DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 2, 2017

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

September 29, 2016

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-04.txt

Abstract

This document describes the solution of data-plane separation from control-plane which enables a flexible mobility management system using agent and client functions. To configure data-plane nodes and functions, the data-plane is abstracted by an agent interface to the client. The data-plane abstraction model is extensible in order to support many different type of mobility management systems and data-plane functions.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 2, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document describes Forwarding Policy Configuration (FPC), the solution of data-plane separation from control-plane which enables flexible mobility management systems using agent and client functions. To configure data-plane nodes and functions, the data-plane is abstracted in the agent which provides an interface to the client.

Control planes of mobility management systems, and/or any applications which require data-plane control, can utilize the FPC Client in flexible granularities of operation. The configuration operations are capable of configuring not only single Data-Plane Node (DPN) directly, but also multiple DPNs from abstracted data-plane models on the FPC agent.

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

Topology: DPNs are grouped and abstracted in terms of roles of mobility management such as access, anchors and domains. FPC Agent abstracts DPN-groups and consists of forwarding plane topology, such as access nodes assigned to a DPN-group which peers to a DPN-group of anchor nodes.

Policy: Policy abstracts policies which handle specific traffic flows or packets such as QoS, packet processing to rewrite headers, etc. A policy consists of one or multiple rules which are composed of Descriptors and Actions. Descriptors in a rule identify traffic flows and Actions apply treatments to packets matched to the Descriptors in the rule. An arbitrary set of policies is abstracted as a Policy-group which is applied to Ports.

Mobility: 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), 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 Ports which abstract those policiy sets. A Context can belong to multiple Ports which serve different kinds of purpose and policy. Monitors aprovide a mechanism to produce reports when events regarding Ports, Sessions, DPNs or the Agent occurs.

The Agent collects 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. Specific protocols and configurations to configure DPN from FPC Agent are out of scope of this document.

The data-plane abstraction model is extensible in order to support many different types of mobility management systems and data-plane functions. The architecture and protocol design of FPC intends not to tie to specific types of access technologies and mobility protocols.

2. Conventions and 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].

3. FPC Architecture

In accordance with the requirements of flexible data-plane functions deployment described in [RFC7333], FPC provides a means for mobility control-plane and applications to handle DPNs that must be configured with various roles of the mobility management aspect described in [I-D.ietf-dmm-deployment-models].

FPC uses building blocks of Agent, Client and data-plane abstraction models as the interface between the agent and the client.

Mobility control-plane and applications integrate the FPC Client function and connect to FPC Agent functions. The Client and the Agent communicate based on data-plane abstraction models described in Section 4. Along with models, the control-plane and the applications put forwarding policies for their mobility sessions on the Agent.

The Agent connects to DPN(s) to manage their configuration. These configurations are rendered from the forwarding policies by the Agent. FPC Agent may be implemented in a network controller that handles multiple DPNs or it also may be integrated into a DPN.

The FPC architecture supports multi-tenancy where the FPC enabled data-plane supports multiple tenants of mobile operator networks and/or applications. DPNs on the data-plane run in multiple data-plane roles which are defined per session, domain and tenant.

This architecture is illustrated in Figure 1. This document does not adopt a specific protocol for the FPC envelope protocol and it is out of scope. However it must be capable of supporting FPC protocol messages and transactions described in Section 5.

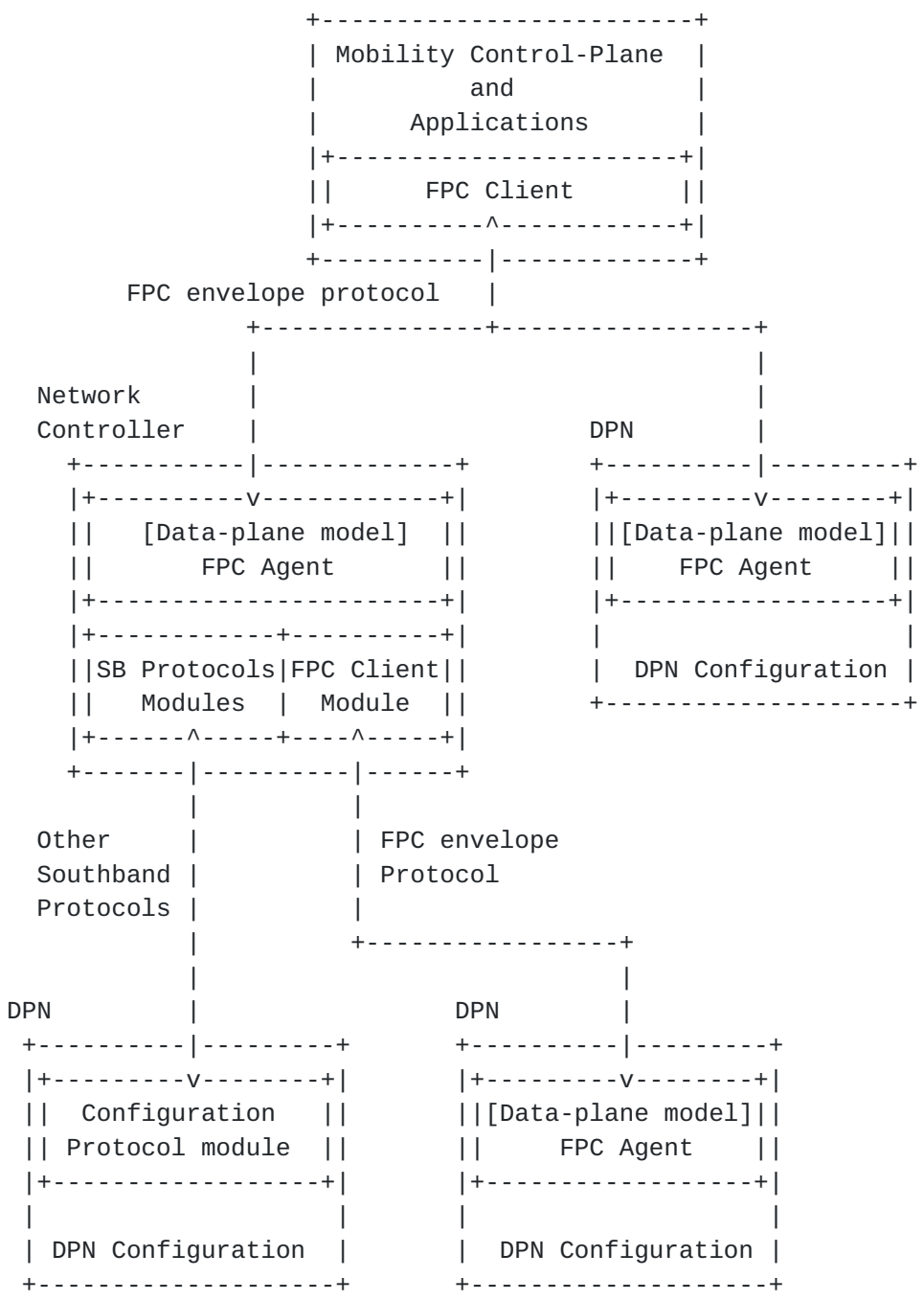

Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

Note that the FPC envelope protocol is only required to handle runtime data in the Mobility model. The rest of the FPC models, namely Topology and Policy, are pre-configured, therefore real-time data handling capabilities are not required for them. Operators that are tenants in the FPC data-plane can configure Toplogy and Policy on

the Agent through other means, such as Restconf [I-D.ietf-netconf-restconf] or Netconf [RFC6241].

4. Information Model

This section describes information model that represents the concept of FPC which is language and protocol neutral. Figure 2 is an overview of FPC data-plane abstraction model.

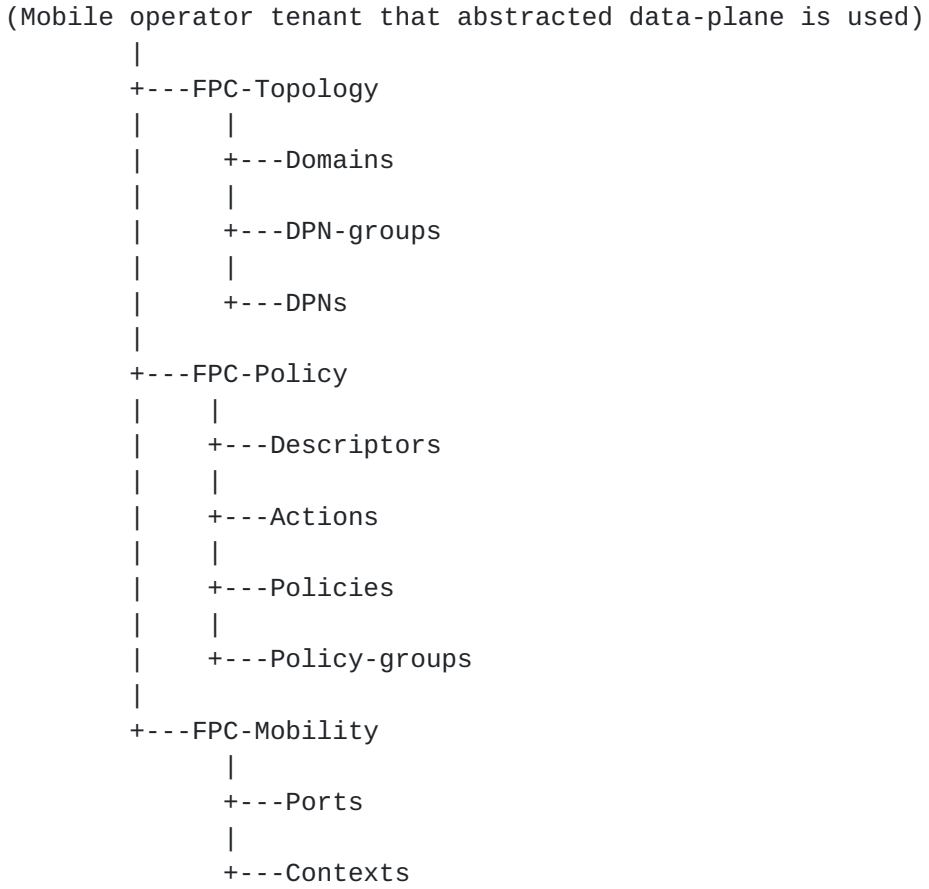

Figure 2: FPC Data-plane Abstraction Model

4.1. FPC-Topology

Topology abstraction enables an actual data-plane network to support multiple mobile operator's topologies of their data-plane. The FPC-Topology consists of DPNs, DPN-groups and Domains which abstract data-plane topologies for the Client's mobility control-planes and applications.

A mobile operator who utilizes a FPC enabled data-plane network can virtually create their DPNs along with their data-plane design on the Agent. The operator also creates a DPN-group of which the DPNs are attributed roles of mobility management such as access, anchors and domains.

4.1.1. Domains

A domain is defined by the operators to attribute DPN-groups to the domain. Domains may represent services or applications within the operator.

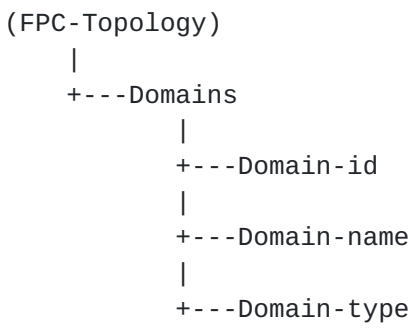

Figure 3: Domain Model Structure

Domain-id: Identifier of Domain. The ID format SHOULD refer to Section 4.4.

Domain-name: Defines Domain name.

Domain-type: Specifies which type of communication allowed within the domain, such as ipv4, ipv6, ipv4v6 or ieee802.

4.1.2. DPN-groups

A DPN-group defines a set of DPNs which share common data-plane attributes. DPN-groups consist data-plane topology that consists of a DPN-group of access nodes connecting to an anchor nodes DPN-group.

DPN Group has attributes such as the data-plane role, supported access technologies, mobility profiles, connected peer groups and domain.

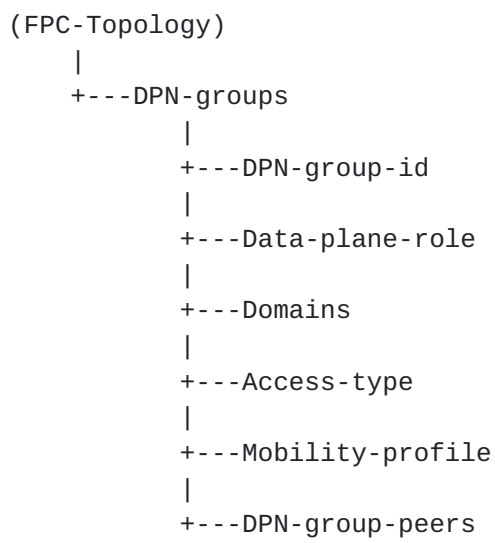

Figure 4: DPN-groups Model Structure

DPN-group-id: Defines identifier of DPN-group. The ID format SHOULD refer to Section 4.4.

Data-plane-role: Defines data-plane role of the DPN-group, such as access-dpn, L2/L3 or anchor-dpn.

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

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

Mobility-profile: Defines supported mobility profile, such as ietf-pmip, 3gpp, or new profiles defined as extensions of this specification. When those profiles are correctly defined, some or all data-plane parameters of contexts can be automatically derived from this profile by FPC Agent.

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

4.1.2.1. DPN-group Peers

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

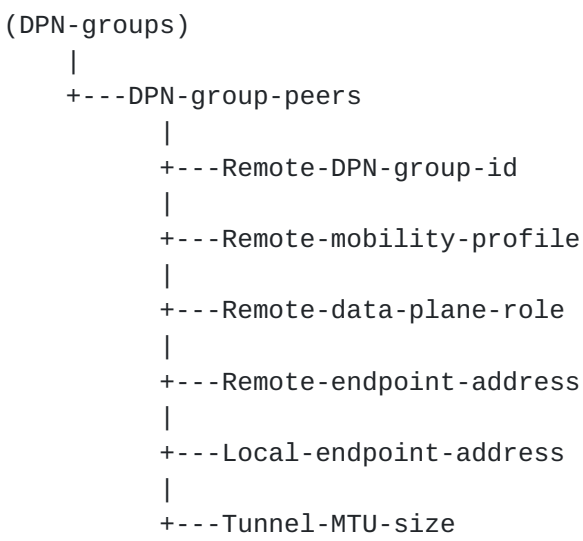

Figure 5: DPN-groups Peer Model Structure

Remote-DPN-group-id: Indicates peering DPN-Group.

Remote-mobility-profile: Defines mobility-profile used for this peer, currently defined profiles are ietf-pmip, 3gpp, or new profiles defined as extensions of this specification.

Remote-data-plane-role: Defines forwarding-plane role of 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.

Tunnel-MTU-size: Defines MTU size of tunnel.

4.1.3. DPNs

List of DPNs which defines all available nodes for a tenant of the FPC data-plane network. Role of a DPN in the data-plane is not determined until the DPN is attributed to a DPN-group.

A DPN may have multiple DPN-groups which are in different data-plane roles or domains. Mobility sessions of that DPN-groups are installed into actual data-plane nodes. The Agent defines DPN binding to actual nodes.

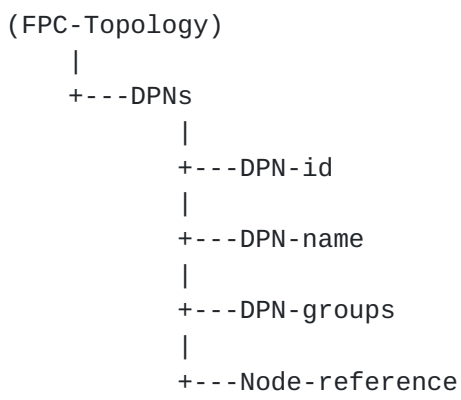

Figure 6: DPNs Model Structure

DPN-id: Defines identifier of DPN. The ID format SHOULD refer to Section 4.4.

DPN-name: Defines name of DPN.

DPN-groups: List of DPN-group which the DPN belongs to.

Node-reference: Indicates an actual node to which the Agent binds the DPN. The Agent SHOULD maintain that nodes information including IP address of management and control protocol to connect them.

4.2. FPC-Policy

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

4.2.1. Descriptors

List of Descriptors which defines classifiers of specific traffic flow, such as those based on source and destination addresses, protocols, port numbers of TCP/UDP/SCTP/DCCP or any packet. Note that Descriptors are extensibly defined by specific profiles which 3gpp, ietf or other SDOs produce. 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 is consumed by the Descriptor. Descriptors are assigned an identifier and contain a type and value.

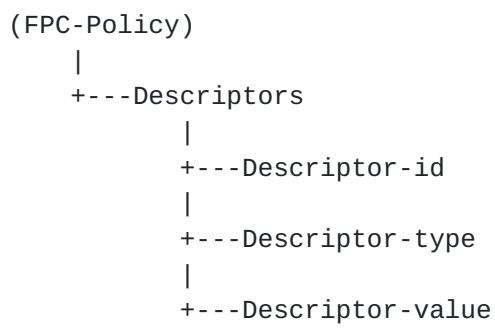

Figure 7: Descriptor Model Structure

Descriptor-id: Identifier of Descriptor. The ID format SHOULD refer to Section 4.4.

Descriptor-type: Defines descriptor type, which classifies specific traffic flow, such as source and destination addresses, protocols, port numbers of TCP/UDP/SCTP/DCCP or any packet.

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

4.2.2. Actions

List of Actions which defines treatment/actions to apply to classified traffic meeting the criteria defined by Descriptors. Actions include traffic management related activity such as shaping, policing based on given bandwidth, and connectivity management actions such as pass, drop, forward to given nexthop. Note that Actions are extensibly defined by specific profiles which 3gpp, ietf or other SDOs produce.

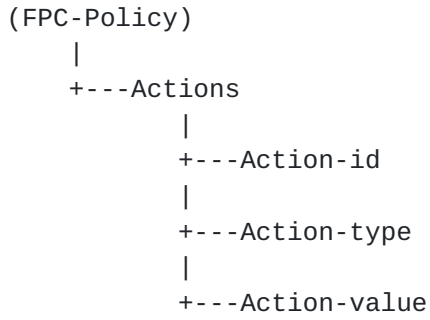

Figure 8: Action Model Structure

Action-id: Identifier of Action. The ID format SHOULD refer to Section 4.4.

Action-type: Defines action type, i.e. how to treat the specified traffic flow, e.g. pass, drop, forward to given nexthop value and shape, police based on given bandwidth value, etc.

Action-value: Specifies value of Action, such as bandwidth, nexthop address or drop explicitly, 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 as opposed to references to Rules.

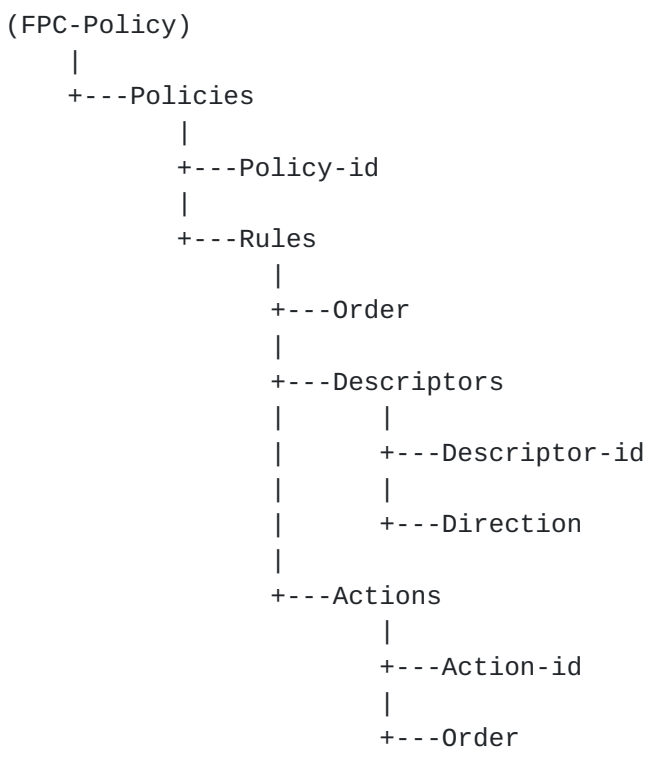

Figure 9: Policies Model Structure

Policy-id: Identifier of Policy. The ID format SHOULD refer 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 MUST be satisfied for the Rule to apply. These are internal structure to the Policy, i.e. it is not a first class, visible object at the top level of an Agent.

Order: Specifies ordering if the Rule has multiple Descriptors and Action sets.

Descriptors: List of Descriptors.

Descriptor-id: Indicates each Descriptor in the Rule.

Direction: Specifies which direction applies, such as upstream, downstream or both.

Actions: List of Actions.

Action-id: Indicates each Action in the rule.

Order: Specifies Action ordering if the Rule has multiple actions.

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.

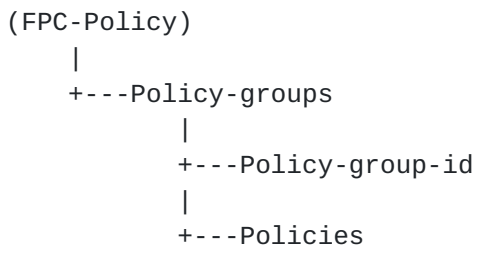

Figure 10: Policy-group Model Structure

Policy-group-id: Identifier of Policy-group. The ID format SHOULD refer to Section 4.4.

Policies: List of Policies in the Policy-group.

4.3. FPC-Mobility

The FPC-Mobility consists of Port and Context. 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 Port abstracts a set of policies applied to the Context.

4.3.1. Port

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

When a Port is indicated in a Context, the set of Descriptors and Actions in the Policies of the Port 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.

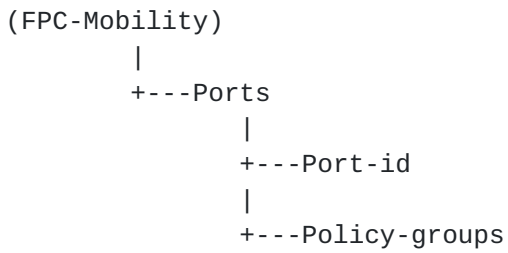

Figure 11: Port Model Structure

Port-id: Identifier of Port. The ID format SHOULD refer to Section 4.4.

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

4.3.2. Context

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

A Context directly describes traffic treatment policies in QoS profile and Mobility profiles or indirectly via Ports. Parameters in these profiles may be set by the FPC Client directly or indirectly derived from the set of Descriptors and Actions when the Ports indicate Policies which specify those descriptors and actions. If a Context doesn't have any Port, all parameters of the Context must be set by the Client.

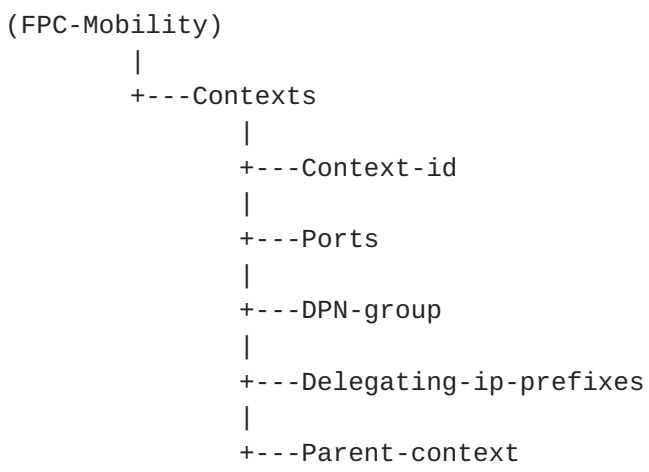

Figure 12: Common Context Model Structure

Context-id: Identifier of Context. The ID format SHOULD refer to Section 4.4.

Ports: List of Ports. When a Context is applied to Port(s), the context is configured by policies of those Port(s). Port-id references indicate Ports which apply to the Context. Context can be a part of multiple Ports which have different policies.

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

Delegating-ip-prefixes: List of IP prefixes to be delegated to the mobile node of the context.

Parent-context: Indicates context which the 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. The Context in single DPN case consists of following parameters for both direction of uplink and downlink.

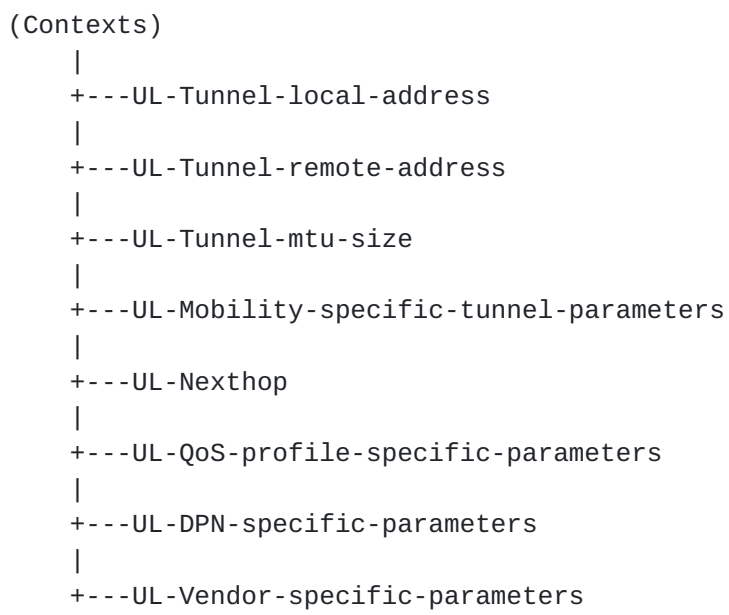

Figure 13: Uplink Context Model of Single DPN Structure

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

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

UL-Tunnel-Mtu-size: Specifies uplink MTU size of tunnel.

UL-Mobility-specific-tunnel-parameters: Specifies profile specific uplink 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 profiles defined by extensions of this specification.

UL-Nexthop: Indicates nexthop information of uplink in external network such as IP address, MAC address, SPI of service function chain, SID of segment routing, etc.

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

UL-DPN-specific-parameters: Specifies optional node specific parameters of uplink in need, 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 uplink.

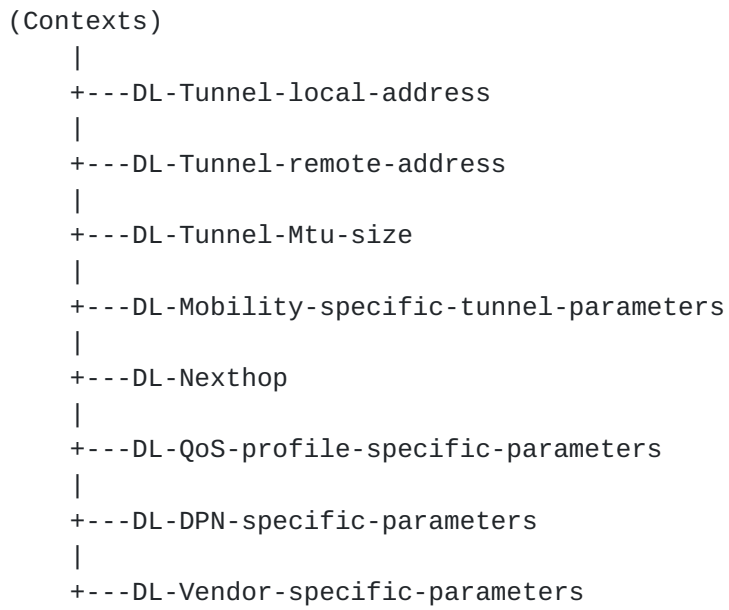

Figure 14: Downlink Context Model of Single DPN Structure

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

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

DL-Tunnel-Mtu-size: Specifies downlink MTU size of tunnel.

DL-Mobility-specific-tunnel-parameters: Specifies profile specific downlink 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 profiles defined by extensions of this specification.

DL-Nexthop: Indicates nexthop information of downlink in external network such as IP address, MAC address, SPI of service function chain, SID of segment routing, etc.

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

DL-DPN-specific-parameters: Specifies optional node specific parameters of downlink in need 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 downlink.

4.3.2.2. Multiple DPN Agent Case

Another case is when a FPC Agent connects to multiple DPNs. This Agent MUST maintain a set of Context data for each DPN. The Context contains a DPNs list where each entry of the list consists of the parameters in Figure 15. A Context data for one DPN has two entries for each direction of uplink and downlink.

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" or "downlink".

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.

Tunnel-mtu-size: Specifies the MTU size of tunnel on uplink or downlink.

Mobility-specific-tunnel-parameters: Specifies profile specific tunnel parameters for uplink or downlink of the DPN. The profiles includes GTP/TEID for 3gpp profile, GRE/Key for ietf-pmip profile, or new profiles defined by extensions of this specification.

Nexthop: Indicates nexthop information for uplink or downlink in external network of the DPN such as IP address, MAC address, SPI of service function chain, SID of segment routing, etc.

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

DPN-specific-parameters: Specifies optional node specific parameters for uplink or downlink of the DPN in need, 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.

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, e.g. a Monitor using a Threshold configuration cannot be applied to a context but it can be applied to a numeric property.

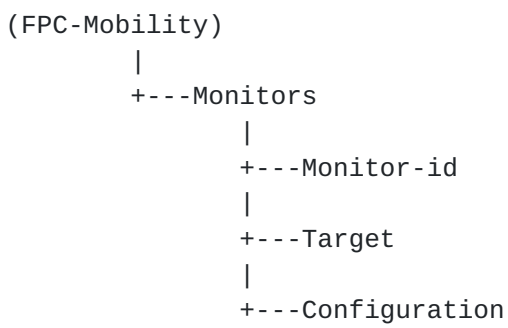

Figure 16: Common Monitor Model Structure

Monitor-id: Name of the Monitor. The ID format SHOULD refer to Section 4.4.

Target: Target to be monitored. This may be an event, a Context, a Port 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 even types that, if they occur and are related to the monitored attribute, will result in sending a notfication to the Client

* Scheduled reporting specifies the time (in seconds since Jan 1, 1970) when a notificaiton 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 uniquenes 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 assign 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 MUST NOT limit the types of format that dictate the choice of FPC protocol. It is noted that the choice of identifiers which are used in Mobility model should be suitable to handle runtime parameters in real-time. The Topology and Policy models are not restricted to meet that requirement as described in Section 3.

5. Protocol

5.1. Protocol Messages and Semantics

Five message types are supported:

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

Table 1: Client to Agent Messages

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

Messages that create or update Monitors and Entities, i.e. CONF, CONF_BUNDLES 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, then an operation will occur on the DPN IF the entity exists on the DPN. If set to 'active' the DPN will be provisioned. Values are 'enabled', 'disabled' or 'virtual'.

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 5.1.1.5.2. When returning an 'ok' of any kind, optional data may be present.

Two Agent notifications are supported:

Message	Type	Description
CONFIG_RESULT_NOTIFY	See Table 15	An asynchronous notification from Agent to Client based upon a previous CONFIG or CONFIG_BUNDLES request.
NOTIFY	See Table 16	An asynchronous notification from Agent to Client based upon a registered MONITOR.

Table 2: Agent to Client Messages (Notfications)

5.1.1. CONF and CONF_BUNDLES Messages

CONF and CONF_BUNDLES 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: specifies the Command Set IF the feature is supported (see Section 5.1.1.3).

BODY A list of Clones, if supported, Ports 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. Ports

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 attibutes.

5.1.1.2. 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 CONFIG_BUNDLES 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 Ports and Contexts can be cloned.

5.1.1.3. 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.

5.1.1.4. Reference Scope

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

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

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

- bundle - All references in exist in bundle (inter-operation/intra-bundle). NOTE - If this value comes in CONFIG call it is equivalent to 'op'.

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

- 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 Ports are present the scope MUST be 'storage' or 'uknown'.

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.

5.1.1.5. Operation Response

5.1.1.5.1. 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 a 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 5.1.1.3).

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.

5.1.1.5.2. Asynchronous Notification

A CONFIG_RESULT_NOTIFY occurs after the Agent has completed processing related to a CONFIG or CONFIG_BUNDLES 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.

5.2. Protocol Operation

5.2.1. Simple RPC Operation

An FPC Client and Agent MUST identify themself 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 Port, 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 paritially 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 paritially 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.

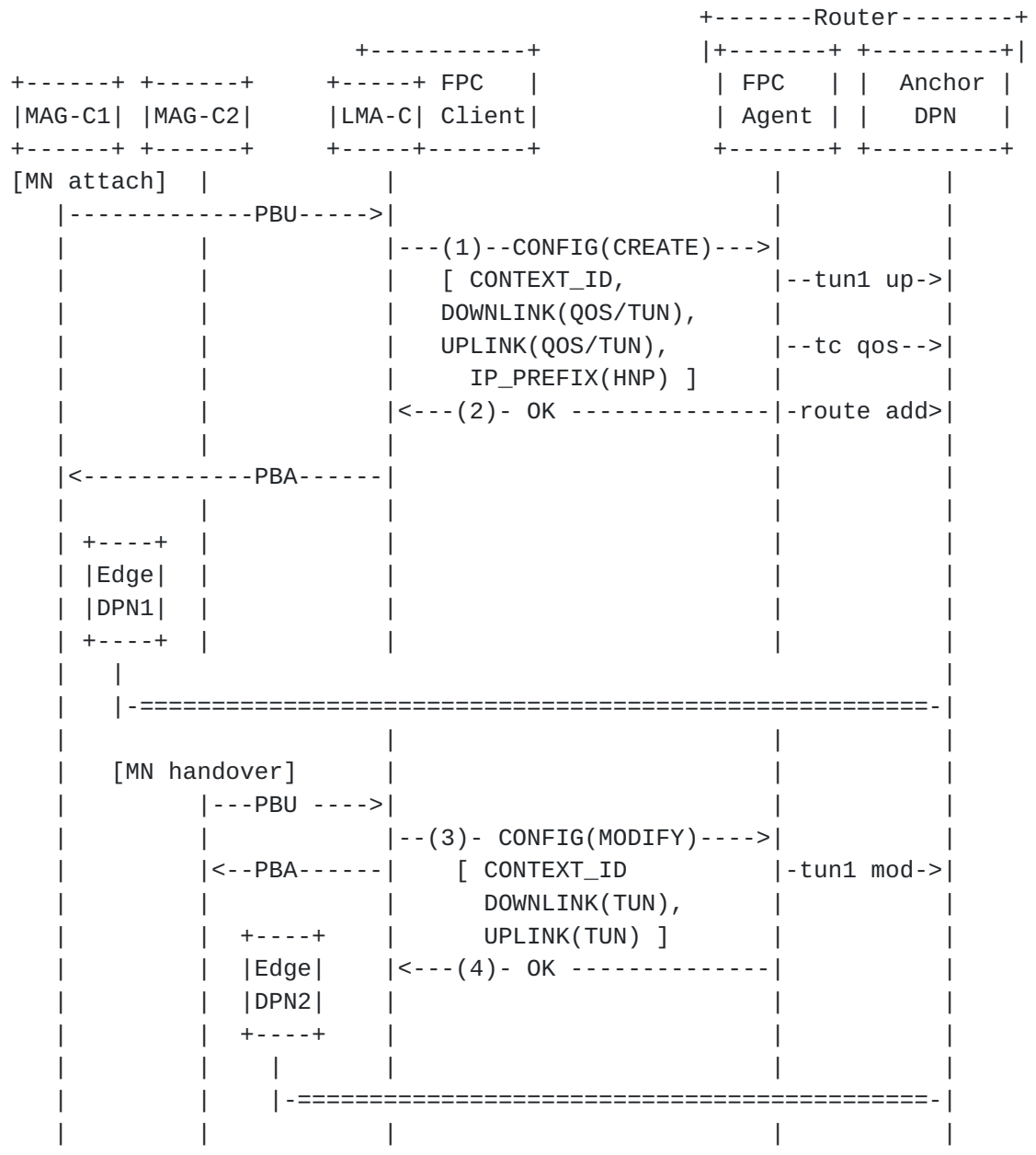

Figure 17: Exemplary Message Sequence (focus on FPC reference point)

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 CONFIGURE command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

The LMA-C adds properties during the creaton 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 requried. The LMA-C sends a CONFIGURE 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 CONFIGURE message, the Agent applies updated tunnel property to the local configuration and responds to the Client (4).

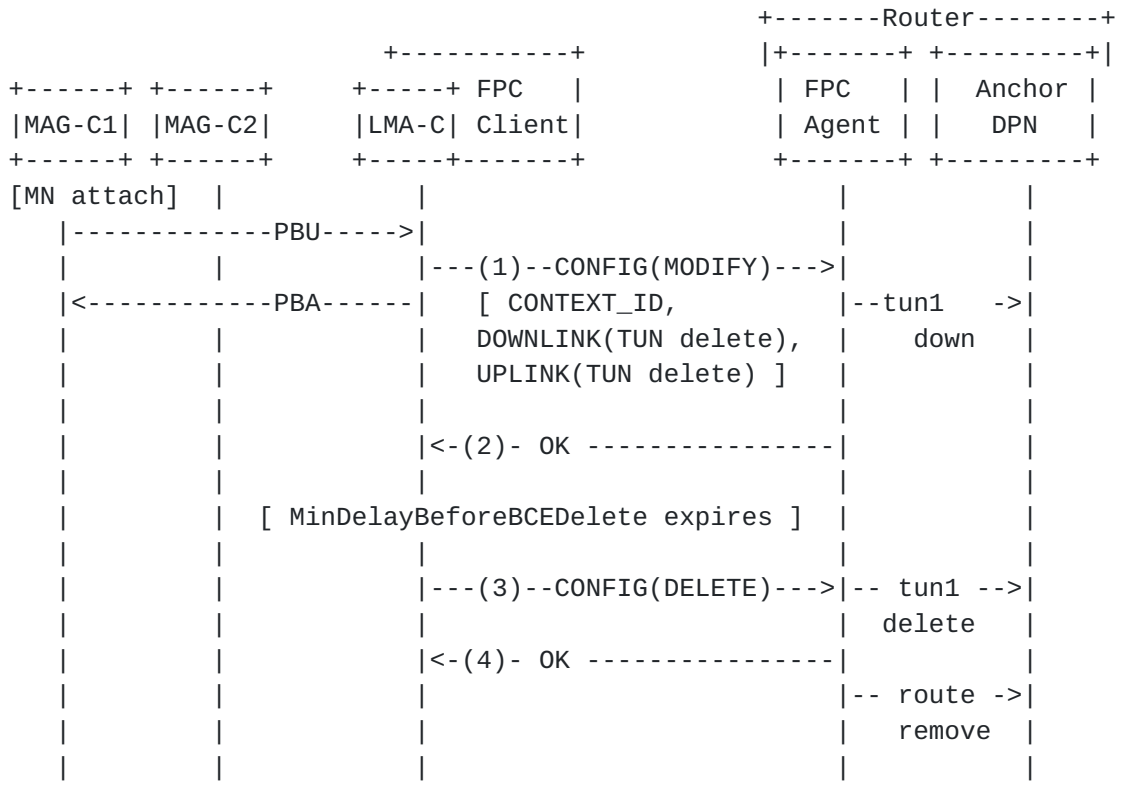

Figure 18: Exemplary Message Sequence (focus on FPC reference point)

When a teardown of the session occurs, MAG-C1 will send a PBU with a lifteime value of zero. The LMA-C sends a CONFIGURE message (1) to

the Agent to modify the existing tunnel property of the existing Context to delete the tunnel information.) Upon reception of the CONFIGRE 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 recieved after the expiration of the MinDelayBeforeBCEDelete timer (see [RFC5213]), the LMA-C will send a CONFIGURE (3) message with a deletion request for the Context. Upon reception of the message, the Agent deletes the tunnel and route on the DPN and responds to the Client (4).

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 Ports 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 19 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.

Figure 19: Structure of Policies and Ports

As depicted in Figure 19, the Port represents the anchor of Rules through the Policy-group, Policy, Rule heirarchy 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 Port.

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

Figure 20 illustrates the generic context configuration model as used between a FPC Client and a FPC Agent.

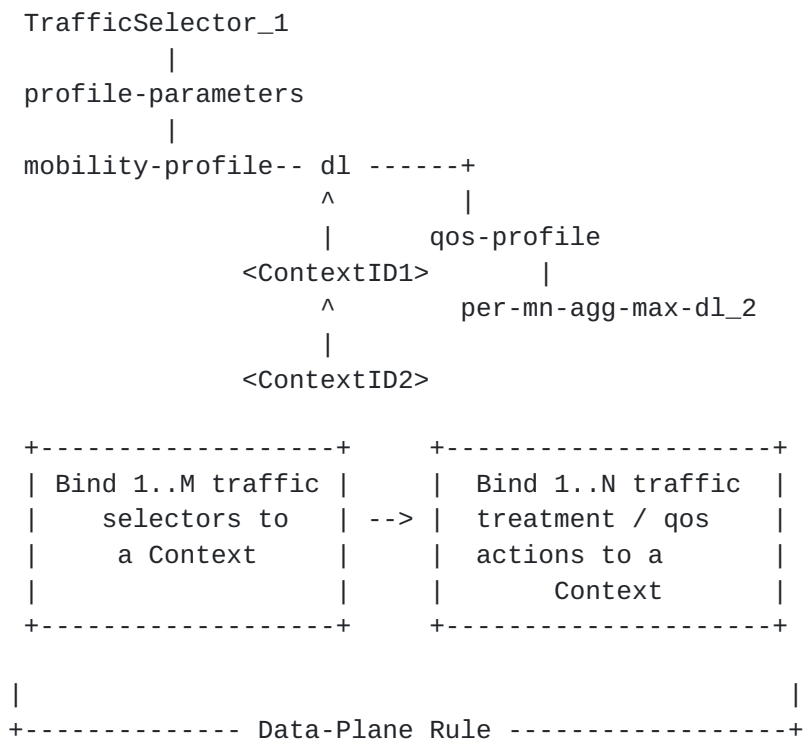

Figure 20: Structure of Contexts

As depicted in Figure 20, the Context represents a mobiility session heirarchy. A Client and Agent directly assigns values such as dowlink 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 techology 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 CONFIGURE message (1). Upon receiveing the

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

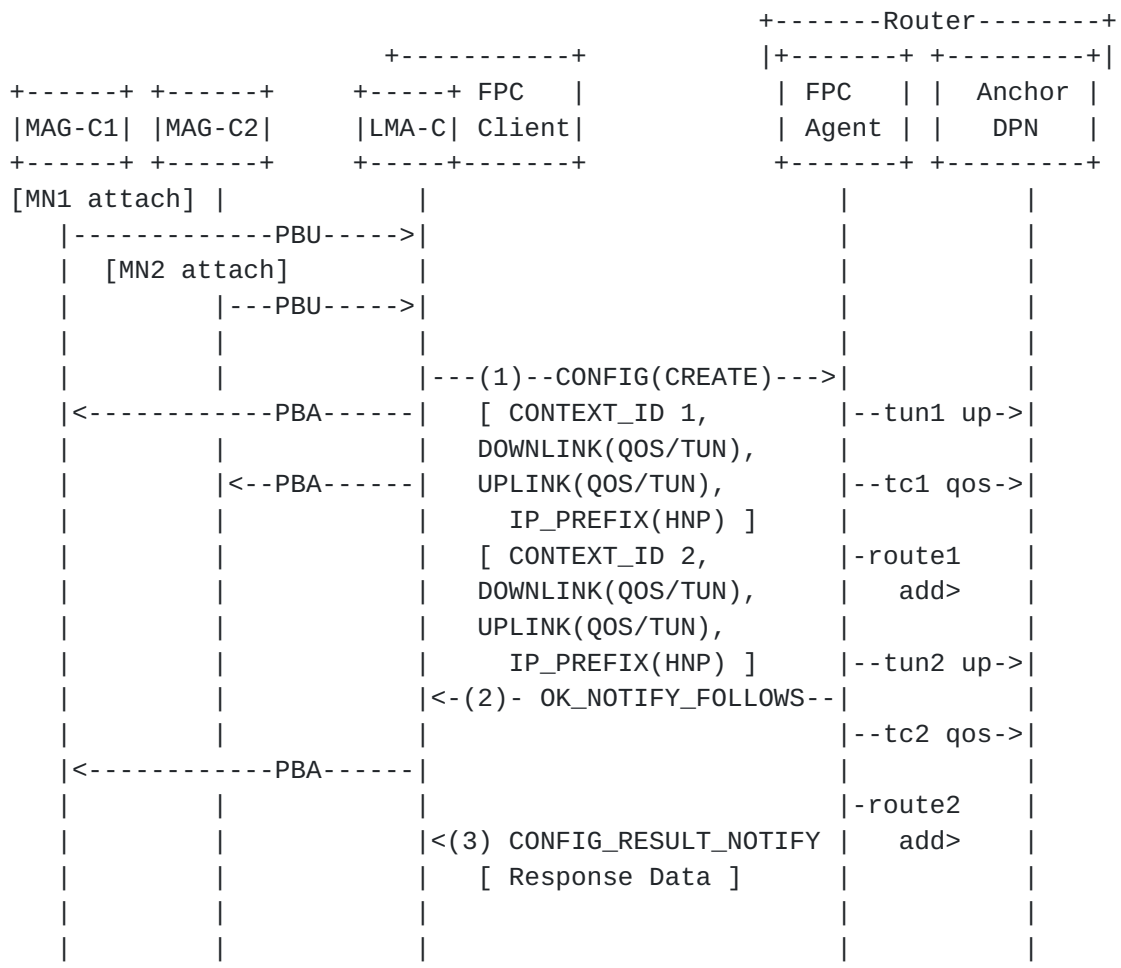

Figure 21: Exemplary Bulk Entity with Asynchronous Notification Sequence (focus on FPC reference point)

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 CONFIGURE_BUNDLES is preservation of dependency orders in a single message as opposed to sending multiple CONFIGURE messages and awaiting results from the Agent.

When a CONFIGURE_BUNDLES 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.

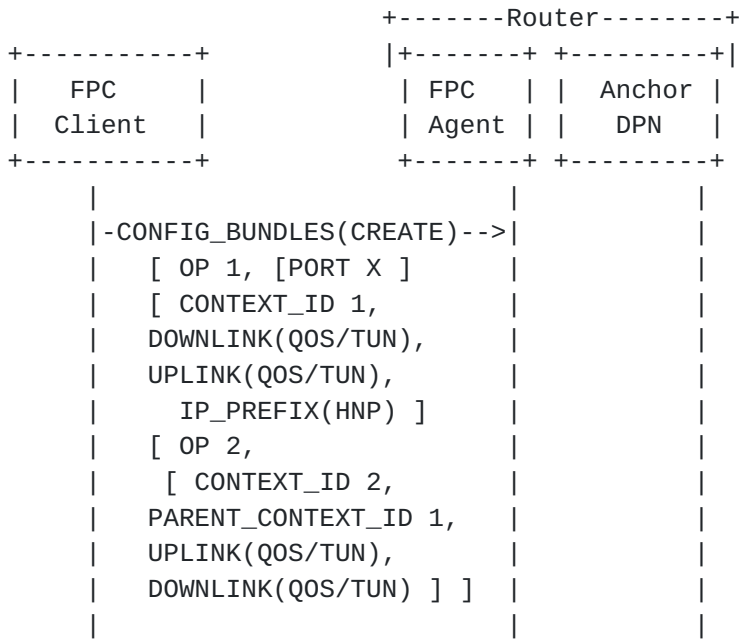

Figure 22: Exemplary Bundle Message (focus on FPC reference point)

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 then overrides the value. The avoid the accidental activation of the Contexts on the DPN, the CONFIGURE (1) message with the cloning instruction has a SESSION_STATE with a value of 'incomplete' and OP_TYPE of 'CREATE'. A second CONFIGURE (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.

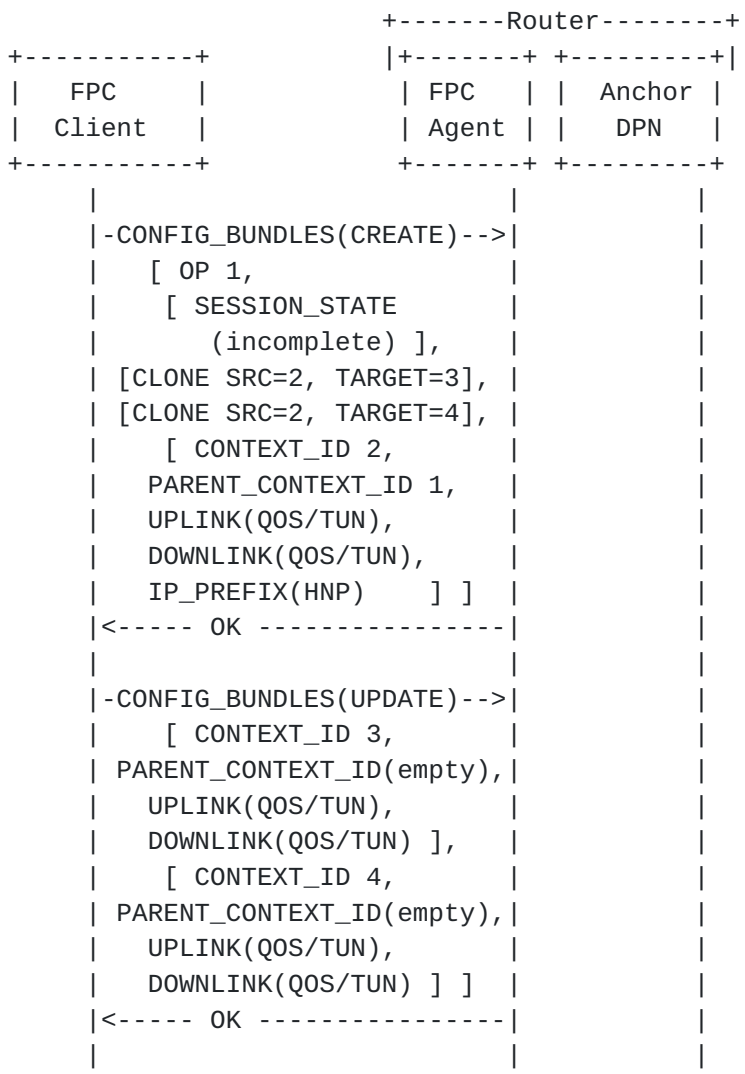

Figure 23: Exemplary Bundle Message (focus on FPC reference point)

Cloning has the added advantage of reducing the over the wire data size required to create multiple entities. This can improve performance if serializaiton / 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

- 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.

- 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 sitations can increase the time to error detection and overall system load withouth the Command Set present.

- Unambiguous provisioning specification. The Agent is exactily in sync with the expectations of the Client as opposed to guessing what DPN work could be done based upon data present at the Agent. This greatly increases the speed by which the Agent can complete work.

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

As Command Bitsets are technology specfic, 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 statless 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 Ports 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 CONFIG_BUNDLES 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 CONFIG_BUNDLES 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 CONFIG_BUNDLES 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 heirarchy 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 24 shows an example cache where each Cache where a containment heirarchy is provided for all caches.

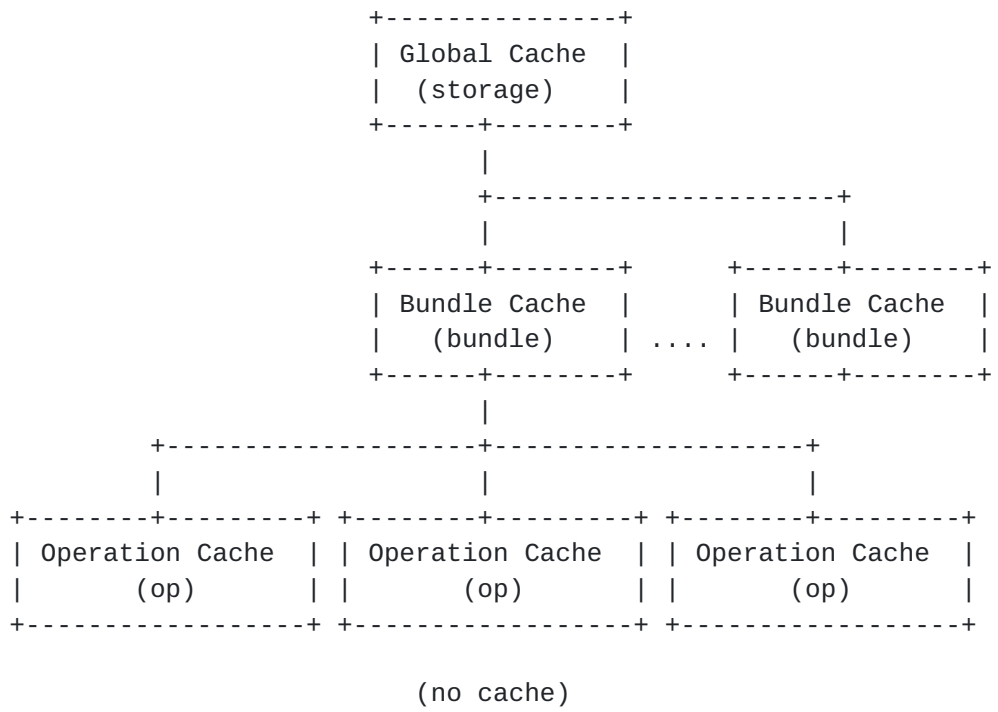

Figure 24: Exemplary Heirarchical Cache

5.2.4. Pre-provisioning

Although Contexts are used for Session based lifecycle elements, Ports may exist outside of a specific lifecycle and represent more general policies that may affect multiple Contexts (sessions). The use of pre-provisioning of Ports permits policy and administrative use cases to be exected. 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 Ports.

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

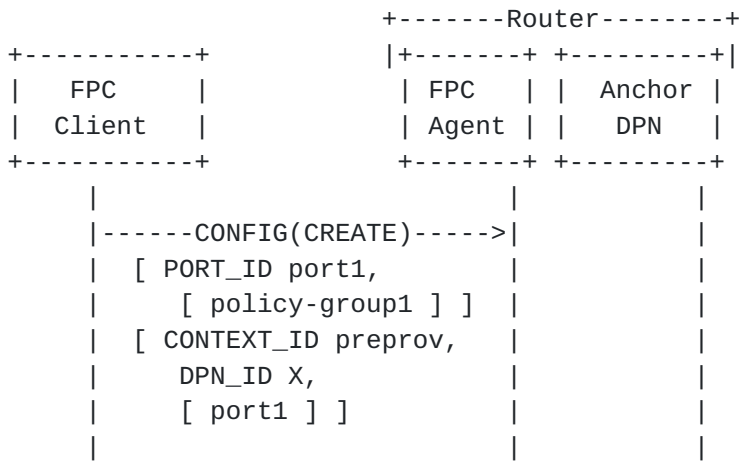

Figure 25: Exemplary Bundle Message (focus on FPC reference point)

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	*[DESCRIPTOR_ID DIRECTION]	DIRECTION [2, unsigned bits] is an ENUMERATION (uplink, downlink or both).
RULE_ACTIONS	*[ACTION_ID ORDER]	ORDER [8, unsigned integer] specifies action execution order.

Table 4: Rule Fields

6.1.2. Mobilty Structures

Field	Type
PORT_ID	FPC-Identity (Section 4.4)
POLICIES	*[POLICY_GROUP_ID]

Table 5: Port Fields

Field	Type
CONTEXT_ID	FPC-Identity (Section 4.4)
PORTS	*[PORT_ID]
DPN_GROUP_ID	FPC-Identity (Section 4.4)
DELEGATING 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 fields 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]

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:

- Events specified in the Event List of an EVENTS CONFIG
- LOW_THRESHOLD_CROSSED
- HIGH_THRESHOLD_CROSSED
- PERIODIC_REPORT
- SCHEDULED_REPORT
- PROBED
- 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 4.4)	All
DELAY	[32, unsigned integer]	All
OP_ID	[64, unsigned integer]	All
ADMIN_STATE	[8, admin state]	CONF, CONF_BUNDLES and REG_MONITOR
OP_TYPE	[8, op type]	CONF and CONF_BUNDLES

Table 12: Message Header Fields

6.2.2. CONF and CONF_BUNDLES 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 (1)	FPC Command Bitset. See Section 5.1.1.3.	C,U
CLONES (1)	*[FPC-Identity FPC-Identity] (Section 4.4)	C,U
PORTS	*[PORT]	C,U
CONTEXTS	*[CONTEXT [COMMAND_SET (1)]]	C,U
TARGETS	FPC-Identity (Section 4.4) *[DPN_ID]	Q,D

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

Table 13: CONF and CONF_BUNDLES OP_BODY Fields

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
PORTS	*[PORT]	C,U
CONTEXTS	*[CONTEXT [COMMAND_SET (1)]]	C,U
TARGETS	*[FPC-Identity (Section 4.4) *[DPN_ID]]	Q,D

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

Table 14: Immediate Response RESPONSE_BODY Fields

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.

Field	Type	Description
AGENT_ID	FPC-Identity (Section 4.4)	
NOTIFICATION_ID	[32, unsigned integer]	A Notification Identifier used to determine notification order.
TIMESTAMP	[32, unsigned integer]	The time that the notification occured.
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 MONITOR_ID TRIGGER [32, timestamp] [NOTIFICATION_DATA]	Timestamp notes when the event occurred. Notification Data is TRIGGER and Monitor type specific.

Table 16: Monitor Notifications

7. Derived and Subtyped Attributes

This section notes derived attributes.

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

Table 17: Descriptor Subtypes

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

Table 18: Action Subtypes

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

Table 19: Next Hop Subtypes

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

Table 20: QoS Subtypes

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

Table 21: Tunnel Subtypes

The following COMMAND_SET values are supported for IETF_PMIP.

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

- assign-ip - Assign the IP Address for the mobile session.
- assign-dpn - Assign the Dataplane Node.
- assign-fteid-ip - Assign the Fully Qualified TEID (F-TEID) LOCAL IP address.
- assign-fteid-teid - Assign the Fully Qualified TEID (F-TEID) LOCAL TEID.
- session - Assign values for the Session Level. When this involves 'assign-fteid-ip' and 'assign-fteid-teid' this implies the values are part of the default bearer.
- uplink - Command applies to uplink.
- downlink - Comman 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 optimiziations, 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 implementration improved throughput by an order of magnitude but was not useful based upon FPC's Version 03 design using either model. During this time the features of version 04 and its converged model became attractive and teh fpcagent project was

closed in August 2016. fpcagent will no longer be developed and will remain a proprietary implemenation.

The learnings of fpcagent has influenced the second project, fpc. Fpc is also an OpenDaylight project but is intended for open source release, if circumstances permit. It is also 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 developed by the FPC co-authors 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. As of this writing the project is 4 weeks old and currently supports CONFIGURE and CONFIGURE_BUNDLES based upon the effort of 3 part time engineers.

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

The addition of the DPN List in a Context Delete operation permits the Agent to avoid lookups of Context data in the Cache entirely. For 3GPP support, extra attributes are required in the Delete to avoid any cache lookup.

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 adapater 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 perforamnce 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 CONFIGURE_BUNDLES 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 Mac laptop sourced

in 2013. 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 after 3 weeks of development by two part time engineers:

1 proprietary DPN API

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

CONFIGURE and CONFIGURE_BUNDLES (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 CONFIGURE 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 CONFIGURE_BUNDLES 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 requried
  if an error occurs undo all assigments 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 nofication (CONFIG_RESULT_NOTIFY)
```

Figure 26: fpc pseudo code

As of this writing (Sept 2016) the implementation is 3 weeks old an considered pre-alpha. It is scheduled to be conformant to all FPC version 04 aspects within 60 days.

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.

10. IANA Considerations

This document provides a data model and protocol operation for DMM Forwarding Policy Configuration. YANG models are currently included in the Appendix and will be updated per the next revision of this document to specify the data model as well as to enable an implementation of the FPC protocol using RPC.

No actions from IANA are required. In case the semantics of this specification will be mapped to a particular wire protocol, authors of an associated separate document will approach IANA for the associated action to create a registry or add registry entries.

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

[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>.

[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-01 (work in progress), July 2016.

[I-D.ietf-dmm-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-00 (work in progress), August 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-16 (work in progress), August 2016.

[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>.

[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:

- ietf-dmm-fpcbase (fpcbase) - Defines the base model for model as defined in this document
- ietf-dmm-fpcagent (fpcagent) - Defines the FPC Agent entites and messages as defined in this document
- ietf-pmip-qos (pmip-qos) - Defines proxy mobile IPv6 QoS parameters per RFC 7222
- ietf-traffic-selectors-types (traffic-selectors) - Defines Traffic Selectors per RFC 6088
- ietf-dmm-threegpp - Defines the base structures for 3GPP based IP mobility and augments fpcagent to support these parameters.
- 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.
- ietf-dmm-fpc-policyext - defines basic policy extensions, e.g. Actions and Descriptors, to fpcbase and as defined in this document.

A.1. YANG Models

A.1.1. FPC Base YANG Model

```
module ietf-dmm-fpcbase {
    namespace "urn:ietf:params:xml:ns:yang:fpcbase";
    prefix fpcbase;

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

    organization "IETF DMM Working Group";
    contact "Satoru Matsushima <satoru.matsushima@g.softbank.co.jp>";

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

    revision 2016-08-03 {
        description "Changes based on -04 version of FPC draft.";
        reference "draft-ietf-dmm-fpc-cpdp-04";
    }
```

```
    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;
        }
    }

    grouping target-value {
      leaf target {
          type fpc-identity;
      }
    }

    grouping targets-value {
      list targets {
          key "target";
          leaf target {
            type fpc-identity;
          }
          leaf dpn-id {
                type fpcbase:fpc-dpn-id;
          }
      }
    }

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

```
    grouping fpc-descriptor {
        uses fpcbase:fpc-descriptor-id;
        leaf descriptor-type {
          mandatory true;
          type identityref {
            base "fpc-descriptor-type";
          }
          description "Descriptor Type";
        }
        choice descriptor-value {
          case all-traffic {
            leaf all-traffic {
              type empty;
            }
          }
        }
    }

    // Action Structure
    typedef fpc-action-id-type {
        type fpcbase:fpc-identity;
        description "Action-ID";
    }
    identity fpc-action-type {
        description "Action Type";
    }
    grouping fpc-action-id {
      leaf action-id {
        type fpcbase:fpc-action-id-type;
      }
    }
    grouping fpc-action {
        uses fpcbase:fpc-action-id;
        leaf action-type {
          mandatory true;
          type identityref {
            base "fpc-action-type";
          }
          description "Action Type";
        }
        choice action-value {
          case drop {
            leaf drop {
              type empty;
            }
          }
        }
    }
```

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

    // Policy Structures
    typedef fpc-policy-id {
        type fpcbase:fpc-identity;
    }
    grouping fpc-policy {
        leaf policy-id {
            type fpcbase:fpc-policy-id;
        }
        list rules {
            key order;
            leaf order {
              type uint32;
            }
            uses fpcbase:fpc-rule;
        }
    }

    // Policy Group
    typedef fpc-policy-group-id {
        type fpcbase:fpc-identity;
    }
    grouping fpc-policy-group {
      leaf policy-group-id {
        type fpcbase:fpc-policy-group-id;
      }
      leaf-list policies {
        type fpcbase:fpc-policy-id;
```

```
      }
    }

    // Mobility Structures
    // Port Group
    typedef fpc-port-id {
        type fpcbase:fpc-identity;
    }
    grouping fpc-port {
        leaf port-id {
            type fpcbase:fpc-port-id;
        }
        leaf-list policy-groups {
            type fpcbase:fpc-policy-group-id;
        }
    }

    // Context Group
    typedef fpc-context-id {
        type fpcbase:fpc-identity;
    }
    grouping fpc-context-profile {
        description "A profile that applies to a specific direction";
        leaf tunnel-local-address {
            type inet:ip-address;
            description "Uplink endpoint address of the DPN which agent
exists.";
        }
        leaf tunnel-remote-address {
            type inet:ip-address;
            description "Uplink endpoint address of the DPN which agent
exists.";
        }
        leaf tunnel-mtu-size {
            type uint32;
            description "Tunnel MTU size";
        }
        container mobility-tunnel-parameters {
            description "Specifies profile specific uplink 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.";
            uses fpcbase:mobility-info;
        }
        container nexthop {
            uses fpcbase:fpc-nexthop;
        }
        container qos-profile-parameters {
```

```
            uses fpcbase:fpc-qos-profile;
        }
        container dpn-parameters {
        }
        list vendor-parameters {
```

```
            key "vendor-id vendor-type";
            uses fpcbase:vendor-attributes;
        }
    }

    typedef fpc-direction {
       type enumeration {
         enum uplink;
         enum downlink;
       }
    }

    grouping fpc-context {
        leaf context-id {
            type fpcbase:fpc-context-id;
        }
        leaf-list ports {
            type fpcbase:fpc-port-id;
        }
        leaf dpn-group {
          type fpcbase:fpc-dpn-group-id;
        }
        leaf-list delegating-ip-prefixes {
            type inet:ip-prefix;
        }
        container ul {
            if-feature fpcbase:fpc-basic-agent;
            uses fpcbase:fpc-context-profile;
        }
        container dl {
            if-feature fpcbase:fpc-basic-agent;
            uses fpcbase:fpc-context-profile;
        }
        list dpns {
            if-feature fpcbase:fpc-multi-dpn;
            key "dpn-id direction";
            leaf dpn-id {
                type fpcbase:fpc-dpn-id;
            }
            leaf direction {
                mandatory true;
                type fpcbase:fpc-direction;
            }
            uses fpcbase:fpc-context-profile;
        }
        leaf parent-context {
            type fpcbase:fpc-context-id;
        }
```

```
    }

    // Mobility (Tunnel) Information
    grouping mobility-info {
        choice profile-parameters {
            case nothing {
              leaf none {
                type empty;
              }
            }
        }
    }

    // Next Hop Structures
    typedef fpcp-service-path-id {
        type uint32 {
            range "0..33554431";
        }
        description "SERVICE_PATH_ID";
    }

    identity fpc-nexthop-type {
        description "Next Hop Type";
    }
    identity fpc-nexthop-ip {
        base "fpcbase:fpc-nexthop-type";
    }
    identity fpc-nexthop-servicepath {
        base "fpcbase:fpc-nexthop-type";
    }
    grouping fpc-nexthop {
        leaf nexthop-type {
            type identityref {
              base "fpcbase:fpc-nexthop-type";
            }
        }
        choice nexthop-value {
            case ip {
                leaf ip {
                  type inet:ip-address;
                }
            }
            case servicepath {
                leaf servicepath {
                    type fpcbase:fpcp-service-path-id;
                }
            }
        }
```

```
    }

    // 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 fpcbase:fpc-qos-type;
            }
            description "the profile type";
        }
        choice value {
        }
    }

    // Vendor Specific Attributes
    identity vendor-specific-type {
        description "Vendor Specific Attribute Type";
    }
    grouping vendor-attributes {
        leaf vendor-id {
            type fpcbase:fpc-identity;
        }
        leaf vendor-type {
            type identityref {
                base "fpcbase:vendor-specific-type";
            }
        }
        choice value {
            case empty-type {
                leaf empty-type {
                    type empty;
                }
            }
        }
    }

    // Topology
    typedef fpc-domain-id {
        type fpcbase:fpc-identity;
    }
    grouping fpc-domain {
      leaf domain-id {
        type fpcbase:fpc-domain-id;
      }
```

```
      leaf domain-name {
```

```
        type string;
      }
      leaf domain-type {
        type string;
      }
    }


    typedef fpc-dpn-id {
        type fpcbase:fpc-identity;
        description "DPN Identifier";
    }
    identity fpc-dpn-control-protocol {
        description "DPN Control Protocol";
    }
    grouping fpc-dpn {
        leaf dpn-id {
          type fpcbase:fpc-dpn-id;
        }
        leaf dpn-name {
          type string;
        }
        leaf-list dpn-groups {
          type fpcbase:fpc-dpn-group-id;
        }
        leaf node-reference {
          type instance-identifier;
        }
    }

    typedef fpc-dpn-group-id {
        type fpcbase:fpc-identity;
        description "DPN Group Identifier";
    }
    identity fpc-forwaridingplane-role {
        description "Role of DPN Group in the Forwarding Plane";
    }
    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 fpcbase:fpc-dpn-group-id;
        }
```

```
        leaf remote-mobility-profile {
            type identityref {
                base "fpcbase:fpc-mobility-profile-type";
            }
        }
        leaf remote-data-plane-role {
            type identityref {
                base "fpcbase:fpc-forwaridingplane-role";
            }
        }
        leaf remote-endpoint-address {
            type inet:ip-address;
        }
        leaf local-endpoint-address {
            type inet:ip-address;
        }
        leaf tunnel-mtu-size {
            type uint32;
        }
    }

    // Events, Probes & Notifications
    identity event-type {
        description "Base Event Type";
    }

    typedef event-type-id {
        type uint32;
    }

    grouping monitor-id {
      leaf monitor-id {
        type fpcbase:fpc-identity;
      }
    }

    identity report-type {
      description "Type of Report";
    }
    identity periodic-report {
      base "fpcbase:report-type";
    }
    identity threshold-report {
      base "fpcbase:report-type";
    }
    identity scheduled-report {
      base "fpcbase:report-type";
    }
```

```
    identity events-report {
      base "fpcbase:report-type";
    }

    grouping report-config {
      choice event-config-value {
        case periodic-config {
            leaf period {
              type uint32;
            }
        }
        case threshold-config {
            leaf lo-thresh {
              type uint32;
            }
            leaf hi-thresh {
              type uint32;
            }
        }
        case scheduled-config {
            leaf report-time {
              type uint32;
            }
        }
        case events-config-ident {
            leaf-list event-identities {
              type identityref {
                base "fpcbase:event-type";
              }
            }
        }
        case events-config {
            leaf-list event-ids {
              type uint32;
            }
        }
      }
    }

    grouping monitor-config {
      uses fpcbase:monitor-id;
      uses fpcbase:target-value;
      uses fpcbase:report-config;
    }

    grouping report {
      uses fpcbase:monitor-config;
      choice report-value {
```

```
        leaf trigger {
          type fpcbase:event-type-id;
        }
        case simple-empty {
          leaf nothing {
            type empty;
          }
        }
        case simple-val32 {
          leaf val32 {
            type uint32;
          }
        }
      }
    }
}
```

Figure 27: FPC YANG base

A.1.2. FPC Agent YANG Model

```
module ietf-dmm-fpcagent {
    namespace "urn:ietf:params:xml:ns:yang:fpcagent";
    prefix fpcagent;

    import ietf-dmm-fpcbase { prefix fpcbase; revision-date 2016-08-03; }
    import ietf-inet-types { prefix inet; revision-date 2013-07-15; }

    organization "IETF DMM Working Group";
    contact "Satoru Matsushima <satoru.matsushima@g.softbank.co.jp>";

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

    revision 2016-08-03 {
        description "Changes based on -04 version of FPC draft.";
        reference "draft-ietf-dmm-fpc-cpdp-04";
    }
    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.";
    }

    typedef agent-identifier {
        type fpcbase:fpc-identity;
    }

    typedef client-identifier {
        type fpcbase:fpc-identity;
    }

    grouping basename-info {
          leaf basename {
            if-feature fpcagent:fpc-basename-registry;
            description "Rules Basename";
            type fpcbase:fpc-identity;
          }
          leaf base-state {
            if-feature fpcagent:fpc-basename-registry;
            type string;
          }
          leaf base-checkpoint {
            if-feature fpcagent:fpc-basename-registry;
            type string;
          }
    }

    // Top Level Structures
    container tenants {
        description "";

        list tenant {
            description "";
            key "tenant-id";
```

```
            leaf tenant-id {
                type fpcbase:fpc-identity;
```

```
            }

            container fpc-policy {
              list policy-groups {
                  key "policy-group-id";
                  uses fpcbase:fpc-policy-group;
              }
              list policies {
                  key "policy-id";
                  uses fpcbase:fpc-policy;
              }
              list descriptors {
                key descriptor-id;
                uses fpcbase:fpc-descriptor;
              }
              list actions {
                  key action-id;
                  uses fpcbase:fpc-action;
              }
            }

            container fpc-mobility {
              config false;
              list contexts {
                  key context-id;
                  uses fpcbase:fpc-context;
              }
              list ports {
                  key port-id;
                  uses fpcbase:fpc-port;
              }
              list monitors {
                  uses fpcbase:monitor-config;
              }
            }
            container fpc-topology {
              // Basic Agent Topology Structures
              list domains {
                key domain-id;
                uses fpcbase:fpc-domain;
                uses fpcagent:basename-info;
              }

              list dpn-group-peers {
                  if-feature fpcbase:fpc-basic-agent;
                  key "remote-dpn-group-id";
                  uses fpcbase:fpc-dpn-peer-group;
              }
```

```
              leaf dpn-id {
                if-feature fpcbase:fpc-basic-agent;
                type fpcbase:fpc-dpn-id;
              }
              leaf-list control-protocols {
                if-feature fpcbase:fpc-basic-agent;
                type identityref {
                  base "fpcbase:fpc-dpn-control-protocol";
                }
              }

              list dpn-groups {
                  if-feature fpcbase:fpc-multi-dpn;
                  key dpn-group-id;
                  uses fpcagent:fpc-dpn-group;
                  list domains {
                    key domain-id;
                    uses fpcbase:fpc-domain;
                    uses fpcagent:basename-info;
                  }
              }
              list dpns {
                  if-feature fpcbase:fpc-multi-dpn;
                  key dpn-id;
                  uses fpcbase:fpc-dpn;
              }
            }

        }
    }

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

      // Common Agent Info
      list supported-events {
        key event;
        leaf event {
          type identityref {
            base "fpcbase:event-type";
          }
        }
        leaf event-id {
          type fpcbase:event-type-id;
        }
```

```
      }

      list supported-error-types {
        key error-type;
        leaf error-type {
          type identityref {
            base "fpcagent:error-type";
          }
        }
        leaf error-type-id {
          type fpcagent:error-type-id;
        }
      }
    }


    // Multi-DPN Agent Structures
    grouping fpc-dpn-group {
        leaf dpn-group-id {
            type fpcbase:fpc-dpn-group-id;
        }
        leaf data-plane-role {
            type identityref {
                base "fpcbase:fpc-forwaridingplane-role";
            }
        }
        leaf access-type {
            type identityref {
                base "fpcbase:fpc-access-type";
            }
        }
        leaf mobility-profile {
            type identityref {
                base "fpcbase:fpc-mobility-profile-type";
            }
        }
        list dpn-group-peers {
            key "remote-dpn-group-id";
            uses fpcbase:fpc-dpn-peer-group;
        }
    }


    // RPC
    // RPC Specific Structures
    //Input Structures
    typedef admin-status {
        type enumeration {
```

```
            enum enabled { value 0; }
            enum disabled { value 1; }
            enum virtual { value 2; }
        }
    }

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

    typedef op-delay {
        type uint32;
    }

    typedef op-identifier {
        type uint64;
    }

    typedef ref-scope {
      description "Search scope for references in the operation.
        op - All references are contained in the operation body (intra-op)
        bundle - All references in exist in bundle (inter-operation/intra-
bundle).
          NOTE - If this value comes in CONFIG call it is equivalen to 'op'.
        storage - One or more references exist outside of the operation and
bundle.
          A lookup to a cache / storage is required.
        unknown - the location of the references are unknown.  This is treated
as
          a 'storage' type.";
      type enumeration {
        enum none { value 0; }
        enum op { value 1; }
        enum bundle { value 2; }
        enum storage { value 3; }
        enum unknown { value 4; }
      }
    }

    grouping instructions {
      container instructions {
        if-feature instruction-bitset;
        choice instr-type {
        }
```

```
      }
    }
```

```
    grouping op-header {
      leaf client-id {
        type fpcagent:client-identifier;
      }
      leaf delay {
        type op-delay;
      }
      leaf session-state {
        type session-status;
      }
      leaf admin-state {
        type admin-status;
      }
      leaf op-type {
        type enumeration {
          enum create { value 0; }
          enum update { value 1; }
          enum query { value 2; }
          enum delete { value 3; }
        }
      }
      leaf op-ref-scope {
          if-feature operation-ref-scope;
          type fpcagent:ref-scope;
      }
      uses fpcagent:instructions;
    }

    grouping clone-ref {
      leaf entity {
        type fpcbase:fpc-identity;
      }
      leaf source {
        type fpcbase:fpc-identity;
      }
    }

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

    grouping context-operation {
      uses fpcbase:fpc-context;
      uses fpcagent:instructions;
    }

    grouping port-operation {
      uses fpcbase:fpc-port;
```

```
      uses fpcagent:instructions;
    }

    // Output Structure
    grouping payload {
      list ports {
        uses fpcagent:port-operation;
      }
      list contexts {
        uses fpcagent:context-operation;
      }
    }

    grouping op-input {
      uses fpcagent:op-header;
      leaf op-id {
        type op-identifier;
      }
      choice op_body {
        case create_or_update {
          list clones {
            if-feature fpc-cloning;
            key entity;
            uses fpcagent:clone-ref;
          }
          uses fpcagent:payload;
        }
        case delete_or_query {
          uses fpcbase:targets-value;
        }
      }
    }

    typedef result {
      type enumeration {
        enum ok { value 0; }
        enum err { value 1; }
        enum ok-notify-follows { value 2; }
      }
    }

    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 {
      description "Integer form of the Error Type";
      type uint32;
    }

    grouping op-status-value {
      leaf op-status {
        type enumeration {
          enum ok { value 0; }
          enum err { value 1; }
        }
      }
    }

    grouping result-body {
      leaf op-id {
        type op-identifier;
      }
      choice result-type {
        case err {
          leaf error-type-id {
            type fpcagent:error-type-id;
          }
          leaf error-info {
            type string {
              length "1..1024";
            }
          }
        }
        case create-or-update-success {
          uses fpcagent:payload;
        }
        case delete_or_query-success {
          uses fpcbase:targets-value;
        }
        case empty-case {
        }
      }
    }

    // Common RPCs
    rpc configure {
      input {
        uses fpcagent:op-input;
      }
      output {
        leaf result {
          type result;
```

```
        }
        uses fpcagent:result-body;
      }
    }

    rpc configure-bundles {
      if-feature fpcagent:fpc-bundles;
      input {
        leaf highest-op-ref-scope {
            if-feature operation-ref-scope;
            type fpcagent:ref-scope;
        }
        list bundles {
          key op-id;
          uses fpcagent:op-input;
        }
      }
      output {
        list bundles {
          key op-id;
          uses fpcagent:result-body;
        }
      }
    }

    rpc bind-dpn {
      if-feature fpcagent:fpc-client-binding;
      input {
        leaf node-id {
          type inet:uri;
        }
        uses fpcbase:fpc-dpn;
      }
      output {
        uses fpcagent:result-body;
      }
    }
    rpc unbind-dpn {
      if-feature fpcagent:fpc-client-binding;
      input {
        leaf dpn-id {
          type fpcbase:fpc-dpn-id;
        }
      }
      output {
        uses fpcagent:result-body;
      }
    }
```

```
    // Notification Messages & Structures
    typedef notification-id {
      type uint32;
    }

    grouping notification-header {
      leaf notification-id {
          type fpcagent:notification-id;
      }
      leaf timestamp {
          type uint32;
      }
    }

    notification config-result-notification {
      uses fpcagent:notification-header;
      choice value {
        case config-result {
          uses fpcagent:op-status-value;
          uses fpcagent:result-body;
        }
        case config-bundle-result {
          list bundles {
            uses fpcagent:op-status-value;
            uses fpcagent:result-body;
          }
        }
      }
    }

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

    rpc event_deregister {
        description "Used to de-register monitoring of parameters/events";
        input {
          list monitors {
            uses fpcbase:monitor-id;
          }
```

```
        }
        output {
          leaf monitor-result {
            type fpcagent:result;
          }
        }
    }

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

    notification notify {
        uses fpcagent:notification-header;
        choice value {
            case dpn-candidate-available {
              if-feature fpcagent:fpc-auto-binding;
              leaf node-id {
                type inet:uri;
              }
              leaf-list access-types {
                type identityref {
                  base "fpcbase:fpc-access-type";
                }
              }
              leaf-list mobility-profiles {
                type identityref {
                  base "fpcbase:fpc-mobility-profile-type";
                }
              }
              leaf-list forwarding-plane-roles {
                type identityref {
                  base "fpcbase:fpc-forwaridingplane-role";
                }
              }
            }
            case monitor-notification {
              choice monitor-notification-value {
                case simple-monitor {
                  uses fpcbase:report;
```

```
                }
                case bulk-monitors {
                  list reports {
                    uses fpcbase:report;
                  }
                }
              }
            }
        }
    }
}
```

Figure 28: FPC YANG agent

A.1.3. PMIP QoS Model

```
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 DMM (Dynamic Mobility Management) Working Group";

    contact
      "WG Web:   <https://datatracker.ietf.org/wg/dmm/>
     WG List:  <mailto:dmm@ietf.org>

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

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

     Editor:
               <mailto:>";

    description
      "This module contains a collection of YANG definitions for
```

```
     quality of service paramaters used in Proxy Mobile IPv6.

     Copyright (c) 2015 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module was created as part of the IETF
        DMM FPC YANG modules; see the RFC itself for full legal notices.";

    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; }
                enum ALLOCATE { value 1; }
                enum DE-ALLOCATE { value 2; }
                enum MODIFY { value 3; }
                enum QUERY { value 4; }
                enum NEGOTIATE { value 5; }
        }
        description
                "1-octet Operational code indicates the type of QoS request.

              RESPONSE:   (0)
                 Response to a QoS request

              ALLOCATE:   (1)
                 Request to allocate QoS resources

              DE-ALLOCATE:   (2)
                 Request to de-Allocate QoS resources

              MODIFY:   (3)
                 Request to modify QoS parameters for a previously negotiated
                 QoS Service Request

              QUERY:   (4)
                 Query to list the previously negotiated QoS Service Requests
                 that are still active

              NEGOTIATE:   (5)
                Response to a QoS Service Request with a counter QoS proposal

              Reserved:   (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; }
```

```
            enum Per-MN-Agg-Max-DL-Bit-Rate { value 1; }
            enum Per-MN-Agg-Max-UL-Bit-Rate { value 2; }
            enum Per-Session-Agg-Max-DL-Bit-Rate { value 3; }
            enum Per-Session-Agg-Max-UL-Bit-Rate { value 4; }
      enum Allocation-Retention-Priority { value 5; }
            enum Aggregate-Max-DL-Bit-Rate { value 6; }
            enum Aggregate-Max-UL-Bit-Rate { value 7; }
            enum Guaranteed-DL-Bit-Rate { value 8; }
            enum Guaranteed-UL-Bit-Rate { value 9; }
            enum QoS-Traffic-Selector { value 10; }
            enum QoS-Vendor-Specific-Attribute { value 11; }
    }
            description
            "8-bit unsigned integer indicating the type of the QoS
          attribute.  This specification reserves the following values.
          (0) -  Reserved
             This value is reserved and cannot be used

          (1) -  Per-MN-Agg-Max-DL-Bit-Rate
             Per-Mobile-Node Aggregate Maximum Downlink Bit Rate.

          (2) -  Per-MN-Agg-Max-UL-Bit-Rate
             Per-Mobile-Node Aggregate Maximum Uplink Bit Rate.

          (3) -  Per-Session-Agg-Max-DL-Bit-Rate
             Per-Mobility-Session Aggregate Maximum Downlink Bit Rate.

                (4) -  Per-Session-Agg-Max-UL-Bit-Rate
          Per-Mobility-Session Aggregate Maximum Uplink Bit Rate.

          (5) -  Allocation-Retention-Priority
             Allocation and Retention Priority.

          (6) -  Aggregate-Max-DL-Bit-Rate
             Aggregate Maximum Downlink Bit Rate.

          (7) -  Aggregate-Max-UL-Bit-Rate
             Aggregate Maximum Uplink Bit Rate.

          (8) -  Guaranteed-DL-Bit-Rate
             Guaranteed Downlink Bit Rate.

          (9) -  Guaranteed-UL-Bit-Rate
             Guaranteed Uplink Bit Rate.

          (10) -  QoS-Traffic-Selector
             QoS Traffic Selector.
```

```
              (11) -  QoS-Vendor-Specific-Attribute
                QoS Vendor-Specific Attribute.

              (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.";
                }
        }

        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; }
                                enum disabled { value 1; }
                                enum reserved1 { value 2; }
                                enum reserved2 { value 3; }
                        }
                        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; }
                                enum disabled { value 1; }
                                enum reserved1 { value 2; }
                                enum reserved2 { value 3; }
                        }
                        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.";
                }
        }

        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
[SMI].";
                        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;
                }
            }
            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;
                }
            }
            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;
                }
            }
            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;
                }
            }
            case allocation-retention-priority-case {
                when "../attributetype = 'Allocation-Retention-Priority-type'";
                uses qos-pmip:Allocation-Retention-Priority-Value;
            }
```

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

        grouping qosoption {
        leaf srid {
            type sr-id;
            mandatory true;
        }
        leaf trafficclass {
            type traffic-class;
            mandatory true;
        }
        leaf operationcode {
            type operational-code;
            mandatory true;
        }
        list attributes {
            unique "attributetype";
            uses qosattribute;
            min-elements 1;
        }
```

```
        }
}
```

Figure 29: FPC YANG PMIP QoS

A.1.4. Traffic Selectors YANG Model

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

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

    prefix "ietf-traffic-selectors";

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

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

    contact
      "WG Web:   <https://datatracker.ietf.org/wg/dmm/>
     WG List:  <mailto:dmm@ietf.org>

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

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

     Editor:
               <mailto:>";

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

     Copyright (c) 2015 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
```

```
     (http://trustee.ietf.org/license-info).

     This version of this YANG module was created as part of the IETF
        DMM FPC YANG modules; see the RFC itself for full legal notices.";

    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
services 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.";
                }
          }
        }

        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.";
                }
          }
```

```
        }

        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;
                        }
                        uses traffic-selector;
                }
        }
}
```

Figure 30: FPC YANG Traffic Selectors

A.1.5. FPC 3GPP Mobility YANG Model

```
module ietf-dmm-threegpp {
    namespace "urn:ietf:params:xml:ns:yang:threegpp";
```

```
    prefix threegpp;

    import ietf-inet-types { prefix inet; revision-date 2013-07-15; }
    import ietf-dmm-fpcagent { prefix fpcagent; }
    import ietf-dmm-fpcbase { prefix fpcbase; revision-date 2016-08-03; }
    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 DMM Working Group";
    contact "Satoru Matsushima <satoru.matsushima@g.softbank.co.jp>";

    description
    "This module contains YANG definition for
     3GPP Related Mobility Structures";

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

    identity threeGPP-access-type {
      base "fpcbase:fpc-access-type";
    }

    // Profile Type
    identity threeGPP-mobility {
         base "fpcbase:fpc-mobility-profile-type";
    }

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

    identity gtpv1 {
        base "threegpp:threeGPP-tunnel-type";
    }

    identity gtpv2 {
        base "threegpp:threeGPP-tunnel-type";
    }

    grouping teid-value {
         leaf tunnel-identifier {
            description "TEID";
            type uint32;
        }
    }

    grouping threeGPP-tunnel {
        leaf tunnel-type {
            type identityref  {
              base "threegpp:threeGPP-tunnel-type";
```

```
            }
        }
        uses threegpp:teid-value;
    }

    // QoS Profile
    identity threeGPP-qos-profile-parameters {
        base "fpcbase:fpc-qos-type";
    }

    typedef fpc-qos-class-identifier {
        type uint8 {
            range "1..9";
        }
        description "QCI";
    }

    grouping threeGPP-QoS {
        leaf qci {
            type fpc-qos-class-identifier;
        }
        leaf gbr {
            type uint32;
        }
        leaf mbr {
            type uint32;
        }
        leaf apn-ambr {
            type uint32;
        }
        leaf ue-ambr {
            type uint32;
        }
        container arp {
            uses pmipqos:Allocation-Retention-Priority-Value;
        }
    }

    typedef ebi-type {
      type uint8 {
        range "0..15";
      }
    }

   // From 3GPP TS 24.008 version 13.5.0 Release 13
   typedef component-type-enum {
        type enumeration {
            enum ipv4RemoteAddress { value 16; }
```

```
            enum ipv4LocalAddress  { value 17; }
            enum ipv6RemoteAddress { value 32; }
            enum ipv6RemoteAddressPrefix { value 33; }
            enum ipv6LocalAddressPrefix { value 35; }
            enum protocolNextHeader { value 48; }
            enum localPort { value 64; }
            enum localPortRange { value 65; }
            enum reomotePort { value 80; }
            enum remotePortRange { value 81; }
            enum secParamIndex { value 96; }
            enum tosTraffClass { value 112; }
            enum flowLabel { value 128; }
        }
    }

    typedef packet-filter-direction {
        type enumeration {
          enum preRel7Tft { value 0; }
          enum uplink { value 1; }
          enum downlink { value 2; }
          enum bidirectional { value 3; }
        }
    }

    typedef component-type-id {
        type uint8 {
          range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 | 80 | 81 | 96 | 112 |
128";
        }
    }

    grouping packet-filter {
      leaf direction {
          type threegpp:packet-filter-direction;
      }
      leaf identifier {
          type uint8 {
            range "1..15";
          }
      }
      leaf evaluation-precedence {
          type uint8;
      }
      list contents {
        key component-type-identifier;
        leaf component-type-identifier {
            type threegpp:component-type-id;
        }
```

```
        choice value {
```

```
          case ipv4-local {
            leaf ipv4-local {
              type inet:ipv4-address;
            }
          }
          case ipv6-prefix-local {
            leaf ipv6-prefix-local {
              type inet:ipv6-prefix;
            }
          }
          case ipv4-ipv6-remote {
            leaf ipv4-ipv6-remote {
              type inet:ip-address;
            }
          }
          case ipv6-prefix-remote {
            leaf ipv6-prefix-remote {
              type inet:ipv6-prefix;
            }
          }
          case next-header {
            leaf next-header {
              type uint8;
            }
          }
          case local-port {
            leaf local-port {
              type inet:port-number;
            }
          }
          case local-port-range {
            leaf local-port-lo {
              type inet:port-number;
            }
            leaf local-port-hi {
              type inet:port-number;
            }
          }
          case remote-port {
            leaf remote-port {
              type inet:port-number;
            }
          }
          case remote-port-range {
            leaf remote-port-lo {
              type inet:port-number;
            }
            leaf remote-port-hi {
```

```
            type inet:port-number;
          }
        }
        case ipsec-index {
          leaf ipsec-index {
            type traffic-selectors:ipsec-spi;
          }
        }
        case traffic-class {
          leaf traffic-class {
            type inet:dscp;
          }
        }
        case traffic-class-range {
            leaf traffic-class-lo {
              type inet:dscp;
            }
            leaf traffic-class-hi {
              type inet:dscp;
            }
        }
        case flow-label-type {
          leaf-list flow-label-type {
            type inet:ipv6-flow-label;
          }
        }
      }
    }
  }

  grouping tft {
    list packet-filters {
        key identifier;
        uses threegpp:packet-filter;
    }
  }

  typedef imsi-type {
      type uint64;
  }

  typedef threegpp-instr {
    description "Instruction Set for 3GPP R11";
    type bits {
      bit assign-ip {
        position 0;
      }
      bit assign-fteid-ip {
```

```
          position 1;
        }
        bit assign-fteid-teid {
          position 2;
        }
        bit session {
          position 3;
        }
        bit uplink {
          position 4;
        }
        bit downlink {
          position 5;
        }
        bit assign-dpn {
          position 6;
        }
      }
    }

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

    // Contexts Update - Contexts / UL / mob-profile
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:ul/fpcagent:mobility-tunnel-parameters/
fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:ul/fpcagent:mobility-
tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:ul/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
```

```
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:ul/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
```

```
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
fpcagent:ul/fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }

    // Contexts Update - Contexts / DL / mob-profile
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:dl/fpcagent:mobility-tunnel-parameters/
fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/fpcagent:mobility-
tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:dl/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
```

```
fpcagent:dl/fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }


    // Contexts Update -  Contexts / dpns / mobility-tunnel-parameters
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:dpns/fpcagent:mobility-tunnel-parameters/
fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
```

```
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:dpns/fpcagent:mobility-
tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:dpns/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:dpns/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
fpcagent:dpns/fpcagent:mobility-tunnel-parameters/fpcagent:profile-
parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
      }
    }


    // QoS Updates - Context / UL / qosprofile
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:ul/fpcagent:qos-profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:ul/fpcagent:qos-profile-
parameters/fpcagent:value" {
```

```
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:ul/
fpcagent:qos-profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:ul/fpcagent:qos-
profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
```

```
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
fpcagent:ul/fpcagent:qos-profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }

    // QoS Updates -  Context / DL / QoS Profile
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:dl/fpcagent:qos-profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/fpcagent:qos-profile-
parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/
fpcagent:qos-profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:dl/fpcagent:qos-
profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
fpcagent:dl/fpcagent:qos-profile-parameters/fpcagent:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
      }
    }

    grouping threegpp-properties {
      leaf imsi {
```

```
        type threegpp:imsi-type;
      }
      leaf ebi {
        type threegpp:ebi-type;
      }
      leaf lbi {
        type threegpp:ebi-type;
      }
    }

    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts" {
```

```
      uses threegpp:threegpp-properties;
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts" {
      uses threegpp:threegpp-properties;
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts" {
       uses threegpp:threegpp-properties;
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts" {
      uses threegpp:threegpp-properties;
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts" {
       uses threegpp:threegpp-properties;
    }

    grouping threegpp-commandset {
      leaf instr-3gpp-mob {
        type threegpp:threegpp-instr;
      }
    }

    augment "/fpcagent:configure/fpcagent:input/fpcagent:instructions/
fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:instructions/
fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:instructions/
fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:ports/fpcagent:instructions/fpcagent:instr-
type" {
```

```
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:ports/fpcagent:instructions/
fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }

    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:instructions/fpcagent:instr-type" {
```

```
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/
fpcagent:instructions/fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
fpcagent:instructions/fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:ports/
fpcagent:instructions/fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:ports/
fpcagent:instructions/fpcagent:instr-type" {
      case instr-3gpp-mob {
        uses threegpp:threegpp-commandset;
      }
    }

    // Deletion Augments - We add the TEID to speed up deletion
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:delete_or_query/fpcagent:targets" {
      uses threegpp:teid-value;
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:delete_or_query-success/fpcagent:targets" {
      uses threegpp:teid-value;
    }

    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:delete_or_query/fpcagent:targets" {
      uses threegpp:teid-value;
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
```

```
fpcagent:result-type/fpcagent:delete_or_query-success/fpcagent:targets" {
      uses threegpp:teid-value;
    }
}
```

Figure 31: FPC YANG 3GPP Mobility

A.1.6. FPC / PMIP Integration YANG Model

```
module ietf-dmm-fpc-pmip {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-pmip";
    prefix fpc-pmip;
```

```
    import ietf-dmm-fpcbase { prefix fpcbase; }
    import ietf-dmm-fpcagent { prefix fpcagent; }
    import ietf-pmip-qos { prefix qos-pmip; }
    import ietf-traffic-selector-types { prefix traffic-selectors; }

    organization "IETF DMM Working Group";
    contact "Satoru Matsushima <satoru.matsushima@g.softbank.co.jp>";

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

    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 "fpcbase:fpc-access-type";
    }

    identity fpcp-qos-index-pmip {
        base "fpcbase:fpc-qos-type";
    }
    identity traffic-selector-mip6 {
        base "fpcbase:fpc-descriptor-type";
    }
    identity ietf-pmip {
        base "fpcbase:fpc-mobility-profile-type";
    }

    identity pmip-tunnel-type {
        description "PMIP Tunnel Type";
    }
    identity grev1 {
        base "fpc-pmip:pmip-tunnel-type";
    }
    identity grev2 {
        base "fpc-pmip:pmip-tunnel-type";
    }
    identity ipinip {
        base "fpc-pmip:pmip-tunnel-type";
    }
    grouping pmip-mobility {
        leaf type {
            type identityref {
                base "fpc-pmip:pmip-tunnel-type";
            }
```

```
        }
        choice value {
            case gre {
                leaf key {
                    type uint32;
                    description "GRE_KEY";
                }
            }
        }
    }

    typedef pmip-instr {
      description "Instruction Set for PMIP";
      type bits {
        bit assign-ip {
          position 0;
        }
        bit assign-dpn {
          position 1;
        }
        bit session {
          position 2;
        }
        bit uplink {
          position 3;
        }
        bit downlink {
          position 4;
        }
      }
    }

    // Descriptors update - goes to Entities, Configure and Configure Bundles
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-policy/
fpcagent:descriptors/fpcagent:descriptor-value" {
      case pmip-selector {
          uses traffic-selectors:traffic-selector;
      }
    }

    // Contexts Update - Contexts / UL / mob-profile, Contexts / DL / mob-
profile and Contexts / dpns / mobility-tunnel-parameters
    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:ul/fpcagent:mobility-tunnel-parameters/
fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
```

```
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:ul/fpcagent:mobility-
tunnel-parameters/fpcagent:profile-parameters" {
      case pmip-tunnel {
```

```
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:ul/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }

    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:dl/fpcagent:mobility-tunnel-parameters/
fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/fpcagent:mobility-
tunnel-parameters/fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }

    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:dpns/fpcagent:mobility-tunnel-parameters/
fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:dpns/fpcagent:mobility-
```

```
tunnel-parameters/fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:dpns/
fpcagent:mobility-tunnel-parameters/fpcagent:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
      }
    }
```

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

    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-mobility/
fpcagent:contexts/fpcagent:dl/fpcagent:qos-profile-parameters/fpcagent:value" {
      case qos-pmip {
          uses qos-pmip:qosattribute;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/fpcagent:qos-profile-
parameters/fpcagent:value" {
      case qos-pmip {
          uses qos-pmip:qosattribute;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/fpcagent:dl/
fpcagent:qos-profile-parameters/fpcagent:value" {
      case qos-pmip {
          uses qos-pmip:qosattribute;
      }
    }

    grouping pmip-commandset {
      leaf instr-pmip {
        type fpc-pmip:pmip-instr;
      }
    }
```

```
    // Instructions Update - OP BODY, Context, Port
    augment "/fpcagent:configure/fpcagent:input/fpcagent:instructions/
fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:contexts/fpcagent:instructions/
fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
```

```
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:contexts/fpcagent:instructions/
fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:input/fpcagent:op_body/
fpcagent:create_or_update/fpcagent:ports/fpcagent:instructions/fpcagent:instr-
type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
    augment "/fpcagent:configure/fpcagent:output/fpcagent:result-type/
fpcagent:create-or-update-success/fpcagent:ports/fpcagent:instructions/
fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }

    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:instructions/fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:contexts/
fpcagent:instructions/fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:contexts/
fpcagent:instructions/fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }

    augment "/fpcagent:configure-bundles/fpcagent:input/fpcagent:bundles/
fpcagent:op_body/fpcagent:create_or_update/fpcagent:ports/
fpcagent:instructions/fpcagent:instr-type" {
```

```
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
    augment "/fpcagent:configure-bundles/fpcagent:output/fpcagent:bundles/
fpcagent:result-type/fpcagent:create-or-update-success/fpcagent:ports/
fpcagent:instructions/fpcagent:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
      }
    }
}
```

Figure 32: FPC YANG FPC / PMIP Integration

A.1.7. FPC Policy Extension YANG Model

```
module ietf-dmm-fpc-policyext {
    namespace "urn:ietf:params:xml:ns:yang:fpcpolicyext";
    prefix fpcpolicyext;

    import ietf-dmm-fpcbase { prefix fpcbase; revision-date 2016-08-03; }
    import ietf-dmm-fpcagent { prefix fpcagent; revision-date 2016-08-03; }
    import ietf-inet-types { prefix inet; revision-date 2013-07-15; }

    organization "IETF DMM Working Group";
    contact "Satoru Matsushima <satoru.matsushima@g.softbank.co.jp>";

    description
    "This module contains YANG definition for
     Forwarding Policy Configuration Protocol (FPCP)
     common Policy Action and Descriptor extensions";

    revision 2016-08-03 {
        description "Changes based on -04 version of FPC draft.";
        reference "draft-ietf-dmm-fpc-cpdp-04";
    }

    identity service-function {
        base "fpcbase:fpc-descriptor-type";
        description "Base Identifier for Service Functions.";
    }

    identity napt-service {
        base "service-function";
    }
    grouping simple-nat {
      leaf outbound-nat-address {
        type inet:ip-address;
      }
    }

    identity nat-service {
        base "service-function";
    }
    grouping simple-napt {
      leaf source-port {
        type inet:port-number;
      }
      leaf outbound-napt-address {
        type inet:ip-address;
      }
      leaf destination-port {
```

```
        type inet:port-number;
      }
    }

    identity copy-forward {
      base "fpcbase: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 fpcbase:fpc-port-id;
            }
          }
          case context-ref {
            leaf context-ref {
              type fpcbase:fpc-context-id;
            }
          }
        }
      }
    }

    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-policy/
fpcagent:actions/fpcagent:action-value" {
      case simple-nat {
          uses fpcpolicyext:simple-nat;
      }
      case simple-napt {
          uses fpcpolicyext:simple-napt;
      }
      case copy-forward {
          uses fpcpolicyext:copy-forward;
      }
    }

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

        leaf destination-ip {
            type inet:ip-prefix;
            description "Rule of destination IP";
        }
```

```
        leaf source-ip {
```

```
            type inet:ip-prefix;
            description "Rule of source IP";
        }
    }

    augment "/fpcagent:tenants/fpcagent:tenant/fpcagent:fpc-policy/
fpcagent:descriptors/fpcagent:descriptor-value" {
      case prefix-descriptor {
          uses fpcpolicyext:prefix-traffic-descriptor;
      }
    }
}
```

Figure 33: FPC YANG 3GPP FPC Policy Extensions

A.2. FPC Agent Information Model YANG Tree

This section only shows the YANG tree for the information model.

```
module: ietf-dmm-fpcagent
   +--rw tenants
   |  +--rw tenant* [tenant-id]
   |     +--rw tenant-id       fpcbase:fpc-identity
   |     +--rw fpc-policy
   |     |  +--rw policy-groups* [policy-group-id]
   |     |  |  +--rw policy-group-id    fpcbase:fpc-policy-group-id
   |     |  |  +--rw policies*          fpcbase:fpc-policy-id
   |     |  +--rw policies* [policy-id]
   |     |  |  +--rw policy-id    fpcbase:fpc-policy-id
   |     |  |  +--rw rules* [order]
   |     |  |     +--rw order          uint32
   |     |  |     +--rw descriptors* [descriptor-id]
   |     |  |     |  +--rw descriptor-id    fpcbase:fpc-identity
   |     |  |     |  +--rw direction?       fpc-direction
   |     |  |     +--rw actions* [action-id]
   |     |  |        +--rw order?       uint32
   |     |  |        +--rw action-id    fpcbase:fpc-action-id-type
   |     |  +--rw descriptors* [descriptor-id]
   |     |  |  +--rw descriptor-id      fpcbase:fpc-identity
   |     |  |  +--rw descriptor-type    identityref
   |     |  |  +--rw (descriptor-value)?
   |     |  |     +--:(all-traffic)
   |     |  |        +--rw all-traffic?       empty
   |     |  +--rw actions* [action-id]
   |     |     +--rw action-id      fpcbase: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               fpcbase:fpc-context-id
|     |  |  +--ro ports*                   fpcbase:fpc-port-id
|     |  |  +--ro dpn-group?               fpcbase:fpc-dpn-group-id
|     |  |  +--ro delegating-ip-prefixes*  inet:ip-prefix
|     |  |  +--ro ul {fpcbase:fpc-basic-agent}?
|     |  |  |  +--ro tunnel-local-address?         inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
|     |  |  |  +--ro tunnel-mtu-size?              uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?   empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?   identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip)
|     |  |  |  |     |  +--ro ip?             inet:ip-address
|     |  |  |  |     +--:(servicepath)
|     |  |  |  |        +--ro servicepath?    fpcbase:fpcp-service-path-id
|     |  |  |  +--ro qos-profile-parameters
|     |  |  |  |  +--ro qos-type?   identityref
|     |  |  |  |  +--ro (value)?
|     |  |  |  +--ro dpn-parameters
|     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
|     |  |  |     +--ro vendor-id      fpcbase:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?    empty
|     |  |  +--ro dl {fpcbase:fpc-basic-agent}?
|     |  |  |  +--ro tunnel-local-address?         inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
|     |  |  |  +--ro tunnel-mtu-size?              uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?   empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?   identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip)
|     |  |  |  |     |  +--ro ip?             inet:ip-address
|     |  |  |  |     +--:(servicepath)
|     |  |  |  |        +--ro servicepath?    fpcbase:fpcp-service-path-id
|     |  |  |  +--ro qos-profile-parameters
|     |  |  |  |  +--ro qos-type?   identityref
```

```
|     |  |  |  |  +--ro (value)?
|     |  |  |  +--ro dpn-parameters
|     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
|     |  |  |     +--ro vendor-id      fpcbase:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?    empty
|     |  |  +--ro dpns* [dpn-id direction] {fpcbase:fpc-multi-dpn}?
|     |  |  |  +--ro dpn-id                       fpcbase:fpc-dpn-id
|     |  |  |  +--ro direction                    fpcbase:fpc-direction
|     |  |  |  +--ro tunnel-local-address?        inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?       inet:ip-address
|     |  |  |  +--ro tunnel-mtu-size?             uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?   empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?   identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip)
|     |  |  |  |     |  +--ro ip?             inet:ip-address
|     |  |  |  |     +--:(servicepath)
|     |  |  |  |        +--ro servicepath?    fpcbase:fpcp-service-path-id
|     |  |  |  +--ro qos-profile-parameters
|     |  |  |  |  +--ro qos-type?   identityref
|     |  |  |  |  +--ro (value)?
|     |  |  |  +--ro dpn-parameters
|     |  |  |  +--ro vendor-parameters* [vendor-id vendor-type]
|     |  |  |     +--ro vendor-id      fpcbase:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?    empty
|     |  |  +--ro parent-context?          fpcbase:fpc-context-id
|     |  +--ro ports* [port-id]
|     |  |  +--ro port-id          fpcbase:fpc-port-id
|     |  |  +--ro policy-groups*   fpcbase:fpc-policy-group-id
|     |  +--ro monitors*
|     |     +--ro monitor-id?         fpcbase: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          fpcbase:fpc-domain-id
   |        |  +--rw domain-name?       string
   |        |  +--rw domain-type?       string
   |        |  +--rw basename?          fpcbase:fpc-identity {fpcagent:fpc-
basename-registry}?
   |        |  +--rw base-state?        string {fpcagent:fpc-basename-
registry}?
   |        |  +--rw base-checkpoint?   string {fpcagent:fpc-basename-
registry}?
   |        +--rw dpn-group-peers* [remote-dpn-group-id] {fpcbase:fpc-basic-
agent}?
   |        |  +--rw remote-dpn-group-id        fpcbase: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 tunnel-mtu-size?           uint32
   |        +--rw dpn-id?              fpcbase:fpc-dpn-id {fpcbase:fpc-basic-
agent}?
   |        +--rw control-protocols*   identityref {fpcbase:fpc-basic-agent}?
   |        +--rw dpn-groups* [dpn-group-id] {fpcbase:fpc-multi-dpn}?
   |        |  +--rw dpn-group-id        fpcbase: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        fpcbase: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 tunnel-mtu-size?           uint32
   |        |  +--rw domains* [domain-id]
   |        |     +--rw domain-id          fpcbase:fpc-domain-id
   |        |     +--rw domain-name?       string
   |        |     +--rw domain-type?       string
   |        |     +--rw basename?          fpcbase:fpc-identity {fpcagent:fpc-
basename-registry}?
   |        |     +--rw base-state?        string {fpcagent:fpc-basename-
registry}?
```

```
   |        |     +--rw base-checkpoint?   string {fpcagent:fpc-basename-
registry}?
   |        +--rw dpns* [dpn-id] {fpcbase:fpc-multi-dpn}?
   |           +--rw dpn-id            fpcbase:fpc-dpn-id
   |           +--rw dpn-name?         string
   |           +--rw dpn-groups*       fpcbase: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?   fpcbase:event-type-id
      +--rw supported-error-types* [error-type]
         +--rw error-type       identityref
         +--rw error-type-id?   fpcagent:error-type-id
rpcs: ...
```

Figure 34: 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: lyleb551144@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