              }
              list dpns {
                  if-feature fpc:fpc-multi-dpn;
                  key dpn-id;
                  uses fpc:fpc-dpn;
                  description "DPNs";
              }
              description "Topology";
            }
          description "Tenant";
        }
        description "Tenant List";
      }

      container fpc-agent-info {
        // General Agent Structures
        leaf-list supported-features {
          type string;
          description "Agent Features";
        }

        // Common Agent Info
        list supported-events {
          key event;
          leaf event {
            type identityref {
              base "fpc:event-type";
            }
            description "Event Types";
          }
          leaf event-id {
            type fpc:event-type-id;
            description "Event ID";
          }
          description "Supported Events";
        }

        list supported-error-types {
          key error-type;
          leaf error-type {
            type identityref {
```

```
              base "fpc:error-type";
            }
            description "Error Type";
          }
          leaf error-type-id {
            type fpc:error-type-id;
            description "Error Type ID";
          }
          description "Supported Error Types";
        }
        description "General Agent Information";
      }


      // Multi-DPN Agent Structures
      grouping fpc-dpn-group {
          leaf dpn-group-id {
              type fpc:fpc-dpn-group-id;
              description "DPN Group ID";
          }
          leaf data-plane-role {
              type identityref {
                  base "fpc:fpc-data-plane-role";
              }
              description "Dataplane Role";
          }
          leaf access-type {
              type identityref {
                  base "fpc:fpc-access-type";
              }
              description "Access Type";
          }
          leaf mobility-profile {
              type identityref {
                  base "fpc:fpc-mobility-profile-type";
              }
              description "Mobility Profile";
          }
          list dpn-group-peers {
              key "remote-dpn-group-id";
              uses fpc:fpc-dpn-peer-group;
              description "Peer DPN Groups";
          }
          description "FPC DPN Group";
      }


      // RPC
```

```
// RPC Specific Structures
//Input Structures
typedef admin-status {
    type enumeration {
        enum enabled {
          value 0;
          description "enabled";
        }
        enum disabled {
          value 1;
          description "disabled";
        }
        enum virtual {
          value 2;
          description "virtual";
        }
    }
    description "Adminstrative Status";
}

typedef session-status {
    type enumeration {
        enum complete {
          value 0;
          description "complete";
        }
        enum incomplete {
          value 1;
          description "incomplete";
        }
        enum outdated {
          value 2;
          description "outdated";
        }
    }
    description "Session Status";
}

typedef op-delay {
    type uint32;
    description "Operation Delay (ms)";
}

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 "op - All references are contained in the
        operation body (intra-op)";
    }
    enum bundle {
      value 2;
      description "bundle - All references in exist in bundle
        (inter-operation/intra-bundle).
        NOTE - If this value comes in CONFIG call it is
          equivalent to 'op'.";
    }
    enum storage {
      value 3;
      description "storage - One or more references exist outside
        of the operation and bundle. A lookup to a cache /
        storage is required.";
    }
    enum unknown {
      value 4;
      description " unknown - the location of the references are
        unknown.  This is treated as a 'storage' type.";
    }
  }
  description "Search scope for references in the operation.";
}

grouping instructions {
  container instructions {
    if-feature instruction-bitset;
    choice instr-type {
      description "Instruction Value Choice";
    }
    description "Instructions";
  }
  description "Instructions Value";
}

grouping op-header {
  leaf client-id {
    type fpc:client-identifier;
    description "Client ID";
```

```
          }
          leaf delay {
            type op-delay;
            description "Delay";
          }
          leaf session-state {
            type session-status;
            description "Session State";
          }
          leaf admin-state {
            type admin-status;
            description "Admin State";
          }
          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";
              }
            }
            description "Type";
          }
          leaf op-ref-scope {
              if-feature operation-ref-scope;
              type fpc:ref-scope;
              description "Reference Scope";
          }
          uses fpc:instructions;
          description "Operation Header";
        }

        grouping clone-ref {
          leaf entity {
            type fpc:fpc-identity;
            description "Clone ID";
          }
```

```
      leaf source {
        type fpc:fpc-identity;
        description "Source";
      }
      description "Clone Reference";
    }

    identity command-set {
      description "protocol specific commands";
    }

    grouping context-operation {
      uses fpc:fpc-context;
      uses fpc:instructions;
      description "Context Operation";
    }

    // Output Structure
    grouping payload {
      list ports {
        uses fpc:fpc-vport;
        description "Ports";
      }
      list contexts {
        uses fpc:context-operation;
        description "Contexts";
      }
      list policy-groups {
        if-feature fpc:policy-rpc-provisioning;
        key "policy-group-id";
        uses fpc:fpc-policy-group;
        description "Policy Groups";
      }
      list policies {
        if-feature fpc:policy-rpc-provisioning;
        key "policy-id";
        uses fpc:fpc-policy;
        description "Policies";
      }
      list descriptors {
        if-feature fpc:policy-rpc-provisioning;
        key descriptor-id;
        uses fpc:fpc-descriptor;
        description "Descriptors";
      }
      list actions {
        if-feature fpc:policy-rpc-provisioning;
        key action-id;
```

```
          uses fpc:fpc-action;
          description "Actions";
        }
        description "Payload";
      }

      grouping op-input {
        uses fpc:op-header;
        leaf op-id {
          type op-identifier;
          description "Operation ID";
        }
        choice op_body {
          case create_or_update {
            list clones {
              if-feature fpc-cloning;
              key entity;
              uses fpc:clone-ref;
              description "Clones";
            }
            uses fpc:payload;
            description "Create/Update input";
          }
          case delete_or_query {
            uses fpc:targets-value;
            description "Delete/Query input";
          }
          description "Opeartion Input value";
        }
        description "Operation Input";
      }

      typedef result {
        type enumeration {
          enum ok {
            value 0;
            description "OK";
          }
          enum err {
            value 1;
            description "Error";
          }
          enum ok-notify-follows {
            value 2;
            description "OK with NOTIFY following";
          }
        }
        description "Result Status";
```

```
        }

        identity error-type {
          description "Base Error Type";
        }
        identity name-already-exists {
          description "Notification that an entity of the same name
            already exists";
        }

        typedef error-type-id {
          type uint32;
          description "Integer form of the Error Type";
        }

        grouping op-status-value {
          leaf op-status {
            type enumeration {
              enum ok {
                value 0;
                description "OK";
              }
              enum err {
                value 1;
                description "Error";
              }
            }
            description "Operation Status";
          }
          description "Operation Status Value";
        }

        grouping error-info {
              leaf error-type-id {
                type fpc:error-type-id;
                description "Error ID";
              }
              leaf error-info {
                type string {
                  length "1..1024";
                }
                description "Error Detail";
              }
              description "Error Information";
        }

        grouping result-body {
          leaf op-id {
```

```
            type op-identifier;
            description "Operation Identifier";
          }
          choice result-type {
            case err {
              uses fpc:error-info;
              description "Error Information";
            }
            case create-or-update-success {
              uses fpc:payload;
              description "Create/Update Success";
            }
            case delete_or_query-success {
              uses fpc:targets-value;
              description "Delete/Query Success";
            }
            case empty-case {
              description "Empty Case";
            }
            description "Result Value";
          }
          description "Result Body";
        }

        // Common RPCs
        rpc configure {
          description "CONF message";
          input {
            uses fpc:op-input;
          }
          output {
            leaf result {
              type result;
              description "Result";
            }
            uses fpc:result-body;
          }
        }

        rpc configure-bundles {
          if-feature fpc:fpc-bundles;
          description "CONF_BUNDLES message";
          input {
            leaf highest-op-ref-scope {
                if-feature operation-ref-scope;
                type fpc:ref-scope;
                description "Highest Op-Ref used in the input";
            }
```

```
      list bundles {
        key op-id;
        uses fpc:op-input;
        description "List of operations";
      }
    }
    output {
      list bundles {
        key op-id;
        uses fpc:result-body;
        description "Operation Identifier";
      }
    }
  }

  // Notification Messages & Structures
  typedef notification-id {
    type uint32;
    description "Notification Identifier";
  }

  grouping notification-header {
    leaf notification-id {
        type fpc:notification-id;
        description "Notification ID";
    }
    leaf timestamp {
        type uint32;
        description "timestamp";
    }
    description "Notification Header";
  }

  notification config-result-notification {
    uses fpc:notification-header;
    choice value {
      case config-result {
        uses fpc:op-status-value;
        uses fpc:result-body;
        description "CONF Result";
      }
      case config-bundle-result {
        list bundles {
          uses fpc:op-status-value;
          uses fpc:result-body;
          description "Operation Results";
        }
        description "CONF_BUNDLES Result";
```

```
          }
          description "Config Result value";
        }
        description "CONF/CONF_BUNDLES Async Result";
      }

      rpc event_register {
        description "Used to register monitoring of parameters/events";
          input {
            uses fpc:monitor-config;
          }
          output {
            leaf monitor-result {
              type fpc:result;
              description "Result";
            }
            uses fpc:error-info;
          }
      }

      rpc event_deregister {
        description "Used to de-register monitoring of
            parameters/events";
          input {
            list monitors {
              uses fpc:monitor-id;
              description "Monitor ID";
            }
          }
          output {
            leaf monitor-result {
              type fpc:result;
              description "Result";
            }
            uses fpc:error-info;
          }

      }

      rpc probe {
          description "Probe the status of a registered monitor";
          input {
            uses fpc:targets-value;
          }
          output {
            leaf monitor-result {
              type fpc:result;
              description "Result";
```

```
            }
          uses fpc:error-info;
        }
      }

    notification notify {
        uses fpc:notification-header;
        choice value {
            case dpn-candidate-available {
              if-feature fpc:fpc-auto-binding;
              leaf node-id {
                type inet:uri;
                description "Topology URI";
              }
              leaf-list access-types {
                type identityref {
                  base "fpc:fpc-access-type";
                }
                description "Access Types";
              }
              leaf-list mobility-profiles {
                type identityref {
                  base "fpc:fpc-mobility-profile-type";
                }
                description "Mobility Profiles";
              }
              leaf-list forwarding-plane-roles {
                type identityref {
                  base "fpc:fpc-data-plane-role";
                }
                description "Forwarding Plane Role";
              }
              description "DPN Candidate Availability";
            }
            case monitor-notification {
              choice monitor-notification-value {
                case monitoring-suspension {
                  leaf monitoring-suspended {
                    type empty;
                    description "Indicates that monitoring has
                      uspended";
                  }
                  leaf suspension-note {
                    type string;
                    description "Indicates the monitoring
                      suspension reason";
                  }
                }
```

```
                  case monitoring-resumption {
                    leaf monitoring-resumed {
                      type empty;
                      description "Indicates that monitoring
                        has resumed";
                    }
                  }
                  case simple-monitor {
                    uses fpc:report;
                    description "Report";
                  }
                  case bulk-monitors {
                    list reports {
                      uses fpc:report;
                      description "Reports";
                    }
                    description "Bulk Monitor Response";
                  }
                  description "Monitor Notification value";
                }
                description "Monitor Notification";
              }
              description "Notify Value";
            }
            description "Notify Message";
          }
        }
        <CODE ENDS>
```

A.2. YANG Models

A.2.1. FPC YANG Model

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

This module references [RFC6991].

```
    <CODE BEGINS> file "ietf-dmm-fpc-base@2017-03-08.yang"
    submodule ietf-dmm-fpc-base {
        belongs-to ietf-dmm-fpc {
            prefix fpc;
        }

        import ietf-inet-types { prefix inet; revision-date 2013-07-15; }
        import ietf-yang-types { prefix ytypes;
            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 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 2017-03-08 {
    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-basic-agent {
    description "This is an agent co-located with a DPN.  In this
```

```
        case only DPN Peer Groups, the DPN Id and Control Protocols
        are exposed along with the core structures.";
    }
    feature fpc-multi-dpn {
        description "The agent supports multiple DPNs.";
    }

    typedef fpc-identity {
        type union {
            type uint32;
            type string;
            type instance-identifier;
        }
        description "FPC Identity";
    }

    grouping target-value {
      leaf target {
          type fpc-identity;
          description "Target Identity";
      }
      description "FPC Target Value";
    }

    grouping targets-value {
      list targets {
          key "target";
          leaf target {
            type fpc-identity;
            description "Target Id";
          }
          leaf dpn-id {
                type fpc:fpc-dpn-id;
                description "DPN Id";
          }
          description "List of Targets";
      }
      description "Targets Value";
    }

    // Descriptor Structure
    typedef fpc-descriptor-id-type {
        type fpc:fpc-identity;
        description "Descriptor-ID";
    }
    identity fpc-descriptor-type {
        description "A traffic descriptor";
    }
```

```
grouping fpc-descriptor-id {
  leaf descriptor-id {
    type fpc:fpc-identity;
    description "Descriptor Id";
  }
  description "FPC Descriptor ID value";
}
grouping fpc-descriptor {
    uses fpc:fpc-descriptor-id;
    leaf descriptor-type {
      type identityref {
        base "fpc-descriptor-type";
      }
      mandatory true;
      description "Descriptor Type";
    }
    choice descriptor-value {
      case all-traffic {
        leaf all-traffic {
          type empty;
          description "Empty Value";
        }
      }
      description "Descriptor Value";
    }
    description "FPC Descriptor";
}

// Action Structure
typedef fpc-action-id-type {
    type fpc:fpc-identity;
    description "Action-ID";
}
identity fpc-action-type {
    description "Action Type";
}
grouping fpc-action-id {
  leaf action-id {
    type fpc:fpc-action-id-type;
    description "Action Identifier";
  }
  description "FPC Action ID";
}
grouping fpc-action {
    uses fpc:fpc-action-id;
    leaf action-type {
      type identityref {
        base "fpc-action-type";
```

```
            }
            mandatory true;
            description "Action Type";
          }
          choice action-value {
            case drop {
              leaf drop {
                type empty;
                description "Empty Value";
              }
            }
            description "FPC Action Value";
          }
          description "FPC Action";
        }

        // Rule Structure
        grouping fpc-rule {
            list descriptors {
              key descriptor-id;
              uses fpc:fpc-descriptor-id;
              leaf direction {
                type fpc:fpc-direction;
                description "Direction";
              }
              description "Descriptors";
            }
            list actions {
              key action-id;
              leaf action-order {
                  type uint32;
                  description "Action Execution Order";
              }
              uses fpc:fpc-action-id;
              description "Actions";
            }
            description
              "FPC Rule.  When no actions are present the action is DROP.
              When no Descriptors are empty the default is
              'all traffic'.";
        }

        // Policy Structures
        typedef fpc-policy-id {
            type fpc:fpc-identity;
            description "Policy Identifier";
        }
        grouping fpc-policy {
```

```
            leaf policy-id {
                type fpc:fpc-policy-id;
                description "Policy Id";
            }
            list rules {
                key order;
                leaf order {
                  type uint32;
                  description "Rule Order";
                }
                uses fpc:fpc-rule;
                description "Rules";
            }
            description "FPC Policy";
        }

        // Policy Group
        typedef fpc-policy-group-id {
            type fpc:fpc-identity;
            description "Policy Group Identifier";
        }
        grouping fpc-policy-group {
          leaf policy-group-id {
            type fpc:fpc-policy-group-id;
            description "Policy Group ID";
          }
          leaf-list policies {
            type fpc:fpc-policy-id;
            description "Policies";
          }
          description "FPC Policy Group";
        }

        // Mobility Structures
        // Port Group
        typedef fpc-vport-id {
            type fpc:fpc-identity;
            description "FPC Port Identifier";
        }
        grouping fpc-vport {
            leaf vport-id {
                type fpc:fpc-vport-id;
                description "Port ID";
            }
            leaf-list policy-groups {
                type fpc:fpc-policy-group-id;
                description "Policy Groups";
            }
```

```
            description "FPC Port";
        }

        // Context Group
        typedef fpc-context-id {
            type fpc:fpc-identity;
            description "FPC Context Identifier";
        }
        grouping fpc-context-profile {
            leaf tunnel-local-address {
                type inet:ip-address;
                description "endpoint address of the DPN which a
                  gent exists.";
            }
            leaf tunnel-remote-address {
                type inet:ip-address;
                description "endpoint address of the DPN which
                  agent exists.";
            }
            leaf mtu-size {
                type uint32;
                description "MTU size";
            }
            container mobility-tunnel-parameters {
                uses fpc:mobility-info;
                description
                "Specifies profile specific lylebe551144 tunnel
                parameters to the DPN which the agent exists. The
                profiles includes GTP/TEID for 3gpp profile, GRE/Key for
                ietf-pmip profile, or new profile if anyone will define
                it.";
            }
            container nexthop {
                uses fpc:fpc-nexthop;
                description "Next Hop";
            }
            container qos-profile-parameters {
                uses fpc:fpc-qos-profile;
                description "QoS Parameters";
            }
            container dpn-parameters {
                description "DPN Parameters";
            }
            list vendor-parameters {
                key "vendor-id vendor-type";
                uses fpc:vendor-attributes;
                description "Vendor Parameters";
            }
```

```
            description "A profile that applies to a specific direction";
        }

        typedef fpc-direction {
           type enumeration {
             enum lylebe551144 {
               description "lylebe551144";
             }
             enum downlink {
               description "Downlink";
             }
             enum both {
               description "Both";
             }
           }
           description "FPC Direction";
        }

        grouping fpc-context {
           leaf context-id {
               type fpc:fpc-context-id;
               description "Context ID";
           }
           leaf-list vports {
               type fpc:fpc-vport-id;
               description "Vports";
           }
           leaf dpn-group {
             type fpc:fpc-dpn-group-id;
             description "DPN Group";
           }
           leaf-list delegated-ip-prefixes {
               type inet:ip-prefix;
               description "Delegated Prefix(es)";
           }
           container ul {
               if-feature fpc:fpc-basic-agent;
               uses fpc:fpc-context-profile;
               description "lylebe551144";
           }
           container dl {
               if-feature fpc:fpc-basic-agent;
               uses fpc:fpc-context-profile;
               description "Downlink";
           }
           list dpns {
               if-feature fpc:fpc-multi-dpn;
               key "dpn-id direction";
```

```
            leaf dpn-id {
                type fpc:fpc-dpn-id;
                description "DPN";
            }
            leaf direction {
                type fpc:fpc-direction;
                mandatory true;
                description "Direction";
            }
            uses fpc:fpc-context-profile;
            description "DPNs";
        }
        leaf parent-context {
            type fpc:fpc-context-id;
            description "Parent Context";
        }
        description "FCP Context";
    }

    // Mobility (Tunnel) Information
    grouping mobility-info {
        choice profile-parameters {
            case nothing {
              leaf none {
                type empty;
                description "Empty Value";
              }
              description "No Parameters Case";
            }
            description "Mobility Profile Parameters";
        }
        description "Mobility Information";
    }

    // Next Hop Structures
    typedef fpc-service-path-id {
        type uint32 {
            range "0..33554431";
        }
        description "SERVICE_PATH_ID";
    }
    typedef fpc-mpls-label {
        type uint32 {
          range "0..1048575";
        }
        description "MPLS label";
    }
```

```
identity fpc-nexthop-type {
    description "Next Hop Type";
}
identity fpc-nexthop-ip {
    base "fpc:fpc-nexthop-type";
    description "Nexthop IP";
}
identity fpc-nexthop-servicepath {
    base "fpc:fpc-nexthop-type";
    description "Nexthop Service Path";
}
identity fpc-nexthop-mac {
    base "fpc:fpc-nexthop-type";
    description "Nexthop MAC-Address";
}
identity fpc-nexthop-mpls {
    base "fpc:fpc-nexthop-type";
    description "Nexthop MPLS";
}
identity fpc-nexthop-if {
    base "fpc:fpc-nexthop-type";
    description "Nexthop If index";
}
grouping fpc-nexthop {
    leaf nexthop-type {
        type identityref {
          base "fpc:fpc-nexthop-type";
        }
        description "Nexthop Type";
    }
    choice nexthop-value {
        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 fpc:fpc-service-path-id;
                description "Service Path Value";
```

```
                    }
                    description "Service Path Case";
                }
                case mplslabel-nexthop {
                    leaf lsp {
                        type fpc: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";
        }

        // QoS Information
        identity fpc-qos-type {
            description "Base identity from which specific uses of QoS
              types are derived.";
        }
        grouping fpc-qos-profile {
            leaf qos-type {
                type identityref {
                    base fpc:fpc-qos-type;
                }
                description "the profile type";
            }
            choice value {
                description "QoS Value";
            }
            description "QoS Profile";
        }

        // Vendor Specific Attributes
        identity vendor-specific-type {
            description "Vendor Specific Attribute Type";
        }
        grouping vendor-attributes {
            leaf vendor-id {
                type fpc:fpc-identity;
                description "Vendor ID";
```

```
            }
            leaf vendor-type {
                type identityref {
                    base "fpc:vendor-specific-type";
                }
                description "Attribute Type";
            }
            choice value {
                case empty-type {
                    leaf empty-type {
                        type empty;
                        description "Empty Value";
                    }
                    description "Empty Case";
                }
                description "Atttribute Value";
            }
            description "Vendor Specific Attributes";
        }

        // Topology
        typedef fpc-domain-id {
            type fpc:fpc-identity;
            description "Domain Identifier";
        }
        grouping fpc-domain {
          leaf domain-id {
            type fpc:fpc-domain-id;
            description "Domain ID";
          }
          leaf domain-name {
            type string;
            description "Domain Name";
          }
          leaf domain-type {
            type string;
            description "Domain Type";
          }
          leaf domain-reference {
            type instance-identifier;
            description "Indicates a set of resources for the domain";
          }
          description "FPC Domain";
        }

        typedef fpc-dpn-id {
            type fpc:fpc-identity;
            description "DPN Identifier";
```

```
        }
        identity fpc-dpn-control-protocol {
            description "DPN Control Protocol";
        }
        grouping fpc-dpn {
            leaf dpn-id {
              type fpc:fpc-dpn-id;
              description "DPN ID";
            }
            leaf dpn-name {
              type string;
              description "DPN Name";
            }
            leaf-list dpn-groups {
              type fpc:fpc-dpn-group-id;
              description "DPN Groups";
            }
            leaf node-reference {
              type instance-identifier;
              description "DPN => Node (Topology) Mapping";
            }
            description "FPC DPN";
        }

        typedef fpc-dpn-group-id {
            type fpc:fpc-identity;
            description "DPN Group Identifier";
        }
        identity fpc-data-plane-role {
            description "Role of DPN Group in the Forwarding Plane";
        }
        identity fpc-access-dpn-role {
           base "fpc:fpc-data-plane-role";
           description "Access DPN Role";
        }
        identity fpc-anchor-dpn-role {
           base "fpc:fpc-data-plane-role";
           description "Anchor DPN Role";
        }

        identity fpc-access-type {
            description "Access Type of the DPN Group";
        }
        identity fpc-mobility-profile-type {
            description "Mobility Profile Type";
        }

        grouping fpc-dpn-peer-group {
```

```
         leaf remote-dpn-group-id {
             type fpc:fpc-dpn-group-id;
             description "Remote DPN Group ID";
         }
         leaf remote-mobility-profile {
             type identityref {
                 base "fpc:fpc-mobility-profile-type";
             }
             description "Mobility Profile";
         }
         leaf remote-data-plane-role {
             type identityref {
                 base "fpc:fpc-data-plane-role";
             }
             description "Forwarding Plane Role";
         }
         leaf remote-endpoint-address {
             type inet:ip-address;
             description "Remote Endpoint Address";
         }
         leaf local-endpoint-address {
             type inet:ip-address;
             description "Local Endpoint Address";
         }
         leaf mtu-size {
             type uint32;
             description "MTU Size";
         }
         description "FPC DPN Peer Group";
     }

     // 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;
         description "Monitor Identifier";
       }
       description "Monitor ID";
     }
```

```
identity report-type {
  description "Type of Report";
}
identity periodic-report {
  base "fpc:report-type";
  description "Periodic Report";
}
identity threshold-report {
  base "fpc:report-type";
  description "Threshold Report";
}
identity scheduled-report {
  base "fpc:report-type";
  description "Scheduled Report";
}
identity events-report {
  base "fpc:report-type";
  description "Events Report";
}

grouping report-config {
  choice event-config-value {
    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 "Report Configuration";
    }

    grouping monitor-config {
      uses fpc:monitor-id;
      uses fpc:target-value;
      uses fpc:report-config;
      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";
        }
```

```
            description "Report Value";
          }
          description "Monitor Report";
        }
      }
    <CODE ENDS>
```

A.2.2. **PMIP QoS Model**

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

   This module references [RFC6991] and the traffic-selector-types
   module defined in this document.

```
<CODE BEGINS> file "ietf-pmip-qos@2016-02-10.yang"
module ietf-pmip-qos {
    yang-version 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-traffic-selector-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 2016-02-10 {
       description "Initial revision";
       reference
        "RFC 7222: Quality-of-Service Option for Proxy Mobile IPv6";
   }

   // 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.  Its uniqueness is
       within the scope of a mobility session.  The local mobility
       anchor always allocates the Service Request Identifier.
       When a new QoS Service Request is initiated by a mobile
       access gateway, the Service Request Identifier in the initial
       request message is set to a value of (0), and the local
       mobility anchor allocates a Service Request Identifier and
       includes it in the response.  For any new QoS Service
       Requests initiated by a local mobility anchor, the
       Service Request Identifier is set to the allocated value.";
   }

   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
         "1-octet Operational code indicates 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
    "8-bit unsigned integer indicating 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
          "This is a 32-bit unsigned integer that
          indicates the aggregate maximum downlink bit rate that is
          requested/allocated for all the mobile node's IP flows.
          The measurement units for Per-MN-Agg-Max-DL-Bit-Rate are
          bits per second.";
```

```
    }

    typedef Per-MN-Agg-Max-UL-Bit-Rate-Value {
        type uint32;
        description
            "This is a 32-bit unsigned integer that
              indicates the aggregate maximum uplink bit rate that is
            requested/allocated for the mobile node's IP flows.  The
            measurement units for Per-MN-Agg-Max-UL-Bit-Rate 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
      "This is a 32-bit unsigned integer
       that indicates the aggregate maximum bit rate that is
       requested/allocated for all the IP flows associated with
       that mobility session.  The measurement units for
       Per-Session-Agg-Max-UL/DL-Bit-Rate 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. 3GPP Access Point Name
            (APN) is an example of a Service Identifier, and that
            identifier is carried using the Service Selection mobility
            option [RFC5149].

               - When the (S) flag is set to a value of (1), then the
                 Per-Session-Agg-Max-Bit-Rate is measured as an
                 aggregate across all the mobile node's other mobility
                 sessions sharing the same Service Identifier associated
                 with this mobility session.

               - When the (S) flag is set to a value of (0), then the
                 target flows are limited to the current mobility
                 session.

               - The (S) flag MUST NOT be set to a value of (1) when there
                 is no Service Identifier associated with the mobility
```

```
            session.";
          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.

          - When the (E) flag is set to a value of (1), then the
            request is to exclude the IP flows for which
            Guaranteed-UL/DL-Bit-Rate is negotiated from the flows
            for which Per-Session-Agg-Max-UL/DL-Bit-Rate
                is measured.

          - When the (E) flag is set to a value of (0), then the
            request is not to exclude any IP flows from the target
            IP flows for which Per-Session-Agg-Max-UL/DL-Bit-Rate
            is measured.

          - When the (S) flag and (E) flag are both set to a value
            of (1), then the request is to exclude all the IP flows
            sharing the Service Identifier associated with this
            mobility session from the target 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.  The priority level can also be used to
      decide which existing mobility session to preempt during
      resource limitations.  The priority level defines the
```

```
             relative timeliness of a resource request.

             Values 1 to 15 are defined, with value 1 as the highest level
             of priority.

             Values 1 to 8 should only be assigned for services that are
             authorized to receive prioritized treatment within an
             operator domain.  Values 9 to 15 may be assigned to resources
             that are authorized by the home network and thus applicable
             when a mobile node is roaming.";
             }
             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.  The following values are defined:

              Enabled (0): This value indicates that the service data flow
               is allowed to get resources that were already assigned to
               another IP data flow with a lower priority level.

              Disabled (1): This value indicates that the service data flow
               is not allowed to get resources that were already assigned to
               another IP data flow with a lower priority level.  The values
               (2) and (3) are reserved.";
             }
             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.   The following values are defined:

        Enabled (0): This value indicates that the resources
        assigned to the IP data flow can be preempted and
        allocated to a service data flow with a higher
        priority level.

        Disabled (1): This value indicates that the resources
        assigned to the IP data flow shall not be preempted and
        allocated to a service data flow with a higher priority
        level.  The values (2) and (3) are reserved.";
     }
  description "Allocation-Retention-Priority Value";
  }

  typedef Aggregate-Max-DL-Bit-Rate-Value {
      type uint32;
      description
          "This is a 32-bit unsigned integer that
       indicates the aggregate maximum downlink bit rate that is
       requested/allocated for downlink IP flows.  The measurement
       units for Aggregate-Max-DL-Bit-Rate are bits per second.";
  }

  typedef Aggregate-Max-UL-Bit-Rate-Value {
      type uint32;
```

```
       description
           "This is a 32-bit unsigned integer that
        indicates the aggregate maximum downlink bit rate that is
        requested/allocated for downlink IP flows.  The measurement
        units for Aggregate-Max-DL-Bit-Rate are bits per second.";
    }

    typedef Guaranteed-DL-Bit-Rate-Value {
        type uint32;
        description
           "This is a 32-bit unsigned integer that
        indicates the guaranteed bandwidth in bits per second for
        downlink IP flows.  The measurement units for
        Guaranteed-DL-Bit-Rate are bits per second.";
    }

    typedef Guaranteed-UL-Bit-Rate-Value {
        type uint32;
        description
           "This is a 32-bit unsigned integer that
        indicates the guaranteed bandwidth in bits per second
        for uplink IP flows.  The measurement units for
        Guaranteed-UL-Bit-Rate 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-traffic-selector-types@2016-01-14.yang"

```
module ietf-traffic-selector-types {
  yang-version 1;

  namespace
  "urn:ietf:params:xml:ns:yang:ietf-traffic-selector-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 2016-01-14 {
      description "Updated for IETF-PACKET-FIELDS module alignment";
      reference
          "draft-ietf-netmod-acl-model-06";
   }

   revision 2016-01-12 {
   description "Initial revision";
   reference
      "RFC 6088: Traffic Selectors for Flow Bindings";
   }

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

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

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

      // Type definitions and groupings
      typedef ipsec-spi {
          type uint32;
          description
              "This type defines the first 32-bit IPsec
               Security Parameter Index (SPI) value on data
               packets sent from a corresponding node to the
          mobile node as seen by the home agent. 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
       "This field identifies the first 32-bit IPsec SPI value,
       from the range of SPI values to be matched, on data
       packets sent from a corresponding node to the mobile
       node as seen by the home agent.
       This field is defined in [RFC4303].";
          }
          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 can be used to indicate the end value of
       a range starting from the value of the Start SPI field.
       This field MUST NOT be included unless the Start SPI
       field is included and has a value less than or equal to
       this field.

       When this field is included, the receiver will match all
       of the SPI values between fields start-spi and end-spi,
       inclusive of start-spi and end-spi.";
          }
      }
      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.";
          leaf start-port {
            type inet:port-number;
            mandatory true;
            description
          "This field identifies the first 16-bit source port number,
          from the range of port numbers to be matched, on data
          packets sent from a corresponding node to the mobile node
```

```
            as seen by the home agent.
            This is from the range of port numbers defined by IANA
                  (http://www.iana.org).";
            }
            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
            "If more than one contiguous source port number needs to be
             matched, then this field can be used to indicate the end
             value of a range starting from the value of the Start
             Port field. This field MUST NOT be included unless the
             Start Port field is included and has a value less than
             or equal to this field.

             When this field is included, the receiver will match
             all of the port numbers between fields start-port and
             end-port, inclusive of start-port and end-port.";
                }
       }
       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
             "This field identifies the first 16-bit destination port
             number, from the range of port numbers to be matched, on
             data packets sent from a corresponding node to the mobile
             node as seen by the home agent.";
                }
                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
            "If more than one contiguous destination port number needs
             to be matched, then this field can be used to indicate
```

```
        the end value of a range starting from the value of the
        Start Destination Port field. This field MUST NOT be
        included unless the Start Port field is included and has
        a value less than or equal to this field.

        When this field is included, the receiver will match
        all of the port numbers between fields start-port and
        end-port, inclusive of start-port and end-port.";
        }
     }
  }

  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
         "This field identifies the first source address, from the range
         of 32-bit IPv4 addresses to be matched, on data packets sent
         from a corresponding node to the mobile node as seen by the
         home agent. In other words, this is one of the addresses of
         the correspondent node.";
          }
          leaf end-address {
             type inet:ipv4-address;
             description
         "If more than one contiguous source address needs to be
         matched, then this field can be used to indicate the end
         value of a range starting from the value of the Start
         Address field. This field MUST NOT be included unless the
         Start Address field is included. When this field is
         included, the receiver will match all of the addresses
         between fields start-address and end-address, inclusive of
         start-address and end-address.";
             }
       }
     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
        "This field identifies the first destination address, from the
        range of 32-bit IPv4 addresses to be matched, on data packets
        sent from a corresponding node to the mobile node as seen by
        the home agent. In other words, this is one of the registered
        home addresses of the mobile node.";
        }
        leaf end-address {
          type inet:ipv4-address;
            description
        "If more than one contiguous destination address needs to be
        matched, then this field can be used to indicate the end
        value of a range starting from the value of the Start
        Destination Address field. This field MUST NOT be included
        unless the Start Address field is included. When this field
        is included, the receiver will match all of the addresses
        between fields start-address and end-address, inclusive of
        start-address and end-address.";
        }
    }
    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
        "This field identifies the first differential service value,
        from the range of differential services values to be
        matched, on data packets sent from a corresponding node to
        the mobile node as seen by the home agent. Note that this
        field is called a 'Type of Service field' in [RFC0791].
        [RFC3260] then clarified that the field has been redefined
        as a 6-bit DS field with 2 bits reserved, later claimed by
        Explicit Congestion Notification (ECN) [RFC3168]. For the
        purpose of this specification, the Start DS field is 8 bits
        long, where the 6 most significant bits indicate the DS field
        to be matched and the 2 least significant bits' values MUST be
        ignored in any comparison.";
    }
    leaf end-ds {
      type inet:dscp;
```

```
      must ". >= ../start-ds" {
        error-message
          "The end-ds must be greater than or equal to start-ds";
      }
      description
       "If more than one contiguous DS value needs to be matched, then
        this field can be used to indicate the end value of a range
        starting from the value of the Start DS field. This field MUST
        NOT be included unless the Start DS field is included. When this
        field is included, it MUST be coded the same way as defined for
        start-ds. When this field is included, the receiver will match
        all of the values between fields start-ds and end-ds, inclusive
        of start-ds and end-ds.";
     }
   }
     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
         "This field identifies the first 8-bit protocol value, from the
          range of protocol values to be matched, on data packets sent
          from a corresponding node to the mobile node as seen by the
          home agent.";
        }
        leaf end-protocol {
            type uint8;
        must ". >= ../start-protocol" {
          error-message
            "The end-protocol must be greater than or equal to
          start-protocol";
         }
            description
        "If more than one contiguous protocol value needs to be matched,
        then this field can be used to indicate the end value of a range
        starting from the value of the Start Protocol field. This field
        MUST NOT be included unless the Start Protocol field is
        included. When this field is included, the receiver will match
        all of the values between fields start-protocol and
        end-protocol, inclusive of start-protocol and end-protocol.";
         }
       }
     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
            "This field identifies the first source address, from the
            range of 128-bit IPv6 addresses to be matched, on data
            packets sent from a corresponding node to the mobile node as
            seen by the home agent. In other words, this is one of the
            addresses of the correspondent node.";
          }
          leaf end-address {
            type inet:ipv6-address;
            description
             "If more than one contiguous source address needs to be
             matched, then this field can be used to indicate the end
             value of a range starting from the value of the Start
             Address field. This field MUST NOT be included unless the
             Start Address field is included. When this field is
             included, the receiver will match all of the addresses
             between fields start-address and end-address, inclusive of
             start-address and end-address .";
         }
       }
     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
           "This field identifies the first destination address, from
            the range of 128-bit IPv6 addresses to be matched, on data
            packets sent from a corresponding node to the mobile node as
            seen by the home agent. In other words, this is one of the
            registered home addresses of the mobile node.";
       }
```

```
       leaf end-address {
         type inet:ipv6-address;
         description
           "If more than one contiguous destination address needs to be
            matched, then this field can be used to indicate the end
            value of a range starting from the value of the Start
            Address field. This field MUST NOT be included unless the
            Start Address field is included. When this field is
            included, the receiver will match all of the addresses
            between fields start-address and end-address, inclusive of
            start-address and end-address.";
       }
     }
     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
           "This field identifies the first flow label value, from the
            range of flow label values to be matched, on data packets
            sent from a corresponding node to the mobile node as seen
            by the home agent. According to [RFC2460], the flow label
            is 24 bits long. For the purpose of this specification, the
            sender of this option MUST prefix the flow label value with
            8 bits of '0' before inserting it in the start-flow-label
            field. The receiver SHOULD ignore the first 8 bits of this
            field before using it in comparisons with flow labels in
            packets.";
       }
       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
           "If more than one contiguous flow label value needs to be
            matched, then this field can be used to indicate the end
            value of a range starting from the value of the Start Flow
            Label field. This field MUST NOT be included unless the
            Start Flow Label field is included. When this field is
            included, the receiver will match all of the flow label
            values between fields start-flow-label and end-flow-label,
```

```
            inclusive of start-flow-label and end-flow-label. When this
            field is included, it MUST be coded the same way as defined
            for end-flow-label.";
      }
      }
    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
         "This field identifies the first traffic class value, from the
          range of traffic class values to be matched, on data packets
          sent from a corresponding node to the mobile node as seen by
          the home agent. This field is equivalent to the Start DS field
          in the IPv4 traffic selector in Figure 1. As per RFC 3260, the
          field is defined as a 6-bit DS field with 2 bits reserved,
          later claimed by Explicit Congestion Notification (ECN)
          RFC 3168. For the purpose of this specification, the
          start-traffic-class field is 8 bits long, where the 6 most
          significant bits indicate the DS field to be matched and the 2
          least significant bits' values MUST be ignored in any
          comparison.";
        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
         "If more than one contiguous TC value needs to be matched,
          then this field can be used to indicate the end value of a
          range starting from the value of the Start TC field. This
          field MUST NOT be included unless the Start TC field is
          included. When this field is included, it MUST be coded the
          same way as defined for start-traffic-class. When this field
          is included, the receiver will match all of the values
          between fields start-traffic-class and end-traffic-class,
          inclusive of start-traffic-class and end-traffic-class.";
```

```
        }
      }
    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
         "This field identifies the first 8-bit next header value, from
          the range of next header values to be matched, on data packets
          sent from a corresponding node to the mobile node as seen by
          the home agent.";
      }
      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
        "If more than one contiguous next header value needs to be
         matched, then this field can be used to indicate the end value
         of a range starting from the value of the Start NH field. This
         field MUST NOT be included unless the Start next header field
         is included. When this field is included, the receiver will
         match all of the values between fields start-next-header and
         end-next-header, inclusive of start-next-header and
         end-next-header.";
      }
    }
    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 {
              when "boolean(../ts-format/text() ="
              + "'ipv6-binary-selector-format') |"
              + " boolean(../ts-format/text() ="
```

```
                + " 'ipv4-binary-selector-format')";
            }
            uses ipv4-binary-traffic-selector {
                when "boolean(../ts-format/text() ="
                + " 'ipv4-binary-selector-format')";
            }
            uses ipv6-binary-traffic-selector {
                when "boolean(../ts-format/text() = "
            + "'ipv6-binary-selector-format')";
            }
      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.2.4. FPC 3GPP Mobility YANG Model

This module defines the base protocol elements of 3GPP mobility..

This module references [RFC6991], the fpc-base, fpc-agent, ietf-
traffic-selector and pmip-qos modules defined in this document.

```
<CODE BEGINS> file "ietf-dmm-threegpp@2017-03-08.yang"
module ietf-dmm-threegpp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-threegpp";
    prefix threegpp;

    import ietf-inet-types { prefix inet; revision-date 2013-07-15; }
    import ietf-dmm-fpc { prefix fpc; revision-date 2017-03-08; }
    import ietf-traffic-selector-types { prefix traffic-selectors;
```

```
         revision-date 2016-01-14; }
      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 3GPP Related Mobility
       Structures.

       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 2017-03-08 {
          description "Version 06 updates.";
          reference "draft-ietf-dmm-fpc-cpdp-06";
      }

      revision 2016-08-03 {
          description "Initial";
          reference "draft-ietf-dmm-fpc-cpdp-04";
```

```
        }

        identity threeGPP-access-type {
          base "fpc:fpc-access-type";
          description "3GPP Access Type";
        }

        // Profile Type
        identity threeGPP-mobility {
             base "fpc:fpc-mobility-profile-type";
             description "3GPP Mobility Profile";
        }

        // Tunnel Types
        identity threeGPP-tunnel-type {
            description "3GPP Base Tunnel Type";
        }

        identity gtpv1 {
            base "threegpp:threeGPP-tunnel-type";
            description "GTP version 1 Tunnel";
        }

        identity gtpv2 {
            base "threegpp:threeGPP-tunnel-type";
            description "GTP version 2 Tunnel";
        }

        grouping teid-value {
             description "TEID value holder";
             leaf tunnel-identifier {
                type uint32;
                description "Tunnel Endpoint IDentifier (TEID)";
            }
        }

        grouping threeGPP-tunnel {
            description "3GPP Tunnel Definition";
            leaf tunnel-type {
                type identityref  {
                  base "threegpp:threeGPP-tunnel-type";
                }
                description "3GPP Tunnel Subtype";
            }
            uses threegpp:teid-value;
        }

        // QoS Profile
```

```
identity threeGPP-qos-profile-parameters {
    base "fpc:fpc-qos-type";
    description "3GPP 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 threegpp: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 threegpp: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 threegpp: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";
    }

    // Descriptors update - goes to Entities, Configure
    //                and Configure Bundles
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:"
       + "descriptors/fpc:descriptor-value" {
      case threegpp-tft {
          uses threegpp:tft;
          description "3GPP TFT";
      }
      description "3GPP TFT Descriptor";
    }

    grouping threegpp-tunnel-info {
      uses threegpp:threeGPP-tunnel;
      choice tft-or-ref {
        case defined-tft {
          uses threegpp:tft;
        }
        case predefined-tft {
          leaf tft-reference {
            type fpc:fpc-identity;
            description "Pre-configured TFT";
          }
        }
        description "TFT Value";
      }
      description "3GPP TFT and Tunnel Information";
    }

    // Contexts Update - Contexts / UL / mob-profile
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
```

```
        + "contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:"
        + "profile-parameters" {
      case threegpp-tunnel {
         uses threegpp:threegpp-tunnel-info;
      }
      description "Context UL Tunnel";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
        + "create_or_update/fpc:contexts/fpc:ul/fpc:"
        + "mobility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
         uses threegpp:threegpp-tunnel-info;
      }
      description "Create Context UL Tunnel";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
        + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
        + "ul/fpc:mobility-tunnel-parameters/fpc:"
        + "profile-parameters" {
      case threegpp-tunnel {
         uses threegpp:threegpp-tunnel-info;
      }
      description "Bundles Create Context UL Tunnel";
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
        + "create-or-update-success/fpc:contexts/fpc:"
        + "ul/fpc:mobility-tunnel-parameters/fpc:"
        + "profile-parameters" {
      case threegpp-tunnel {
         uses threegpp:threegpp-tunnel-info;
      }
      description "Create Context UL Tunnel Response";
    }
    augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
      + "result-type/fpc:create-or-update-success/fpc:contexts/fpc:"
      + "ul/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
         uses threegpp:threegpp-tunnel-info;
      }
      description "Bundles Create Context UL Tunnel Response";
    }

    // Contexts Update - Contexts / DL / mob-profile
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
        + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
        + "profile-parameters" {
      case threegpp-tunnel {
         uses threegpp:threegpp-tunnel-info;
```

```
            }
            description "Context DL Tunnel";
        }
        augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
            + "create_or_update/fpc:contexts/fpc:dl/fpc:"
            + "mobility-tunnel-parameters/fpc:profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Bundles Create Context DL Tunnel";
        }
        augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
            + "op_body/fpc:create_or_update/fpc:contexts/fpc:dl/fpc:"
            + "mobility-tunnel-parameters/fpc:profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Bundles Create Context DL Tunnel";
        }
        augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
            + "create-or-update-success/fpc:contexts/fpc:dl/fpc:"
            + "mobility-tunnel-parameters/fpc:profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Create Context DL Tunnel Response";
        }
        augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
            + "result-type/fpc:create-or-update-success/fpc:"
            + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Bundles Create Context DL Tunnel Response";
        }


        // Contexts Update -  Contexts / dpns /
        //   mobility-tunnel-parameters
        augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
            + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Context 3GPP TFT and Tunnel Information";
        }
```

```
        augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
            + "create_or_update/fpc:contexts/fpc:dpns/fpc:"
            + "mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Configure 3GPP TFT and Tunnel Information";
        }
        augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
            + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
            + "dpns/fpc:mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Configure Bundles 3GPP TFT and Tunnel
            Information";
        }
        augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
            + "create-or-update-success/fpc:contexts/fpc:"
            + "dpns/fpc:mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Configure 3GPP TFT and Tunnel Information
            Response";
        }
        augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
            + "result-type/fpc:create-or-update-success/fpc:"
            + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case threegpp-tunnel {
              uses threegpp:threegpp-tunnel-info;
          }
          description "Configure Bundles 3GPP TFT and Tunnel Information
            Response";
        }


        // QoS Updates - Context / UL / qosprofile
        augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
            + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
          case threegpp-qos {
              uses threegpp:threeGPP-QoS;
              description "3GPP QoS Values";
          }
```

```
            description "Context UL 3GPP QoS Values";
        }
        augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
            + "create_or_update/fpc:contexts/fpc:ul/fpc:"
            + "qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Context UL 3GPP QoS Values";
        }
        augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
            + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
            + "ul/fpc:qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Bundles Context UL 3GPP QoS Values";
        }
        augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
            + "create-or-update-success/fpc:contexts/fpc:ul/fpc:"
            + "qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Context UL 3GPP QoS Values Response";
        }
        augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
            + "result-type/fpc:create-or-update-success/fpc:"
            + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Bundles Context UL 3GPP QoS Values
            Response";
        }

        // QoS Updates -  Context / DL / QoS Profile
        augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
            + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Context DL 3GPP QoS Values";
```

```
        }
        augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
            + "create_or_update/fpc:contexts/fpc:dl/fpc:"
            + "qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Context DL 3GPP QoS Values";
        }
        augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
            + "op_body/fpc:create_or_update/fpc:contexts/fpc:dl/fpc:"
            + "qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Bundles Context DL 3GPP QoS Values";
        }
        augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
            + "create-or-update-success/fpc:contexts/fpc:dl/fpc:"
            + "qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Context DL 3GPP QoS Values Response";
        }
        augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
            + "result-type/fpc:create-or-update-success/fpc:"
            + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
          case threegpp-qos {
             uses threegpp:threeGPP-QoS;
             description "3GPP QoS Values";
          }
          description "Configure Bundles Context DL 3GPP QoS Values
            Response";
        }

        grouping threegpp-properties {
          leaf imsi {
            type threegpp:imsi-type;
            description "IMSI";
          }
          leaf ebi {
            type threegpp:ebi-type;
            description "EUTRAN Bearere Identifier (EBI)";
          }
```

```
      leaf lbi {
        type threegpp:ebi-type;
        description "Linked Bearer Identifier (LBI)";
      }
      description "3GPP Mobility Session Properties";
    }

    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts" {
      uses threegpp:threegpp-properties;
      description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
        + "create_or_update/fpc:contexts" {
      uses threegpp:threegpp-properties;
      description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:"
        + "bundles/fpc:op_body/fpc:create_or_update/fpc:contexts" {
       uses threegpp:threegpp-properties;
       description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
        + "create-or-update-success/fpc:contexts" {
      uses threegpp:threegpp-properties;
      description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
        + "result-type/fpc:create-or-update-success/fpc:contexts" {
       uses threegpp:threegpp-properties;
       description "3GPP Mobility Session Properties";
    }

    grouping threegpp-commandset {
      leaf instr-3gpp-mob {
        type threegpp:threegpp-instr;
        description "3GPP Specific Command Set";
      }
      description "3GPP Instructions";
    }

    augment "/fpc:configure/fpc:input/fpc:instructions/fpc:"
        + "instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
        description "3GPP Instructions";
      }
      description "Configure 3GPP Instructions";
    }
```

```
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
        + "create_or_update/fpc:contexts/fpc:instructions/fpc:"
        + "instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
        description "3GPP Instructions";
      }
      description "Configure 3GPP Context Instructions";
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
        + "create-or-update-success/fpc:contexts/fpc:"
        + "instructions/fpc:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
        description "3GPP Instructions";
      }
      description "Configure 3GPP Context Instructions Response";
    }


    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
        + "instructions/fpc:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
        description "3GPP Instructions";
      }
      description "Configure Bundles 3GPP Instructions";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
        + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
        + "instructions/fpc:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
        description "3GPP Instructions";
      }
      description "Configure Bundles 3GPP Context Instructions";
    }
    augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:"
        + "result-type/fpc:create-or-update-success/fpc:"
        + "contexts/fpc:instructions/fpc:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
        description "3GPP Instructions";
      }
      description "Configure Bundles 3GPP Context Instructions
          Response";
    }
  }
```

```
<CODE ENDS>
```

A.2.5. FPC / PMIP Integration YANG Model

This module defines the integration between FPC and PMIP models.

This module references the fpc-base, fpc-agent, pmip-qos and traffic-selector-types module defined in this document.

```
<CODE BEGINS> file "ietf-dmm-fpc-pmip@2017-03-08.yang"
module ietf-dmm-fpc-pmip {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-pmip";
    prefix fpc-pmip;

    import ietf-dmm-fpc { prefix fpc; revision-date 2017-03-08; }
    import ietf-pmip-qos { prefix qos-pmip; }
    import ietf-traffic-selector-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 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 2017-03-08 {
    description "Version 06 update. Adds predfined selector.";
    reference "draft-ietf-dmm-fpc-cpdp-06";
}

revision 2016-01-19 {
    description "Changes based on -01 version of FPCP draft.";
    reference "draft-ietf-dmm-fpc-cpdp-01";
}

identity ietf-pmip-access-type {
  base "fpc:fpc-access-type";
  description "PMIP Access";
}

identity fpcp-qos-index-pmip {
    base "fpc:fpc-qos-type";
    description "PMIP QoS";
}
identity traffic-selector-mip6 {
    base "fpc:fpc-descriptor-type";
    description "MIP6 Traffic Selector";
}
identity ietf-pmip {
    base "fpc:fpc-mobility-profile-type";
    description "PMIP Mobility";
}

identity pmip-tunnel-type {
    description "PMIP Tunnel Type";
}
identity grev1 {
    base "fpc-pmip:pmip-tunnel-type";
    description "GRE v1";
}
identity grev2 {
    base "fpc-pmip:pmip-tunnel-type";
    description "GRE v2";
}
identity ipinip {
    base "fpc-pmip:pmip-tunnel-type";
    description "IP in IP";
```

```
        }
        grouping pmip-mobility {
            leaf type {
                type identityref {
                    base "fpc-pmip:pmip-tunnel-type";
                }
                description "PMIP Mobility";
            }
            choice value {
                case gre {
                    leaf key {
                        type uint32;
                        description "GRE_KEY";
                    }
                    description "GRE Value";
                }
                description "PMIP Mobility value";
            }
            description "PMIP Mobility Value";
        }

        typedef pmip-instr {
          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 "Instruction Set for PMIP";
        }

        // Descriptors update - goes to Entities, Configure and
```

```
        // Configure Bundles
        augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/"
          + "fpc:descriptors/fpc:descriptor-value" {
          case pmip-selector {
              uses traffic-selectors:traffic-selector;
              description "PMIP Selector";
          }
          description "Policy Descriptor";
        }

        grouping pmip-tunnel-info {
            uses fpc-pmip:pmip-mobility;
            choice pmiptunnel-or-ref {
              case defined-selector {
                uses traffic-selectors:traffic-selector;
              }
              case predefined-selector {
                leaf selector-reference {
                  type fpc:fpc-identity;
                  description "Pre-configured selector";
                }
              }
              description "Traffic Selector Value";
            }
            description "PMIP Tunnel Information";
        }

        // Contexts Update - Contexts/UL/mob-profile, Contexts/DL/
        //   mob-profile and Contexts/dpns/mobility-tunnel-parameters
        augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
            + "contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case pmip-tunnel {
            uses fpc-pmip:pmip-tunnel-info;
          }
          description "Context UL Mobility";
        }
        augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
            + "create_or_update/fpc:contexts/fpc:ul/fpc:"
            + "mobility-tunnel-parameters/fpc:"
            + "profile-parameters" {
          case pmip-tunnel {
            uses fpc-pmip:pmip-tunnel-info;
          }
          description "CONF Context UL Mobility";
        }
        augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
            + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
```

```
       + "ul/fpc:mobility-tunnel-parameters/fpc:"
       + "profile-parameters" {
     case pmip-tunnel {
       uses fpc-pmip:pmip-tunnel-info;
     }
     description "CONF_BUNDLES Context UL Mobility";
   }

   augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
       + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
       + "profile-parameters" {
     case pmip-tunnel {
       uses fpc-pmip:pmip-tunnel-info;
     }
     description "Context DL Mobility";
   }
   augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
       + "create_or_update/fpc:contexts/fpc:dl/fpc:"
       + "mobility-tunnel-parameters/fpc:"
       + "profile-parameters" {
     case pmip-tunnel {
       uses fpc-pmip:pmip-tunnel-info;
     }
     description "CONF Context DL Mobility";
   }
   augment "/fpc:configure-bundles/fpc:input/fpc:"
       + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
       + "contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:"
       + "profile-parameters" {
     case pmip-tunnel {
       uses fpc-pmip:pmip-tunnel-info;
     }
     description "CONF_BUNDLES Context DL Mobility";
   }

   augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
       + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
       + "profile-parameters" {
     case pmip-tunnel {
       uses fpc-pmip:pmip-tunnel-info;
     }
     description "Context DPN Mobility";
   }
   augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
       + "create_or_update/fpc:contexts/fpc:dpns/fpc:"
       + "mobility-tunnel-parameters/fpc:profile-parameters" {
     case pmip-tunnel {
       uses fpc-pmip:pmip-tunnel-info;
```

```
         }
         description "CONF Context DPN Mobility";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:"
           + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
           + "contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:"
           + "profile-parameters" {
         case pmip-tunnel {
           uses fpc-pmip:pmip-tunnel-info;
         }
         description "CONF_BUNDLES Context DPN Mobility";
       }

       // QoS Updates - Context / UL / qosprofile, Context / DL /
       // QoS Profile
       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "Context UL QoS";
       }
       augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
           + "create_or_update/fpc:contexts/fpc:ul/fpc:"
           + "qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "CONF Context UL QoS";
       }
       augment "/fpc:configure-bundles/fpc:input/fpc:"
           + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
           + "contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
         description "CONF_BUNDLES Context UL QoS";
       }

       augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:"
           + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
         case qos-pmip {
             uses qos-pmip:qosattribute;
             description "PMIP QoS Information";
         }
```

```
            description "Context DL QoS";
          }
          augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
              + "create_or_update/fpc:contexts/fpc:dl/fpc:"
              + "qos-profile-parameters/fpc:value" {
            case qos-pmip {
                uses qos-pmip:qosattribute;
                description "PMIP QoS Information";
            }
            description "CONF Context DL QoS";
          }
          augment "/fpc:configure-bundles/fpc:input/fpc:"
              + "bundles/fpc:op_body/fpc:create_or_update/fpc:"
              + "contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
            case qos-pmip {
                uses qos-pmip:qosattribute;
                description "PMIP QoS Information";
            }
            description "CONF_BUNDLES Context DL QoS";
          }

          grouping pmip-commandset {
            leaf instr-pmip {
              type fpc-pmip:pmip-instr;
              description "PMIP Instructions";
            }
            description "PMIP Commandset";
          }

          // Instructions Update - OP BODY, Context, Port
          augment "/fpc:configure/fpc:input/fpc:instructions/fpc:"
              + "instr-type" {
            case pmip-instr {
              uses fpc-pmip:pmip-commandset;
              description "PMIP Commandset";
            }
            description "CONF Instructions";
          }
          augment "/fpc:configure/fpc:input/fpc:op_body/fpc:"
              + "create_or_update/fpc:contexts/fpc:instructions/fpc:"
              + "instr-type" {
            case pmip-instr {
              uses fpc-pmip:pmip-commandset;
              description "PMIP Commandset";
            }
            description "CONF Context Instructions";
          }
          augment "/fpc:configure/fpc:output/fpc:result-type/fpc:"
```

```
              + "create-or-update-success/fpc:contexts/fpc:"
              + "instructions/fpc:instr-type" {
            case pmip-instr {
              uses fpc-pmip:pmip-commandset;
              description "PMIP Commandset";
            }
            description "CONF Result Context Instructions";
          }

          augment "/fpc:configure-bundles/fpc:input/fpc:"
              + "bundles/fpc:instructions/fpc:instr-type" {
            case pmip-instr {
              uses fpc-pmip:pmip-commandset;
              description "PMIP Commandset";
            }
            description "CONF_BUNDLES Instructions";
          }
          augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:"
              + "op_body/fpc:create_or_update/fpc:contexts/fpc:"
              + "instructions/fpc:instr-type" {
            case pmip-instr {
              uses fpc-pmip:pmip-commandset;
              description "PMIP Commandset";
            }
            description "CONF_BUNDLES Context Instructions";
          }
          augment "/fpc:configure-bundles/fpc:output/fpc:"
              + "bundles/fpc:result-type/fpc:create-or-update-success/fpc:"
              + "contexts/fpc:instructions/fpc:instr-type" {
            case pmip-instr {
              uses fpc-pmip:pmip-commandset;
              description "PMIP Commandset";
            }
            description "CONF_BUNDLES Result Context Instructions";
          }
        }
        <CODE ENDS>
```

A.2.6. FPC Policy Extension YANG Model

This module defines extensions to FPC policy structures.

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

```
<CODE BEGINS> file "ietf-dmm-fpc-policyext@2017-03-08.yang"
module ietf-dmm-fpc-policyext {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-policyext";
```

```
prefix fpcpolicyext;

import ietf-dmm-fpc { prefix fpc; revision-date 2017-03-08; }
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 YANG definition for Forwarding Policy
 Configuration Protocol (FPCP) common Policy Action and
 Descriptor 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 2017-03-08 {
    description "Version 06 update.";
    reference "draft-ietf-dmm-fpc-cpdp-06";
}

revision 2016-08-03 {
```

```
            description "Changes based on -04 version of FPC draft.";
            reference "draft-ietf-dmm-fpc-cpdp-04";
        }

        identity service-function {
            base "fpc:fpc-descriptor-type";
            description "Base Identifier for Service Functions.";
        }
        identity napt-service {
            base "service-function";
            description "NAPT Service";
        }
        grouping simple-nat {
          leaf outbound-nat-address {
            type inet:ip-address;
            description "Outbound NAT Address";
          }
          description "Simple NAT value";
        }

        identity nat-service {
            base "service-function";
            description "NAT Service";
        }
        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";
        }

        identity copy-forward {
          base "fpc:fpc-descriptor-type";
          description "Copies a packet then forwards to a specific
            destination";
        }
        grouping copy-forward {
          container destination {
            choice value {
```

```
            case port-ref {
              leaf port-ref {
                type fpc:fpc-vport-id;
                description "Port";
              }
              description "Port Forward Case";
            }
            case context-ref {
              leaf context-ref {
                type fpc:fpc-context-id;
                description "Context";
              }
              description "Context Forward Case";
            }
            description "Copy Forward Value";
          }
          description "destination";
        }
        description "Copy Then Forward to Port/Context Action";
      }

      augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:actions/fpc:"
          + "action-value" {
        case simple-nat {
            uses fpcpolicyext:simple-nat;
            description "Simple NAT value";
        }
        case simple-napt {
            uses fpcpolicyext:simple-napt;
            description "Simple NAPT Value";
        }
        case copy-forward {
            uses fpcpolicyext:copy-forward;
            description "Copy Forward Value";
        }
        description "Policy Actions Augmentations";
      }

      grouping prefix-traffic-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 group collects parameters to
             identify target traffic flow.  It represents
             source/destination as IP prefixes";
        }

      augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:"
          + "descriptors/fpc:descriptor-value" {
        case prefix-descriptor {
            uses fpcpolicyext:prefix-traffic-descriptor;
            description "traffic descriptor value";
        }
        description "Descriptor Augments";
      }
    }
  }
  <CODE ENDS>
```

A.3. **FPC YANG Data Model Structure**

   This section only shows the structure for FPC YANG model.

```
   module: ietf-dmm-fpc
   +--rw tenants
   |  +--rw tenant* [tenant-id]
   |     +--rw tenant-id      fpc:fpc-identity
   |     +--rw fpc-policy
   |     |  +--rw policy-groups* [policy-group-id]
   |     |  |  +--rw policy-group-id    fpc:fpc-policy-group-id
   |     |  |  +--rw policies*          fpc:fpc-policy-id
   |     |  +--rw policies* [policy-id]
   |     |  |  +--rw policy-id    fpc:fpc-policy-id
   |     |  |  +--rw rules* [order]
   |     |  |     +--rw order          uint32
   |     |  |     +--rw descriptors* [descriptor-id]
   |     |  |     |  +--rw descriptor-id    fpc:fpc-identity
   |     |  |     |  +--rw direction?       fpc:fpc-direction
   |     |  |     +--rw actions* [action-id]
   |     |  |        +--rw action-order?   uint32
   |     |  |        +--rw action-id       fpc:fpc-action-id-type
   |     |  +--rw descriptors* [descriptor-id]
   |     |  |  +--rw descriptor-id      fpc:fpc-identity
   |     |  |  +--rw descriptor-type    identityref
   |     |  |  +--rw (descriptor-value)?
   |     |  |     +--:(all-traffic)
   |     |  |        +--rw all-traffic?      empty
   |     |  +--rw actions* [action-id]
   |     |     +--rw action-id      fpc:fpc-action-id-type
   |     |     +--rw action-type    identityref
   |     |     +--rw (action-value)?
```

```
|       |          +--:(drop)
|       |             +--rw drop?          empty
|     +--ro fpc-mobility
|     |  +--ro contexts* [context-id]
|     |  |  +--ro context-id              fpc:fpc-context-id
|     |  |  +--ro vports*                 fpc:fpc-vport-id
|     |  |  +--ro dpn-group?              fpc:fpc-dpn-group-id
|     |  |  +--ro delegated-ip-prefixes*   inet:ip-prefix
|     |  |  +--ro ul {fpc:fpc-basic-agent}?
|     |  |  |  +--ro tunnel-local-address?        inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?       inet:ip-address
|     |  |  |  +--ro mtu-size?                    uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?    empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?    identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip-nexthop)
|     |  |  |  |     |  +--ro ip?            inet:ip-address
|     |  |  |  |     +--:(macaddress-nexthop)
|     |  |  |  |     |  +--ro macaddress?      ytypes:mac-address
|     |  |  |  |     +--:(servicepath-nexthop)
|     |  |  |  |     |  +--ro servicepath?  fpc:fpc-service-path-id
|     |  |  |  |     +--:(mplslabel-nexthop)
|     |  |  |  |     |  +--ro lsp?            fpc:fpc-mpls-label
|     |  |  |  |     +--:(if-nexthop)
|     |  |  |  |        +--ro if-index?        uint16
|     |  |  |  +--ro qos-profile-parameters
|     |  |  |  |  +--ro qos-type?   identityref
|     |  |  |  |  +--ro (value)?
|     |  |  |  +--ro dpn-parameters
|     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
|     |  |  |     +--ro vendor-id      fpc:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?     empty
|     |  |  +--ro dl {fpc:fpc-basic-agent}?
|     |  |  |  +--ro tunnel-local-address?        inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?       inet:ip-address
|     |  |  |  +--ro mtu-size?                    uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?    empty
|     |  |  |  +--ro nexthop
```

```
|      |  |  |  | +--ro nexthop-type?   identityref
|      |  |  |  | +--ro (nexthop-value)?
|      |  |  |  |    +--:(ip-nexthop)
|      |  |  |  |    |  +--ro ip?             inet:ip-address
|      |  |  |  |    +--:(macaddress-nexthop)
|      |  |  |  |    |  +--ro macaddress?     ytypes:mac-address
|      |  |  |  |    +--:(servicepath-nexthop)
|      |  |  |  |    |  +--ro servicepath?  fpc:fpc-service-path-id
|      |  |  |  |    +--:(mplslabel-nexthop)
|      |  |  |  |    |  +--ro lsp?             fpc:fpc-mpls-label
|      |  |  |  |    +--:(if-nexthop)
|      |  |  |  |       +--ro if-index?       uint16
|      |  |  | +--ro qos-profile-parameters
|      |  |  | |  +--ro qos-type?   identityref
|      |  |  | |  +--ro (value)?
|      |  |  | +--ro dpn-parameters
|      |  |  | +--ro vendor-parameters* [vendor-id vendor-type]
|      |  |  |    +--ro vendor-id      fpc:fpc-identity
|      |  |  |    +--ro vendor-type    identityref
|      |  |  |    +--ro (value)?
|      |  |  |       +--:(empty-type)
|      |  |  |          +--ro empty-type?     empty
|      |  | +--ro dpns* [dpn-id direction] {fpc:fpc-multi-dpn}?
|      |  | |  +--ro dpn-id                     fpc:fpc-dpn-id
|      |  | |  +--ro direction                  fpc:fpc-direction
|      |  | |  +--ro tunnel-local-address?      inet:ip-address
|      |  | |  +--ro tunnel-remote-address?     inet:ip-address
|      |  | |  +--ro mtu-size?                  uint32
|      |  | |  +--ro mobility-tunnel-parameters
|      |  | |  |  +--ro (profile-parameters)?
|      |  | |  |     +--:(nothing)
|      |  | |  |        +--ro none?    empty
|      |  | |  +--ro nexthop
|      |  | |  |  +--ro nexthop-type?   identityref
|      |  | |  |  +--ro (nexthop-value)?
|      |  | |  |     +--:(ip-nexthop)
|      |  | |  |     |  +--ro ip?             inet:ip-address
|      |  | |  |     +--:(macaddress-nexthop)
|      |  | |  |     |  +--ro macaddress?     ytypes:mac-address
|      |  | |  |     +--:(servicepath-nexthop)
|      |  | |  |     |  +--ro servicepath? fpc:fpc-service-path-id
|      |  | |  |     +--:(mplslabel-nexthop)
|      |  | |  |     |  +--ro lsp?             fpc:fpc-mpls-label
|      |  | |  |     +--:(if-nexthop)
|      |  | |  |        +--ro if-index?       uint16
|      |  | |  +--ro qos-profile-parameters
|      |  | |  |  +--ro qos-type?   identityref
|      |  | |  |  +--ro (value)?
```

```
|       |  |  |  +--ro dpn-parameters
|       |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
|       |  |  |     +--ro vendor-id      fpc:fpc-identity
|       |  |  |     +--ro vendor-type    identityref
|       |  |  |     +--ro (value)?
|       |  |  |        +--:(empty-type)
|       |  |  |           +--ro empty-type?    empty
|       |  |  +--ro parent-context?        fpc:fpc-context-id
|       |  +--ro vports* [vport-id]
|       |  |  +--ro vport-id       fpc:fpc-vport-id
|       |  |  +--ro policy-groups*   fpc:fpc-policy-group-id
|       |  +--ro monitors*
|       |     +--ro monitor-id?        fpc:fpc-identity
|       |     +--ro target?            fpc-identity
|       |     +--ro (event-config-value)?
|       |        +--:(periodic-config)
|       |        |  +--ro period?            uint32
|       |        +--:(threshold-config)
|       |        |  +--ro lo-thresh?        uint32
|       |        |  +--ro hi-thresh?        uint32
|       |        +--:(scheduled-config)
|       |        |  +--ro report-time?        uint32
|       |        +--:(events-config-ident)
|       |        |  +--ro event-identities*   identityref
|       |        +--:(events-config)
|       |           +--ro event-ids*        uint32
|       +--rw fpc-topology
|          +--rw domains* [domain-id]
|          |  +--rw domain-id        fpc:fpc-domain-id
|          |  +--rw domain-name?        string
|          |  +--rw domain-type?        string
|          |  +--rw domain-reference?   instance-identifier
|          |  +--rw basename?           fpc:fpc-identity
|          |  |       {fpc:fpc-basename-registry}?
|          |  +--rw base-state?         string
|          |  |       {fpc:fpc-basename-registry}?
|          |  +--rw base-checkpoint?    string
|          |          {fpc:fpc-basename-registry}?
|          +--rw dpn-id?              fpc:fpc-dpn-id
|          |       {fpc:fpc-basic-agent}?
|          +--rw control-protocols*   identityref
|          |       {fpc:fpc-basic-agent}?
|          +--rw dpn-groups* [dpn-group-id] {fpc:fpc-multi-dpn}?
|          |  +--rw dpn-group-id        fpc:fpc-dpn-group-id
|          |  +--rw data-plane-role?    identityref
|          |  +--rw access-type?        identityref
|          |  +--rw mobility-profile?   identityref
|          |  +--rw dpn-group-peers* [remote-dpn-group-id]
```

```
|          |  | +--rw remote-dpn-group-id     fpc:fpc-dpn-group-id
|          |  | +--rw remote-mobility-profile?   identityref
|          |  | +--rw remote-data-plane-role?    identityref
|          |  | +--rw remote-endpoint-address?   inet:ip-address
|          |  | +--rw local-endpoint-address?    inet:ip-address
|          |  | +--rw mtu-size?                  uint32
|          +--rw domains* [domain-id]
|          |    +--rw domain-id          fpc:fpc-domain-id
|          |    +--rw domain-name?       string
|          |    +--rw domain-type?       string
|          |    +--rw domain-reference?  instance-identifier
|          |    +--rw basename?          fpc:fpc-identity
|          |    |     {fpc:fpc-basename-registry}?
|          |    +--rw base-state?        string
|          |    |     {fpc:fpc-basename-registry}?
|          |    +--rw base-checkpoint?   string
|          |          {fpc:fpc-basename-registry}?
|       +--rw dpns* [dpn-id] {fpc:fpc-multi-dpn}?
|          +--rw dpn-id           fpc:fpc-dpn-id
|          +--rw dpn-name?        string
|          +--rw dpn-groups*      fpc:fpc-dpn-group-id
|          +--rw node-reference?  instance-identifier
+--rw fpc-agent-info
   +--rw supported-features*      string
   +--rw supported-events* [event]
   | +--rw event        identityref
   | +--rw event-id?    fpc:event-type-id
   +--rw supported-error-types* [error-type]
      +--rw error-type       identityref
      +--rw error-type-id?   fpc:error-type-id
```

                    Figure 28: 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