Distributed Mobility Management [dmm]
Internet-Draft
Expires: December 11, 2016

C. Perkins
Futurewei
V. Devarapalli
Vasona Networks

June 9, 2016

MN Identifier Types for RFC 4283 Mobile Node Identifier Option
draft-ietf-dmm-4283mnids-02.txt

Abstract

Additional Identifier Types are proposed for use with the Mobile Node Identifier Option for MIPv6 (RFC 4283).

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 December 11, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Mobile Node Identifier Option for MIPv6 [RFC4283] has proved to be a popular design tool for providing identifiers for mobile nodes during authentication procedures with AAA protocols such as Diameter [RFC3588]. To date, only a single type of identifier has been specified, namely the MN NAI. Other types of identifiers are in common use, and even referenced in RFC 4283. In this document, we propose adding some basic types that are defined in various telecommunications standards, including types for IMSI [ThreeGPP-IDS], P-TMSI [ThreeGPP-IDS], IMEI [ThreeGPP-IDS], and GUTI [ThreeGPP-IDS]. In addition, we specify the IPv6 address itself and IEEE MAC-layer addresses as mobile node identifiers. Defining

identifiers that are tied to the physical elements of the device (RFID, MAC address etc.) help in deployment of Mobile IP because in many cases such identifiers are the most natural means for uniquely identifying the device, and will avoid additional look-up steps that might be needed if other identifiers were used.

2. New Mobile Node Identifier Types

The following types of identifiers are commonly used to identify mobile nodes. For each type, references are provided with full details on the format of the type of identifer.

The Tag Data standard promoted by Electronic Product Code(TM) (abbreviated EPC) supports several encoding systems or schemes including

- RFID-GID (Global Identifier),
- RFID-SGTIN (Serialized Global Trade Item Number),
- RFID-SSCC (Serial Shipping Container),
- RFID-SGLN (Global Location Number),
- RFID-GRAI (Global Returnable Asset Identifier),
- RFID-DOD (Department of Defense ID), and
- RFID-GIAI (Global Individual Asset Identifier).

For each RFID scheme except GID, there are two variations: a 64-bit scheme (for example, SGLN-64) and a 96-bit scheme (SGLN-96). GID has only a 96-bit scheme. Within each scheme, an EPC identifier can be represented in a binary form or other forms such as URI.

The following list includes the above RFID types as well as various other common identifiers and several different types of DUIDs.

Mobile Node Identifier Description

Identifier Type	Description	Reference
IPv6 Address		[RFC4291]
IMSI	International Mobile Subscriber Identity	[ThreeGPP-IDS]
P-TMSI	Packet-Temporary Mobile Subscriber Identity	[ThreeGPP-IDS]
GUTI	Globally Unique Temporary ID	[ThreeGPP-IDS]
EUI-48 address	48-bit Extended Unique Identifier	[IEEE802]
EUI-64 address	64-bit Extended Unique Identifier-64 bit	[IEEE802]

DUID-LLT	DHCPv6 Unique Identifier: Link-Layer address plus timestamp	[RFC3315]
DUID-EN	DHCPv6 Unique Identifier: Enterprise Number plus add'l data	[RFC3315]
DUID-LL	DHCPv6 Unique Identifier: Link-Layer address	[RFC3315]
DUID-UUID	DHCPv6 Unique Identifier: other conformant format	[RFC6355]
RFID-SGTIN-64	64-bit Serialized Global Trade Item Number	[EPC-Tag-Data]
RFID-SSCC-64	64-bit Serial Shipping Container	[EPC-Tag-Data]
RFID-SGLN-64	64-bit Serialized Global Location Number	[EPC-Tag-Data]
RFID-GRAI-64	64-bit Global Returnable Asset Identifier	[EPC-Tag-Data]
RFID-DOD-64	64-bit Department of Defense ID	[RFID-DoD-spec]
RFID-GIAI-64	64-bit Global Individual Asset Identifier	[EPC-Tag-Data]
RFID-GID-96	96-bit Global Identifier	[EPC-Tag-Data]
RFID-SGTIN-96	96-bit Serialized Global Trade Item Number	[EPC-Tag-Data]
RFID-SSCC-96	96-bit Serial Shipping Container	[EPC-Tag-Data]
RFID-SGLN-96	96-bit Serialized Global Location Number	[EPC-Tag-Data]
RFID-GRAI-96	96-bit Global Returnable Asset Identifier	[EPC-Tag-Data]
RFID-DOD-96	96-bit Department of Defense ID	[RFID-DoD-spec]
RFID-GIAI-96	96-bit Global Individual Asset Identifier	[EPC-Tag-Data]
RFID-GID-URI	Global Identifier represented as URI	[EPC-Tag-Data]
RFID-SGTIN-URI	Serialized Global Trade Item Number represented as URI	[EPC-Tag-Data]
RFID-SSCC-URI	Serial Shipping Container represented as URI	[EPC-Tag-Data]
RFID-SGLN-URI	Global Location Number represented as URI	[EPC-Tag-Data]
RFID-GRAI-URI	Global Returnable Asset Identifier represented as URI	[EPC-Tag-Data]
RFID-DOD-URI	Department of Defense ID represented as URI	[RFID-DoD-spec]
RFID-GIAI-URI	Global Individual Asset	[EPC-Tag-Data]

| | Identifier represented as URI | |
+----------------+--------------------------------+----------------+

Table 1

3. Descriptions of MNID types

In this section descriptions for the various MNID types are provided.

3.1. Description of the IPv6 address type

The IPv6 address [RFC4291] is encoded as a 16 octet string containing the full IPv6 address.

3.2. Description of the IMSI MNID type

The International Mobile Subscriber Identity (IMSI) [ThreeGPP-IDS] is at most 15 decimal digits (i.e., digits from 0 through 9). The IMSI MUST be encoded as a string of octets in network order, where each digit occupies 4 bits. The last digit MUST be zero padded, if needed, for full octet size. For example an example IMSI 123456123456789 would be encoded as follows:

0x12, 0x34, 0x56, 0x12, 0x34, 0x56, 0x78, 0x90

3.3. Description of the EUI-48 address type

The IEEE EUI-48 address [IEEE802-eui48] is encoded as a 6 octet string containing the IEEE EUI-48 address.

3.4. Description of the EUI-64 address type

The IEEE EUI-64 address [IEEE802-eui64] is encoded as a 8 octet string containing the full IEEE EUI-64 address.

3.5. Description of the DUID-LLT type

The DUID-LLT is the DHCPv6 Unique Identifier (DUID) formulated by concatenating the link-layer address plus a timestamp [RFC3315]. This type of DUID consists of a two octet type field containing the value 1, a two octet hardware type code, four octets containing a time value, followed by link-layer address of any one network interface that is connected to the DHCP device at the time that the DUID is generated. The time value is the time that the DUID is generated represented in seconds since midnight (UTC), January 1, 2000, modulo 2^32. Since the link-layer address can be of variable length [RFC2464], the DUID-LLT is of variable length.

3.6. Description of the DUID-EN type

The DUID-EN is the DHCPv6 Unique Identifier (DUID) formulated by concatenating the Enterprise Number plus some additional data [RFC3315]. This form of DUID is assigned by the vendor to the device. It consists of a two octet type field containing the value 2, the vendor's registered Private Enterprise Number as maintained by IANA, followed by a unique identifier assigned by the vendor. Since the vendor's unique identifier can be of variable length, the DUID-EN is of variable length.

3.7. Description of the DUID-LL type

The DUID-LL is the DHCPv6 Unique Identifier (DUID) formulated by concatenating the network hardware type code and the link-layer address [RFC3315]. This type of DUID consists of two octets containing the DUID type 3, a two octet network hardware type code, followed by the link-layer address of any one network interface that is permanently connected to the client or server device. For example, a host that has a network interface implemented in a chip that is unlikely to be removed and used elsewhere could use a DUID-LL. Since the link-layer address can be of variable length, the DUID-LL is of variable length.

3.8. Description of the DUID-UUID type

The DUID-UUID [RFC6355] is the DHCPv6 Unique Identifier based on the Universally Unique IDentifier (UUID) [RFC4122]. This type of DUID consists of two octets containing the DUID type 4, followed by 128-bit UUID.

3.9. Description of the RFID types

The General Identifier (GID) that is used with RFID is composed of three fields - the General Manager Number, Object Class and Serial Number. The General Manager Number identifies an organizational entity that is responsible for maintaining the numbers in subsequent fields. GID encodings include a fourth field, the header, to guarantee uniqueness in the namespace defined by EPC.

Some of the RFID types depend on the Global Trade Item Number (GTIN) code defined in the General EAN.UCC Specifications [EANUCCGS]. A GTIN identifies a particular class of object, such as a particular kind of product or SKU.

The EPC encoding scheme for SGTIN permits the direct embedding of EAN.UCC System standard GTIN and Serial Number codes on EPC tags. In

all cases, the check digit is not encoded. Two encoding schemes are specified, SGTIN-64 (64 bits) and SGTIN-96 (96 bits).

The Serial Shipping Container Code (SSCC) is defined by the EAN.UCC Specifications. Unlike the GTIN, the SSCC is already intended for assignment to individual objects and therefore does not require additional fields to serve as an EPC pure identity. Two encoding schemes are specified, SSCC-64 (64 bits) and SSCC-96 (96 bits).

The Global Location Number (GLN) is defined by the EAN.UCC Specifications. A GLN can represent either a discrete, unique physical location such as a warehouse slot, or an aggregate physical location such as an entire warehouse. In addition, a GLN can represent a logical entity that performs a business function such as placing an order. The Serialized Global Location Number (SGLN) includes the Company Prefix, Location Reference, and Serial Number.

The Global Returnable Asset Identifier is (GRAI) is defined by the General EAN.UCC Specifications. Unlike the GTIN, the GRAI is already intended for assignment to individual objects and therefore does not require any additional fields to serve as an EPC pure identity. The GRAI includes the Company Prefix, Asset Type, and Serial Number.

The Global Individual Asset Identifier (GIAI) is defined by the General EAN.UCC Specifications. Unlike the GTIN, the GIAI is already intended for assignment to individual objects and therefore does not require any additional fields to serve as an EPC pure identity. The GRAI includes the Company Prefix, and Individual Asset Reference.

The DoD Construct identifier is defined by the United States Department of Defense (DoD). This tag data construct may be used to encode tags for shipping goods to the DoD by a supplier who has already been assigned a CAGE (Commercial and Government Entity) code.

3.9.1. Description of the RFID-SGTIN-64 type

The RFID-SGTIN-64 is encoded as specified in [EPC-Tag-Data]. The SGTIN-64 includes five fields: Header, Filter Value (additional data that is used for fast filtering and pre-selection), Company Prefix Index, Item Reference, and Serial Number. Only a limited number of Company Prefixes can be represented in the 64-bit tag.

3.9.2. Description of the RFID-SGTIN-96 type

The RFID-SGTIN-96 is encoded as specified in [EPC-Tag-Data]. The SGTIN-96 includes six fields: Header, Filter Value, Partition (an indication of where the subsequent Company Prefix and Item Reference

numbers are divided), Company Prefix Index, Item Reference, and Serial Number.

3.9.3. Description of the RFID-SSCC-64 type

The RFID-SSCC-64 is encoded as specified in [EPC-Tag-Data]. The SSCC-64 includes four fields: Header, Filter Value, Company Prefix Index, and Serial Reference. Only a limited number of Company Prefixes can be represented in the 64-bit tag.

3.9.4. Description of the RFID-SSCC-96 type

The RFID-SSCC-96 is encoded as specified in [EPC-Tag-Data]. The SSCC-96 includes six fields: Header, Filter Value, Partition, Company Prefix, and Serial Reference, as well as 24 bits that remain Unallocated and must be zero.

3.9.5. Description of the RFID-SGLN-64 type

The RFID-SGLN-64 type is encoded as specified in [EPC-Tag-Data]. The SGLN-64 includes five fields: Header, Filter Value, Company Prefix Index, Location Reference, and Serial Number.

3.9.6. Description of the RFID-SGLN-96 type

The RFID-SGLN-96 type is encoded as specified in [EPC-Tag-Data]. The SGLN-96 includes six fields: Header, Filter Value, Partition, Company Prefix, Location Reference, and Serial Number.

3.9.7. Description of the RFID-GRAI-64 type

The RFID-GRAI-64 type is encoded as specified in [EPC-Tag-Data]. The GRAI-64 includes five fields: Header, Filter Value, Company Prefix Index, Asset Type, and Serial Number.

3.9.8. Description of the RFID-GRAI-96 type

The RFID-GRAI-96 type is encoded as specified in [EPC-Tag-Data]. The GRAI-96 includes six fields: Header, Filter Value, Partition, Company Prefix, Asset Type, and Serial Number.

3.9.9. Description of the RFID-GIAI-64 type

The RFID-GIAI-64 type is encoded as specified in [EPC-Tag-Data]. The GIAI-64 includes four fields: Header, Filter Value, Company Prefix Index, and Individual Asset Reference.

3.9.10. Description of the RFID-GIAI-96 type

The RFID-GIAI-96 type is encoded as specified in [EPC-Tag-Data]. The GIAI-96 includes five fields: Header, Filter Value, Partition, Company Prefix, and Individual Asset Reference.

3.9.11. Description of the RFID-DoD-64 type

The RFID-DoD-64 type is encoded as specified in [RFID-DoD-spec]. The DoD-64 type includes four fields: Header, Filter Value, Government Managed Identifier, and Serial Number.

3.9.12. Description of the RFID-DoD-96 type

The RFID-DoD-96 type is encoded as specified in [RFID-DoD-spec]. The DoD-96 type includes four fields: Header, Filter Value, Government Managed Identifier, and Serial Number.

3.9.13. Description of the RFID URI types

In some cases, it is desirable to encode in URI form a specific encoding of an RFID tag. For example, an application may prefer a URI representation for report preparation. Applications that wish to manipulate any additional data fields on tags may need some representation other than the pure identity forms.

For this purpose, the fields as represented the previous sections are associated with specified fields in the various URI types. For instance, the URI may have fields such as CompanyPrefix, ItemReference, or SerialNumber. For details and encoding specifics, consult [EPC-Tag-Data].

4. Security Considerations

This document does not introduce any security mechanisms, and does not have any impact on existing security mechanisms. Insofar as the selection of a security association may be dependent on the exact form of a mobile node identifier, additional specification may be necessary when the new identifier types are employed with the general AAA mechanisms for mobile node authorizations.

Some identifiers (e.g., IMSI) are considered to be private information. If used in the MNID extension as defined in this document, the packet including the MNID extension should be encrypted so that personal information or trackable identifiers would not be inadvertently disclosed to passive observers. Operators can potentially apply IPsec Encapsulating Security Payload (ESP) with

confidentiality and integrity protection for protecting the location information.

Moreover, MNIDs containing sensitive identifiers might only be used for signaling during initial network entry. Subsequent binding update exchanges might then rely on a temporary identifier allocated during the initial network entry, perhaps using mechanisms not standardized within the IETF. Managing the association between long-lived and temporary identifiers is outside the scope of this document.

5. IANA Considerations

The new mobile node identifier types defined in the document should be assigned values from the "Mobile Node Identifier Option Subtypes" registry. The following values should be assigned.

New Mobile Node Identifier Types

Identifier Type	Identifier Type Number
IPv6 Address	2
IMSI	3
P-TMSI	4
EUI-48 address	5
EUI-64 address	6
GUTI	7
DUID-LLT	8
DUID-EN	9
DUID-LL	10
DUID-UUID	11
	12-15 reserved
	16 reserved
RFID-SGTIN-64	17
RFID-SSCC-64	18
RFID-SGLN-64	19
RFID-GRAI-64	20
RFID-DOD-64	21
RFID-GIAI-64	22
	23 reserved
RFID-GID-96	24
RFID-SGTIN-96	25
RFID-SSCC-96	26
RFID-SGLN-96	27
RFID-GRAI-96	28
RFID-DOD-96	29
RFID-GIAI-96	30
	31 reserved
RFID-GID-URI	32
RFID-SGTIN-URI	33
RFID-SSCC-URI	34
RFID-SGLN-URI	35
RFID-GRAI-URI	36
RFID-DOD-URI	37
RFID-GIAI-URI	38
	39-255 reserved

Table 2

See Section 3 for additional information about the identifier types.

6. Acknowledgements

The authors wish to acknowledge Hakima Chaouchi, Jouni Korhonen and Sri Gundavelli for their helpful comments.

7. References

7.1. Normative References

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <http://www.rfc-editor.org/info/rfc2464>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC4283] Patel, A., Leung, K., Khalil, M., Akhtar, H., and K. Chowdhury, "Mobile Node Identifier Option for Mobile IPv6 (MIPv6)", RFC 4283, DOI 10.17487/RFC4283, November 2005, <http://www.rfc-editor.org/info/rfc4283>.

[RFC4285] Patel, A., Leung, K., Khalil, M., Akhtar, H., and K. Chowdhury, "Authentication Protocol for Mobile IPv6", RFC 4285, DOI 10.17487/RFC4285, January 2006, <http://www.rfc-editor.org/info/rfc4285>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <http://www.rfc-editor.org/info/rfc6355>.

7.2. Informative References

[EANUCCGS] EAN International and the Uniform Code Council, , "General EAN.UCC Specifications Version 5.0", Jan 2004.

[EPC-Tag-Data]
EPCglobal Inc., , "EPC(TM) Generation 1 Tag Data Standards Version 1.1 Rev.1.27 http://www.gs1.org/gsmp/kc/epcglobal/tds/ tds_1_1_rev_1_27-standard-20050510.pdf", January 2005.

[IEEE802] IEEE, , "IEEE Std 802: IEEE Standards for Local and Metropolitan Networks: Overview and Architecture", 2001.

[IEEE802-eui]
IEEE, , "Guidelines for Use Organizationally Unique Identifier (OUI) and Company ID (CID) https://standards.ieee.org/develop/regauth/tut/eui.pdf", 2001.

[IEEE802-eui48]
IEEE, , "Guidelines for 48-Bit Global Identifier (EUI-48) https://standards.ieee.org/develop/regauth/tut/eui48.pdf", 2001.

[IEEE802-eui64]
IEEE, , "Guidelines for 64-Bit Global Identifier (EUI-64) https://standards.ieee.org/develop/regauth/tut/eui.pdf64", 2001.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, DOI 10.17487/RFC3588, September 2003, <http://www.rfc-editor.org/info/rfc3588>.

[RFID-DoD-spec]
Department of Defense, , "United States Department of Defense Suppliers Passive RFID Information Guide (Version 15.0)", January 2010.

[ThreeGPP-IDS]
3rd Generation Partnership Project, , "3GPP Technical Specification 23.003 V8.4.0: Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 8)", March 2009.

Authors' Addresses

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

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

Vijay Devarapalli
Vasona Networks
2900 Lakeside Drive, Suite 180
Santa Clara, CA 95054
USA

DMM WG
Internet-Draft
Intended status: Informational
Expires: February 23, 2017

S. Gundavelli
Cisco
S. Jeon
Sungkyunkwan University
August 22, 2016

DMM Deployment Models and Architectural Considerations
draft-ietf-dmm-deployment-models-00.txt

Abstract

This document identifies the deployment models for Distributed Mobility Management architecture.

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 February 23, 2017.

Copyright Notice

Table of Contents

1. Overview

One of the key aspects of the Distributed Mobility Management (DMM) architecture is the separation of control plane (CP) and data plane (DP) functions of a network element. While data plane elements continue to reside on customized networking hardware, the control plane resides as a software element in the cloud. This is usually referred to as CP-DP separation and is the basis for the IETF's DMM Architecture. This approach of centralized control plane and distributed data plane allows elastic scaling of control plane and efficient use of common data plane that is agnostic to access architectures.

This document identifies the functions in the DMM architecture and the supported deployment models.

2. Conventions and Terminology

2.1. Conventions

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 RFC 2119 [RFC2119].

2.2. Terminology

All the mobility related terms are to interpreted as defined in [RFC6275], [RFC5213], [RFC5844], [RFC7333], [RFC7429], [I-D.ietf-sfc-nsh] and [I-D.ietf-dmm-fpc-cpdp]. Additionally, this document uses the following terms:

Home Control-Plane Anchor (H-CPA)

: The Home-CPA function hosts the mobile node's mobility session. There can be more than one mobility session for a mobile node [MN] and those sessions may be anchored on the same or different Home-CPA's. The home-CPA will interface with the home-dpa for managing the forwarding state.

Home Data Plane Anchor (Home-DPA)

: The Home-DPA is the topological anchor for the mobile node's IP address/prefix(es). The Home-DPA is chosen by the Home-CPA on a session-basis. The Home-DPA is in the forwarding path for all the mobile node's IP traffic.

Access Control Plane Node (Access-CPN)

The Access-CPN is responsible for interfacing with the mobile node's Home-CPA and with the Access-DPN. The Access-CPN has a protocol interface to the Home-CPA.

Access Data Plane Node (Access-DPN)

The Access-DPN function is hosted on the first-hop router where the mobile node is attached. This function is not hosted on a layer-2 bridging device such as a eNode(B) or Access Point.

3. DMM Architectural Overview

Following are the key goals of the Distributed Mobility Management architecture.

1. Separation of control and data Plane
2. Aggregation of control plane for elastic scaling
3. Distribution of the data plane for efficient network usage
4. Elimination of mobility state from the data plane
5. Dynamic selection of control and data plane nodes
6. Enabling the mobile node with network properties
7. Relocation of anchor functions for efficient network usage

3.1. DMM Service Primitives

The functions in the DMM architecture support a set of service primitives. Each of these service primitives identifies a specific service capability with the exact service definition. The functions in the DMM architecture are required to support a specific set of service primitives that are mandatory for that service function. Not all service primitives are applicable to all DMM functions. The below table identifies the service primitives that each of the DMM function SHOULD support. The marking "X" indicates the service primitive on that row needs to be supported by the identified DMM function on the corresponding column; for example, the IP address management must be supported by Home-CPA function.

Service Primitive	H-CPA	H-DPA	A-CPN	A-DPN	MC	RC
IP Management	X				X	
IP Anchoring		X				
MN Detect			X	X		
Routing		X		X		
Tunneling		X		X		
QoS Enforcement		X		X		
FPC Client	X		X		X	
FPC Agent		X		X		X
NSH Classifier		X		X		

Figure 1: Mapping of DMM functions

3.2. DMM Functions and Interfaces

3.2.1. Home Control-Plane Anchor (H-CPA):

The Home-CPA function hosts the mobile node's mobility session. There can be more than one mobility session for a mobile node and those sessions may be anchored on the same or different Home-CPA's. The home-CPA will interface with the homd-dpa for managing the forwarding state.

There can be more than one Home-CPA serving the same mobile node at a given point of time, each hosting a different control plane session.

The Home-CPA is responsible for life cycle management of the session, interfacing with the policy infrastructure, policy control and interfacing with the Home-DPA functions.

The Home-CPA function typically stays on the same node. In some special use-cases (Ex: Geo-Redundancy), the session may be migrated to a different node and with the new node assuming the Home-CPA role for that session.

3.2.2. Home Data-Plane Anchor (H-DPA):

The Home-DPA is the topological anchor for the mobile node's IP address/prefix(es). The Home-DPA is chosen by the Home-CPA/MC on a session-basis. The Home-DPA is in the forwarding path for all the mobile node's IP traffic.

As the mobile node roams in the mobile network, the mobile node's access-DPN may change, however, the Home-DPA does not change, unless the session is migrated to a new node.

The Home-DPA interfaces with the Home-CPA/MC for all IP forwarding and QoS rules enforcement.

The Home-DPA and the Access-DPN functions may be collocated on the same node.

3.2.3. Access Control Plane Node (Access-CPN)

The Access-CPN is responsible for interfacing with the mobile node's Home-CPA and with the Access-DPN. The Access-CPN has a protocol interface to the Home-CPA.

The Access-CPN is responsible for the mobile node's Home-CPA selection based on: Mobile Node's Attach Preferences, Access and Subscription Policy, Topological Proximity and Other Considerations.

The Access-CPN function is responsible for MN's service authorization. It will interface with the access network authorization functions.

3.2.4. Access Data Plane Node (Access-DPN)

The Access-DPN function is hosted on the first-hop router where the mobile node is attached. This function is not hosted on a layer-2 bridging device such as a eNode(B) or Access Point.

The Access-DPA will have a protocol interface to the Access-CPA.

The Access-DPN and the Home-DPA functions may be collocated on the same node.

3.2.5. DMM Function Mapping to other Architectures

Following table identifies the potential mapping of DMM functions to protocol functions in other system architectures.

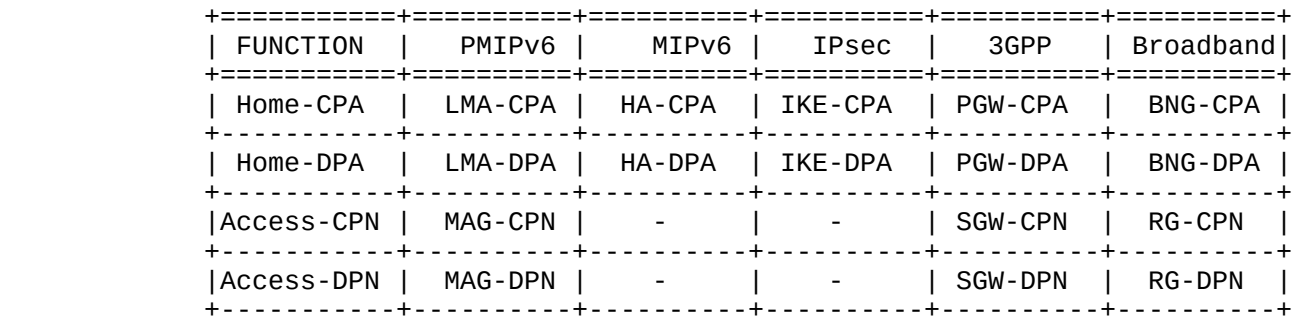


Figure 2: Mapping of DMM functions

4. Deployment Models

This section identifies the key deployment models for the DMM architecture.

4.1. Model-1: Split Home Anchor Mode

In this model, the control and the data plane functions of the home anchor are separated and deployed on different nodes. The control plane function of the Home anchor is handled by the Home-CPA and where as the data plane function is handled by the Home-DPA. In this model, the access node operates in the legacy mode with the integrated control and user plane functions.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the control plane functions to interact with the data plane for the subscriber’s forwarding state management.

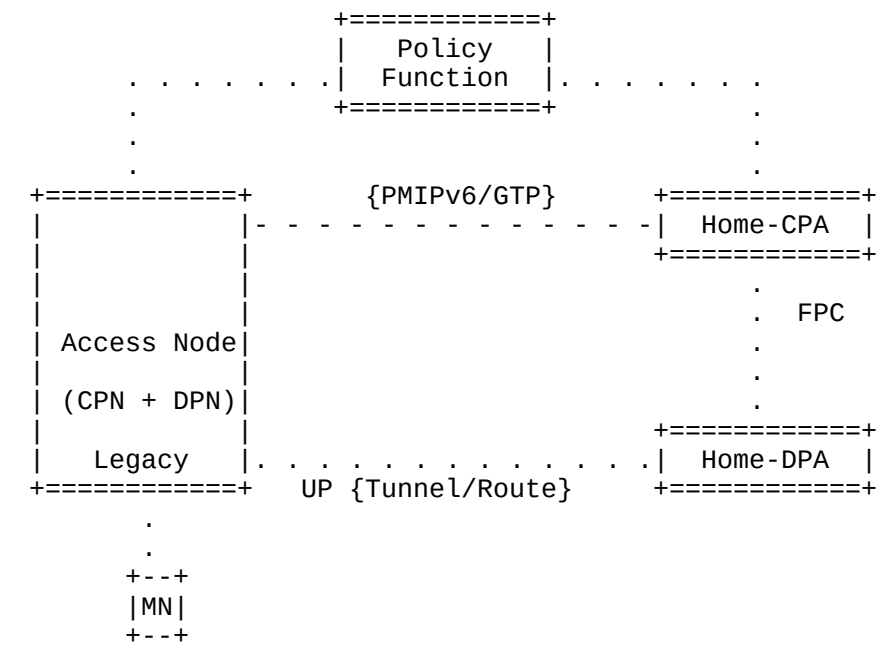


Figure 3: Split Home Anchor Mode

4.2. Model-2: Seperated Control and User Plane Mode

In this model, the control and the data plane functions on both the home anchor and the access node are seperated and deployed on different nodes. The control plane function of the Home anchor is handled by the Home-CPA and where as the data plane function is handled by the Home-DPA. The control plane function of the access node is handled by the Access-CPN and where as the data plane function is handled by the Access-DPN.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the control plane functions of the home and access nodes to interact with the respective data plane functions for the subscriber’s forwarding state management.

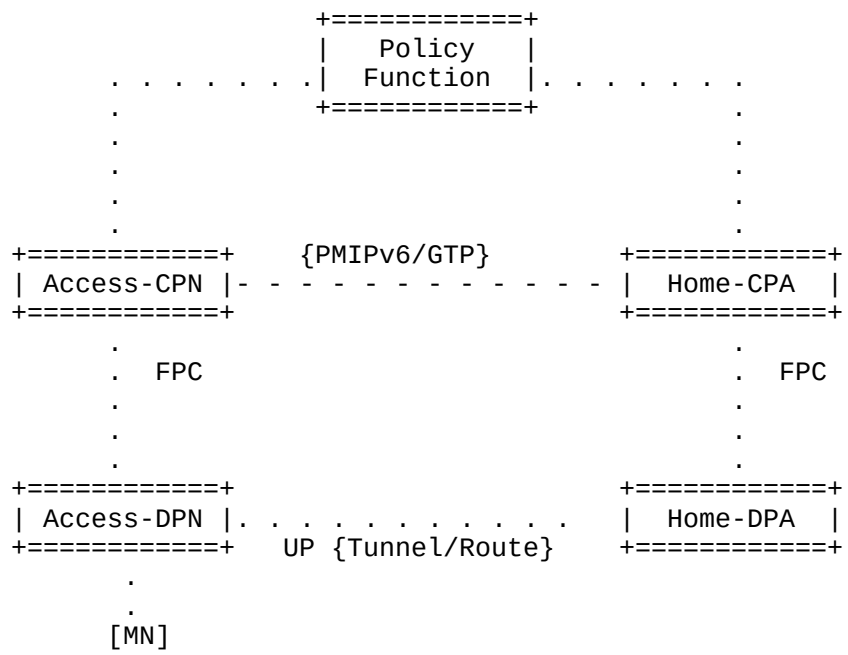


Figure 4: Seperated Control and User Plane Mode

4.3. Model-3: Centralized Control Plane Mode

In this model, the control-plane functions of the home and the access nodes are collapsed. This is a flat architecture with no signaling protocol between the access node and home anchors. The interface between the Home-CPA and the Access-DPN is internal to the system.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the mobility controller to interact with the respective data plane functions for the subscriber’s forwarding state management.

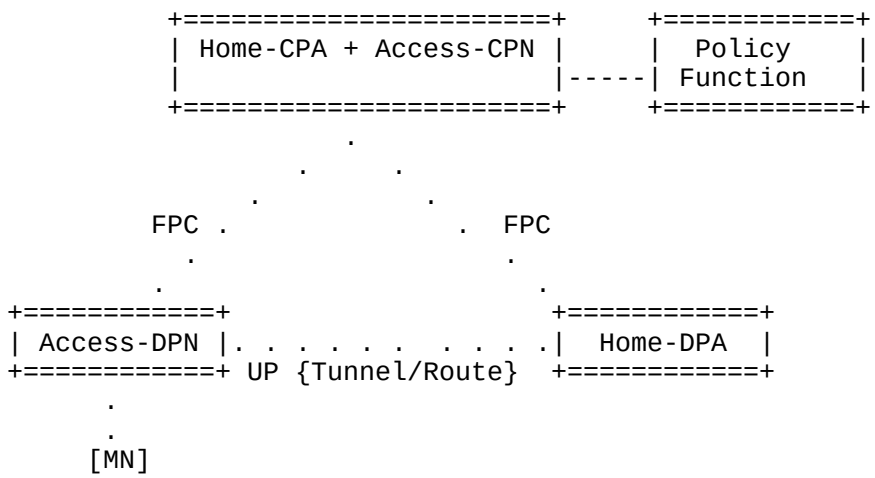


Figure 5: Centralized Control Plane Mode

4.4. Model-4: Data Plane Abstraction Mode

In this model, the data plane network is completely abstracted from the control plane. There is a new network element, Routing Controller which abstracts the entire data plane network and offers data plane services to the control plane functions. The control plane functions, Home-CPA and the Access-CPN interface with the Routing Controller for the forwarding state management.

The FPC interface defined in [I-D.ietf-dmm-fpc-cpdp] allows the Home-CPA and Access-CPN functions to interface with the Routing Controller for subscriber's forwarding state management.

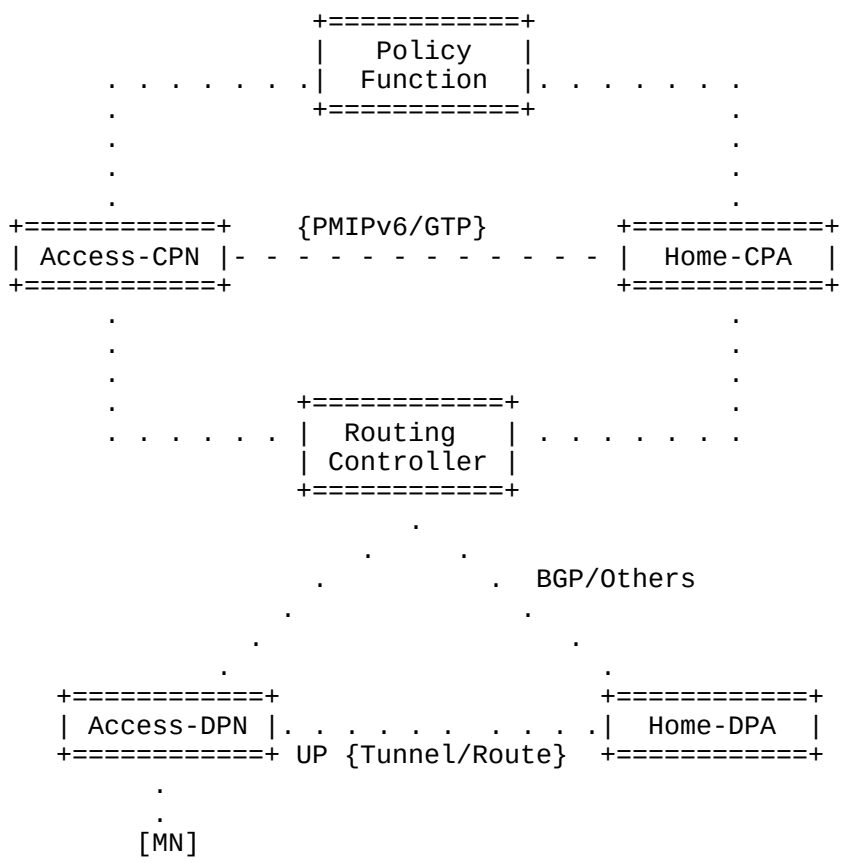


Figure 6: Data Plane Abstraction Mode

4.5. On-Demand Control Plane Orchestration Mode

In this model, there is a new function Mobility Controller which manages the orchestration of Access-CPN and Home-CPA functions. The Mobility Controller allocates the Home-CPA and Access-DPN

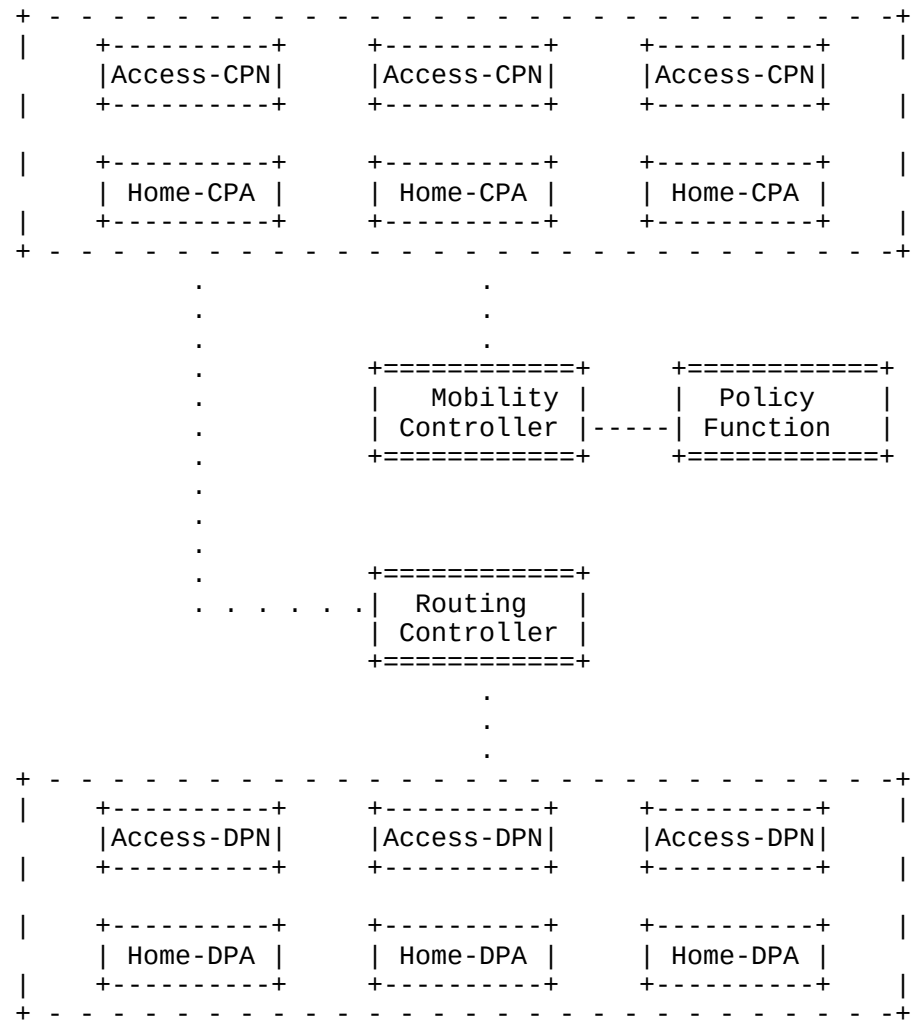


Figure 7: On-Demand CP Orchestration Mode

5. IANA Considerations

This document does not require any IANA actions.

6. Security Considerations

The control-plane messages exchanged between a Home-CPA and the Home-DPA must be protected using end-to-end security associations with data-integrity and data-origination capabilities.

IPsec ESP in transport mode with mandatory integrity protection should be used for protecting the signaling messages. IKEv2 should be used to set up security associations between the Home-CPA and Home-DPA.

There are no additional security considerations other than what is presented in the document.

7. Work Team

This document reflects contributions from the following work team members:

Younghan Kim

younghak@ssu.ac.kr

Vic Liu

liuzhiheng@chinamobile.com

Danny S Moses

danny.moses@intel.com

Marco Liebsch

liebsch@neclab.eu

Carlos Jesus Bernardos Cano

cjbc@it.uc3m.es

8. Acknowledgements

This document is a result of DMM WT#4 team discussions and ideas taken from several DMM WG presentations and documents including, draft-sijeon-dmm-deployment-models, draft-liu-dmm-deployment-scenario and others. The work teams would like to thank the authors of these documents and additionally the discussions in DMM Working group that

helped shape this document.

9. References

9.1. 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>.

9.2. Informative References

[I-D.ietf-dmm-fpc-cpdp] Liebsch, M., Matsushima, S., Gundavelli, S., Moses, D., and L. Bertz, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-03 (work in progress), March 2016.

[I-D.ietf-sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-05 (work in progress), May 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>.

[RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy Mobile IPv6", RFC 5844, DOI 10.17487/RFC5844, May 2010, <http://www.rfc-editor.org/info/rfc5844>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[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>.

[RFC7429] Liu, D., Ed., Zuniga, JC., Ed., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, DOI 10.17487/RFC7429, January 2015, <http://www.rfc-editor.org/info/rfc7429>.

Authors' Addresses

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

Email: sgundave@cisco.com

Seil Jeon
Sungkyunkwan University
2066 Seobu-ro, Jangan-gu
Suwon, Gyeonggi-do
Korea

Email: seiljeon@skku.edu

DMM
Internet-Draft
Intended status: Informational
Expires: March 27, 2017

H. Chan, Ed.
X. Wei
Huawei Technologies
J. Lee
Sangmyung University
S. Jeon
Sungkyunkwan University
A. Petrescu
CEA, LIST
F. Templin
Boeing Research and Technology

September 23, 2016

Distributed Mobility Anchoring
draft-ietf-dmm-distributed-mobility-anchoring-02

Abstract

This document defines distributed mobility anchoring to meet diverse mobility needs in 5G Wireless and beyond. Multiple anchors and nodes with mobility functions work together to provide IP mobility support. A network or network slice may be configured with distributed mobility anchoring depending on the needs of mobility support. In the distributed mobility anchoring environment, multiple anchors are available for mid-session switching of an IP prefix anchor. Without an ongoing session, i.e., no IP session continuity required, a flow of a mobile node can be re-started using a new IP prefix which is allocated from a new network of the mobile node and is therefore anchored to the new network. With an ongoing session, the anchoring of the prior IP prefix may be relocated to the new network to enable IP session continuity.

This Internet-Draft will expire on March 27, 2017.

Table of Contents

1. Introduction

A key requirement in distributed mobility management [RFC7333] is to enable traffic to avoid traversing a single mobility anchor far from an optimal route. Distributed mobility management solutions do not make use of centrally deployed mobility anchor for a data plane [Paper-Distributed.Mobility]. As such, the traffic of a flow SHOULD be able to change from traversing one mobility anchor to traversing another mobility anchor as a mobile node (MN) moves, or when changing operation and management requirements call for mobility anchor switching, thus avoiding non-optimal routes. This draft proposes distributed mobility anchoring to enable making such route changes.

Distributed mobility anchoring employs multiple anchors in the data plane. In general, control plane functions may be separate from data plane functions and be centralized but may also be co-located with the data plane functions at the distributed anchors. Different configurations of distributed mobility anchoring are described in Section 3.1. For instance, the configurations for network-based mobility support in a flat network, for network-based mobility support in a hierarchical network, for host-based mobility support, and for NEtwork MObility (NEMO) basic support are described respectively in Section 3.1.1, Section 3.1.2, Section 3.1.3 and Section 3.1.4. Required operations and parameters for distributed mobility anchoring are presented in Section 3.2. For instance, location management is described in Section 3.2.1, forwarding management is described in Section 3.2.2.

An MN attached to an access router of a network or network slice may be allocated an IP prefix which is anchored to that router. It may then use an IP address configured from this prefix as the source IP address to run a flow with its correspondent node (CN). When there are multiple mobility anchors, an address selection for a given flow

is first required before the flow is initiated. Using an anchor in an MN's network of attachment has the advantage that the packets can simply be forwarded according to the forwarding table. Although the anchor is in the MN's network of attachment when the flow was initiated, the MN may later move to another network, so that the IP no longer belongs to the current network of attachment of the MN.

Whether the flow needs IP session continuity will determine how to ensure that the IP address of the flow will be anchored to the new network of attachment. If the ongoing IP flow can cope with an IP prefix/address change, the flow can be reinitiated with a new IP address anchored in the new network as shown in Section 4.1. On the other hand, if the ongoing IP flow cannot cope with such change, mobility support is needed as shown in Section 4.2. A network or network slice supporting a mix of flows requiring and not requiring IP mobility support will need to distinguish these flows. The guidelines for such network or network slice are described in Section 4.1.1. The general guidelines for such network or network slice to provide IP mobility support are described in Section 4.2.1.

Specifically, IP mobility support can be provided by changing the anchoring of the IP prefix/address of the flow from the home network of the flow to the new network of attachment. The basic case may be with network-based mobility for a flat network configuration described in Section 5.1 with the guidelines described in Section 5.1.1. This case is discussed further with a centralized control plane in Section 5.2 with additional guidelines described in Section 5.2.1. A level of hierarchy of nodes may then be added to the network configuration. Mobility involving change in the Data Plane Node (DPN) without changing the Data Plane Anchor (DPA) is described in Section 5.3 with additional guidelines described in Section 5.3.1 Mobility involving change in the DPN without changing the DPA is described in Section 5.4 with additional guidelines described in Section 5.4.1

2. Conventions and Terminology

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

All general mobility-related terms and their acronyms used in this document are to be interpreted as defined in the Mobile IPv6 (MIPv6) base specification [RFC6275], the Proxy Mobile IPv6 (PMIPv6) specification [RFC5213], the "Mobility Related Terminologies" [RFC3753], and the DMM current practices and gap analysis [RFC7429]. These include terms such as mobile node (MN), correspondent node

(CN), home agent (HA), home address (HoA), care-of-address (CoA), local mobility anchor (LMA), and mobile access gateway (MAG).

In addition, this document uses the following terms:

Home network of an application session or a home address: the network that has allocated the HoA used for the session identifier by the application running in an MN. The MN may be running multiple application sessions, and each of these sessions can have a different home network.

IP prefix/address anchoring: An IP prefix, i.e., Home Network Prefix (HNP), or address, i.e., HoA, allocated to an MN is topologically anchored to an anchor node when the anchor node is able to advertise a connected route into the routing infrastructure for the allocated IP prefix.

Location Management (LM) function: managing and keeping track of the internetwork location of an MN. The location information may be a binding of the IP advertised address/prefix, e.g., HoA or HNP, to the IP routing address of the MN or of a node that can forward packets destined to the MN.

When the MN is a mobile router (MR) carrying a mobile network of mobile network nodes (MNN), the location information will also include the mobile network prefix (MNP), which is the IP prefix delegated to the MR. The MNP is allocated to the MNNs in the mobile network.

LM is a control plane function.

In a client-server protocol model, location query and update messages may be exchanged between a Location Management client (LMc) and a Location Management server (LMs).

Optionally, there may be a Location Management proxy (LMp) between LMc and LMs.

With separation of control plane and data plane, the LM function is in the control plane. It may be a logical function at the control plane node, control plane anchor, or mobility controller.

It may be distributed or centralized.

Forwarding Management (FM) function: packet interception and forwarding to/from the IP address/prefix assigned to the MN, based on the internetwork location information, either to the destination or to some other network element that knows how to forward the packets to their destination.

This function may be used to achieve traffic indirection. With separation of control plane and data plane, the FM function may split into a FM function in the data plane (FM-DP) and a FM function in the control plane (FM-CP).

FM-DP may be distributed with distributed mobility management. It may be a function in a data plane anchor or data plane node.

FM-CP may be distributed or centralized. It may be a function in a control plane node, control plane anchor or mobility controller.

Security Management (SM) function: The security management function controls security mechanisms/protocols providing access control, integrity, authentication, authorization, confidentiality, etc. for the control plane and data plane.

This function resides in all nodes such as control plane anchor, data plane anchor, mobile node, and correspondent node.

3. Distributed Mobility Anchoring

3.1. Configurations for Different Networks or Network Slices

The mobility functions may be implemented in different configurations of distributed mobility anchoring in architectures separating the control and data planes. The separation described in [I-D.ietf-dmm-deployment-models] has defined the home control plane anchor (Home-CPA), home data plane anchor (Home-DPA), access control plane node (Access-CPN), and access data plane node (Access-DPN), which are respectively abbreviated as CPA, DPA, CPN, and DPN here. Some configurations are described in [I-D.sijeon-dmm-deployment-models].

Different networks or different network slices may have different configurations in distributed mobility anchoring.

The configurations also differ depending on the desired mobility supports: network-based mobility support for a flat network in Section 3.1.1, network-based mobility support for a hierarchical network in Section 3.1.2, host-based mobility support

(Section 3.1.3), and NEtwork MObility (NEMO) based support in Section 3.1.4.

3.1.1. Network-based Mobility Support for a Flat Network

Figure 1 shows two different configurations of network-based mobility management for a flat network.

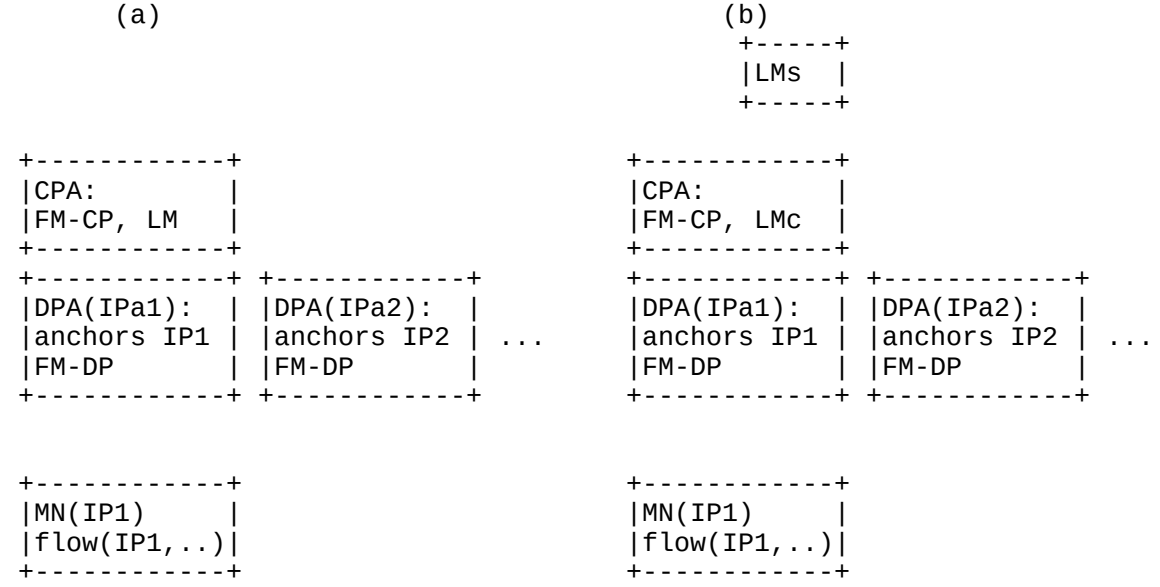


Figure 1. Configurations of network-based mobility management for a flat network (a) FM-CP and LM at CPA, FM-DP at DPA; (b) Separate LMs, FM-CP and LMc at CPA, FM-DP at DPA.

Figure 1 also shows a distributed mobility anchoring environment with multiple instances of the DPA.

There is an FM-DP function at each of the distributed DPA.

The control plane may either be distributed (not shown) or centralized. When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

There is an FM-CP function at the CPA.

An MN is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1. The MN uses IP1 to communicate with a CN not shown in the figure. The flow of this communication

session is shown as flow(IP1, ...) which uses IP1 and other parameters.

In Figure 1(a), LM and FM-CP co-locate at CPA.

Then LM may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

Figure 1(b) differs from Figure 1(a) in that the LM function is split into a server LMs and a client LMc.

LMc and FM-CP co-locate at the CPA.

The LMs may be centralized whereas the LMc may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

3.1.2. Network-based Mobility Support for a Hierarchical Network

Figure 2 shows two different configurations of network-based mobility management for a hierarchical network.

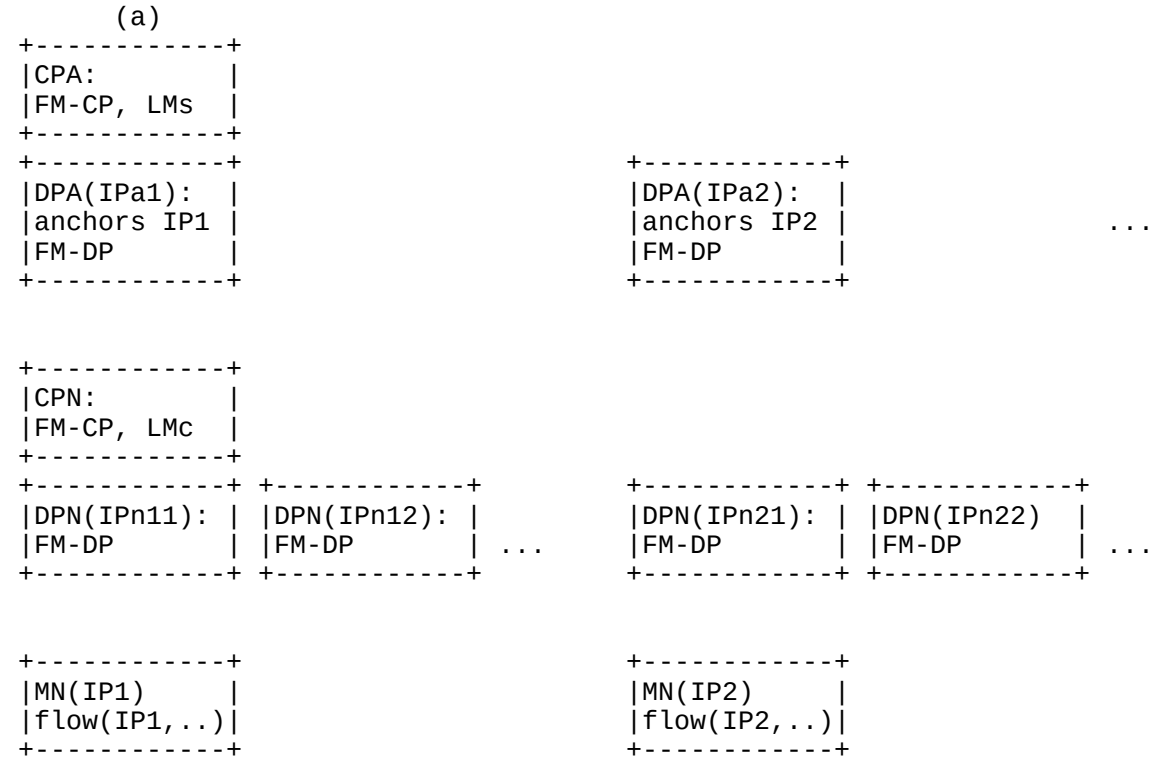


Figure 2(a). Configurations of network-based mobility management for a hierarchical network with FM-CP and LMs at CPA, FM-DP at DPA; FM-CP and LMc at CPN, FM-DP at DPN.

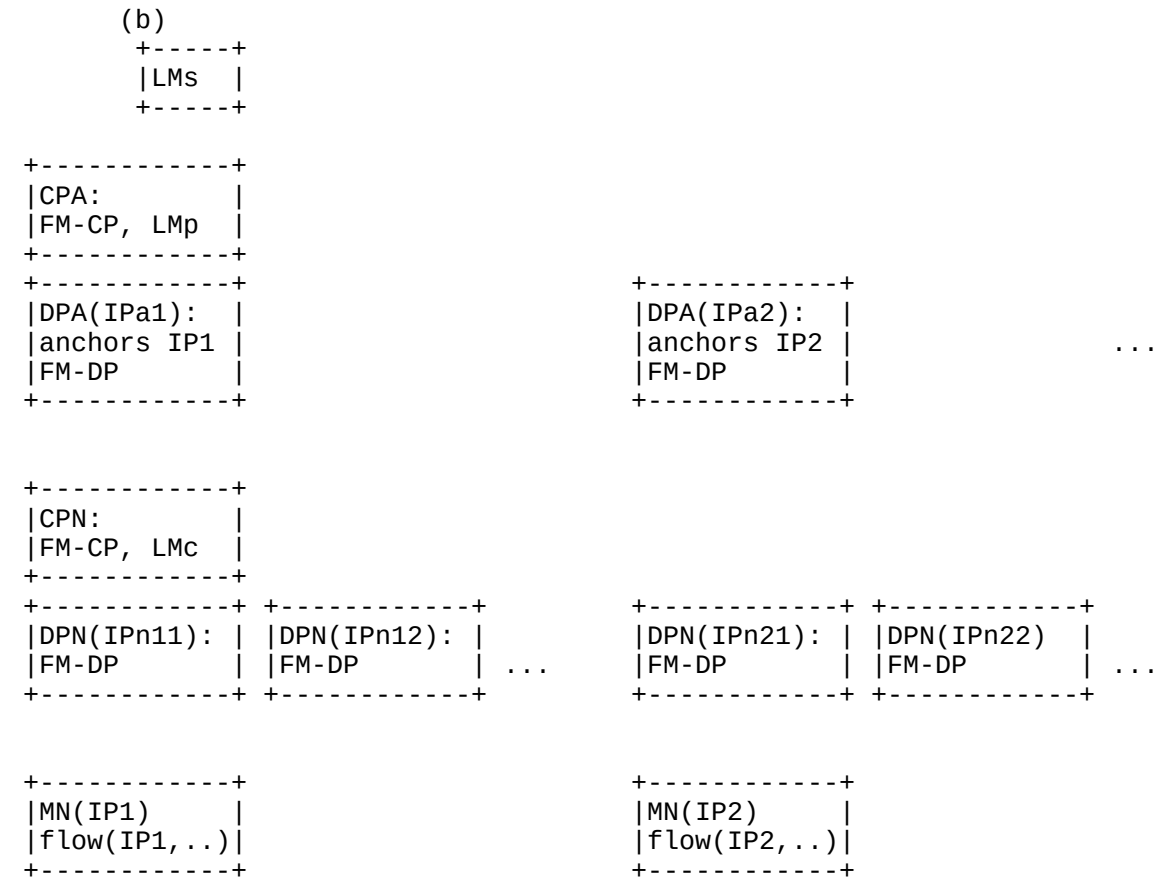


Figure 2(b). Configurations of network-based mobility management for a hierarchical network with separate LMs, FM-CP and LMp at CPA, FM-DP at DPA; FM-CP and LMc at CPN, FM-DP at DPN.

Figures 2 also shows a distributed mobility anchoring environment with multiple instances of the DPA.

In the hierarchy, there may be multiple DPN’s for each DPA.

There is FM-DP at each of the distributed DPA and at each of the distributed DPN.

The control plane may either be distributed (not shown) or centralized.

When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

When the CPN co-locates with the distributed DPN there will be multiple instances of the co-located CPN and DPN (not shown).

There is FM-CP function at the CPA and at the CPN.

MN is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1. It is using IP1 to communicate with a correspondent node (CN) not shown in the figure. The flow of this communication session is shown as flow(IP1, ...) which uses IP1 and other parameters.

In Figure 2(a), LMs and FM-CP are at the CPA. In addition, there are FM-CP and LMc at the CPN.

LMs may be distributed or centralized according to whether the CPA is distributed or centralized. The CPA may co-locate with DPA or may separate.

Figure 2(b) differs from Figure 2(a) in that the LMs is separated out, and a proxy LMp is added between the LMs and LMc.

LMp and FM-CP co-locate at the CPA.

FM-CP and LMc co-locate at the CPN.

The LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed or centralized.

3.1.3. Host-based Mobility Support

Host-based variants of the mobility function configurations from Figures 2(a) and 2(b) are respectively shown in Figures 3(a) and 3(b) where the role to perform mobility functions by CPN and DPN are now taken by the MN. The MN then needs to possess the mobility functions FM and LMc.

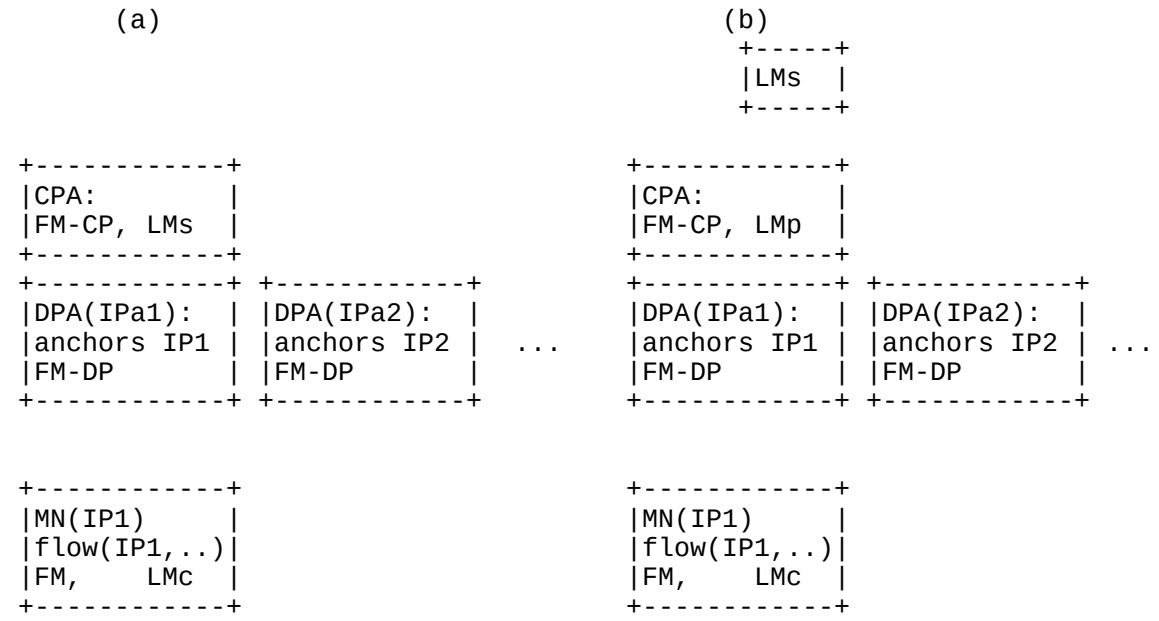


Figure 3. Configurations of host-based mobility management (a) FM-CP and LMs at CPA, FM-DP at DPA, FM and LMc at MN; (b) Separate LMs, FM-CP and LMp at CPA, FM-DP at DPA, FM and LMc at MN.

Figure 3 shows 2 configurations of host-based mobility management with multiple instances of DPA for a distributed mobility anchoring environment.

There is an FM-DP function at each of the distributed DPA.

The control plane may either be distributed (not shown) or centralized.

When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

There is an FM-CP function at the CPA.

The MN possesses the mobility functions such as FM and LMc.

The MN is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1. It is using IP1 to communicate with a CN not shown in the figure. The flow of this communication session is shown as flow(IP1, ...) which uses IP1 and other parameters.

In Figure 3(a), LMs and FM-CP co-locate at the CPA.

The LMs may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

Figure 3(b) differs from Figure 3(a) in that the LMs is separated out and the proxy LMp is added between the LMs and LMc.

LMp and FM-CP co-locate at the CPA.

The LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

3.1.4. NEtwork MObility (NEMO) Basic Support

Figure 4 shows two configurations of NEMO basic support for a mobile router.

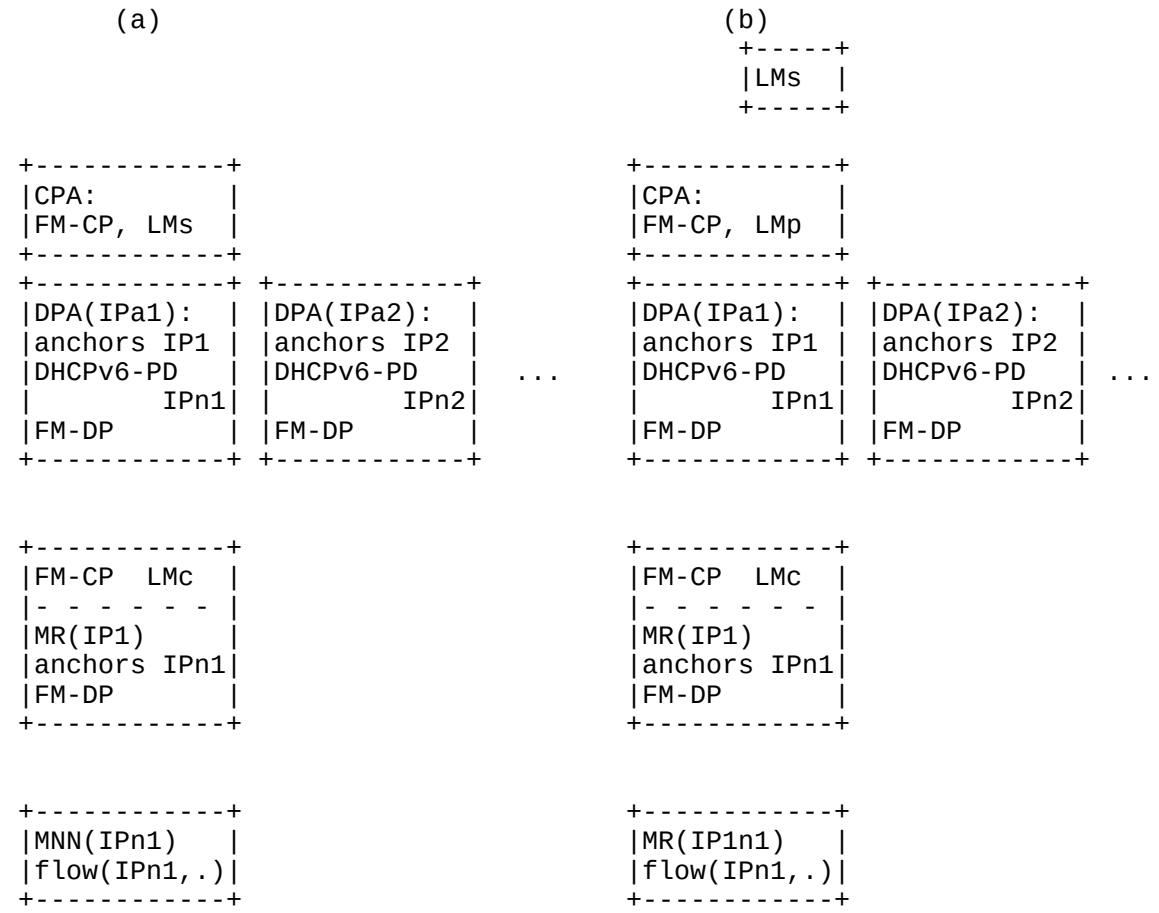


Figure 4. Configurations of NEMO basic support for a MR. (a) FM-CP and LMs at CPA, FM-DP at DPA, FM and LMc at MR; (b) Separate LMs, FM-CP and LMp at CPA, FM-DP at DPA, FM and LMc at MR.

Figure 4 shows 2 configurations of host-based mobility management for a MR with multiple instances of DPA for a distributed mobility anchoring environment.

There is an FM-DP function at each of the distributed DPA.

The control plane may either be distributed (not shown) or centralized.

When the CPA co-locates with the distributed DPA there will be multiple instances of the co-located CPA and DPA (not shown).

There is FM-CP function at the CPA.

The MR possesses the mobility functions FM and LMc.

MR is allocated an IP prefix/address IP1 which is anchored to the DPA with the IP prefix/address IPa1.

A mobile network node (MNN) in the mobile network is allocated an IP prefix/address IPn1 which is anchored to the MR with the IP prefix/address IP1.

The MNN is using IPn1 to communicate with a correspondent node (CN) not shown in the figure. The flow of this communication session is shown as flow(IPn1, ...) which uses IPn1 and other parameters.

In Figure 4(a), LMs and FM-CP co-locate at the CPA.

The LMs may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

Figure 4(b) differs from Figure 4(a) in that the LMs is separated out and the proxy LMp is added between the LMs and LMc.

LMp and FM-CP co-locate at the CPA.

The LMs may be centralized whereas the LMp may be distributed or centralized according to whether the CPA is distributed (not shown) or centralized.

3.2. Operations and Parameters

The operations of distributed mobility anchoring are defined in order that they may work together in expected manners to produce a distributed mobility solution. The needed information is passed as mobility message parameters, which must be protected in terms of integrity. Some parameters may require a means to support privacy of an MN or MR.

The mobility needs in 5G Wireless and beyond are diverse. Therefore operations needed to enable different distributed mobility solutions in different distributed mobility anchoring configurations are extensive as illustrated below. It is however not necessary for every distributed mobility solution to exhibit all the operations listed in this section. A given distributed mobility solution may exhibit the operations as needed.

3.2.1. Location Management

An example LM design consists of a distributed database with multiple LMs servers. The location information about the prefix/address of an MN is primarily at a given LMs. Peer LMs may exchange the location information with each other. LMc may retrieve a given record or send a given record update to LMs.

Location management configurations:

LM-cfg: As shown in Section 3.1:

LMs may be implemented at CPA, may co-locate with LMc at CPA, or may be a separate server.

LMc may be at CPA, CPN, or MN.

LMp may proxy between LMs and LMc.

Specifically:

Location management operations and parameters:

LM-cfg:1 LMs may co-locate with LMc at CPA in a flat network with network-based mobility as shown in Figure 1(a) in Section 3.1.1.

LM-cfg:2 LMs may be a separate server whereas LMc is implemented in CPA in a flat network with network-based mobility as shown in Figure 1(b) in Section 3.1.1.

LM-cfg:3 LMs may be implemented at CPA, whereas LMc is implemented at CPN in a hierarchical network with network-based mobility as shown in Figure 2(a) in Section 3.1.2 or at MN for host-based mobility as shown in Figure 3(a) in Section 3.1.3.

LM-cfg:4 LMs may be a separate server with LMp implemented at CPA whereas LMc is implemented at CPN in a hierarchical network with network-based mobility as shown in Figure 2(b) in Section 3.1.2 or at MN for host-based mobility as shown in Figure 3(b) in Section 3.1.3.

LM-db: LM may manage the location information in a client-server database system.

Example LM database functions are as follows:

LM-db:1 LMc may query LMs about location information for a prefix of MN (pull).
Parameters:
- IP prefix of MN: integrity support required and privacy support may be required.

LM-db:2 LMs may reply to LMc query about location information for a prefix of MN (pull).
Parameters:
- IP prefix of MN: integrity support required and privacy support may be required
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required.

LM-db:3 LMs may inform LMc about location information for a prefix of MN (push).
Parameters:
- IP prefix of MN: integrity support required and privacy support may be required
- IP address of FM-DP/DPA/DPN to forward the packets of the flow.

This function in the PMIPv6 protocol is the Update Notification (UPN) together with the Update Notification Acknowledgment (UPA) as defined in [RFC7077].

LM-db:4 LMc may inform LMs about update location information for a prefix of MN.
Parameters:
- IP prefix of MN: integrity support required and privacy support may be required
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required

This function in the MIPv6 / PMIPv6 protocol is the Binding Update (BU) / Proxy Binding Update (PBU) together with the Binding Acknowledgment (BA) / Proxy Binding Acknowledgment (PBA) as defined in [RFC6275] / [RFC5213] respectively.

LM-db:5 The MN may be a host or a router. When the MN is an MR, the prefix information may include the MNP delegated to the MR.
Additional parameters:
MNP: integrity support required and privacy support may be required

LM-svr: The LM may be a distributed database with multiple LMs servers.

For example:

LM-svr:1 A LMs may join a pool of LMs servers.
Parameters:
- IP address of the LMs: integrity support required
- IP prefixes for which the LMs will host the primary location information: integrity support required.

LM-svr:2 LMs may query a peer LMs about location information for a prefix of MN.
Parameters:
- IP prefix: integrity support required and privacy support may be required.

LM-svr:3 LMs may reply to a peer LMs about location information for a prefix of MN.
Parameters:
- IP prefix of MN: integrity support required and privacy support may be required
- IP address of FM-DP/DPA/DPN to forward the packets of the flow: integrity support required.

The parameters indicated above are only the minimal. In a specific mobility protocol, additional parameters should be added as needed. Examples of these additional parameters are those passed in the mobility options of the mobility header for MIPv6 [RFC6275] and for PMIPv6 [RFC5213].

3.2.2. Forwarding Management

Forwarding management configurations:

FM-cfg: As shown in Section 3.1:

FM-CP may be implemented at CPA, CPN, MN depending on the configuration chosen.

FM-DP may also be implemented at CPA, CPN, MN depending on the configuration chosen.

Specifically:

FM-cfg:1 FM-CP and FM-DP may be implemented at CPA and DPA respectively in a flat network with network-based mobility as shown in Figure 1(a) and Figure 1(b) in Section 3.1.1.

FM-cfg:2 FM-CP may be implemented at both CPA and CPN and FM-DP is implemented at both DPA and DPN in a hierarchical network

with network-based mobility as shown in Figure 2(a) and Figure 2(b) in Section 3.1.2.

FM-cfg:3 FM-CP and FM-DP may be implemented at CPA and DPA respectively and also both implemented at MN for host-based mobility as shown in Figure 3(a) and Figure 3(b) in Section 3.1.3.

Forwarding management operations and parameters:

FM-find:1 An anchor may discover and be discovered such as through an anchor registration system as follows:

FM-find:2 FM registers and authenticates itself with a centralized mobility controller.
Parameters:
- IP address of DPA and its CPA: integrity support required
- IP prefix anchored to the DPA: integrity support required

registration reply: acknowledge of registration and echo the input parameters.

FM-find:3 FM discovers the FM of another IP prefix by querying the mobility controller based on the IP prefix.
Parameters:
- IP prefix of MN: integrity support required and privacy support may be required

FM-find:4 when making anchor discovery FM expects the answer parameters:
- IP address of DPA to which IP prefix of MN is anchored: integrity support required
- IP prefix of the corresponding CPA: integrity support required

FM-flow:1 The FM may be carried out on the packets to/from an MN up to the granularity of a flow.

FM-flow:2 Example matching parameters are in the 5-tuple of a flow.

FM-cpdp: With separation of control plane function and data plane function, FM-CP and FM-DP communicate with each other. Such communication may be realized by the appropriate messages in [I-D.ietf-dmm-fpc-cpdp].

For example:

FM-cpdp:1 CPA/FM-CP sends forwarding table updates to DPA/FM-DP. Parameters:
- New forwarding table entries to add: integrity support required
- Expired forwarding table entries to delete: integrity support required

FM-cpdp:2 DPA/FM-DP sends to CPA/FM-CP about its status and load. Parameters:
- State of forwarding function being active or not: integrity support required
- Loading percentage: integrity support required

FM-path:1 FM may change the forwarding path of a flow upon a change of point of attachment of a MN. Prior to the changes, packets coming from the CN to the MN would traverse from the CN to the home network anchor of the flow for the MN before reaching the MN. Changes are from this original forwarding path or paths to a new forwarding path or paths from the CN to the current AR of the MN and then the MN itself.

FM-path:2 As an incoming packet is forwarded from the CN to the MN, the far end where forwarding path change begins may in general be any node in the original forwarding path from the CN to the home network DPA. The packet is forwarded to the MN for host-based mobility and to a node in the network which will deliver the packets to the MN for network-based mobility. The near-end is generally a DPN with a hierarchical network but may also be another node with DPA capability in a flattened network.

FM-path:3 The mechanisms to accomplish such changes may include changes to the forwarding table and indirection such as tunneling, rewriting packet header, or NAT.

Note: An emphasis in this document in distributed mobility anchoring is to explain the use of multiple anchors to avoid unnecessarily long route which may be encountered in centralized mobility anchoring. It is therefore not the emphasis of this document on which particular mechanism to choose from.

FM-path-tbl:4 With forwarding table updates, changes to the forwarding table are needed at each of the affected forwarding switches in order to change the forwarding path of the packets for the flow from that originally between the CN and the home network anchor to that between the CN and the new AR.

Forwarding table updates may be achieved through BGP update as described in [I-D.templin-aerolink], [I-D.mccann-dmm-flatarch] and also for 3GPP Evolved Packet Core (EPC) network in [I-D.matsushima-stateless-uplane-vepc] when the scope and response time can be managed. Alternatively, a centralized control plane may be used.

When the control plane is centralized, forwarding table updates may be achieved through messaging between the centralized control plane and the distributed forwarding switches as described above (FM-cpdp) in this section.

Forwarding table updates may be triggered using DHCPv6-PD prefix delegation to change the role of IP anchoring from the home network anchor (with FM-DP) to the new anchor (with FM-DP) to which the MN is currently attached. The new anchor will then advertise routes for the delegated prefix.

With a distributed routing protocol, the updates spread out from neighbors to neighbors and will affect all the forwarding switches such that the packets sent from "any" node to MN will go to the new AR.

Yet the scope of such updates for a given flow may be confined to only those forwarding switches such that the packets sent only from the "CN" to MN will go to the new AR. Such confinement may be made when using a centralized central plane possessing a global view of all the forwarding switches.

FM-path-tbl:5 FM reverts the changes previously made to the forwarding path of a flow when such changes are no longer needed, e.g., when all the ongoing flows using an IP prefix/address requiring IP session continuity have closed. When using DHCPv6-PD, the forwarding paths will be reverted upon prefix lease expiration.

FM-path-ind:6 Indirection forwards the incoming packets of the flow from the DPA at the far end to a DPA/DPN at the near end of indirection. Both ends of the indirection needs to know the LM information of the MN for the flow and also needs to possess FM capability to perform indirection.

FM-path-ind:7 The mechanism of changing the forwarding path in [RFC6275] and [RFC5213] is tunneling. In the control plane, the FM-CP sets up the tunnel by instructing the FM-DP at both ends of the tunnel. In the data plane, the FM-DP at the start of the tunnel performs packet encapsulation, whereas the FM-DP at the end of the tunnel decapsulates the packet.

Note that in principle the ends of the indirection path can be any pair of network elements with the FM-DP function.

FM-path-ind:8 FM reverts the changes previously made to the forwarding path of a flow when such changes are no longer needed, e.g., when all the ongoing flows using an IP prefix/address requiring IP session continuity have closed. When tunneling is used, the tunnels will be torn down when they are no longer needed.

FM-DPA:1 Recall from above that for the incoming packets from the CN, forwarding path change by FM is from the DPA at the far end which may be at any forwarding switch (or even CN itself) in the original forwarding path to the near end DPA/DPN.

It is necessary that any incoming packet from the CN of the flow must traverse the DPA (or at least one of the DPAs, e.g., in the case of anycast) at the far end in order for the packet to detour to a new forwarding path.

Therefore a convenient design is to locate the far end DPA at a unique location which is always in the forwarding path. This is the case in a centralized mobility design where the DPA at the far end is the home network anchor of the flow.

Distributed mobility however may place the far end DPA at other locations in order to avoid unnecessarily long route.

FM-DPA:2 With multiple nodes possessing DPA capabilities, the role of FM to begin path change for the incoming packets of a flow at the home network DPA at the far end may be passed to or added to that of another DPA.

In particular, this DPA role may be moved upstream from the home network DPA in the original forwarding path from CN to MN.

FM-DPA:3 Optimization of the new forwarding path may be achieved when the path change for the incoming packets begins at a DPA where the original path and the direct IPv6 path overlaps. Then the new forwarding path will resemble the direct IPv6 path from the CN to the MN.

FM-DPA-tbl:4 Forwarding table updates, such as that triggered using DHCPv6-PD to change the role of IP anchoring from the home network anchor (DPA with FM-DP) to the new anchor (DPA with FM-DP), may put the near end of the path change at the new DPA. Subsequent forwarding table updates may propagates upstream up to a far end where the original path and the direct IPv6 path overlaps.

When that far end is too far upstream the signaling of forwarding table updates may become excessive. An alternative is to use indirection (see FM-DPA-ind) from that far end to the new DPA at the near end.

Still another alternative is to combine forwarding table update with indirection.

FM-DPA-tbl:5 Changes made by FM to the following tables, which are IPv6 nodes, at the ends of the path change for a flow will be reverted when the mobility support for the flow is no longer needed, e.g., when the flows have terminated.

FM-DPA-ind:6 With indirection, locating or moving the FM function to begin indirection upstream along the forwarding path from CN to MN again may help to reduce unnecessarily long path.

FM-DPA-ind:7 Changes made by FM to establish indirection at the DPA and DPN, which are IPv6 nodes, at the ends of the path change for a flow will be reverted when the mobility

support for the flow is no longer needed, e.g., when the flows have terminated.

FM-state:1 In addition to the above, a flow/session may contain states with the required information for QoS, charging, etc. as needed. These states need to be transferred from the old anchor to the new anchor.

FM-buffer:1 An anchor can buffer packets of a flow in a mobility event:

FM-buffer:2 CPA/FM-CP informs DPA/FM-DP to buffer packets of a flow.
Trigger:
- MN leaves DPA in a mobility event.
Parameters:
- IP prefix of the flow for which packets need to be buffered: integrity support required

FM-buffer:3 CPA/FM-CP on behalf of a new DPA/FM-DP informs the CPA/FM-CP of the prior DPA/FM-DP that it is ready to receive any buffered packets of a flow.
Parameters:
- Destination IP prefix of the flow’s packets: integrity support required
- IP address of the new DPA: integrity support required

FM-mr:1 When the MN is a mobile router the access router anchoring the IP prefix of MR will also anchor the IP prefix or prefixes delegated to the MR.

4. IP Mobility Handling in Distributed Anchoring Environments - Mobility Support Only When Needed

IP Mobility Support Only When Needed:

IP mobility support may be provided only when needed instead of being provided by default. The LM and FM functions in the different configurations shown in Section 3.1 are then utilized only when needed.

A straightforward choice of mobility anchoring is for a flow to use the IP prefix of the network to which the MN is attached when the flow is initiated [I-D.seite-dmm-dma].

The IP prefix/address at the MN’s side of a flow may be anchored at the access router to which the MN is attached. For example, when an

MN attaches to a network (Net1) or moves to a new network (Net2), it is allocated an IP prefix from the attached network. In addition to configuring new link-local addresses, the MN configures from this prefix an IP address which is typically a dynamic IP address. It then uses this IP address when a flow is initiated. Packets to the MN in this flow are simply forwarded according to the forwarding table.

There may be multiple IP prefixes/addresses that an MN can select when initiating a flow. They may be from the same access network or different access networks. The network may advertise these prefixes with cost options [I-D.mccann-dmm-prefixcost] so that the mobile node may choose the one with the least cost. In addition, these IP prefixes/addresses may be of different types regarding whether mobility support is needed [I-D.ietf-dmm-ondemand-mobility]. A flow will need to choose the appropriate one according to whether it needs IP mobility support.

4.1. No Need of IP Mobility: Changing to New IP Prefix/Address

When IP mobility support is not needed for a flow, the LM and FM functions are not utilized so that the configurations in Section 3.1 are simplified as shown in Figure 5.

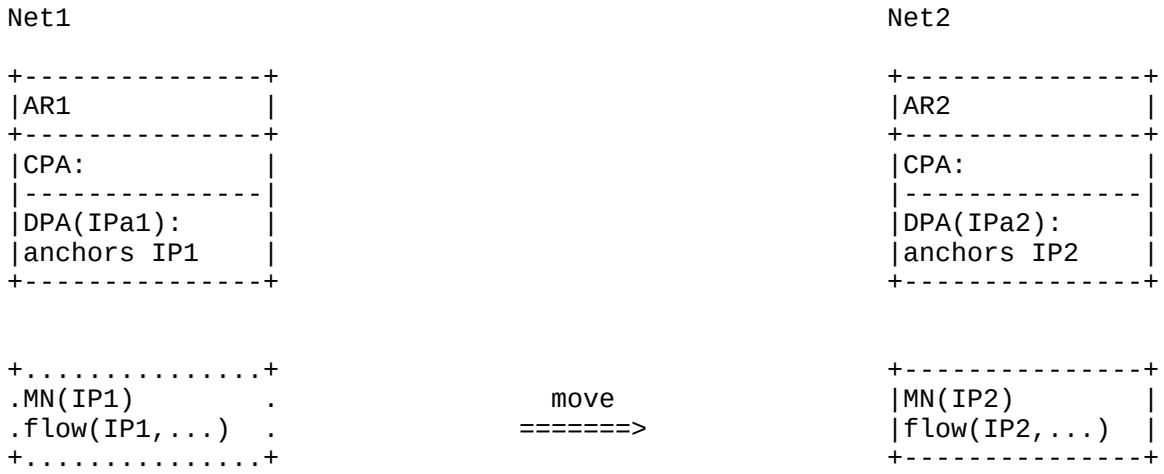


Figure 5. Changing to the new IP prefix/address. MN running a flow using IP1 in a network Net1 changes to running a flow using IP2 in Net2.

When there is no need to provide IP mobility to a flow, the flow may use a new IP address acquired from a new network as the MN moves to the new network.

Regardless of whether IP mobility is needed, if the flow has terminated before the MN moves to a new network, the flow may subsequently restart using the new IP address allocated from the new network.

When IP session continuity is needed, even if a flow is ongoing as the MN moves, it may still be desirable for the flow to change to using the new IP prefix configured in the new network. The flow may then close and then restart using a new IP address configured in the new network. Such a change in the IP address of the flow may be enabled using a higher layer mobility support which is not in the scope of this document.

In Figure 5, a flow initiated while the MN was in a network Net1 has terminated before the MN moves to a new network Net2. After moving to Net2, the MN uses the new IP prefix anchored in Net2 to start a new flow. The packets may then be forwarded without requiring IP layer mobility support.

An example call flow is outlined in Figure 6.

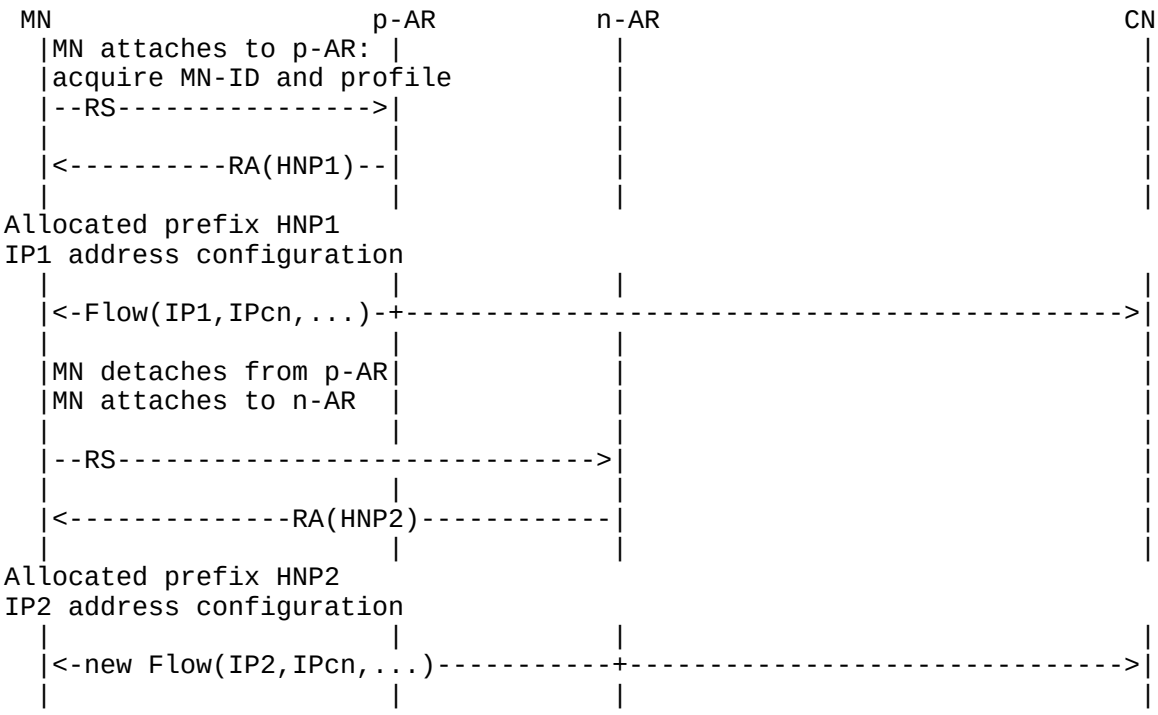


Figure 6. Re-starting a flow to use the IP allocated from the network at which the MN is attached.

4.1.1. Guidelines for IPv6 Nodes: Changing to New IP Prefix/Address

A network or network slice may not need IP mobility support. For example, a network slice for stationary sensors only will never encounter mobility.

The standard functions in IPv6 already include dropping the old IPv6 prefix/address and acquiring new IPv6 prefix/address when the node changes its point of attachment to a new network. Therefore, a network or network slice not providing IP mobility support at all will not need any of the functions with the mobility operations and messages described in Section 3.2.

The guidelines for the IPv6 nodes in a network or network slice supporting a mix of flows requiring and not requiring IP mobility support include the following:

GL-cfg:1 A network or network slice supporting a mix of flows requiring and not requiring mobility support may take any

of the configurations described in Section 3.1 and need to implement in the appropriate IPv6 nodes the mobility functions LM and FM as described respectively in LM-cfg and FM-cfg in Section 3.2 according to the configuration chosen.

GL-mix:1 These mobility functions perform some of the operations with the appropriate messages as described in Section 3.2 depending on which mobility mechanisms are used. Yet these mobility functions must not be invoked for a flow that does not need IP mobility support. It is necessary to be able to distinguish the needs of a flow. The guidelines for the MN and the AR are in the following.

GL-mix:2 Regardless of whether there are flows requiring IP mobility support, when the MN changes its point of attachment to a new network, it needs to configure a new global IP address for use in the new network in addition to configuring the new link-local addresses.

GL-mix:3 The MN needs to check whether a flow needs IP mobility support. This can be performed when the application was initiated. The specific method is not in the scope of this document.

GL-mix:4 The information of whether a flow needs IP mobility support is conveyed to the network such as by choosing an IP address to be provided with mobility support as described in [I-D.ietf-dmm-ondemand-mobility]. Then as the MN attaches to a new network, if the MN was using an IP address that is not supposed to be provided with mobility support, the access router will not invoke the mobility functions described in Section 3.2 for this IP address. That is, the IP address from the prior network is simply not used in the new network.

The above guidelines are only to enable distinguishing whether there is need of IP mobility support for a flow that does not. When the flow needs IP mobility support, the list of guidelines will continue in Section 4.2.1.

4.2. Need of IP Mobility

When IP mobility is needed for a flow, the LM and FM functions in Section 3.1 are utilized. The mobility support may be provided by IP prefix anchor switching to the new network to be described in Section 5 or by using other mobility management methods

([Paper-Distributed.Mobility.PMIP] and [Paper-Distributed.Mobility.Review]). Then the flow may continue to use the IP prefix from the prior network of attachment. Yet some time later, the user application for the flow may be closed. If the application is started again, the new flow may not need to use the prior network's IP address to avoid having to invoke IP mobility support. This may be the case where a dynamic IP prefix/address rather than a permanent one is used. The flow may then use the new IP prefix in the network where the flow is being initiated. Routing is again kept simpler without employing IP mobility and will remain so as long as the MN which is now in the new network has not moved again and left to another new network.

An example call flow in this case is outlined in Figure 7.

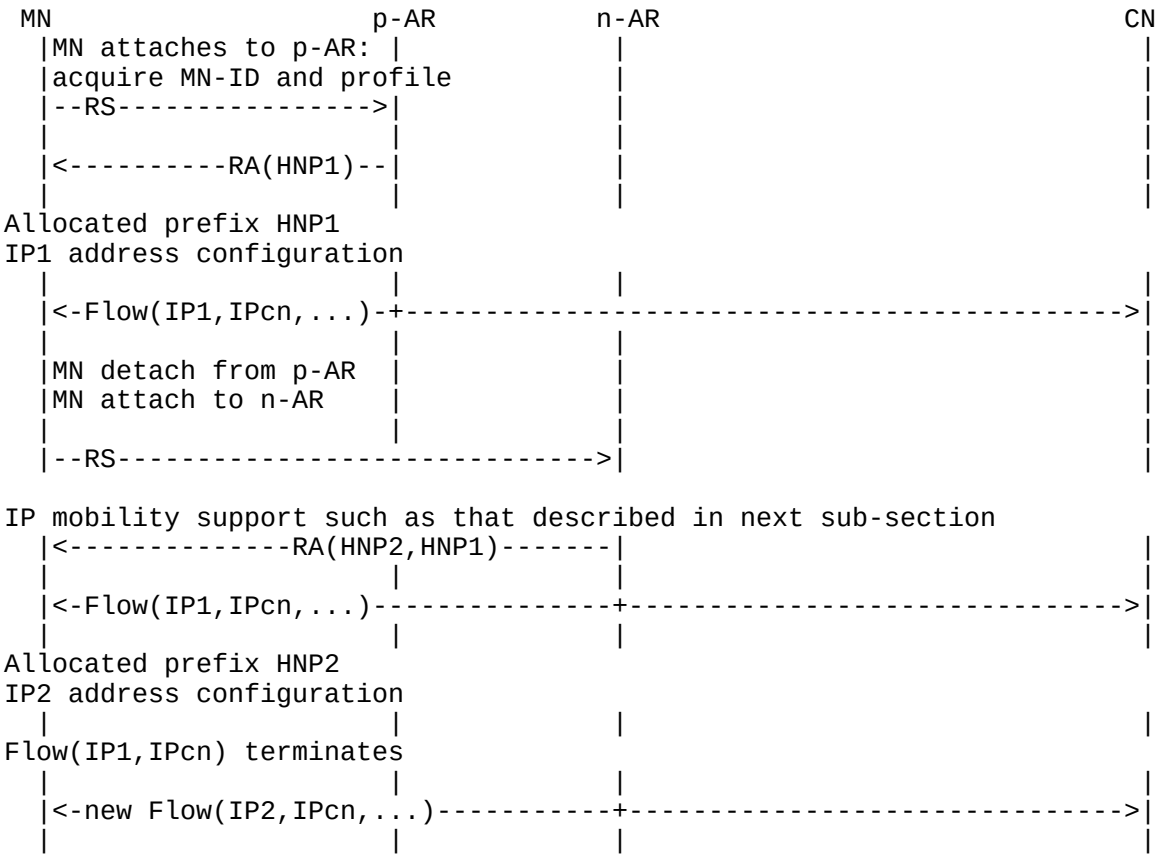


Figure 7. A flow continues to use the IP from its home network after MN has moved to a new network.

4.2.1. Guidelines for IPv6 Nodes: Need of IP Mobility

The configuration guidelines of distributed mobility for the IPv6 nodes in a network or network slice supporting a mix of flows requiring and not requiring distributed mobility support are as follows:

GL-cfg:2 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring in an appropriate configuration such as those in Figure 1 (Section 3.1.1) for network-based distributed mobility or in Figure 3 (Section 3.1.3) for host-based distributed mobility.

The appropriate IPv6 nodes (CPA, DPA, CPN, DPN) are to be implemented the mobility functions LM and FM as described respectively in LM-cfg and FM-cfg in Section 3.2 according to the configuration chosen.

The guidelines of distributed mobility for the IPv6 nodes in a network or network slice supporting a mix of flows requiring and not requiring distributed mobility support had begun with those given as GL-mix in Section 4.1.1 and continue as follows:

GL-mix:5 The distributed anchors may need to message with each other. When such messaging is needed, the anchors may need to discover each other as described in the FM operations and mobility message parameters (FM-find) in Section 3.2.2.

GL-mix:6 The anchors may need to provide mobility support on a per-flow basis as described in the FM operations and mobility message parameters (FM-flow) in Section 3.2.2.

GL-mix:7 Then the anchors need to properly forward the packets of the flows as described in the FM operations and mobility message parameters (FM-path, FM-path-tbl, FM-DPA, FM-DPA-tbl) in Section 3.2.2.

GL-mix:8 If there are in-flight packets toward the old anchor while the MN is moving to the new anchor, it may be necessary to buffer these packets and then forward to the new anchor after the old anchor knows that the new anchor is ready. Such are described in the FM operations and mobility message parameters (FM-buffer) in Section 3.2.2.

5. IP Mobility Handling in Distributed Mobility Anchoring Environments - Anchor Switching to the New Network

IP Prefix/Address Anchor Switching to the New Network:

IP mobility is invoked to enable IP session continuity for an ongoing flow as the MN moves to a new network. Here the anchoring of the IP address of the flow is in the home network of the flow, which is not in the current network of attachment. A centralized mobility management mechanism may employ indirection from the anchor in the home network to the current network of attachment. Yet it may be difficult to avoid unnecessarily long route when the route between the MN and the CN via the anchor in the home network is significantly longer than the direct route between them. An alternative is to switch the IP prefix/address anchoring to the new network.

5.1. IP Prefix/Address Anchor Switching for Flat Network

The IP prefix/address anchoring may move without changing the IP prefix/address of the flow. Here the LM and FM functions in Figures 1(a) and 1(b) in Section 3.1 are implemented as shown in Figure 8.

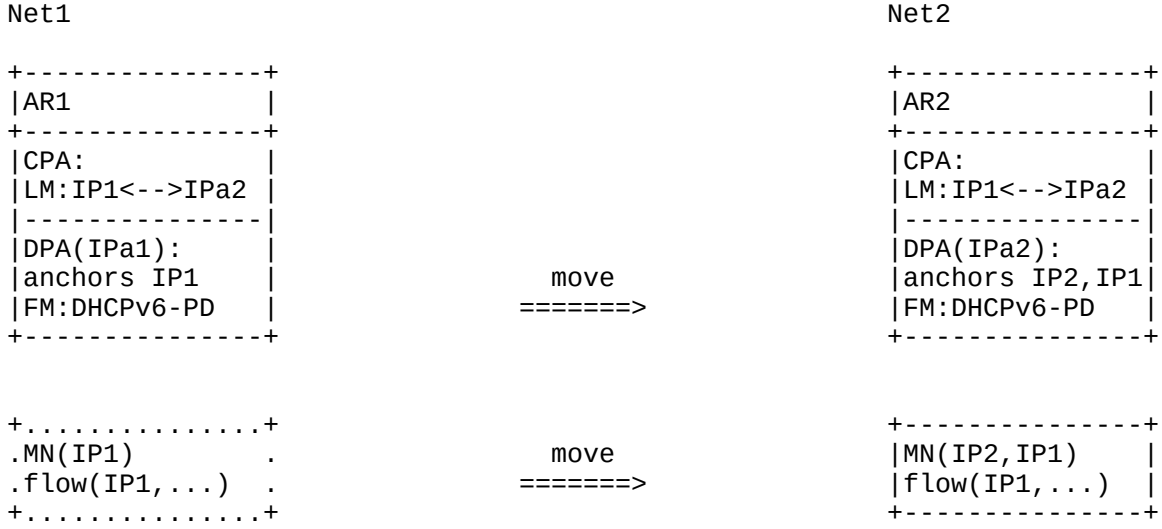


Figure 8. IP prefix/address anchor switching to the new network. MN with flow using IP1 in Net1 continues to run the flow using IP1 as it moves to Net2.

As an MN with an ongoing session moves to a new network, the flow may preserve IP session continuity by moving the anchoring of the original IP prefix/address of the flow to the new network. BGP

UPDATE messages may be used to change the forwarding table entries as described in [I-D.templin-aerolink] and [I-D.mccann-dmm-flatarch] if the response time of such updates does not exceed the handover delay requirement of the flow. An alternative is to use a centralized routing protocol to be described in Section 5.2 with a centralized control plane.

5.1.1. Guidelines for IPv6 Nodes: Switching Anchor for Flat Network

The configuration guideline for a flat network or network slice supporting a mix of flows requiring and not requiring IP mobility support is:

GL-cfg:3 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring according to Figure 1(a) or Figure 1(b)in Section 3.1 for a flat network.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented the mobility functions LM and FM as described respectively in LM-cfg:1 or LM-cfg:2 and FM-cfg:1 in Section 3.2.

The guidelines (GL-mix) in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network or network slice supporting a mix of flows requiring and not requiring IP mobility support apply here. In addition, the following are required.

GL-switch:1 The location management provides information about which IP prefix from an AR in the original network is being used by a flow in which AR in a new network. Such information needs to be deleted or updated when such flows have closed so that the IP prefix is no longer used in a different network. The LM operations are described in Section 3.2.1.

GL-switch:2 The FM functions are implemented through the DHCPv6-PD protocol. Here the anchor operations to properly forward the packets for a flow as described in the FM operations and mobility message parameters in Section 3.2.2 FM-path, FM-path-tbl, FM-DPA, FM-DPA-tbl are realized by changing the anchor with DHCPv6-PD and also by reverting such changes later after the application has already closed and when the DHCPv6-PD timer expires. If there are in-flight packets toward the old anchor while the MN is moving to the new anchor, it may be necessary to buffer these packets and then

forward to the new anchor after the old anchor knows that the new anchor is ready as are described in Section 3.2.2 (FM-buffer). The anchors may also need to discover each other as described also in the FM operations and mobility message parameters (FM-find).

GL-switch:3 The security management function in the anchor node at a new network must allow to assign the original IP prefix/address used by the mobile node at the previous (original) network. As the assigned original IP prefix/address is to be used in the new network, the security management function in the anchor node must allow to advertise the prefix of the original IP address and also allow the mobile node to send and receive data packets with the original IP address.

GL-switch:4 The security management function in the mobile node must allow to configure the original IP prefix/address used at the previous (original) network when the original IP prefix/address is assigned by the anchor node in the new network. The security management function in the mobile node also allows to use the original IP address for the previous flow in the new network.

5.2. IP Prefix/Address Anchor Switching for Flat Network with Centralized Control Plane

An example of IP prefix anchor switching is in the case where Net1 and Net2 both belong to the same operator network with separation of control and data planes ([I-D.liu-dmm-deployment-scenario] and [I-D.matsushima-stateless-uplane-vepc]), where the controller may send to the switches/routers the updated information of the forwarding tables with the IP address anchoring of the original IP prefix/address at AR1 moved to AR2 in the new network. That is, the IP address anchoring in the original network which was advertising the prefix will need to move to the new network. As the anchoring in the new network advertises the prefix of the original IP address in the new network, the forwarding tables will be updated so that packets of the flow will be forwarded according to the updated forwarding tables. The configurations in Figures 1(a) and 1(b) in Section 3.1 for which FM-CP and LM are centralized and FM-DP's are distributed apply here. Figure 9 shows its implementation where LM is a binding between the original IP prefix/address of the flow and the IP address of the new DPA, whereas FM uses the DHCPv6-PD protocol.

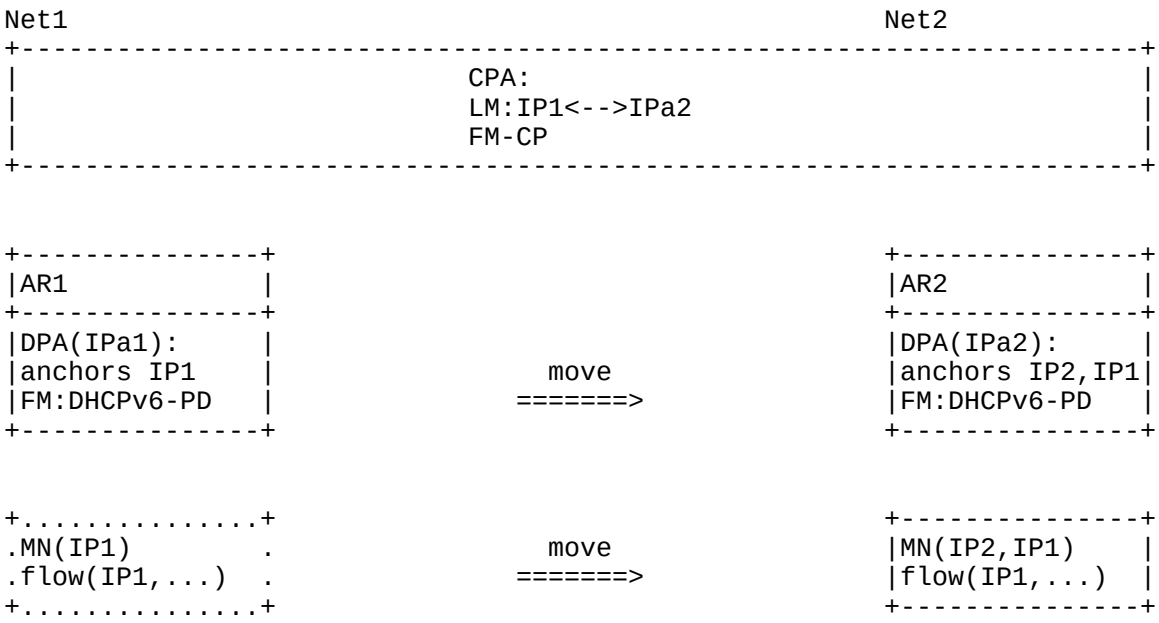


Figure 9. IP prefix/address anchor switching to the new network with with LM and FM-CP in a centralized control plane whereas the FM-DP's are distributed.

The example call flow in Figure 10 shows that MN is allocated HNP1 when it attaches to the p-AR. A flow running in MN and needing IP mobility may continue to use the previous IP prefix by moving the anchoring of the IP prefix to the new network. Yet a new flow to be initiated in the new network may simply use a new IP prefix allocated from the new network.

Figure 10. DMM solution. MN with flow using IP1 in Net1 continues to run the flow using IP1 as it moves to Net2.

As the MN moves from p-AR to n-AR, the p-AR as a DHCPv6 client may send a DHCPv6 release message to release the HNP1. It is now necessary for n-AR to learn the IP prefix of the MN from the previous network so that it will be possible for Net2 to allocate both the previous network prefix and the new network prefix. The network may learn the previous prefix in different methods. For example, the MN

may provide its previous network prefix information by including it to the RS message [I-D.jhlee-dmm-dnpp].

Knowing that MN is using HNP1, the n-AR sends to a DHCPv6 server a DHCPv6-PD request to move the HNP1 to n-AR. The server sends to n-AR a DHCPv6-PD reply to move the HNP1. Then forwarding tables updates will take place here.

In addition, the MN also needs a new HNP in the new network. The n-AR may now send RA to n-AR, with prefix information that includes HNP1 and HNP2. The MN may then continue to use IP1. In addition, the MN is allocated the prefix HNP2 with which it may configure its IP addresses. Now for flows using IP1, packets destined to IP1 will be forwarded to the MN via n-AR.

As such flows have terminated and DHCPv6-PD has timed out, HNP1 goes back to Net1. MN will then be left with HNP2 only, which it will use when it now starts a new flow.

5.2.1. Additional Guidelines for IPv6 Nodes: Switching Anchor with Centralized CP

The configuration guideline for a flat network or network slice with centralized control plane and supporting a mix of flows requiring and not requiring IP mobility support is:

GL-cfg:4 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring according to Figure 1(a) or Figure 1(b)in Section 3.1 with centralized control plane for a flat network.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented the mobility functions LM and FM as described respectively in LM-cfg:1 or LM-cfg:2 and FM-cfg:1 in Section 3.2.

The guidelines (GL-mix) in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network or network slice supporting a mix of flows requiring and not requiring IP mobility support apply here. The guidelines (GL-mix) in Section 5.1.1 also apply here. In addition, the following are required.

GL-switch:5 The anchor operations to properly forward the packets for a flow as described in the FM operations and mobility message parameters in Section 3.2.2 FM-path, FM-path-tbl, FM-DPA, FM-DPA-tbl is realized by changing

the anchoring with DHCPv6-PD and undoing such changes later when its timer expires and the application has already closed. With the anchors being separated in control and data planes with LMs and FM-CP centralized in the same control plane, messaging between anchors and the discovery of anchors become internal to the control plane as described in Section 3.2.2 FM-cpdp. However, the centralized FM-CP needs to communicate with the distributed FM-DP as described as described in the FM operations and mobility message parameters (FM-find). Such may be realized by the appropriate messages in [I-D.ietf-dmm-fpc-cpdp].

GL-switch:6 It was already mentioned before that, if there are in-flight packets toward the old anchor while the MN is moving to the new anchor, it may be necessary to buffer these packets and then forward to the new anchor after the old anchor knows that the new anchor is ready Here, however, the corresponding FM operations and mobility message parameters as described in Section 3.2.2 (FM-buffer) can be realized by the internal operations in the control plane together with signaling between the control plane and distributed data plane. These signaling may be realized by the appropriate messages in [I-D.ietf-dmm-fpc-cpdp].

5.3. IP Prefix/Address Anchor Switching for a Hierarchical Network

The configuration for a hierarchical network is shown in Figures 1(c) and 1(d) in Section 3.1. With centralized control plane, CPA and CPN, with the associated LM and FM-CP are all co-located. There are multiple DPAs (each with FM-DP) in distributed mobility anchoring. In the data plane, there are multiple DPNs (each with FM-DP) hierarchically below each DPA. The DPA at each AR supports forwarding to the DPN at each of a number of forwarding switches (FW's). A mobility event in this configuration belonging to distributed mobility management will be deferred to Section 5.4.

In this distributed mobility configuration, a mobility event involving change of FW only but not of AR as shown in Figure 11 may still belong to centralized mobility management and may be supported using PMIPv6. This configuration of network-based mobility is also applicable to host-based mobility with the modification for the MN directly taking the role of DPN and CPN, and the corresponding centralized mobility event may be supported using MIPv6.

In Figure 11, the IP prefix allocated to the MN is anchored at the access router (AR) supporting indirection to the old FW to which the MN was originally attached as well as to the new FW to which the MN has moved.

The realization of LM may be the binding between the IP prefix/ address of the flow used by the MN and the IP address of the DPN to which MN has moved. The implementation of FM to enable change of FW without changing AR may be accomplished using tunneling between the AR and the FW as described in [I-D.korhonen-dmm-local-prefix] and in [I-D.templin-aerolink] or using some other L2 mobility mechanism.

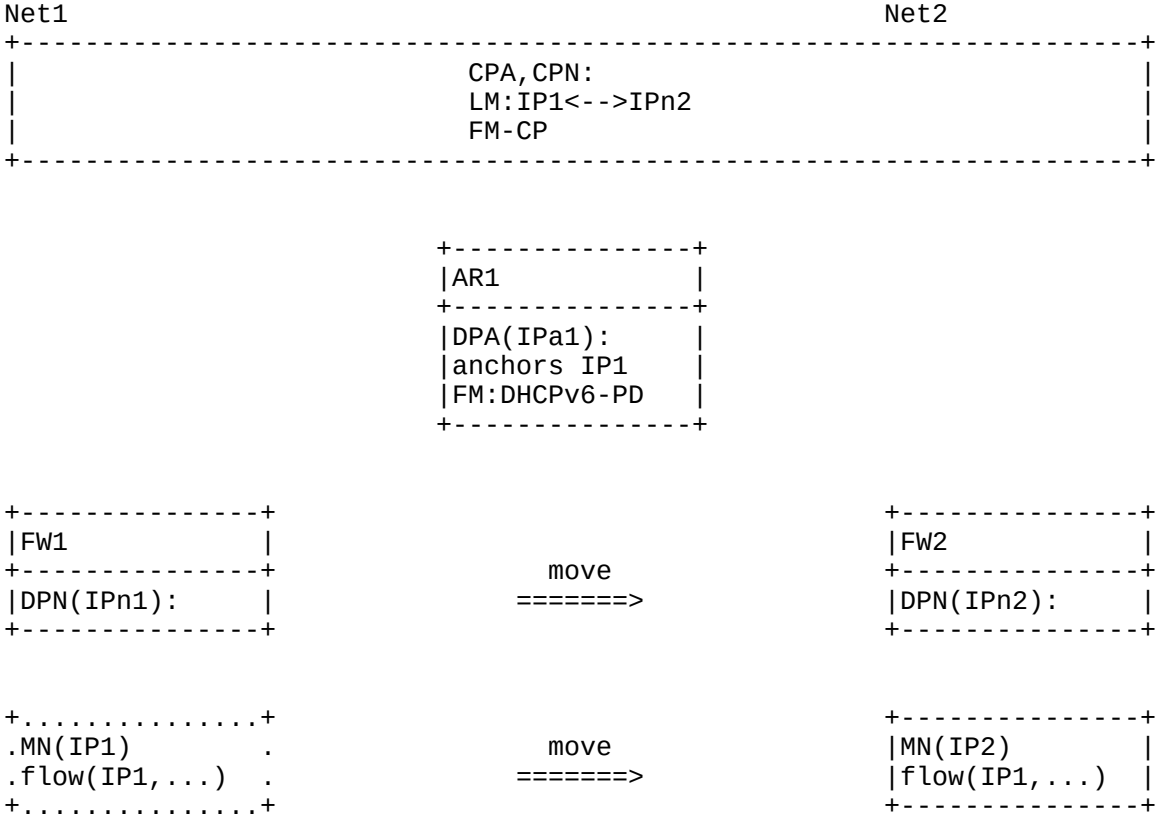


Figure 11. Mobility without involving change of IP anchoring in a network in which the IP prefix allocated to the MN is anchored at an AR which is hierarchically above multiple FWs to which the MN may connect.

5.3.1. Additional Guidelines for IPv6 Nodes: No Anchoring Change with a Hierarchical Network

The configuration guideline ( ) for a hierarchical network or network slice with centralized control plane and supporting a mix of flows requiring and not requiring IP mobility support is:

GL-cfg:5 Multiple instances of DPAs (at access routers) which are providing IP prefix to the MNs are needed to provide distributed mobility anchoring according to Figure 2(a) or Figure 2(b)in Section 3.1.2 with centralized control plane for a hierarchical network.

The appropriate IPv6 nodes (CPA, DPA) are to be implemented the mobility functions LM and FM as described respectively in LM-cfg:3 or LM-cfg:4 and FM-cfg:2 in Section 3.2.

Even when the mobility event does not involve change of anchor, it is still necessary to distinguish whether a flow needs IP mobility support.

The GL-mix guidelines in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network or network slice supporting a mix of flows requiring and not requiring IP mobility support apply here. The guidelines (GL-switch) in Section 5.1.1 and in Section 5.2.1 also apply here. In addition, the following are required.

GL-switch:7 Here, the LM operations and mobility message parameters described in Section 3.2.1 provides information of which IP prefix from its FW needs to be used by a flow using which new FW. The anchor operations to properly forward the packets of a flow described in the FM operations and mobility message parameters (FM-path, FM-path-ind, FM-cpdp in Section 3.2.2) may be realized with PMIPv6 protocol ([I-D.korhonen-dmm-local-prefix]) or with AERO protocol ([I-D.templin-aerolink]) to tunnel between the AR and the FW.

5.4. IP Prefix/Address Anchor Switching for a Hierarchical Network

The configuration for the hierarchical network is again shown in Figures 1(c) and 1(d) in Section 3.1. Again, with centralized control plane, CPA and CPN, with the associated LM and FM-CP are all co-located. There are multiple DPAs (each with FM-DP) in distributed mobility anchoring. In the data plane, there are multiple DPNs (each

with FM-DP) hierarchically below each DPA. The DPA at each AR supports forwarding to the DPN at each of a number of forwarding switches (FW's).

A distributed mobility event in this configuration involves change from a previous DPN which is hierarchically under the previous DPA to a new DPN which is hierarchically under a new DPA. Such an event involving change of both DPA and DPN is shown in Figure 12.

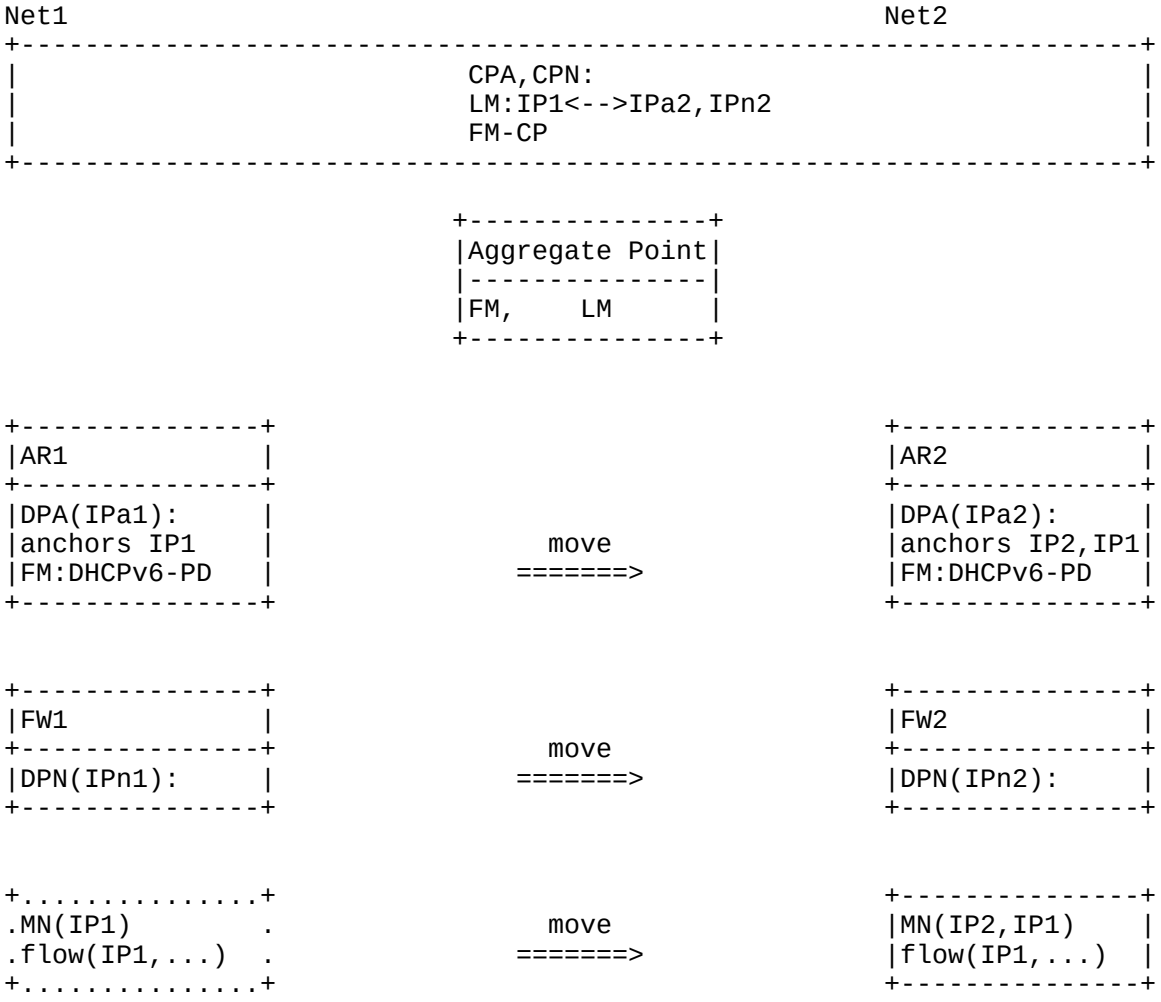


Figure 12. Mobility involving change of IP anchoring in a network with hierarchy in which the IP prefix allocated to the MN is anchored at an Edge Router supporting multiple access routers to which the MN may connect.

This deployment case involves both a change of anchor from AR1 to AR2 and a network hierarchy AR-FW. It can be realized by a combination of changing the IP prefix/address anchoring from AR1 to AR2 with the mechanism as described in Section 5.2 and then forwarding the packets with network hierarchy AR-FW as described in Section 5.3.

To change AR, AR1 acting as a DHCPv6-PD client may exchange message with the DHCPv6 server to release the prefix IP1. Meanwhile, AR2 acting as a DHCPv6-PD client may exchange message with the DHCPv6 server to delegate the prefix IP1 to AR2.

5.4.1. Additional Guidelines for IPv6 Nodes: Switching Anchor with Hierarchical Network

The configuration guideline (GL-cfg) for a hierarchical network or network slice with centralized control plane described in Section 5.3.1 apply here.

The GL-mix guidelines in Section 4.1.1 and in Section 4.2.1 for the IPv6 nodes for a network or network slice supporting a mix of flows requiring and not requiring IP mobility support apply here.

The guidelines (GL-switch) in Section 5.1.1 and in Section 5.2.1 also apply here to change the anchoring of the IP prefix/address with a centralized control plane.

In addition, the guideline for indirection between the new DPA and the new DPN as described in Section 5.3.1 apply here.

6. Security Considerations

TBD

7. IANA Considerations

This document presents no IANA considerations.

8. Contributors

This document has benefited from other work on mobility solutions using BGP update, on mobility support in SDN network, on providing mobility support only when needed, and on mobility support in enterprise network. These work have been referenced. While some of these authors have taken the work to jointly write this document, others have contributed at least indirectly by writing these drafts. The latter include Philippe Bertin, Dapeng Liu, Satoru Matushima, Peter McCann, Pierrick Seite, Jouni Korhonen, and Sri Gundavelli.

Valuable comments have also been received from John Kaippallimalil, ChunShan Xiong, and Dapeng Liu.

9. References

9.1. Normative References

[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-dmm-fpc-cpdp]
Liebsch, M., Matsushima, S., Gundavelli, S., Moses, D., and L. Bertz, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-03 (work in progress), March 2016.

[I-D.ietf-dmm-ondemand-mobility]
Yegin, A., Moses, D., Kweon, K., Lee, J., and J. Park, "On Demand Mobility Management", draft-ietf-dmm-ondemand-mobility-07 (work in progress), July 2016.

[I-D.jhlee-dmm-dnpp]
Lee, J. and Z. Yan, "Deprecated Network Prefix Provision", draft-jhlee-dmm-dnpp-01 (work in progress), April 2016.

[I-D.korhonen-dmm-local-prefix]
Korhonen, J., Savolainen, T., and S. Gundavelli, "Local Prefix Lifetime Management for Proxy Mobile IPv6", draft-korhonen-dmm-local-prefix-01 (work in progress), July 2013.

[I-D.liu-dmm-deployment-scenario]
Liu, V., Liu, D., Chan, A., Lingli, D., and X. Wei, "Distributed mobility management deployment scenario and architecture", draft-liu-dmm-deployment-scenario-05 (work in progress), October 2015.

[I-D.matsushima-stateless-uplane-vepc]
Matsushima, S. and R. Wakikawa, "Stateless user-plane architecture for virtualized EPC (vEPC)", draft-matsushima-stateless-uplane-vepc-06 (work in progress), March 2016.

[I-D.mccann-dmm-flatarch]
McCann, P., "Authentication and Mobility Management in a Flat Architecture", draft-mccann-dmm-flatarch-00 (work in progress), March 2012.

[I-D.mccann-dmm-prefixcost]
McCann, P. and J. Kaippallimalil, "Communicating Prefix Cost to Mobile Nodes", draft-mccann-dmm-prefixcost-03 (work in progress), April 2016.

[I-D.seite-dmm-dma]
Seite, P., Bertin, P., and J. Lee, "Distributed Mobility Anchoring", draft-seite-dmm-dma-07 (work in progress), February 2014.

[I-D.sijeon-dmm-deployment-models]
Jeon, S. and Y. Kim, "Deployment Models for Distributed Mobility Management", draft-sijeon-dmm-deployment-models-03 (work in progress), July 2016.

[I-D.templin-aerolink]
Templin, F., "Asymmetric Extended Route Optimization (AERO)", draft-templin-aerolink-71 (work in progress), September 2016.

[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>.

[RFC3753] Manner, J., Ed. and M. Kojo, Ed., "Mobility Related Terminology", RFC 3753, DOI 10.17487/RFC3753, June 2004, <http://www.rfc-editor.org/info/rfc3753>.

[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>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[RFC7077] Krishnan, S., Gundavelli, S., Liebsch, M., Yokota, H., and J. Korhonen, "Update Notifications for Proxy Mobile IPv6", RFC 7077, DOI 10.17487/RFC7077, November 2013, <http://www.rfc-editor.org/info/rfc7077>.

[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>.

[RFC7429] Liu, D., Ed., Zuniga, JC., Ed., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, DOI 10.17487/RFC7429, January 2015, <http://www.rfc-editor.org/info/rfc7429>.

9.2. Informative References

[Paper-Distributed.Mobility] Lee, J., Bonnin, J., Seite, P., and H. Chan, "Distributed IP Mobility Management from the Perspective of the IETF: Motivations, Requirements, Approaches, Comparison, and Challenges", IEEE Wireless Communications, October 2013.

[Paper-Distributed.Mobility.PMIP] Chan, H., "Proxy Mobile IP with Distributed Mobility Anchors", Proceedings of GlobeCom Workshop on Seamless Wireless Mobility, December 2010.

[Paper-Distributed.Mobility.Review] Chan, H., Yokota, H., Xie, J., Seite, P., and D. Liu, "Distributed and Dynamic Mobility Management in Mobile Internet: Current Approaches and Issues", February 2011.

Authors' Addresses

H Anthony Chan (editor)
Huawei Technologies
5340 Legacy Dr. Building 3
Plano, TX 75024
USA

Email: h.a.chan@ieee.org

Xinpeng Wei
Huawei Technologies
Xin-Xi Rd. No. 3, Haidian District
Beijing, 100095
P. R. China

Email: weixinpeng@huawei.com

Jong-Hyouk Lee
Sangmyung University
31, Sangmyeongdae-gil, Dongnam-gu
Cheonan 31066
Republic of Korea

Email: jonghyouk@smu.ac.kr

Seil Jeon
Sungkyunkwan University
2066 Seobu-ro, Jangan-gu
Suwon, Gyeonggi-do
Republic of Korea

Email: seiljeon@skku.edu

Alexandre Petrescu
CEA, LIST
CEA Saclay
Gif-sur-Yvette, Ile-de-France 91190
France

Phone: +33169089223
Email: Alexandre.Petrescu@cea.fr

Fred L. Templin
Boeing Research and Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 7, 2015

M. Liebsch
NEC
S. Matsushima
Softbank Telecom
S. Gundavelli
Cisco
D. Moses
Intel Corporation

May 6, 2015

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-00.txt

Abstract

The specification as per this document supports the separation of the Control-Plane for mobility- and session management from the actual Data-Plane. The protocol semantics abstract from the actual details for the configuration of Data-Plane nodes and apply between a Client function, which is used by an application of the mobility Control-Plane, and an Agent function, which is associated with the configuration of Data-Plane nodes according to the policies issued by the mobility Control-Plane. The scope of the policies comprises forwarding rules and treatment of packets in terms of encapsulation, IP address re-writing and QoS. Additional protocol semantics are described to support the maintenance of the Data-Plane path.

Status of This Memo

Table of Contents

1. Introduction

One objective of the Distributed Mobility Management (DMM) WG is the separation of the mobility management Control- and Data-Plane to enable flexible deployment, such as decentralized provisioning of Data-Plane nodes (DPN). Data-Plane nodes can be configured to function as anchor for a registered Mobile Node's (MN) traffic, others can be configured to function as Mobile Access Gateway (MAG) as per the Proxy Mobile IPv6 protocol [RFC5213] or a Foreign Agent

(FA) as per the Mobile IPv4 protocol [RFC3344]. Requirements for DMM have been described in [RFC7333], whereas best current practices for DMM are documented in [RFC7429].

The Data-Plane must provide a set of functions to the Mobility Control-Plane, such as support for encapsulation, IP address re-writing, QoS differentiation and traffic shaping. In addition, the configuration of forwarding rules must be provided. These requirements are met by various transport network components, such as IP switches and routers, though configuration semantics differs between them.

Forwarding Policy Configuration (FPC) as per this document enables the configuration of any Data-Plane node and type by the abstraction of configuration details and the use of common configuration semantics. The protocol using the FPC semantics is deployed between a Client function, which is associated with the Mobility Management Control-Plane, and an Agent function. The Agent function enforces the Data-Plane configuration and can be present on a transport network controller or co-located with a Data-Plane node. The Agent applies the generalized configuration semantics to configuration, which is specific to the Data-Plane node and type. The Mobility Control-Plane can select one or multiple DPNs which suit the MN’s mobility management without the need to handle each node’s routing- or switching tables and local interface configurations for potentially many routers serving the Data-Plane, but enforce the policies for traffic treatment and forwarding through the FPC Client and the FPC Agent functions.

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. Model for Policy-based DMM Network Control

3.1. Reference Architecture for DMM Forwarding Policy Configuration

The DMM Forwarding Policy Configuration (FPC) protocol enables DMM use cases in deployments with separated Control-/Data-Plane and is used by applications of the Mobility Control-Plane to enforce rules for forwarding and traffic treatment in the Data-Plane. Figure 1 depicts an exemplary use case where downlink traffic from a Correspondent Node (CN) towards a Mobile Node (MN) traverses multiple DPNs, each applying policies as per the Control-Plane’s request. Policies in the one or multiple DPNs can result in traffic steering according to a host-route, packet scheduling and marking according to

a subscriber's QoS profile, or forwarding rules (e.g. encapsulation within GRE or GTP-U tunnel).

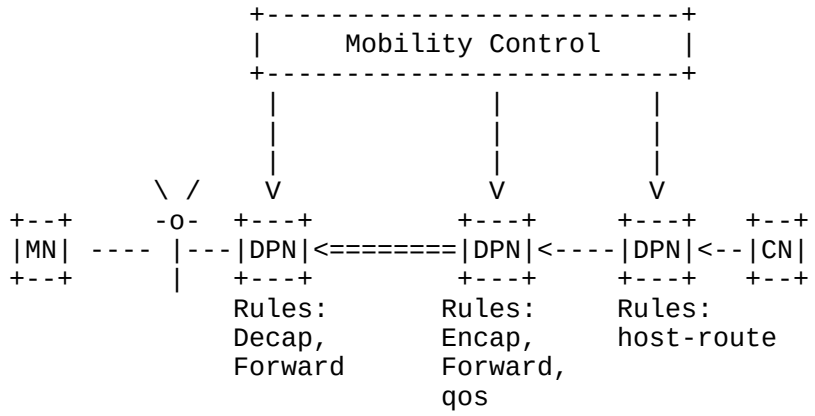


Figure 1: Exemplary illustration of a use case for DMM traffic steering and policy enforcement at Data Plane Nodes (DPN)

Mobility Control-Plane functions have the following roles in common:

- Tracking an MN's location
- Accept requests to set up and maintain mobility-related Data-Plane path between DPNs, taking QoS attributes into account. Such requests can be issued through mobility protocols, such as Proxy Mobile IPv6, and the associated operation with remote Mobility Control-Plane functions.
- Become aware of different DPNs that provide the required Data-plane functions to the Mobility Control-Plane and can be used for mobility traffic forwarding and treatment
- Monitor the DPNs' operation and handle exceptions, e.g. the detection of a partial DPN failure and the diversion of traffic through a different DPN
- Maintain consistency between multiple DPNs which enforce policy rules for an MN

Mobility Data-Plane functions have the following roles in common:

- Forward and treat traffic according to the policies and directives sent by the Mobility Control-Plane

- Provide status (e.g. load, health, statistics and traffic volume) information on request

- Participate in the process for topology acquisition, e.g. by exposing relevant topological and capability information, such as support for QoS differentiation and supported encapsulation protocols

The protocol for DMM FPC applies to the interface between an FPC Client function and an FPC Agent function, as depicted in Figure 2. The FPC Client function is associated with an application function of the mobility management Control-Plane, e.g. a Local Mobility Anchor Control-Plane function as per the Proxy Mobile IPv6 protocol. The FPC Agent function processes the FPC protocol semantics and translates them into configuration commands as per the DPN's technology. In one example, an FPC Agent can be co-located with a Transport Network Controller, which enforces forwarding rules on a set of SDN switches. In another example, the Agent can be co-located with a single router to directly interact with interface management and the router's RIB Manager. The mapping of the common FPC semantics and policy description as per this specification to the configuration commands of a particular DPN is specific to the DPN's technology and the Agent's implementation.

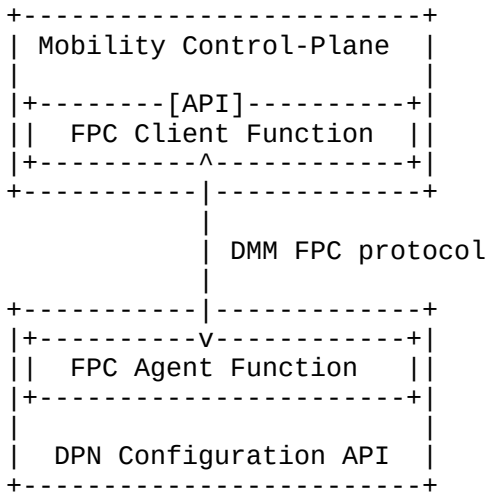


Figure 2: Illustration of the functional reference architecture for DMM Forwarding Policy Configuration (FPC)

3.2. Generalized Rules on the Client-Agent-Interface

To abstract configuration details of an IP switch or IP router on the FPC protocol interface, this specification adopts the model of logical gates (Ports) to bind certain properties, such as a QoS policy. Additional properties can be bound to the same logical Port, e.g. encapsulation of packets, being directed to that logical Port, in a GRE tunnel. The remote tunnel endpoint is configured as part of the property bound to that logical Port. All traffic, which has a forwarding rule in common and should be forwarded according to the properties bound to a particular Port, can be referred to that Port by configuration of a forwarding rule. Multiple IP flows or even aggregated traffic being destined to a given IP prefix can be directed to that logical Port and experiences the same treatment according to the configured properties and forwarding characteristics. Aggregated or per-Host/per-Flow traffic can be identified by a longest prefix match or a Traffic Selector respectively.

Figure 3 illustrates the generic policy configuration model as used between an FPC Client function and an FPC Agent function.

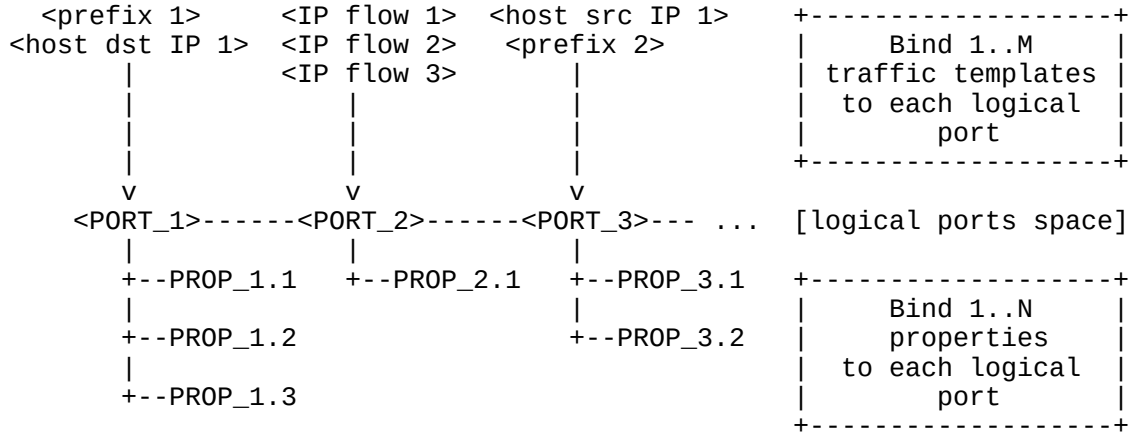


Figure 3: Illustration of generalized rules

3.3. Role of the DMM FPC Client Function

The DMM FPC Client function includes the following tasks:

- Per mobility management transaction or relevant event, build one or multiple Control messages/attributes to control policies on one or multiple DPA(s) according to the application’s directives

- Treat a DPN's policy rules (encapsulation, address re-write, QoS, traffic monitoring) on the basis of properties being bound to logical ports (similar to the bearer concept in cellular networks)
- Build, modify or delete logical ports as needed
- Bind associated policy rules as one or multiple properties to a logical port
- Treat forwarding rules (e.g. per-IP flow, per-MN, per-IP, per-prefix) on the basis of logical ports
- Send each generated message to the DMM FPC Agent associated with the identified DPN
- Keep record of the policy rules/port information and the associated DPN and FPC Agent Function
- Process received Response, Notification and Query messages issued by a DMM FPC Agent Function and notify the application

3.4. Role of the DMM FPC Agent Function

The DMM FPC Agent function includes the following tasks:

- Process the received Control messages issued by a DMM FPC Client Function
- Unambiguously match each logical port with an associated physical port or interface at the identified DPN
- Apply the received properties to local configuration (e.g. encapsulation, NA(P)T, traffic prioritization and scheduling) on the identified DPN according to the DPN's technology
- Monitor scheduled events (e.g. failure or missing rule) and issue an associated message to the FPC Client Function (NOTIFICATION, QUERY)

4. Protocol Messages and Semantics

4.1. Protocol Messages

Message	Description
Messages issued by the FPC Client	
PRT_ADD	Add a logical port
PRT_DEL	Delete an existing logical port
PROP_ADD	Add a property to a logical port
PROP_MOD	Modify a property of a logical port
PROP_DEL	Remove and delete a property from a logical port
RULE_ADD	Add forwarding rule by binding traffic descriptor to a logical port
RULE_MOD	Modify existing forwarding rule by changing the traffic descriptor bound to a logical port
RULE_DEL	Delete a forwarding rule
EVENT_REG	Register an event at an Agent, which is to be monitored by the Agent and to be reported
PROBE	Probe the status of a registered event
Messages issued by the FPC Agent	
NOTIFY	Notify the Client about the status of a monitored attribute at any event kind (periodic / event trigger / probed)
QUERY	Query the Client about missing rules/states

Figure 4: Protocol Messages

4.2. Protocol Attributes

Protocol messages as per Section 4.1 carry attributes to identify an FPC Client- or Agent function, as well as a DPN, logical ports and configuration data. Furthermore, attributes are carried to manage logical ports and describe properties associated with a logical port, as well as to describe per-host-, aggregate or IP flow traffic and refer to a logical port as forwarding information.

This document specifies attributes from the following categories:

- Identifier attributes
- Properties
- Property-specific attributes
- Traffic descriptors

Note on the list of attributes: The list of attributes is not yet complete.

Note on Format Clarification: Meant to provide a first idea on the format and number space and indicates length (bit) and semantics of key information fields.

Attribute	Format Clarification	Description
	Identifiers	
PRT_ID	[16,PTR_ID]	Identifies a logical Port
PRT_PROP_ID	[16,PRT_ID] [8,PROP_ID]	Identifies a logical Port and one of its properties
CLI_ID	[8, Carrier ID] [8, Network ID] [16, Client ID]	Identifies an FPC Client function
AGT_ID	[8, Carrier ID] [8, Network ID] [16, Agent ID]	Identifies an FPC Agent function
DPN_ID	[8, Carrier ID] [8, Network ID] [16, DPN ID]	Identifies a Data Plane Node (DPN)
EVENT_ID	[16, Event ID]	Identifies a registered event

Figure 5: Protocol Attributes: Identifiers

Attribute	Format Clarification	Description
	Properties	
PROP_TUN	[type][src][dst]	Property Encapsulation, indicates type GRE, IP, GTP
PROP_REWR	TBD	Property NAT
PROP_QOS	TBD	Property QoS
PROP_GW	[ip address next hop]	Property Next Hop

Figure 6: Protocol Attributes: Properties

Attribute	Format Clarification	Description
	Property-specific	
IPIP_CONF		IP-encapsulation configuration attribute
GRE_CONF	[prototype][seq-#] [key]..	GRE_encapsulation configuration attribute
GTP_CONF	[TEID_local] [TEID_remote] [seq-#]..	GTP-U encapsulation configuration attribute

Figure 7: Protocol Attributes: Property-specific

4.3. Protocol Operation

The following list comprises a more detailed description of each message's semantic.

- PRT_ADD - Issued by a Client to add a new logical port at an Agent, to which traffic can be directed. An Agent receiving the PRT_ADD message should identify the new logical port according to the included port identifier (PRT_ID). In case the DPN holds already a registration for a logical port with the same identifier, the Agent should throw an error message to the Client. Otherwise the Agent should add a new logical port into its conceptual data structures using the port identifier as key.

- PRT_DEL - Used by a Client to delete an existing logical port. An Agent receiving such message should delete all properties associated with the identified port.

- PROP_ADD - Used by the Client to add a new property to an existing logical port. The property is unambiguously identified through a property identifier (PRT_PROP_ID). All traffic, which is directed to this logical port, experiences the existing and newly added property.

- PROP_MOD - Used by a Client to modify an existing property. For example, a tunnel property can be changed to direct traffic to a different tunnel endpoint in case of an MN’s handover

- PROP_DEL - Used by a Client to delete one or multiple properties, each being identified by a property identifier.

- RULE_ADD - Used by a Client to add a forwarding rule and direct traffic towards a logical port. The rule add command must unambiguously identify aggregated traffic (longest prefix), per host IP traffic or per-flow traffic in the RULE_ADD command and bind the identified traffic to a logical port. An Agent receiving a RULE_ADD command must add the rule to its local conceptual data structures and apply commands for local configuration to add the new forwarding rule on the DPN. Multiple forwarding rules, each identifying different traffic, can direct traffic to the same logical port. All traffic being directed to this logical port will then experience the same properties.

- RULE_MOD - Used by a Client to modify an existing forwarding rule. An Agent receiving such message should apply commands for local configuration to update the forwarding rule on the DPN.

- RULE_DEL - Used to delete an existing forwarding rule on a DPN. The Agent receiving such message should delete the rules from its local conceptual data structures and apply commands for local configuration to remove the forwarding rule on the DPN.

- EVENT_REG - Used by a Client to register an attribute, which is to be monitored, at an Agent. The EVENT_REG provides an attribute to the Agent as well as a reporting kind. The Agent should register the event and an event identifier in the local conceptual data structures. The Agent should start monitoring the registered attribute (e.g. load) and notify the Client about the status according to the registered reporting kind (periodic, event trigger, probed). In case of a periodic reporting kind, the Agent should report the status of the attribute each configured interval using a NOTIFY message. The reporting interval is provided with

the EVENT_REG message. In case of an event triggered reporting kind, the Agent should report the status of the attribute in case of a triggered event, e.g. the monitored attribute's value exceeds a given threshold. The threshold is provided with the EVENT_REG message. In case of probed reporting, the Agent receives a PROBE message and should report the status of a monitored attributes to the Client by means of a NOTIFY message.

- PROBE - Used by a Client to retrieve information about a previously registered event. The PROBE message should identify one or more events by means of including the associated event identifier. An Agent receiving a PROBE message should send the requested information for each event in a single or multiple NOTIFY messages.

- NOTIFY - Used by an Agent to report the status of an event to a Client.

- QUERY - Used by an Agent to request an update of logical port properties via a Client.

Figure 8 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.

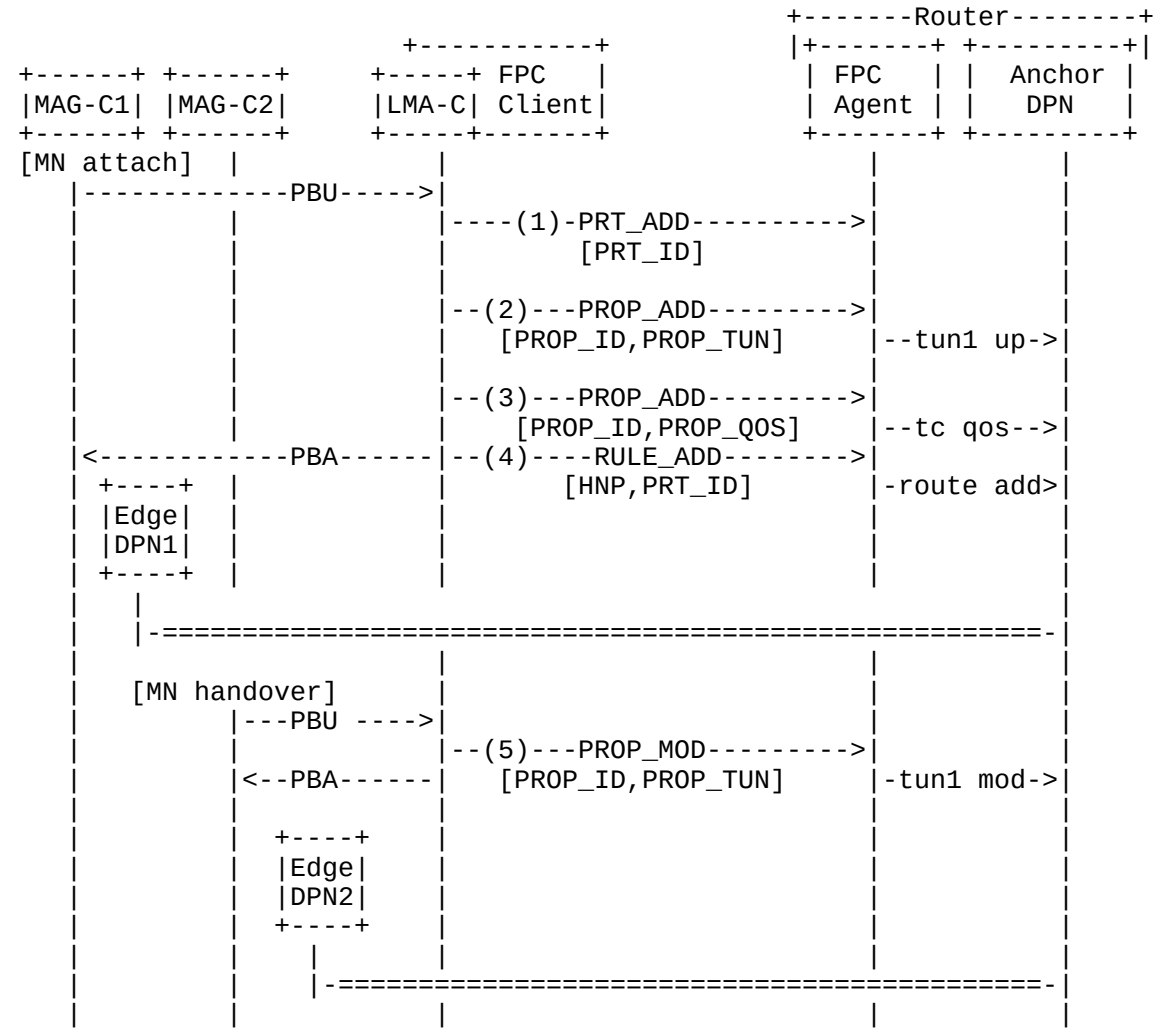


Figure 8: Exemplary Message Sequence (focus on FPC reference point)

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 MN's traffic. The LMA-C adds a new logical port to the DPN to treat the MN's traffic (1) and includes a Port Identifier (PRT_ID) to the PRT_ADD command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

Subsequently, the LMA-C adds properties to the new logical port. One property is added (2) to specify the forwarding tunnel type and endpoints (Anchor DPN, Edge DPN1). Another property is added (3) to specify the QoS differentiation, which the MN's traffic should experience. At reception of the properties, the FPC Agent calls local router commands to enforce the tunnel configuration (tun1) as well as the traffic control (tc) for QoS differentiation. After configuration of port properties have been completed, the LMA can configure the enforcement of the MN's traffic by adding a rule (RULE_ADD) to forward traffic destined to the MN's HNP to the new logical port (4). At the reception of the forwarding rule, the Agent applies a new route to forward all traffic destined to the MN's HNP 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 endpoint. The LMA-C sends a PROP_MOD message (5) to the Agent to modify the existing tunnel property of the existing logical port and to update the tunnel endpoint from Edge DPN1 to Edge DPN2. At reception of the PROP_MOD message, the Agent applies local configuration commands to modify the tunnel.

To reduce the number of protocol handshakes between the LMA-C and the DPN, the LMA-C can append property (PROP_TUN, PROP_QOS) and rules (prefix info HNP) attributes to the PRT_ADD message, as illustrated in Figure 9

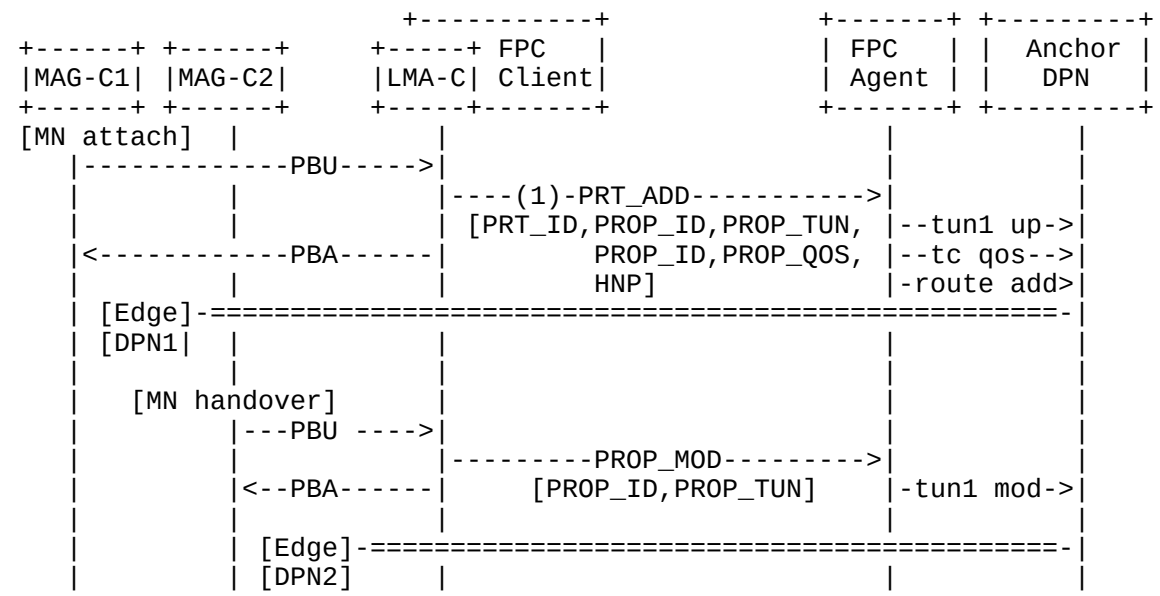


Figure 9: Example: Sequence for Message Aggregation (focus on FPC reference point)

5. Conceptual Data Structures

An FPC Client must keep record about the logical ports, each port's properties as well as configured rules as per the Mobility Control-Plane function's request. Such information must be maintained for each Agent, with which the Client communicates. In case the Mobility Control-Plane function identifies a particular DPN at which the policies should be enforced, the Client must associate the DPN identifier with the logical port configuration.

According to the FPC Agent's role, the Agent translates the generalized model for policy configuration and forwarding rules into semantics and commands for local configuration, which is specific to a DPN. Keeping a local record of DPN configuration attributes/values is implementation specific and out of scope of this document.

Description of detailed data structures and information to be recorded and maintained by an FPC Client and an FPC Agent are TBD and will be added to a revision of this initial document.

6. 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.

7. IANA Considerations

This document provides an information model for DMM Forwarding Policy Configuration. Detailed protocol specifications for DMM Forwarding Policy Configuration will follow the information model as per this document and can be based on, for example, ReST-like or binary protocol formats. Such protocol-specific details will be described in separate documents and may require IANA actions.

8. 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.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7333] Chan, H., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, August 2014.

[RFC7429] Liu, D., Zuniga, JC., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, January 2015.

9.2. Informative References

[RFC3344] Perkins, C., "IP Mobility Support for IPv4", RFC 3344, August 2002.

[RFC5213] Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

Appendix A. YANG Data Model for the FPC Protocol

This appendix provides (so far experimental) formating of some FPC protocol components adopting YANG data modeling. The current FPC information model as per this initial draft version will experience extensions, as it is not yet complete, and may experience changes that need to be reflected in the data model. Whether a detailed data model will be included in this document or solely an information model will be adopted by this document and a detailed data model will be part of a separate document is currently being discussed.

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

    import ietf-inet-types { prefix inet; }

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

    revision 2015-03-09 {}

    typedef fpcp-port-id {
        description "PRT_ID";
        type uint16;
    }

    typedef fpcp-property-id {
        description "PROP_ID";
        type uint8;
    }

    identity tunnel-type {
        description
        "Base identity from which specific use of
         tunnels are derived.";
    }

    identity fpcp-tunnel-type {
        base "tunnel-type";
        description
        "Base identity from which specific tunnel
         types in FPCP uses are derived.";
    }

    identity ip-in-ip {
```

```
            base "fpcp-tunnel-type";
            description "IP-in-IP tunnel";
        }

        identity gtp {
            base "fpcp-tunnel-type";
            description "GTP-U tunnel";
        }

        identity gre {
            base "fpcp-tunnel-type";
            description "GRE tunnel";
        }

        identity ip-protocol {
            description
            "Base identity from which specific
             IP protocol types are derived.";
        }

        identity qos-type {
            description
            "Base identity from which specific
             uses of QoS types are derived.";
        }

        identity fpcp-qos-type {
            base "qos-type";
            description
            "Base identity from which specific
             QoS types in FPCP uses are derived.";
        }

        identity fpcp-qos-type-high {
            base "fpcp-qos-type";
            description
            "An example FPCP QoS Type for high quality class.
             FPCP supported QoS classes are TBD.";
        }

        identity fpcp-qos-type-middle {
            base "fpcp-qos-type";
            description
            "An example FPCP QoS Type for middle quality class.
             FPCP supported QoS classes are TBD.";
        }

        identity fpcp-qos-type-low {
```

```
            base "fpcp-qos-type";
            description
            "An example FPCP QoS Type for low quality class.
             FPCP supported QoS classes are TBD.";
        }

        grouping fpcp-client {
            description "CLI_ID to identify FPCP Client";
            leaf carrier-id {
                type uint8;
            }
            leaf network-id {
                type uint8;
            }
            leaf client-id {
                type uint16;
                mandatory true;
            }
        }

        grouping fpcp-agent {
            description "AGT_ID to identify FPCP Agent";
            leaf carrier-id {
                type uint8;
            }
            leaf network-id {
                type uint8;
            }
            leaf agent-id {
                type uint16;
                mandatory true;
            }
        }

        grouping dpn {
            description "DPN_ID to identify Data-Plane Node";
            leaf carrier-id {
                type uint8;
            }
            leaf network-id {
                type uint8;
            }
            leaf dpn-id {
                type uint16;
                mandatory true;
            }
        }
```

```
    grouping port-property-id {
        description "PRT_PROP_ID";
        leaf port-id {
            mandatory true;
            type fpcp-port-id;
        }
        leaf property-id {
            type fpcp-property-id;
            mandatory true;
        }
    }

    grouping tunnel-endpoints {
        description
        "PROP_TUN property as a set of tunnel endpoints";
        leaf tunnel-type {
            type identityref {
                base "fpcp-tunnel-type";
            }
        }
        leaf remote-address {
            type inet:ip-address;
        }
        leaf local-address {
            type inet:ip-address;
        }
    }

    grouping gtp-attributes {
        description
        "GTP_CONF as GTP tunnel specific attributes";
        leaf remote-teid {
            type uint32;
        }
        leaf local-teid {
            type uint32;
        }
    }

    grouping gre-attributes {
        description
        "GRE_CONF as GRE tunnel specific attribute";
        leaf key {
            type uint32;
        }
    }

    grouping fpcp-identifier-attributes {
```

```
            description
            "Identifiers of protocol attributes";
            leaf port-id {
                type fpcp-port-id;
            }
            container client {
                uses fpcp-client;
            }
            container agent {
                uses fpcp-agent;
            }
            list nodes {
                key dpn-id;
                uses dpn;
            }
        }

        grouping fpcp-traffic-descriptor {
            description
            "Traffic descriptor group collects parameters to
             identify target traffic flow and apply QoS policy";
            leaf destination-ip {
                type inet:ip-prefix;
            }
            leaf source-ip {
                type inet:ip-prefix;
            }
            leaf protocol {
                type identityref {
                    base "ip-protocol";
                }
            }
            leaf destination-port {
                type inet:port-number;
            }
            leaf source-port {
                type inet:port-number;
            }
            leaf qos {
                type identityref {
                    base "fpcp-qos-type";
                }
            }
        }

        grouping fpcp-port-properties {
            description
            "A set of port property attributes";
```

```
            leaf property-id {
                type fpcp-property-id;
            }
            list next-hops {
                container endpoints {
                    uses tunnel-endpoints;
                }
                choice tunnel {
                    case gtp-u {
                        when "tunnel-type = 'gtp'";
                        uses gtp-attributes;
                    }
                    case gre {
                        when "tunnel-type = 'gre'";
                        uses gre-attributes;
                    }
                }
            }
        }


        // Port Entries

        container port-entries {
            description
            "This container binds set of traffic-descriptor and
             port properties to a port and lists them as a port entry.";
            list port-entry {
                key port-id;
                container identifier {
                    uses fpcp-identifier-attributes;
                }
                container trafic-descriptor {
                    uses fpcp-traffic-descriptor;
                }
                list properties {
                    uses fpcp-port-properties;
                }
            }
        }

        // PRT_ADD

        rpc port_add {
            description "PRT_ADD";
            output {
                list fpcp-port-entry {
                    uses fpcp-identifier-attributes;
```

```
            }
        }
    }

    // PRT_DEL

    rpc port_delete {
        description "PRT_DEL";
        input {
            leaf deleting-port {
                type fpcp-port-id;
            }
        }
    }

    // PROP_ADD

    rpc port_property_add {
        description "PROP_ADD";
        input {
            leaf target-port {
                type fpcp-port-id;
                mandatory true;
            }
            container port-properties {
                uses fpcp-port-properties;
            }
        }
    }

    // PROP_MOD

    rpc port_property_modify {
        description "PROP_MOD";
        input {
            leaf target-port {
                type fpcp-port-id;
                mandatory true;
            }
            container port-properties {
                uses fpcp-port-properties;
            }
        }
    }

    // PROP_DEL

    rpc port_property_delete {
```

```
            description "PROP_DEL";
            input {
                container deleting-property {
                    uses port-property-id;
                }
            }
        }

        // RULE_ADD

        rpc rule_add {
            description
            "TBD for input parameters of which RULE_ADD includes
             but now just traffic-descriptor.";
            input {
                leaf target-port {
                    type fpcp-port-id;
                    mandatory true;
                }
                container port-properties {
                    uses fpcp-traffic-descriptor;
                }
            }
        }

        // RULE_MOD

        rpc rule_modify {
            description
            "TBD for input parameters of which RULE_MOD includes
             but now just traffic-descriptor.";
            input {
                leaf target-port {
                    type fpcp-port-id;
                    mandatory true;
                }
                container port-properties {
                    uses fpcp-traffic-descriptor;
                }
            }
        }

        // RULE_DEL

        rpc rule_delete {
            description
            "TBD for input parameters of which RULE_DEL includes
             but now just traffic-descriptor.";
```

```
            input {
                leaf target-port {
                    type fpcp-port-id;
                    mandatory true;
                }
                container port-properties {
                    uses fpcp-traffic-descriptor;
                }
            }
        }

        // EVENT_REG

        rpc event_register {
            description
            "TBD for registered parameters included in EVENT_REG.";
        }

        // PROBE

        rpc probe {
            description
            "TBD for retrieved parameters included in PROBE.";
        }

        // NOTIFY

        notification notify {
            description
            "TBD for which status and event are reported to client.";
        }
    }
```

Figure 10: FPC YANG Data Model

Authors' Addresses

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

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

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

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

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

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

S. Matsushima
SoftBank
L. Bertz
Sprint
M. Liebsch
NEC
S. Gundavelli
Cisco
D. Moses
Intel Corporation
October 31, 2016

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-05.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 May 4, 2017.

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 policy sets. A Context can belong to multiple Ports which serve different kinds of purpose and policy. Monitors provide 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

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. Terminology

DPN: A data-plane node (DPN) is capable of deploying data-plane features. DPNs may be switches or routers regardless of their realiziation, i.e. whether they are hardware or software based.

FPC Agent:	A functional entity in FPC that manages DPNs and provides abstracted data-plane networks to mobility management systems and/or applications through FPC Clients.
FPC Client:	A functional entity in FPC that is integrated with mobility management systems and/or applications to control forwarding policy, mobility sessions and DPNs.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions.
Domain:	One or more DPNs that form a data-plane network. A mobility management system or an application in a tenant may utilize a single or multiple domains.
Port:	A set of forwarding policies.
Context:	An abstracted endpoint of a mobility session associated with runtime attributes. Ports may apply to Context which instantiates those forwarding policies on a DPN.

4. 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 5. 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 6.

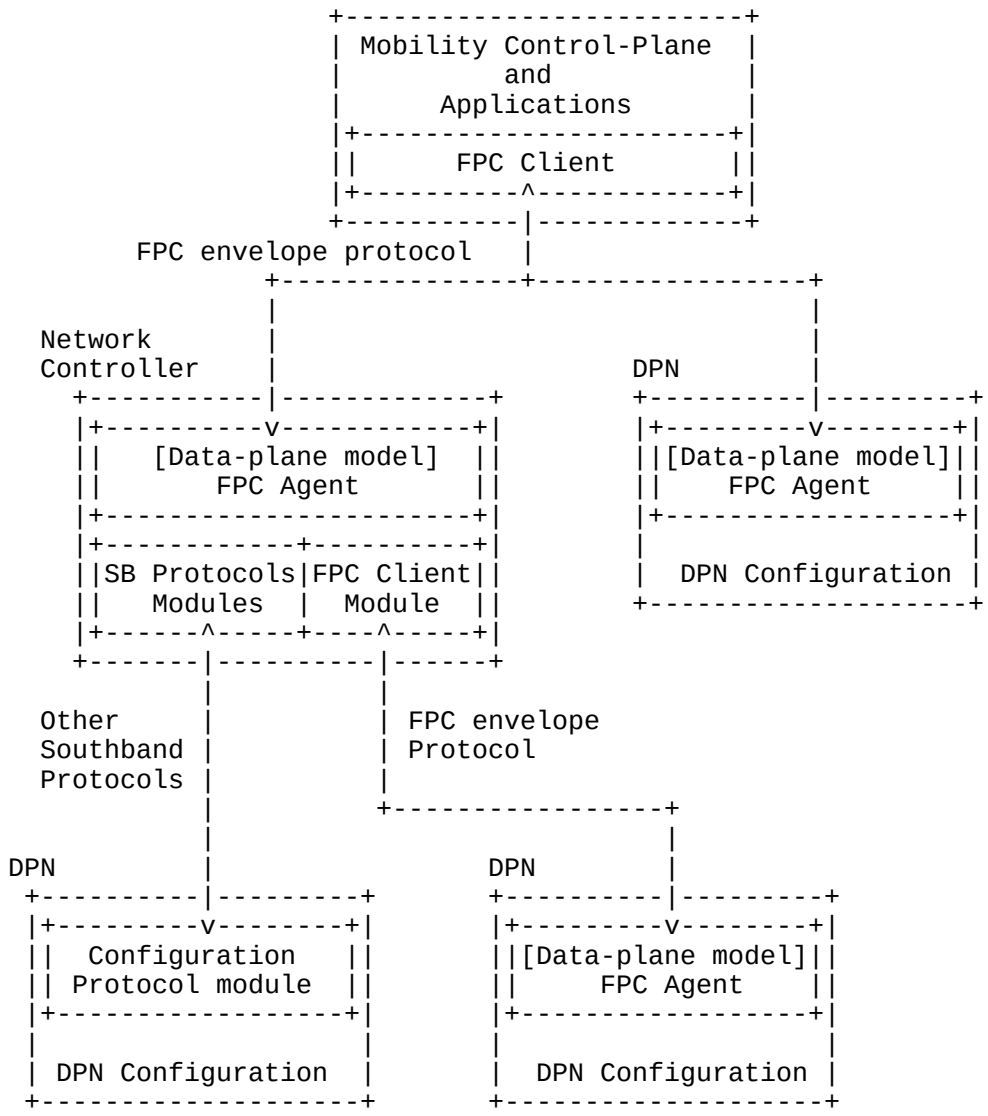


Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

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 Topology and Policy

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

5. 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.

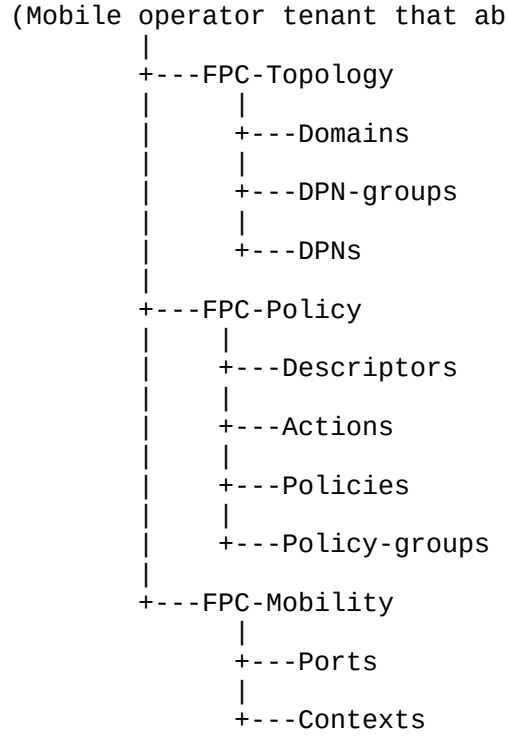


Figure 2: FPC Data-plane Abstraction Model

5.1. 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.

5.1.1. Domains

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

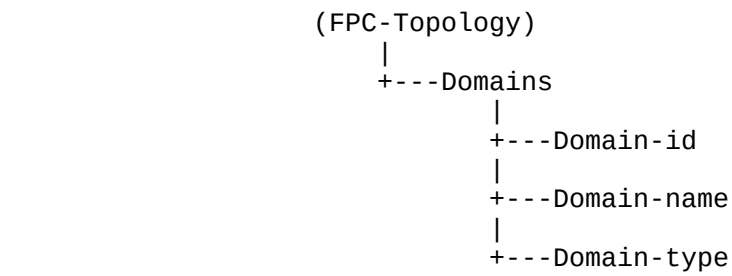


Figure 3: Domain Model Structure

Domain-id: Identifier of Domain. The ID format SHOULD refer to Section 5.4.

Domain-name: Defines Domain name.

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

5.1.2. 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.

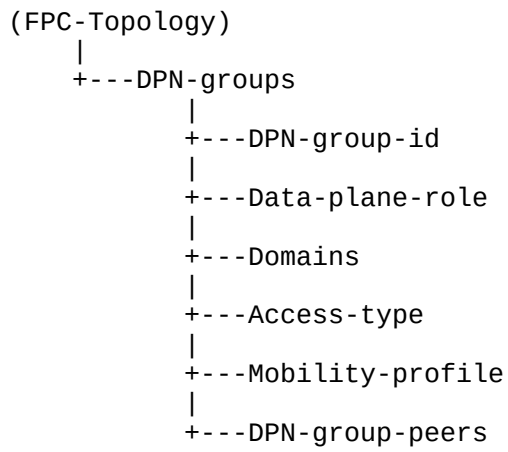


Figure 4: DPN-groups Model Structure

DPN-group-id: Defines identifier of DPN-group. The ID format SHOULD refer to Section 5.4.

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 5.1.2.1.

5.1.2.1. DPN-group Peers

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

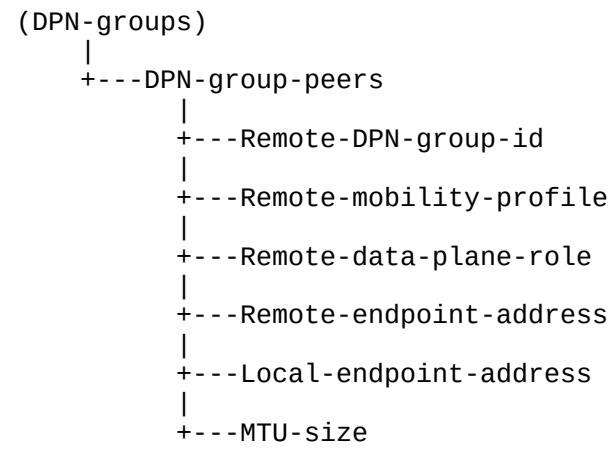


Figure 5: DPN-groups Peer Model Structure

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.

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

5.1.3. 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.

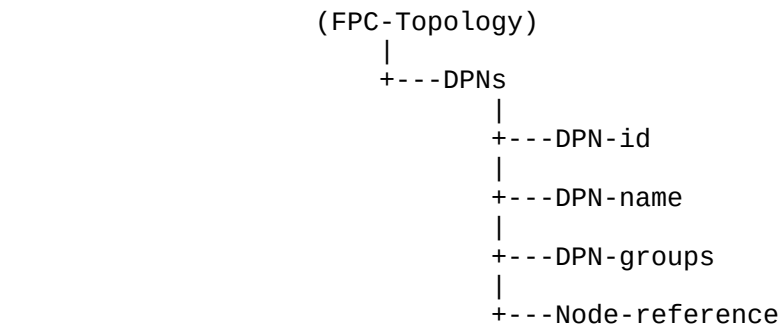


Figure 6: DPNs Model Structure

DPN-id: Defines identifier of DPN. The ID format SHOULD refer to Section 5.4.

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.

5.2. 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.

5.2.1. 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.

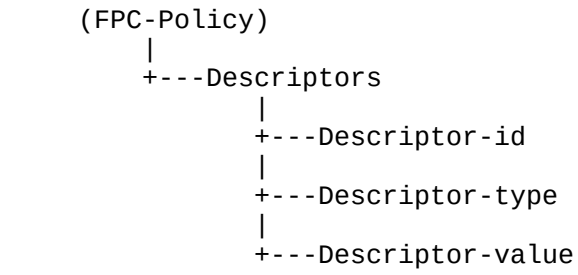


Figure 7: Descriptor Model Structure

Descriptor-id: Identifier of Descriptor. The ID format SHOULD refer to Section 5.4.

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.

5.2.2. 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.

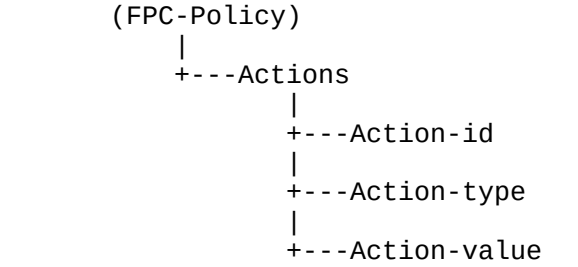


Figure 8: Action Model Structure

Action-id: Identifier of Action. The ID format SHOULD refer to Section 5.4.

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.

5.2.3. 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.

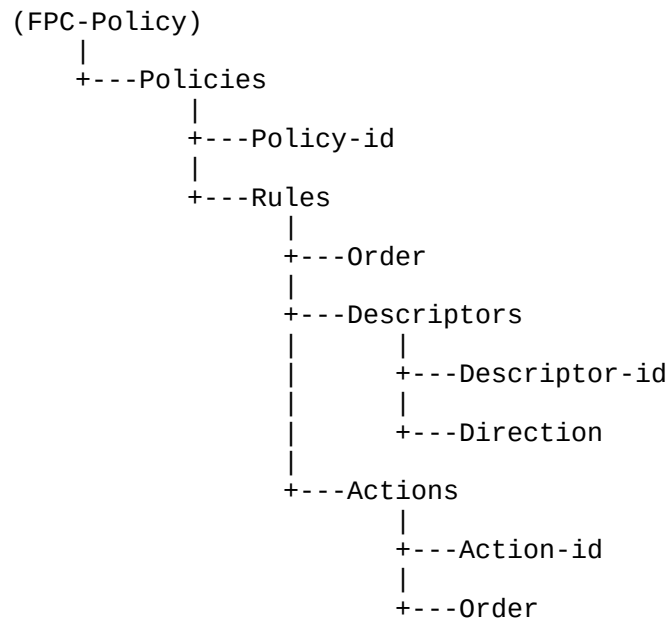


Figure 9: Policies Model Structure

Policy-id: Identifier of Policy. The ID format SHOULD refer to Section 5.4.

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.

5.2.4. 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.

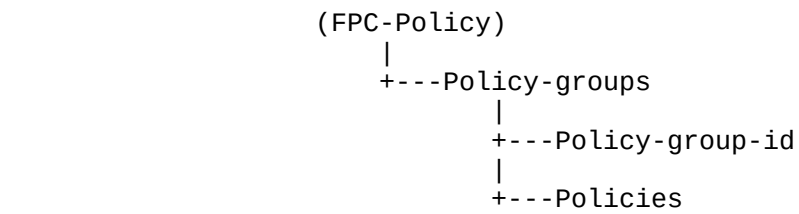


Figure 10: Policy-group Model Structure

Policy-group-id: Identifier of Policy-group. The ID format SHOULD refer to Section 5.4.

Policies: List of Policies in the Policy-group.

5.3. 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.

5.3.1. 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.

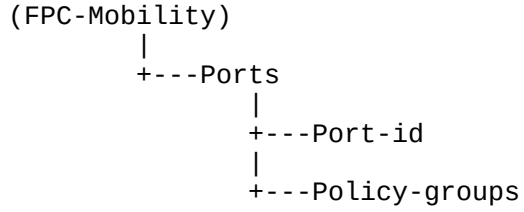


Figure 11: Port Model Structure

Port-id: Identifier of Port. The ID format SHOULD refer to Section 5.4.

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

5.3.2. 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.

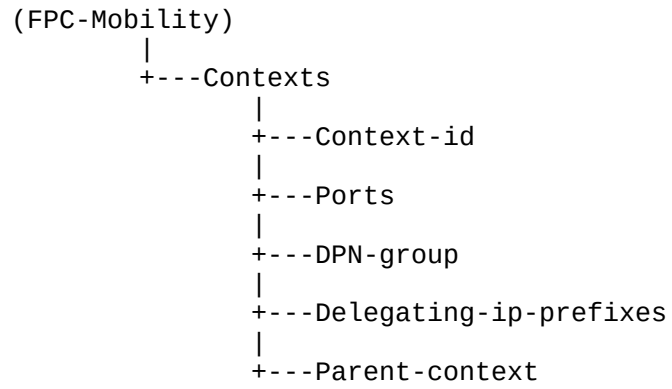


Figure 12: Common Context Model Structure

Context-id: Identifier of Context. The ID format SHOULD refer to Section 5.4.

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.

5.3.2.1. 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.

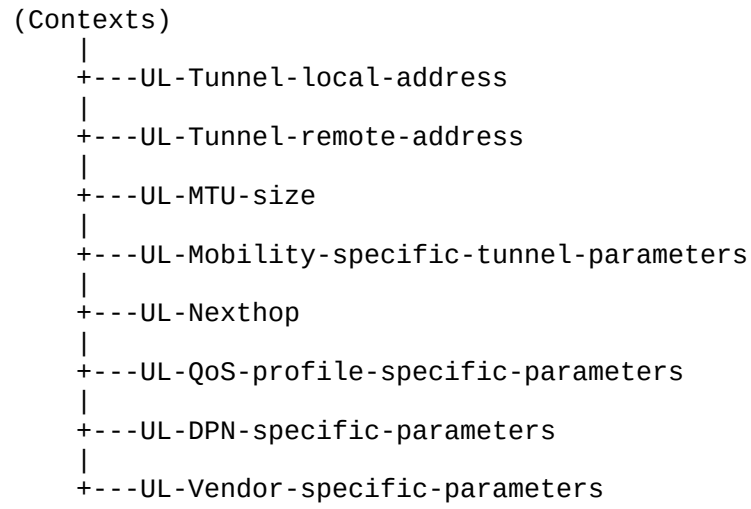


Figure 13: Uplink Context Model of Single DPN Structure

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

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

UL-MTU-size: Specifies uplink MTU size.

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.

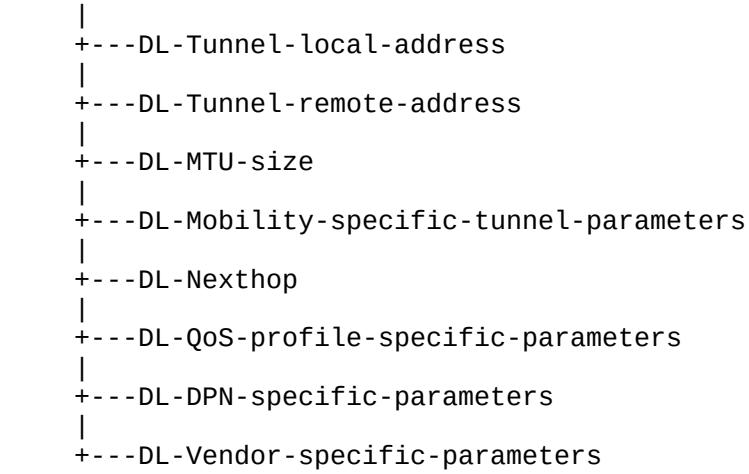


Figure 14: Downlink Context Model of Single DPN Structure

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

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

DL-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.

5.3.2.2. 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 or, where applicable, a direction of 'both'.

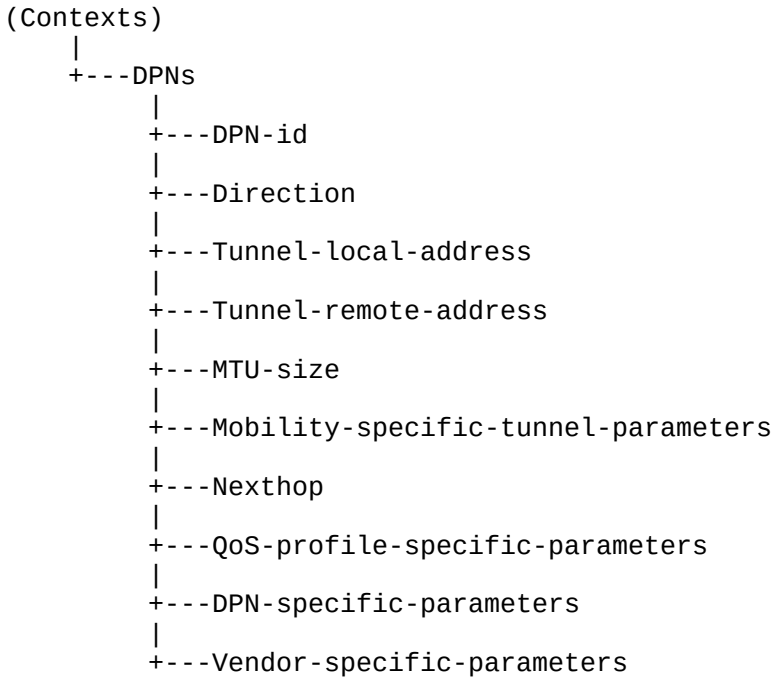


Figure 15: Multiple-DPN Supported Context Model Structure

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

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

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

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

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

Mobility-specific-tunnel-parameters: Specifies profile specific tunnel parameters for uplink or downlink 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.

Multi-DPN Agents will only use the DPNs list of a Context for processing as described in this section. A single-DPN Agent MAY use both the Single Agent DPN model Section 5.3.2.1 and the multi-DPN Agent Context described here. However, Agent feature support MUST be discoverable by the FPC Client in order to determine which option(s) an Agent supports.

5.3.3. 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.

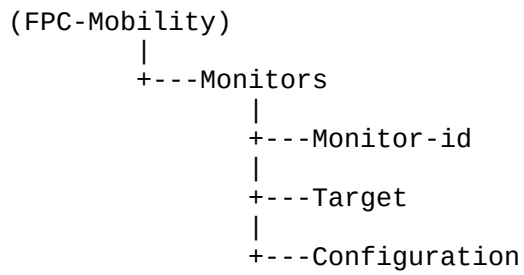


Figure 16: Common Monitor Model Structure

Monitor-id: Name of the Monitor. The ID format SHOULD refer to Section 5.4.

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 notification to the Client

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

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

5.4. Namespace and Format

The identifiers and names in FPC models which reside in the same namespace must be unique. That uniqueness must be kept in agent or data-plane tenant namespace on an Agent. The tenant namespace uniqueness MUST be applied to all elements of the tenant model, i.e. Topology, Policy and Mobility models.

When a Policy needs to be applied to Contexts in all tenants on an Agent, the Agent SHOULD define that policy to be visible from all the tenants. In this case, the Agent 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 4.

5.5. Attribute Application

Attributes in FPC Topology and Policy are pre-configured in a FPC Agent prior to Contexts and Ports. Those pre-configured attributes SHOULD NOT be instantiated on DPN(s) until the Contexts and Ports indicate them.

This is intentional as it provides FPC Clients ability to reuse attributes that helps to minimize over the wire exchanges and reduce system errors by exchanging less information.

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

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

When the attribute is a direct value of the Context, e.g. IMSI defined in the 3GPP extension, only missing values can be provided by the Parent Context.

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

5.6. Policy and Runtime Data

Contexts and Ports that are supporting runtime, realtime mobility sessions which are produced in the mobility control plane. These could be installed using any number of protocols, but in case of they need to be delivered in realtime that Restconf [I-D.ietf-netconf-restconf] and/or Netconf [RFC6241] will not fullfill, an appropriate FPC envelope protocol MUST be required.

When data is delivered as part of the FPC envelop protocol it should be part of a Context. If it is a binding to a generic policy that could be used by multiple Contexts a Port is used. Given the support for pre-configuration of policies and references by identifiers, e.g a Rule ID, most policies do not require realtime delivery.

In case of modifying an existing Context attribute, the Agent MUST overwrite that attribute with the value of which the Client brings to the Agent.

6. Protocol

6.1. 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 TRANS_STRATEGY 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 ascending 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 operation.
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.

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

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

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

It is important to note that an envelope protocol used to support this specification may not need to support CONF_BUNDLES messages or specific TRANS_STRATEGY types beyond 'default' when the protocol provides similar semantics. However, this MUST be clearly defined in the specification that defines how the envelope protocol supports this specificaiton.

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 6.1.1.6.2 for CONF and CONF_BUNDLES requests. When returning an 'ok' of any kind, optional data may be present.

Two Agent notifications are supported:

Message	Type	Description
CONFIG_RESULT_NOTIFY	See Table 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 (notifications)

6.1.1. 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 6.1.1.4).

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 7.2.2 for details.

6.1.1.1. 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 attributes.

6.1.1.2. Policy RPC Support

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

6.1.1.3. Cloning

Cloning is an optional feature that allows a Client to copy one structure to another in an operation. Cloning is always done first within the operation (see Operation Order of Execution for more detail). If a Client wants to build an object then Clone it, use 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.

6.1.1.4. Command Bitsets

The COMMAND_SET is a technology specific bitset that allows for a single entity to be sent in an operation with requested sub-transactions to be completed. For example, a Context could have the Home Network Prefix absent but it is unclear if the Client would like the address to be assigned by the Agent or if this is an error.

Rather than creating a specific command for assigning the IP a bit position in a COMMAND_SET is reserved for Agent based IP assignment. Alternatively, an entity could be sent in an update operation that would be considered incomplete, e.g. missing some required data in for the entity, but has sufficient data to complete the instructions provided in the COMMAND_SET.

6.1.1.5. Reference Scope

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

- o none - all entities have no references to other entities. This implies only Contexts are present Ports MUST have references to Policy-Groups.
- o op - All references are contained in the operation body, i.e. only intra-operaion references exist.
- o bundle - All references in exist in bundle (inter-operation/intra-bundle). NOTE - If this value comes in CONFIG call it is equivalent to ’op’.
- o storage - One or more references exist outside of the operation and bundle. A lookup to a cache / storage is required.
- o unknown - the location of the references are unknown. This is treated as a ’storage’ type.

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

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

6.1.1.6. Operation Response

6.1.1.6.1. Immediate Response

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

OK - SUCCESS

ERR - An Error has occurred

OK_NOTIFY_FOLLOWS - The Operation has been accepted by the Agent but further processing is required. A CONFIG_RESULT_NOTIFY will be sent once the processing has succeeded or failed.

Any result MAY contain nothing or 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 6.1.1.4).

If an error occurs the following information is returned.

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

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

6.1.1.6.2. 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.

6.1.2. 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.

6.1.2.1. Operation Response

6.1.2.1.1. Immediate Response

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

OK - SUCCESS

ERR - An Error has occurred

Any OK result will contain no more information.

If an error occurs the following information is returned.

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

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

6.1.2.1.2. Asynchronous Notification

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

6.2. Protocol Operation

6.2.1. Simple RPC Operation

An FPC Client and Agent MUST identify themselves using the CLI_ID and AGT_ID respectively to ensure that for all transactions a recipient of an FPC message can unambiguously identify the sender of the FPC message. A Client MAY direct the Agent to enforce a rule in a

particular DPN by including a DPN_ID value in a Context. Otherwise the Agent selects a suitable DPN to enforce a Context and notifies the Client about the selected DPN using the DPN_ID.

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all entities as well as status information, which indicates the result of processing the message, using the RESPONSE_BODY property. In case the processing of the message results in a failure, the Agent sets the ERROR_TYPE_ID and ERROR_INFORMATION accordingly and MAY clear the Context or 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 partially completed entities. When an OK_NOTIFY_FOLLOWS is sent, the Agent will, upon completion or failure of the operation, respond with an asynchronous CONFIG_RESULT_NOTIFY to the Client.

A Client MAY add a property to a Context without providing all required details of the attribute’s value. In such case the Agent SHOULD determine the missing details and provide the completed property description back to the Client. If the processing will take too long or based upon Agent configuration, the Agent MAY respond with an OK_NOTIFY_FOLLOWS with a RESPONSE_BODY containing the partially completed entities.

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

Figure 17 illustrates an exemplary session life-cycle based on Proxy Mobile IPv6 registration via MAG Control-Plane function 1 (MAG-C1) and handover to MAG Control-Plane function 2 (MAG-C2). Edge DPN1 represents the Proxy CoA after attachment, whereas Edge DPN2 serves as Proxy CoA after handover. As exemplary architecture, the FPC Agent and the network control function are assumed to be co-located with the Anchor-DPN, e.g. a Router.

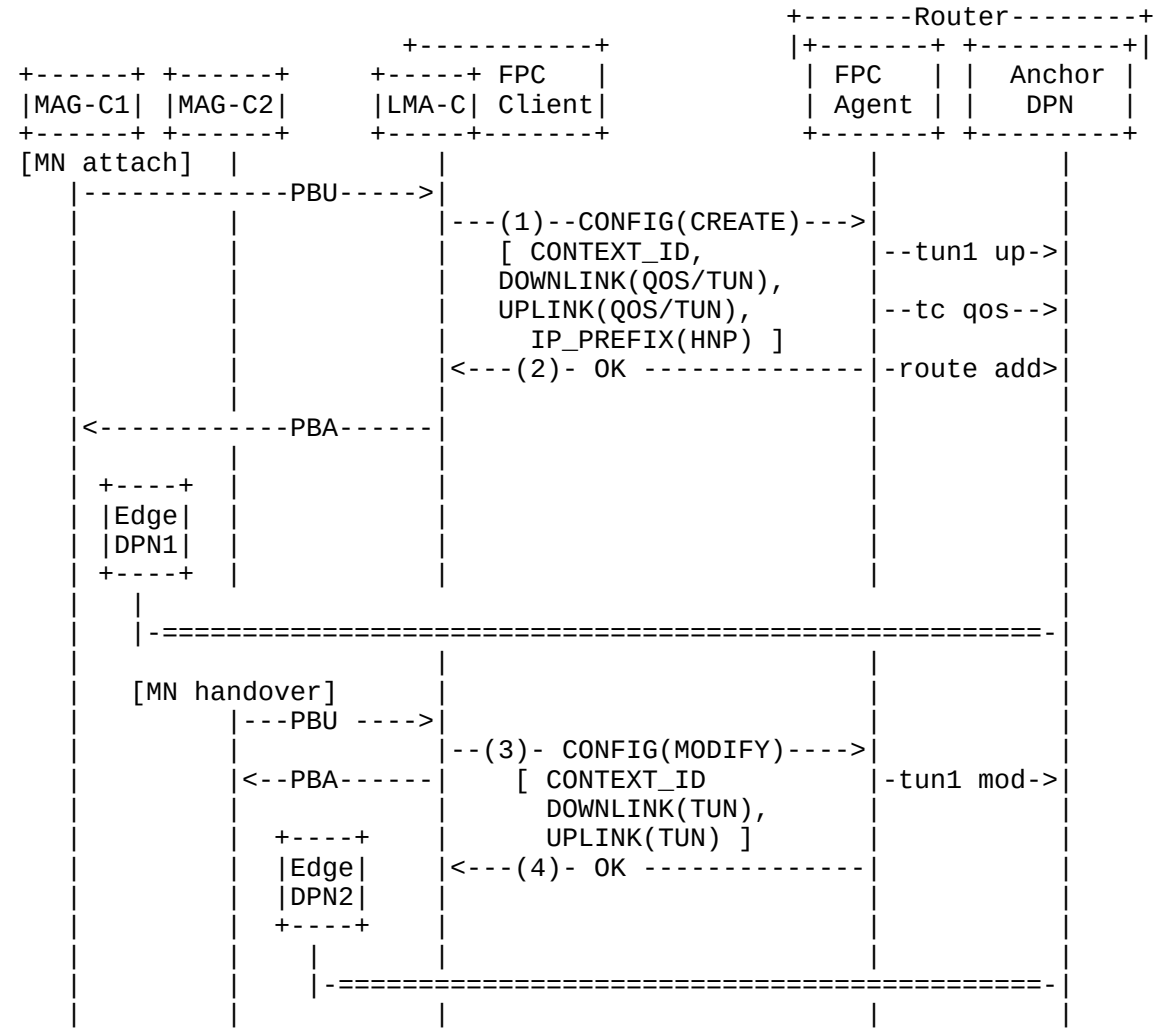


Figure 17: Exemplary Message Sequence (focus on FPC reference point)

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 creation of the new Context. One property is added to specify the forwarding tunnel type and endpoints (Anchor DPN, Edge DPN1) in each direction (as required). Another property is added to specify the QoS differentiation, which the MN's traffic should experience. At reception of the Context, the FPC Agent utilizes local configuration commands to create the tunnel (tun1) as well as the traffic control (tc) to enable QoS differentiation. After configuration has been completed, the Agent applies a new route to forward all traffic destined to the MN's HNP specified as a property in the Context to the configured tunnel interface (tun1).

During handover, the LMA-C receives an updating PBU from the handover target MAG-C2. The PBU refers to a new Data-Plane node (Edge DPN2) to represent the new tunnel endpoints in the downlink and uplink, as required. The LMA-C sends a 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).

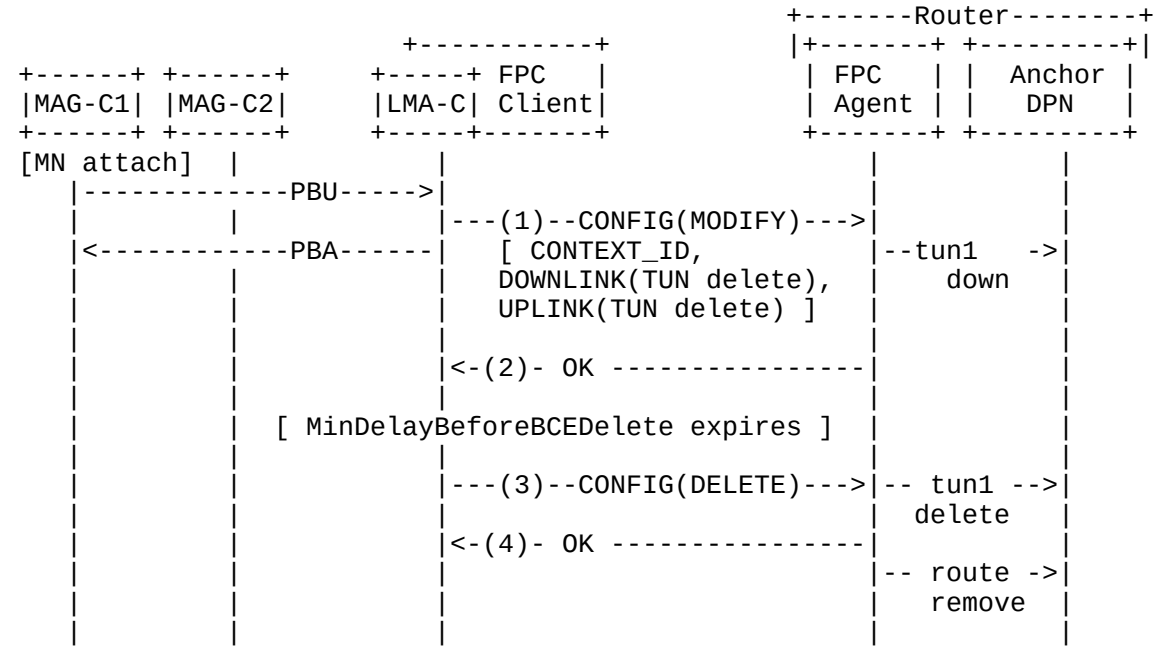


Figure 18: Exemplary Message Sequence (focus on FPC reference point)

When a teardown of the session occurs, MAG-C1 will send a PBU with a lifetime 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 CONFIGURE message, the Agent removes the tunnel configuration and responds to the Client (2). Per [RFC5213], the PBA is sent back immediately after the PBA is received.

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

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

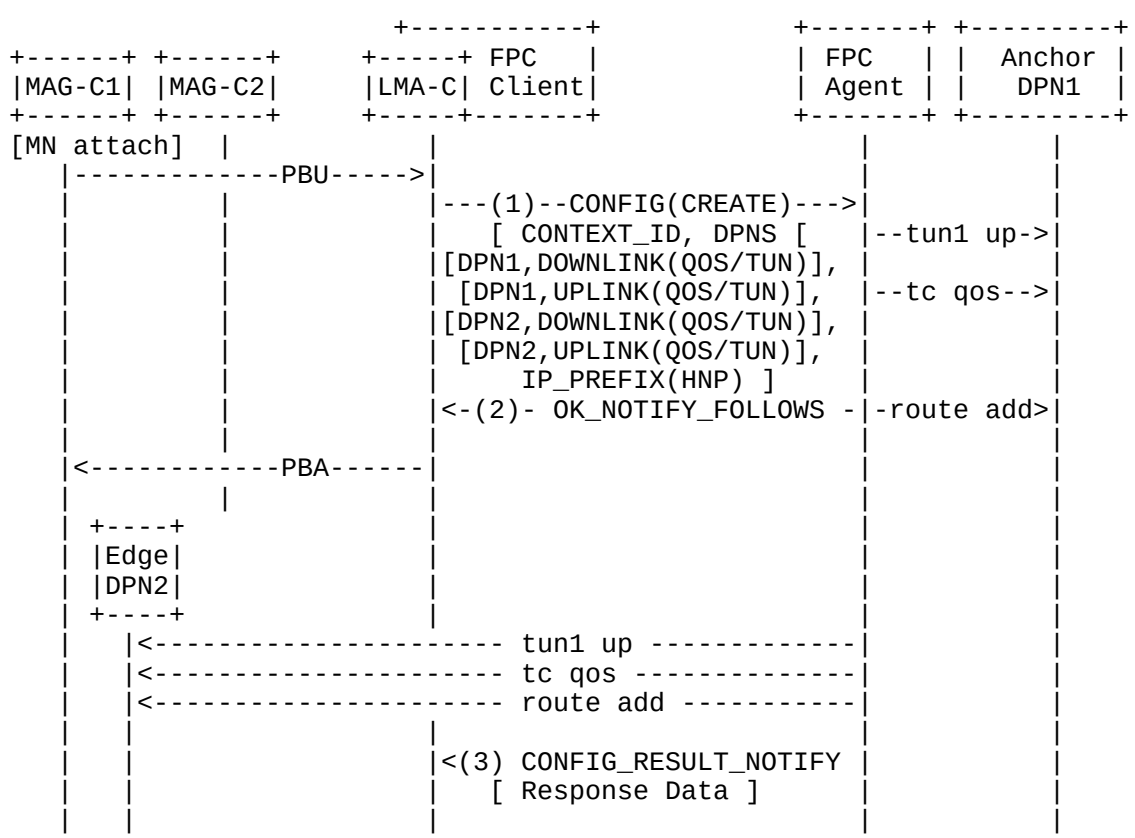


Figure 19: Exemplary Message Sequence for Multi-DPN Agent

Figure 19 shows how the first 2 messages in Figure 17 are supported when a multi-DPN Agent communicates with both Anchor DPN1 and Edge DPN2. In such a case, the FPC Client sends the donwnlink and uplink for both DPNs in the "DPNS" list of the same Context. Message 1 shows the DPNS list with all entries. Each entry identifies the DPN and direction (one of 'uplink', 'downlink' or 'both'). Generally, the 'both' direction is not used for normal mobility session processing. It is commonly used for the instantaition of Policies on a specific DPN (see Section 6.2.4).

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

6.2.2. 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 20 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.

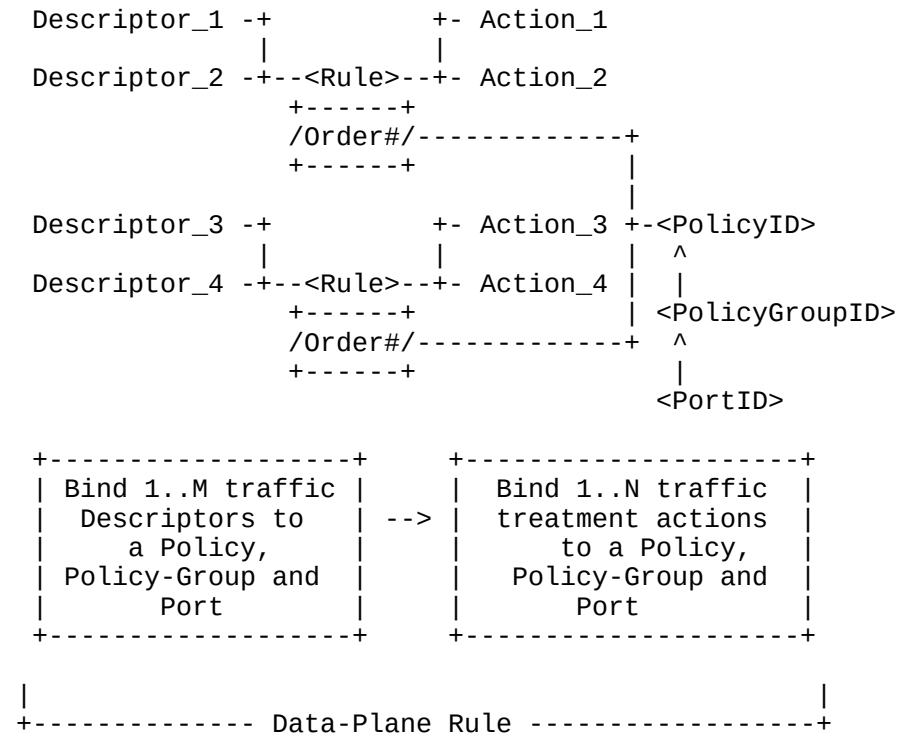


Figure 20: Structure of Policies and Ports

As depicted in Figure 20, the Port represents the anchor of Rules through the Policy-group, Policy, Rule hierarchy configured by any mechanism including RPC or N. A Client and Agent use the identifier of the associated Policy to directly access the Rule and perform modifications of traffic Descriptors or Action references. A Client and Agent use the identifiers to access the Descriptors or Actions to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the treatment Actions specified in the list of properties associated with the 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 21 illustrates the generic context configuration model as used between a FPC Client and a FPC Agent.

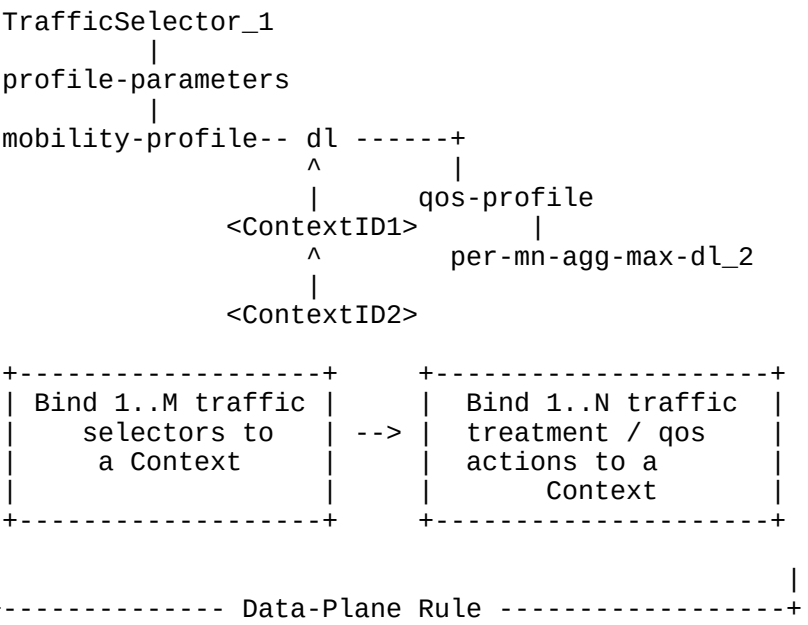


Figure 21: Structure of Contexts

As depicted in Figure 21, the Context represents a mobility session hierarchy. A Client and Agent directly assigns values such as downlink traffic descriptors, QoS information, etc. A Client and Agent use the context identifiers to access the descriptors, qos information, etc. to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic Descriptors and processed according to the qos or other mobility profile related Actions specified in the Context's properties. If present, the final action is to use a Context's tunnel information to encapsulate and forward the packet.

A second Context also references context1 in the figure. Based upon the technology a property in a parent context MAY be inherited by its descendants. This permits concise over the wire representation. When a Client deletes a parent Context all children are also deleted.

6.2.3. Optimization for Current and Subsequent Messages

6.2.3.1. 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 recieveing the

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

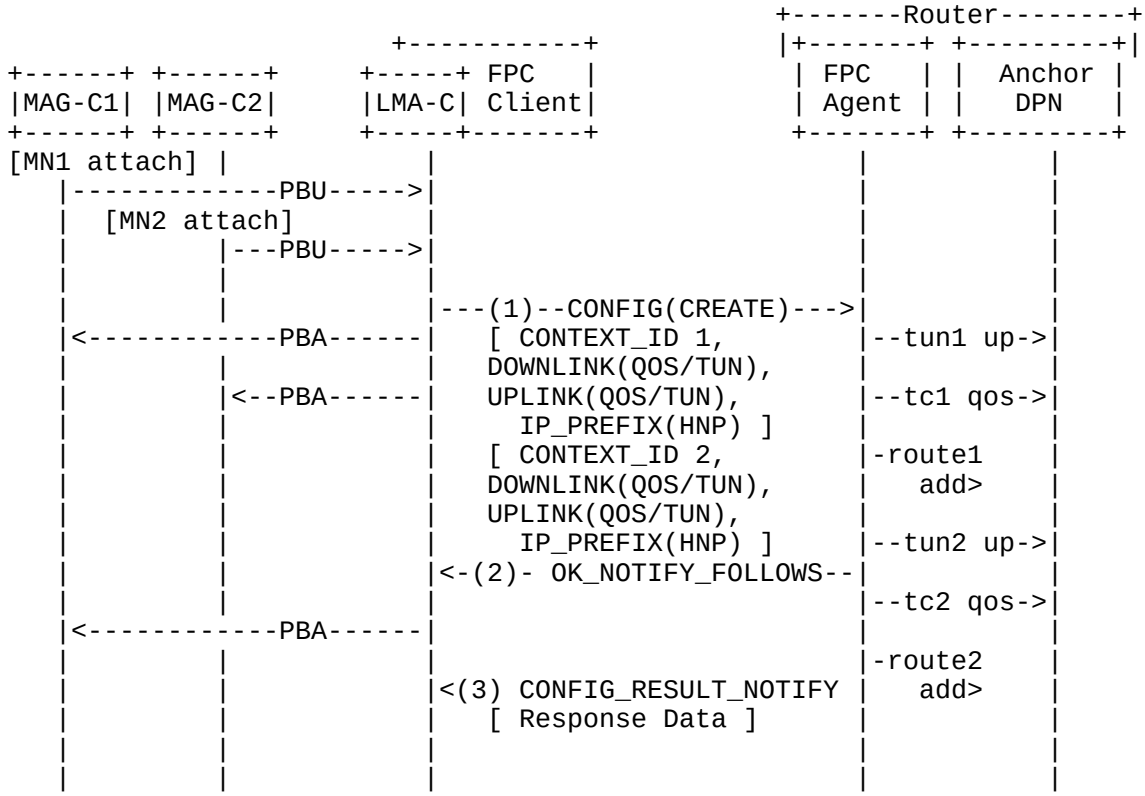


Figure 22: Exemplary Bulk Entity with Asynchronous Notification Sequence (focus on FPC reference point)

6.2.3.2. 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.

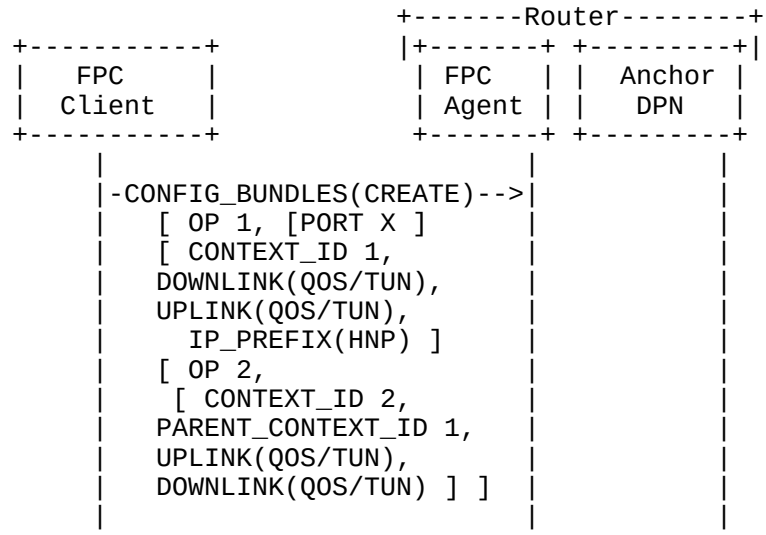


Figure 23: Exemplary Bundle Message (focus on FPC reference point)

6.2.3.3. 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.

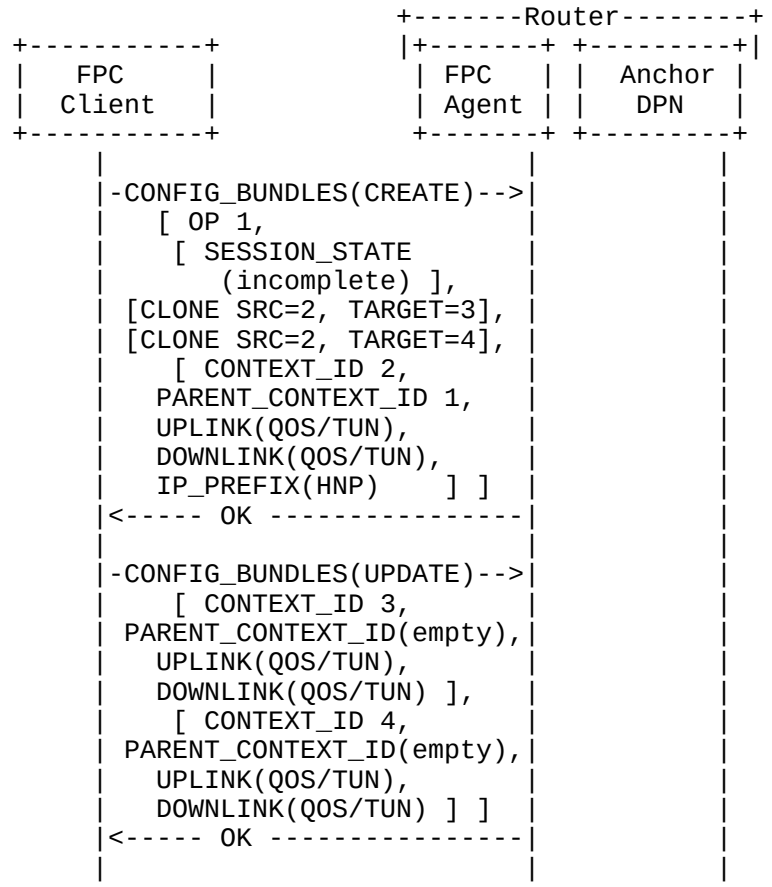


Figure 24: Exemplary Bundle Message (focus on FPC reference point)

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

6.2.3.4. 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.

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

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

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

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

6.2.3.5. 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 hierarchy provided by the Reference Scope

Caches are temporarily created at each level and as the scope includes more caches the amount of entities that are searched increases. Figure 25 shows an example cache where each Cache where a containment hierarchy is provided for all caches.

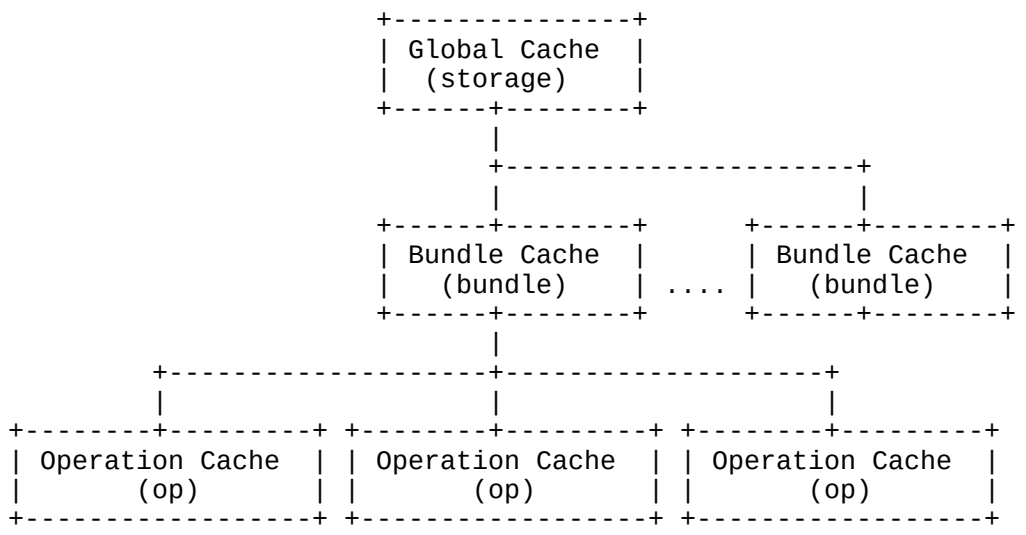


Figure 25: Exemplary Hierarchical Cache

6.2.4. 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 executed. For example, creating tunnels to forward traffic to a trouble management platform and dropping packets to a defective web server can be accomplished via provisioning of 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.

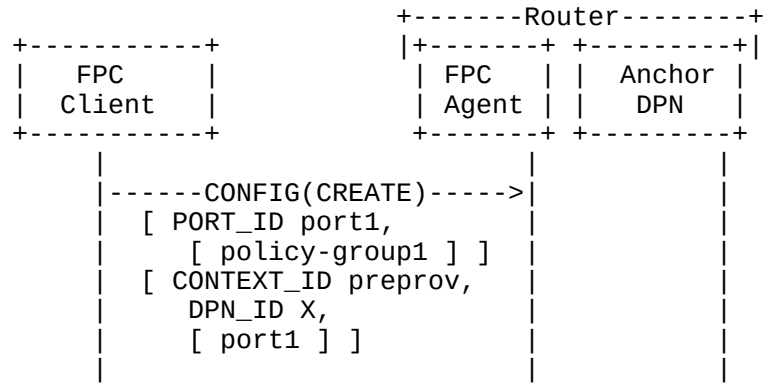


Figure 26: Exemplary Config Message for policy pre-provisioning

6.2.4.1. 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.

7. Protocol Message Details

7.1. Data Structures And Type Assignment

7.1.1. Policy Structures

Structure	Field	Type
ACTION	ACTION_ID	FPC-Identity (Section 5.4)
ACTION	TYPE	[32, unsigned integer]
ACTION	VALUE	Type specific
DESCRIPTOR	DESCRIPTOR_ID	FPC-Identity (Section 5.4)
DESCRIPTOR	TYPE	[32, unsigned integer]
DESCRIPTOR	VALUE	Type specific
POLICY	POLICY_ID	FPC-Identity (Section 5.4)
POLICY	RULES	*[RULE] (See Table 4)
POLICY-GROUP	POLICY_GROUP_ID	FPC-Identity (Section 5.4)
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

7.1.2. Mobilty Structures

Field	Type
PORT_ID	FPC-Identity (Section 5.4)
POLICIES	*[POLICY_GROUP_ID]

Table 5: Port Fields

Field	Type
CONTEXT_ID	FPC-Identity (Section 5.4)
PORTS	*[PORT_ID]
DPN_GROUP_ID	FPC-Identity (Section 5.4)
DELEGATING IP PREFIXES	*[IP_PREFIX]
PARENT_CONTEXT_ID	FPC-Identity (Section 5.4)
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

7.1.3. Topology Structures

Field	Type
DPN_ID	FPC-Identity. See Section 5.4
DPN_NAME	[1024, OCTET STRING]
DPN_GROUPS	* [FPC-Identity] See Section 5.4
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 5.4
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

7.1.4. Monitors

Field	Type	Description
MONITOR	MONITOR_ID TARGET [REPORT_CONFIG]	
MONITOR_ID	FPC-Identity. See Section 5.4	
EVENT_TYPE_ID	[8, Event Type ID]	Event Type (unsigned integer).
TARGET	OCTET STRING (See Section 5.3.3)	
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

7.2. Message Attributes

7.2.1. Header

Each operation contains a header with the following fields:

Field	Type	Messages
CLIENT_ID	FPC-Identity (Section 5.4)	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

7.2.2. 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	FPC Command Bitset. See Section 6.1.1.4.	C,U [NOTE 1]
CLONES	*[FPC-Identity FPC-Identity] (Section 5.4)	C,U [NOTE 1]
PORTS	*[PORT]	C,U
CONTEXTS	*[CONTEXT [COMMAND_SET [NOTE 1]]]	C,U
TARGETS	FPC-Identity (Section 5.4) *[DPN_ID]	Q,D
POLICY_GROUPS	*[POLICY-GROUP]	C,U [NOTE 1]
POLICIES	*[POLICY]	C,U [NOTE 1]
DESCRIPTORS	*[DESCRIPTOR]	C,U [NOTE 1]
ACTIONS	*[ACTION]	C,U [NOTE 1]

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

Table 13: CONF and CONF_BUNDLES OP_BODY Fields

Field	Type	Operation Types Create(C), Update(U), Query(Q) and Delete(D)
PORTS	*[PORT]	C,U [NOTE 2]
CONTEXTS	*[CONTEXT [COMMAND_SET [NOTE 1]]]	C,U [NOTE 2]
TARGETS	*[FPC-Identity (Section 5.4) *[DPN_ID]]	Q,D [NOTE 2]
ERROR_TYPE_ID	[32, unsigned integer]	All [NOTE 3]
ERROR_INFORMATION	[1024, octet string]	All [NOTE 3]

Table 14: Immediate Response RESPONSE_BODY Fields

Notes:

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

NOTE 2 - Present in OK and OK_NOTIFY_FOLLOWS for both CONF and CONF_BUNDLES. MAY also be present in an CONF_BUNDLES Error response (ERR) if one of the operations completed successfully.

NOTE 3 - Present only for Error (ERR) responses.

Field	Type	Description
AGENT_ID	FPC-Identity (Section 5.4)	
NOTIFICATION_ID	[32, unsigned integer]	A Notification Identifier used to determine notification order.
TIMESTAMP	[32, unsigned integer]	The time that the notification occurred.
DATA	*[OP_ID RESPONSE_BODY (Table 14)]	

Table 15: CONFIG_RESULT_NOTIFY Asynchronous Notification Fields

7.2.3. 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

8. 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 5.4.	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.

8.1. 3GPP Specific Extenstions

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

APN-AMBR: Access Point Name Aggregate Maximum Bit Rate

ARP: Allocation of Retention Priority

EBI: EPS Bearer Identity

GBR: Guaranteed Bit Rate

GTP: GPRS (General Packet Radio Service) Tunneling Protocol

IMSI: International Mobile Subscriber Identity

MBR: Maximum Bit Rate

QCI: QoS Class Identifier

TEID: Tunnel Endpoint Identifier.

TFT: Traffic Flow Template (TFT)

UE-AMBR: User Equipment Aggregate Maximum Bit Rate

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

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

Table 22: 3GPP Attributes and Structures

The following COMMAND_SET values are supported for 3GPP.

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

9. Implementation Status

Two FPC Agent implementations have been made to date. The first was based upon Version 03 of the draft and followed Model 1. The second follows Version 04 of the document. Both implementations were OpenDaylight plug-ins developed in Java by Sprint. Version 03 was known as fpcagent and version 04's implementation is simply referred to as 'fpc'.

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

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

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

project was closed in August 2016. fpcagent will no longer be developed and will remain a proprietary implementation.

The learnings of fpcagent has influenced the second project, fpc. Fpc is also an OpenDaylight project but is 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 others developed by the FPC development team have already lead to an order of magnitude improvement.

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

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

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

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

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

Use of Contexts to pre-provision policy has also eliminated any processing of Ports for DPNs which permitted the code for CONFIGURE and 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 2013 Mac laptop. This results in 2x the number of transactions on the southbound interface to a proprietary DPN API on the same machine.

fpc currently supports the following:

1 proprietary DPN API

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

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 required
    if an error occurs undo all assignments in this operation
    return result

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

Figure 27: fpc pseudo code

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

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

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

10. Security Considerations

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

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

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

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

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

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

Nodes under the Mobility Tree are runtime only and manipulated by remote procedure calls. The unwanted deletion or removal of such information would deny users service or provide services ot unauthorized parties.

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

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

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

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

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

CONF and CONF_BUNDLES send Context information which can include information of a sensitive or vulnerable nature in some network environments as described above.

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

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

General usage of FPC MUST consider the following:

FPC Naming Section 5.4 permits arbirtrary string values but a users MUST avoid placing sensitive or vulnerable information in those values.

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

11. IANA Considerations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

12. 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.

13. References

13.1. 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>.

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

13.2. 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-18 (work in progress), October 2016.

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

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

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

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

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

[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>.

Appendix A. YANG Data Model for the FPC protocol

These modules define YANG definitions. Seven modules are defined:

- ietf-dmm-fpc (fpc) - Defines the base model and messages for FPC
- ietf-dmm-fpc-base An FPC submodule that defines the information model that is specified 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. FPC Agent YANG Model

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

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

```
<CODE BEGINS> file "ietf-dmm-fpc@2016-08-03.yang"
module ietf-dmm-fpc {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc";
    prefix fpc;

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

    include ietf-dmm-fpc-base;

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

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

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

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

        Editor:   Satoru Matsushima
```

```
                  <mailto:satoru.matsushima@g.softbank.co.jp>

        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

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

    revision 2016-08-03 {
        description "Initial Revision.";
        reference "draft-ietf-dmm-fpc-cpdp-05";
    }
    feature fpc-cloning {
      description "An ability to support cloning in the RPC.";
    }
    feature fpc-basename-registry {
      description "Ability to track Base Names already provisioned on the Agent"
;
    }
    feature fpc-bundles {
      description "Ability for Client to send multiple bundles of actions to
        an Agent";
    }
    feature fpc-client-binding {
      description "Allows a FPC Client to bind a DPN to an Topology Object";
    }
    feature fpc-auto-binding {
      description "Allows a FPC Agent to advertise Topology Objects that could b
e 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 opt
mize
        the Agent processing.";
```

```
    }
    feature policy-rpc-provisioning {
      description "Enables the ability to send policy elements (Policy Groups, P
olicies,
        Descriptors and Actions) to be sent in CONF or CONF_BUNDLES operations."
;
    }

    typedef agent-identifier {
        type fpc:fpc-identity;
        description "Agent Identifier";
    }

    typedef client-identifier {
        type fpc:fpc-identity;
        description "Client Identifier";
    }

    grouping basename-info {
          leaf basename {
            if-feature fpc:fpc-basename-registry;
            type fpc:fpc-identity;
            description "Rules Basename";
          }
          leaf base-state {
            if-feature fpc:fpc-basename-registry;
            type string;
            description "Current State";
          }
          leaf base-checkpoint {
            if-feature fpc:fpc-basename-registry;
            type string;
            description "Checkpoint";
          }
          description "Basename Information";
    }

    // Top Level Structures
    container tenants {
        list tenant {
            key "tenant-id";
            leaf tenant-id {
                type fpc:fpc-identity;
                description "Tenant ID";
            }

            container fpc-policy {
              list policy-groups {
                  key "policy-group-id";
                  uses fpc:fpc-policy-group;
```

```
                description "Policy Groups";
              }
              list policies {
                  key "policy-id";
                  uses fpc:fpc-policy;
                  description "Policies";
              }
              list descriptors {
                key descriptor-id;
                uses fpc:fpc-descriptor;
                description "Descriptors";
              }
              list actions {
                  key action-id;
                  uses fpc:fpc-action;
                  description "Actions";
              }
              description "Policy";
            }

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

              leaf dpn-id {
```

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

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

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

      // Common Agent Info
      list supported-events {
        key event;
        leaf event {
```

```
          type identityref {
            base "fpc:event-type";
          }
          description "Event Types";
        }
        leaf event-id {
          type fpc:event-type-id;
          description "Event ID";
        }
        description "Supported Events";
      }

      list supported-error-types {
        key error-type;
        leaf error-type {
          type identityref {
            base "fpc:error-type";
          }
          description "Error Type";
        }
        leaf error-type-id {
          type fpc:error-type-id;
          description "Error Type ID";
        }
        description "Supported Error Types";
      }
      description "General Agent Information";
    }


    // Multi-DPN Agent Structures
    grouping fpc-dpn-group {
        leaf dpn-group-id {
            type fpc:fpc-dpn-group-id;
            description "DPN Group ID";
        }
        leaf data-plane-role {
            type identityref {
                base "fpc:fpc-forwaridingplane-role";
            }
            description "Dataplane Role";
        }
        leaf access-type {
            type identityref {
                base "fpc:fpc-access-type";
            }
            description "Access Type";
        }
```

```
        leaf mobility-profile {
            type identityref {
                base "fpc:fpc-mobility-profile-type";
            }
            description "Mobility Profile";
        }
        list dpn-group-peers {
            key "remote-dpn-group-id";
            uses fpc:fpc-dpn-peer-group;
            description "Peer DPN Groups";
        }
        description "FPC DPN Group";
    }


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

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

```
              description "outdated";
            }
        }
        description "Session Status";
    }

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

    typedef op-identifier {
        type uint64;
        description "Operation Identifier";
    }

    typedef ref-scope {
      type enumeration {
        enum none {
          value 0;
          description "no references";
        }
        enum op {
          value 1;
          description "op - All references are contained in the operation body (
intra-op)";
        }
        enum bundle {
          value 2;
          description "bundle - All references in exist in bundle (inter-operati
on/intra-bundle).
          NOTE - If this value comes in CONFIG call it is equivalen to 'op'.";
        }
        enum storage {
          value 3;
          description "storage - One or more references exist outside of the ope
ration and bundle.
          A lookup to a cache / storage is required.";
        }
        enum unknown {
          value 4;
          description " unknown - the location of the references are unknown.  T
his is treated as
          a 'storage' type.";
        }
      }
      description "Search scope for references in the operation.";
    }

    grouping instructions {
      container instructions {
        if-feature instruction-bitset;
```

```
        choice instr-type {
          description "Instruction Value Choice";
        }
        description "Instructions";
      }
      description "Instructions Value";
    }

    grouping op-header {
      leaf client-id {
        type fpc:client-identifier;
        description "Client ID";
      }
      leaf delay {
        type op-delay;
        description "Delay";
      }
      leaf session-state {
        type session-status;
        description "Session State";
      }
      leaf admin-state {
        type admin-status;
        description "Admin State";
      }
      leaf op-type {
        type enumeration {
          enum create {
            value 0;
            description "create";
          }
          enum update {
            value 1;
            description "update";
          }
          enum query {
            value 2;
            description "query";
          }
          enum delete {
            value 3;
            description "delete";
          }
        }
        description "Type";
      }
      leaf op-ref-scope {
          if-feature operation-ref-scope;
```

```
          type fpc:ref-scope;
          description "Reference Scope";
      }
      uses fpc:instructions;
      description "Operation Header";
    }

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

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

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

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

```
        uses fpc:fpc-policy;
        description "Policies";
      }
      list descriptors {
        if-feature fpc:policy-rpc-provisioning;
        key descriptor-id;
        uses fpc:fpc-descriptor;
        description "Descriptors";
      }
      list actions {
        if-feature fpc:policy-rpc-provisioning;
        key action-id;
        uses fpc:fpc-action;
        description "Actions";
      }
      description "Payload";
    }

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

    typedef result {
      type enumeration {
        enum ok {
          value 0;
```

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

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

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

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

    grouping error-info {
          leaf error-type-id {
            type fpc:error-type-id;
            description "Error ID";
          }
```

```
          leaf error-info {
            type string {
              length "1..1024";
            }
            description "Error Detail";
          }
          description "Error Information";
    }

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

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

```
    }

    rpc configure-bundles {
      if-feature fpc:fpc-bundles;
      description "CONF_BUNDLES message";
      input {
        leaf highest-op-ref-scope {
            if-feature operation-ref-scope;
            type fpc:ref-scope;
            description "Highest Op-Ref used in the input";
        }
        list bundles {
          key op-id;
          uses fpc:op-input;
          description "List of operations";
        }
      }
      output {
        list bundles {
          key op-id;
          uses fpc:result-body;
          description "Operation Identifier";
        }
      }
    }

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

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

    notification config-result-notification {
      uses fpc:notification-header;
      choice value {
        case config-result {
```

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

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

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

    }
```

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

    notification notify {
        uses fpc:notification-header;
        choice value {
            case dpn-candidate-available {
              if-feature fpc:fpc-auto-binding;
              leaf node-id {
                type inet:uri;
                description "Topology URI";
              }
              leaf-list access-types {
                type identityref {
                  base "fpc:fpc-access-type";
                }
                description "Access Types";
              }
              leaf-list mobility-profiles {
                type identityref {
                  base "fpc:fpc-mobility-profile-type";
                }
                description "Mobility Profiles";
              }
              leaf-list forwarding-plane-roles {
                type identityref {
                  base "fpc:fpc-forwaridingplane-role";
                }
                description "Forwarding Plane Role";
              }
              description "DPN Candidate Availability";
            }
            case monitor-notification {
              choice monitor-notification-value {
                case simple-monitor {
                  uses fpc:report;
                  description "Report";
```

```
                }
                case bulk-monitors {
                  list reports {
                    uses fpc:report;
                    description "Reports";
                  }
                  description "Bulk Monitor Response";
                }
                description "Monitor Notification value";
              }
              description "Monitor Notification";
            }
            description "Notify Value";
        }
        description "Notify Message";
    }
}
<CODE ENDS>
```

A.2. YANG Models

A.2.1. FPC YANG Model

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

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-dmm-fpc-base@2016-08-03.yang"
submodule ietf-dmm-fpc-base {
    belongs-to ietf-dmm-fpc {
       prefix fpc;
    }

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

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

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

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

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

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

        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

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

    revision 2016-08-03 {
        description "Initial Revision.";
        reference "draft-ietf-dmm-fpc-cpdp-05";
    }

    feature fpc-basic-agent {
        description "This is an agent co-located with a DPN.  In this case
        only DPN Peer Groups, the DPN Id and Control Protocols are exposed
        along with the core structures.";
    }
    feature fpc-multi-dpn {
        description "The agent supports multiple DPNs.";
    }

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

    grouping target-value {
      leaf target {
```

```
          type fpc-identity;
          description "Target Identity";
      }
      description "FPC Target Value";
    }

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

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

```
            leaf all-traffic {
              type empty;
              description "Empty Value";
            }
          }
          description "Descriptor Value";
        }
        description "FPC Descriptor";
    }

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

    // Rule Structure
    grouping fpc-rule {
```

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

    // Policy Structures
    typedef fpc-policy-id {
        type fpc:fpc-identity;
        description "Policy Identifier";
    }
    grouping fpc-policy {
        leaf policy-id {
            type fpc:fpc-policy-id;
            description "Policy Id";
        }
        list rules {
            key order;
            leaf order {
              type uint32;
              description "Rule Order";
            }
            uses fpc:fpc-rule;
            description "Rules";
        }
        description "FPC Policy";
    }

    // Policy Group
    typedef fpc-policy-group-id {
        type fpc:fpc-identity;
```

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

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

    // Context Group
    typedef fpc-context-id {
        type fpc:fpc-identity;
        description "FPC Context Identifier";
    }
    grouping fpc-context-profile {
        leaf tunnel-local-address {
            type inet:ip-address;
            description "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 mtu-size {
            type uint32;
```

```
            description "MTU size";
        }
        container mobility-tunnel-parameters {
            uses fpc:mobility-info;
            description
            "Specifies profile specific uplink tunnel parameters to the DPN
            which the agent exists. The profiles includes GTP/TEID for 3gpp prof
ile,
            GRE/Key for ietf-pmip profile, or new profile if anyone will define
it.";
        }
        container nexthop {
            uses fpc:fpc-nexthop;
            description "Next Hop";
        }
        container qos-profile-parameters {
            uses fpc:fpc-qos-profile;
            description "QoS Parameters";
        }
        container dpn-parameters {
            description "DPN Parameters";
        }
        list vendor-parameters {
            key "vendor-id vendor-type";
            uses fpc:vendor-attributes;
            description "Vendor Parameters";
        }
        description "A profile that applies to a specific direction";
    }

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

    grouping fpc-context {
        leaf context-id {
            type fpc:fpc-context-id;
            description "Context ID";
        }
```

```
        leaf-list ports {
            type fpc:fpc-port-id;
            description "Ports";
        }
        leaf dpn-group {
          type fpc:fpc-dpn-group-id;
          description "DPN Group";
        }
        leaf-list delegating-ip-prefixes {
            type inet:ip-prefix;
            description "Delegating Prefix(es)";
        }
        container ul {
            if-feature fpc:fpc-basic-agent;
            uses fpc:fpc-context-profile;
            description "Uplink";
        }
        container dl {
            if-feature fpc:fpc-basic-agent;
            uses fpc:fpc-context-profile;
            description "Downlink";
        }
        list dpns {
            if-feature fpc:fpc-multi-dpn;
            key "dpn-id direction";
            leaf dpn-id {
                type fpc:fpc-dpn-id;
                description "DPN";
            }
            leaf direction {
                type fpc:fpc-direction;
                mandatory true;
                description "Direction";
            }
            uses fpc:fpc-context-profile;
            description "DPNs";
        }
        leaf parent-context {
            type fpc:fpc-context-id;
            description "Parent Context";
        }
        description "FCP Context";
    }

    // Mobility (Tunnel) Information
    grouping mobility-info {
        choice profile-parameters {
            case nothing {
```

```
              leaf none {
                type empty;
                description "Empty Value";
              }
              description "No Parameters Case";
            }
            description "Mobility Profile Parameters";
        }
        description "Mobility Information";
    }

    // Next Hop Structures
    typedef 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 "fpc:fpc-nexthop-type";
        description "Nexthop IP";
    }
    identity fpc-nexthop-servicepath {
        base "fpc:fpc-nexthop-type";
        description "Nexthop Service Path";
    }
    grouping fpc-nexthop {
        leaf nexthop-type {
            type identityref {
              base "fpc:fpc-nexthop-type";
            }
            description "Nexthop Type";
        }
        choice nexthop-value {
            case ip {
                leaf ip {
                  type inet:ip-address;
                  description "IP Value";
                }
                description "IP Case";
            }
            case servicepath {
                leaf servicepath {
                    type fpc:fpcp-service-path-id;
```

```
                    description "Service Path Value";
                }
                description "Service Path Case";
            }
            description "Value";
        }
        description "Nexthop Value";
    }

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

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

```
            }
            description "Atttribute Value";
        }
        description "Vendor Specific Attributes";
    }

    // Topology
    typedef fpc-domain-id {
        type fpc:fpc-identity;
        description "Domain Identifier";
    }
    grouping fpc-domain {
      leaf domain-id {
        type fpc:fpc-domain-id;
        description "Domain ID";
      }
      leaf domain-name {
        type string;
        description "Domain Name";
      }
      leaf domain-type {
        type string;
        description "Domain Type";
      }
      description "FPC Domain";
    }

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

```
          type instance-identifier;
          description "DPN => Node (Topology) Mapping";
        }
        description "FPC DPN";
    }

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

```
        }
        description "FPC DPN Peer Group";
    }

    // Events, Probes & Notifications
    identity event-type {
        description "Base Event Type";
    }
    typedef event-type-id {
        type uint32;
        description "Event ID Type";
    }

    grouping monitor-id {
      leaf monitor-id {
        type fpc:fpc-identity;
        description "Monitor Identifier";
      }
      description "Monitor ID";
    }

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

    grouping report-config {
      choice event-config-value {
        case periodic-config {
            leaf period {
              type uint32;
              description "Period";
            }
```

```
            description "Periodic Config Case";
          }
          case threshold-config {
              leaf lo-thresh {
                type uint32;
                description "lo threshold";
              }
              leaf hi-thresh {
                type uint32;
                description "hi threshold";
              }
              description "Threshold Config Case";
          }
          case scheduled-config {
              leaf report-time {
                type uint32;
                description "Reporting Time";
              }
              description "Scheduled Config Case";
          }
          case events-config-ident {
              leaf-list event-identities {
                type identityref {
                  base "fpc:event-type";
                }
                description "Event Identities";
              }
              description "Events Config Identities Case";
          }
          case events-config {
              leaf-list event-ids {
                type uint32;
                description "Event IDs";
              }
              description "Events Config Case";
          }
          description "Event Config Value";
        }
        description "Report Configuration";
      }

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

```
    grouping report {
      uses fpc:monitor-config;
      choice report-value {
        leaf trigger {
          type fpc:event-type-id;
          description "Trigger Identifier";
        }
        case simple-empty {
          leaf nothing {
            type empty;
            description "Empty Value";
          }
          description "Empty Case";
        }
        case simple-val32 {
          leaf val32 {
            type uint32;
            description "Unsigned 32 bit value";
          }
          description "Simple Value Case";
        }
        description "Report Value";
      }
      description "Monitor Report";
    }
}
<CODE ENDS>
```

A.2.2. PMIP QoS Model

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

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

```
<CODE BEGINS> file "ietf-pmip-qos@2016-02-10.yang"
module ietf-pmip-qos {
    yang-version 1;

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

    prefix "qos-pmip";

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

```
    }
    import ietf-traffic-selector-types { prefix traffic-selectors; }

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

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

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

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

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

        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

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

    revision 2016-02-10 {
        description "Initial revision";
        reference
         "RFC 7222: Quality-of-Service Option for Proxy Mobile IPv6";
    }

    // Type Definitions

    // QoS Option Field Type Definitions
```

```
        typedef sr-id {
                type uint8;
            description
             "An 8-bit unsigned integer used
              for identifying the QoS Service Request.  Its uniqueness is within
              the scope of a mobility session.  The local mobility anchor always
              allocates the Service Request Identifier.  When a new QoS Service
              Request is initiated by a mobile access gateway, the Service
              Request Identifier in the initial request message is set to a
              value of (0), and the local mobility anchor allocates a Service
              Request Identifier and includes it in the response.  For any new
              QoS Service Requests initiated by a local mobility anchor, the
              Service Request Identifier is set to the allocated value.";
          }

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

    typedef operational-code {
        type enumeration {
                enum RESPONSE {
          value 0;
          description "Response to a QoS request";
        }
                enum ALLOCATE {
          value 1;
          description "Request to allocate QoS resources";
        }
                enum DE-ALLOCATE {
          value 2;
          description "Request to de-Allocate QoS resources";
        }
                enum MODIFY {
          value 3;
          description "Request to modify QoS parameters for a previously negotia
ted
                 QoS Service Request";
        }
```

```
                enum QUERY {
          value 4;
          description "Query to list the previously negotiated QoS Service Reque
sts
                 that are still active";
        }
                enum NEGOTIATE {
          value 5;
          description "Response to a QoS Service Request with a counter QoS prop
osal";
        }
        }
        description
                "1-octet Operational code indicates the type of QoS request.
              Reserved values:   (6) to (255)
                Currently not used.  Receiver MUST ignore the option received
                with any value in this range.";
    }

    // QoS Attribute Types

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

```
          description "Aggregate Maximum Downlink Bit Rate.";
        }
        enum Aggregate-Max-UL-Bit-Rate {
          value 7;
          description "Aggregate Maximum Uplink Bit Rate.";
        }
        enum Guaranteed-DL-Bit-Rate {
          value 8;
          description "Guaranteed Downlink Bit Rate.";
        }
        enum Guaranteed-UL-Bit-Rate {
          value 9;
          description "Guaranteed Uplink Bit Rate.";
        }
        enum QoS-Traffic-Selector {
          value 10;
          description "QoS Traffic Selector.";
        }
        enum QoS-Vendor-Specific-Attribute {
          value 11;
          description "QoS Vendor-Specific Attribute.";
        }
        }
                description
                "8-bit unsigned integer indicating the type of the QoS
              attribute.  This specification reserves the following reserved val
ues.
              (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 mobi
lity session.  The measurement
                                  units for Per-Session-Agg-Max-UL/DL-Bit-Rate a
re bits per second.";
                }
                leaf service-flag {
                        type boolean;
                        mandatory true;
                        description
                                "This flag is used for extending the scope of th
e
                              target flows for Per-Session-Agg-Max-UL/DL-Bit-Rat
e from(UL)/to(DL) the mobile
                              node's other mobility sessions sharing the same Se
rvice
                              Identifier. 3GPP Access Point Name (APN) is an exa
mple 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), the
n the Per-
                                 Session-Agg-Max-Bit-Rate is measured as an aggr
egate across
```

```
                                   all the mobile node’s other mobility sessions s
haring the same
                                   Service Identifier associated with this mobilit
y session.

                                *  When the (S) flag is set to a value of (0), the
n the target
                                   flows are limited to the current mobility sessi
on.

                                *  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/do
wnlink
                               flows for which the network is providing Guarantee
d-Bit-Rate
                               service be excluded from the target IP flows for w
hich Per-
                               Session-Agg-Max-UL/DL-Bit-Rate is measured.

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

                               *  When the (E) flag is set to a value of (0), the
n 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 mobilit
y session from
                                  the target flows for which Per-Session-Agg-Max-
UL/DL-Bit-Rate is
                                  measured.";
                }
    description "Per-Session-Agg-Max-Bit-Rate Value";
        }

        grouping Allocation-Retention-Priority-Value {
                leaf prioirty-level {
                        type uint8 {
                                range "0..15";
                        }
                        mandatory true;
                        description
```

"This is a 4-bit unsigned integer value. It is used to decide whether a mobility session establishment or modification request can be accepted; this is typically used for

```
                                admission control of Guaranteed Bit Rate traffic i
n 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 relat
ive timeliness
                                of a resource request.
                                Values 1 to 15 are defined, with value 1 as the hi
ghest 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 resourc
es that are
                                authorized by the home network and thus applicable
 when a mobile
                                node is roaming.";
                        }
                        leaf premption-capability {
                                type enumeration {
                                        enum enabled {
                  value 0;
                  description "enabled";
                }
                                        enum disabled {
                  value 1;
                  description "disabled";
                }
                                        enum reserved1 {
                  value 2;
                  description "reserved1";
                }
                                        enum reserved2 {
                  value 3;
                  description "reserved2";
                }
                                }
                                mandatory true;
                                description
                                        "This is a 2-bit unsigned integer
                                value.  It defines whether a service data flow can
 get resources
                                that were already assigned to another service data
 flow with a
                                lower priority level.  The following values are de
fined:

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

                                   Disabled (1): This value indicates that the ser
vice data flow
                                   is not allowed to get resources that were alrea
```

dy assigned to

```
                                another IP data flow with a lower priority leve
l.  The values
                                (2) and (3) are reserved.";
                }
                leaf premption-vulnerability {
                        type enumeration {
                                enum enabled {
          value 0;
          description "enabled";
        }
                                enum disabled {
          value 1;
          description "disabled";
        }
                                enum reserved1 {
          value 2;
          description "reserved1";
        }
                                enum reserved2 {
          value 3;
          description "reserved2";
        }
                        }
                        mandatory true;
                        description
                                "This is a 2-bit unsigned integer
                              value.  It defines whether a service data flow can
 lose the
                              resources assigned to it in order to admit a servi
ce data flow
                              with a higher priority level.  The following value
s are defined:

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

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

        typedef Aggregate-Max-DL-Bit-Rate-Value {
                type uint32;
                description
                        "This is a 32-bit unsigned integer that
                indicates the aggregate maximum downlink bit rate that is
                requested/allocated for downlink IP flows.  The measurement unit
s
                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 downli
nk
                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 Manageme
nt
                      Information) Network Management Private Enterprise Code of
 the
                      IANA-maintained 'Private Enterprise Numbers' registry [SMI
].";
                        reference
                                "'PRIVATE ENTERPRISE NUMBERS', SMI Network Manag
ement
                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-sp
ecific
                      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 mod
ule

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

                  //All of the sub-types by constraint
        choice attribute-choice {
            case per-mn-agg-max-dl-case {
                when "../attributetype = 'Per-MN-Agg-Max-DL-Bit-Rate-type'";
                leaf per-mn-agg-max-dl {
                    type qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value;
                    description "Per-MN-Agg-Max-DL-Bit-Rate Value";
                }
                description "Per-MN-Agg-Max-DL-Bit-Rate Case";
            }
            case per-mn-agg-max-ul-case {
                when "../attributetype = 'Per-MN-Agg-Max-UL-Bit-Rate-type'";
                leaf per-mn-agg-max-ul {
                    type qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value;
                    description "Per-MN-Agg-Max-UL-Bit-Rate Value";
                }
                description "Per-MN-Agg-Max-UL-Bit-Rate Case";
            }
            case per-session-agg-max-dl-case {
                when "../attributetype = 'Per-Session-Agg-Max-DL-Bit-Rate-type'"
;
                container per-session-agg-max-dl {
                    uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                    description "Per-Session-Agg-Max-Bit-Rate Value";
                }
                description "Per-Session-Agg-Max-Bit-Rate Case";
            }
            case per-session-agg-max-ul-case {
                when "../attributetype = 'Per-Session-Agg-Max-UL-Bit-Rate-type'"
;
                container per-session-agg-max-ul {
                    uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
```

```
                    description "Per-Session-Agg-Max-Bit-Rate Value";
                }
                description "Per-Session-Agg-Max-Bit-Rate Case";
            }
            case allocation-retention-priority-case {
                when "../attributetype = 'Allocation-Retention-Priority-type'";
                uses qos-pmip:Allocation-Retention-Priority-Value;
                description "Allocation-Retention-Priority Case";
            }
            case agg-max-dl-case {
                when "../attributetype = 'Aggregate-Max-DL-Bit-Rate-type'";
                leaf agg-max-dl {
                    type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
                    description "Aggregate-Max-DL-Bit-Rate Value";
                }
                description "Aggregate-Max-DL-Bit-Rate Case";
            }
            case agg-max-ul-case {
                when "../attributetype = 'Aggregate-Max-UL-Bit-Rate-type'";
                leaf agg-max-ul {
                    type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
                    description "Aggregate-Max-UL-Bit-Rate Value";
                }
                description "Aggregate-Max-UL-Bit-Rate Case";
            }
            case gbr-dl-case {
                when "../attributetype = 'Guaranteed-DL-Bit-Rate-type'";
                leaf gbr-dl {
                    type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
                    description "Guaranteed-DL-Bit-Rate Value";
                }
                description "Guaranteed-DL-Bit-Rate Case";
            }
            case gbr-ul-case {
                when "../attributetype = 'Guaranteed-UL-Bit-Rate-type'";
                leaf gbr-ul {
                    type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
                    description "Guaranteed-UL-Bit-Rate Value";
                }
                description "Guaranteed-UL-Bit-Rate Case";
            }
            case traffic-selector-case {
                when "../attributetype = 'QoS-Traffic-Selector-type'";
                container traffic-selector {
                    uses traffic-selectors:traffic-selector;
                    description "traffic selector";
                }
                description "traffic selector Case";
```

```
            }
            description "Attribute Value";
        }
        description "PMIP QoS Attribute";
    }

        grouping qosoption {
        leaf srid {
            type sr-id;
            mandatory true;
            description "Service Request Identifier";
        }
        leaf trafficclass {
            type traffic-class;
            mandatory true;
            description "Traffic Class";
        }
        leaf operationcode {
            type operational-code;
            mandatory true;
            description "Operation Code";
        }
        list attributes {
            unique "attributetype";
            uses qosattribute;
            min-elements 1;
            description "Attributes";
        }
        description "PMIP QoS Option";
        }
}
<CODE ENDS>
```

A.2.3. Traffic Selectors YANG Model

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

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-traffic-selector-types@2016-01-14.yang"
module ietf-traffic-selector-types {
    yang-version 1;

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

    prefix "traffic-selectors";
```

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

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

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

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

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

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

        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

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

    revision 2016-01-14 {
        description "Updated for IETF-PACKET-FIELDS module alignment";
        reference
                "draft-ietf-netmod-acl-model-06";
    }

    revision 2016-01-12 {
```

```
        description "Initial revision";
        reference
                "RFC 6088: Traffic Selectors for Flow Bindings";
    }

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

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

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

        // Type definitions and groupings
        typedef ipsec-spi {
                type uint32;
                description "This type defines the first 32-bit IPsec Security P
arameter
                                Index (SPI) value on data packets sent from a co
rresponding
                                node to the mobile node as seen by the home agen
t. This field
                                is defined in [RFC4303].";
                        reference
                                "RFC 4303: IP Encapsulating Security Payload (ES
P)";
        }

        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 t
o 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 t
he 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 n
eeds to be matched, then
                                this field can be used to indicate the end value o
f a range
                                starting from the value of the Start SPI field.  T
his field
                                MUST NOT be included unless the Start SPI field is
 included
                                    and has a value less than or equal to this fie
ld.

                                When this field is included, the receiver will mat
ch 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 p
ackets 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-p
ort";
                        }
                                description
                                        "If more than one contiguous source port
 number needs to be
                                matched, then this field can be used to indicate t
he end value of
                                a range starting from the value of the Start Port
field.
                                This field MUST NOT be included unless the Start P
ort 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 d
ata packets sent
                                   from a corresponding node to the mobile node as se
en 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 t
he end value of
                                   a range starting from the value of the Start Desti
nation Port
                                   field.  This field MUST NOT be included unless the
 Start
                                   Port field is included and has a value less than o
r 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 a
ddress.";
                            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 packe
ts sent from a
                                corresponding node to the mobile node as seen by t
he home agent.
                                In other words, this is one of the addresses of th
e correspondent
                                node.";
                        }
                        leaf end-address {
                                type inet:ipv4-address;
                                description
                                        "If more than one contiguous source addr
ess needs to be matched,
                                then this field can be used to indicate the end va
lue of a range
                                starting from the value of the Start Address field
.  This
                                field MUST NOT be included unless the Start Addres
s field
                                is included.  When this field is included, the rec
eiver will match
                                all of the addresses between fields start-address
and
                                    end-address, inclusive of start-address and en
d-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 a
ddress.";
                        leaf start-address {
                                type inet:ipv4-address;
                                mandatory true;
                                description
                                        "This field identifies the first destina
tion address, from the
                                range of 32-bit IPv4 addresses to be matched, on d
ata packets sent
                                from a corresponding node to the mobile node as se
en by the home
                                agent.  In other words, this is one of the registe
red 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 t
he end value of
                                a range starting from the value of the Start Desti
nation Address
                                field.  This field MUST NOT be included unless the
 Start
                                Address field is included.  When this field is inc
luded, the receiver
```

```
                                    will match all of the addresses between fields
 start-address and
                                    end-address, inclusive of start-address and en
d-address.";
                        }
          }
          container ds-range {
              presence "Enables setting dscp range";
```

```
              description
                "Inclusive range representing DiffServ Codepoints to be used. Wh
en
                         only start-ds is present, it represents a single Codepo
int.";
              leaf start-ds {
                        type inet:dscp;
                        mandatory true;
                        description
                                "This field identifies the first differential se
rvices value, from
                       the range of differential services values to be matched, o
n 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 tha
t the
                       field has been redefined as a 6-bit DS field with 2 bits r
eserved,
                       later claimed by Explicit Congestion Notification (ECN) [R
FC3168].
                       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 MUS
T 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-d
s";
                         }
                                 description
                                         "If more than one contiguous DS value ne
eds to be matched, then
                                    this field can be used to indicate the end value o
f a range
                                    starting from the value of the Start DS field.  Th
is field MUST
                                    NOT be included unless the Start DS field is inclu
ded.  When this
                                    field is included, it MUST be coded the same way a
s defined for
                                    start-ds.  When this field is included, the receiv
er will match all of
                                    the values between fields start-ds and end-ds, inc
lusive 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 se
nt from
                       a corresponding node to the mobile node as seen by the hom
e agent.";
```

```
                }
                leaf end-protocol {
                        type uint8;
                must ". >= ../start-protocol" {
                  error-message
                    "The end-protocol must be greater than or equal to start-pro
tocol";
                }
                        description
                                "If more than one contiguous protocol value need
s 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 in
cluded.
                       When this field is included, the receiver will match all o
f the
                       values between fields start-protocol and end-protocol, inc
lusive
                           of start-protocol and end-protocol.";
                }
           }
    description "ipv4 binary traffic selector";
  }

        grouping ipv6-binary-traffic-selector {
         container source-address-range-v6 {
              presence "Enables setting source IPv6 address range";
              description
                "Inclusive range representing IPv6 addresses to be used. When
                         only start-address is present, it represents a single a
ddress.";
                       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 pack
ets sent from a
                                corresponding node to the mobile node as seen by t
he home agent.
                                In other words, this is one of the addresses of th
e correspondent
                                node.";
                       }
                       leaf end-address {
                               type inet:ipv6-address;
                               description
                                       "If more than one contiguous source addr
ess needs to be matched,
                                then this field can be used to indicate the end va
lue of a range
                                starting from the value of the Start Address field
.  This
                                field MUST NOT be included unless the Start Addres
s 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 a
ddress.";
                        leaf start-address {
                                type inet:ipv6-address;
                                mandatory true;
                                description
                                        "This field identifies the first destina
tion 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 re
gistered 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 t
he end value of
                                a range starting from the value of the Start Addre
ss field. This
                                    field MUST NOT be included unless the Start Ad
dress 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 l
abel.";
                leaf start-flow-label {
                        type inet:ipv6-flow-label;
                        description
                                "This field identifies the first flow label valu
e, from the range
                    of flow label values to be matched, on data packets sent f
rom a
                    corresponding node to the mobile node as seen by the home
agent.
                    According to [RFC2460], the flow label is 24 bits long.  F
or the
                    purpose of this specification, the sender of this option M
UST
                    prefix the flow label value with 8 bits of '0' before inse
rting 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-f
low-label";
                }
                        description
                                "If more than one contiguous flow label value ne
eds 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.  Th
is 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-lab
el
                         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 tra
ffic class.";
                leaf start-traffic-class {
                        type inet:dscp;
                        description
                                "This field identifies the first traffic class v
alue, from the
                     range of traffic class values to be matched, on data packe
ts sent
                     from a corresponding node to the mobile node as seen by th
e 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 cl
aimed 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 ma
tched
                     and the 2 least significant bits' values MUST be ignored i
n any
                     comparison.";
                        reference
                                "RFC 3260: New Terminology and Clarifications fo
r Diffserv
                                 RFC 3168: The Addition of Explicit Congestion N
otification (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 star
t-traffic-class";
                }
                    description
                            "If more than one contiguous TC value needs to b
e matched, then
                      this field can be used to indicate the end value of a rang
e
                      starting from the value of the Start TC field.  This field
 MUST
```

```
                        NOT be included unless the Start TC field is included.  Wh
en this
                        field is included, it MUST be coded the same way as define
d for
                            start-traffic-class.  When this field is included, the
 receiver
                            will match all of the values between fields start-traf
fic-class
                            and end-traffic-class, inclusive of start-traffic-clas
s 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 head
er 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 th
e 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 n
eeds 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 incl
uded.
                        When this field is included, the receiver will match all o
f the
                        values between fields start-next-header and end-next-heade
r,
                            inclusive of start-next-header and end-next-header.";
                }
          }
      description "ipv6 binary traffic selector";
         }

         grouping traffic-selector {
                leaf ts-format {
                        type identityref {
                                base traffic-selector-format;
                        }
                        description "Traffic Selector Format";
                }
```

```
                uses traffic-selector-base {
```

```
                        when "boolean(../ts-format/text() = 'ipv6-binary-selecto
r-format') | boolean(../ts-format/text() = 'ipv4-binary-selector-format')";
                }
                uses ipv4-binary-traffic-selector {
                        when "boolean(../ts-format/text() = 'ipv4-binary-selecto
r-format')";
                }
                uses ipv6-binary-traffic-selector {
                        when "boolean(../ts-format/text() = 'ipv6-binary-selecto
r-format')";
                }
                description
                        "The traffic selector includes the parameters used to ma
tch
                           packets for a specific flow binding.";
                reference
                        "RFC 6089: Flow Bindings in Mobile IPv6 and Network Mobi
lity (NEMO) Basic Support";
  }

        grouping ts-list {
                list selectors {
                        key index;
                        leaf index {
                                type uint64;
        description "index";
                        }
                        uses traffic-selector;
      description "traffic selectors";
                }
    description "traffic selector list";
        }
}
<CODE ENDS>
```

A.2.4. FPC 3GPP Mobility YANG Model

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

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

```
<CODE BEGINS> file "ietf-dmm-threegpp@2016-08-03.yang"
module ietf-dmm-threegpp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-threegpp";
    prefix threegpp;

    import ietf-inet-types { prefix inet; revision-date 2013-07-15; }
    import ietf-dmm-fpc { prefix fpc; revision-date 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 Distributed Mobility Management (DMM)
```

```
      Working Group";

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

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

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

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

        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

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

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

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

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

```
         description "3GPP Mobility Profile";
    }

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

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

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

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

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

    // QoS Profile
    identity threeGPP-qos-profile-parameters {
        base "fpc:fpc-qos-type";
        description "3GPP QoS Profile";
    }

    typedef fpc-qos-class-identifier {
        type uint8 {
            range "1..9";
        }
        description "QoS Class Identifier (QCI)";
```

```
    }

    grouping threeGPP-QoS {
        description "3GPP QoS Attributes";
        leaf qci {
            type fpc-qos-class-identifier;
            description "QCI";
        }
        leaf gbr {
            type uint32;
            description "Guaranteed Bit Rate";
        }
        leaf mbr {
            type uint32;
            description "Maximum Bit Rate";
        }
        leaf apn-ambr {
            type uint32;
            description "Access Point Name Aggregate Max Bit Rate";
        }
        leaf ue-ambr {
            type uint32;
            description "User Equipment Aggregate Max Bit Rate";
        }
        container arp {
            uses pmipqos:Allocation-Retention-Priority-Value;
            description "Allocation Retention Priority";
        }
    }

    typedef ebi-type {
      type uint8 {
        range "0..15";
      }
      description "EUTRAN Bearere Identifier (EBI) Type";
    }

   // From 3GPP TS 24.008 version 13.5.0 Release 13
   typedef component-type-enum {
        type enumeration {
            enum ipv4RemoteAddress {
              value 16;
              description "IPv4 Remote Address";
            }
            enum ipv4LocalAddress  {
              value 17;
              description "IPv4 Local Address";
            }
```

```
            enum ipv6RemoteAddress {
              value 32;
              description "IPv6 Remote Address";
            }
            enum ipv6RemoteAddressPrefix {
              value 33;
              description "IPv6 Remote Address Prefix";
            }
            enum ipv6LocalAddressPrefix {
              value 35;
              description "IPv6 Local Address Prefix";
            }
            enum protocolNextHeader {
              value 48;
              description "Protocol (IPv4) or NextHeader (IPv6) value";
            }
            enum localPort {
              value 64;
              description "Local Port";
            }
            enum localPortRange {
              value 65;
              description "Local Port Range";
            }
            enum reomotePort {
              value 80;
              description "Remote Port";
            }
            enum remotePortRange {
              value 81;
              description "Remote Port Range";
            }
            enum secParamIndex {
              value 96;
              description "Security Parameter Index (SPI)";
            }
            enum tosTraffClass {
              value 112;
              description "TOS Traffic Class";
            }
            enum flowLabel {
              value 128;
              description "Flow Label";
            }
        }
        description "TFT Component Type";
    }
```

```
    typedef packet-filter-direction {
        type enumeration {
          enum preRel7Tft {
            value 0;
            description "Pre-Release 7 TFT";
          }
          enum uplink {
            value 1;
            description "uplink";
          }
          enum downlink {
            value 2;
            description "downlink";
          }
          enum bidirectional {
            value 3;
            description "bi-direcitonal";
          }
        }
        description "Packet Filter Direction";
    }

    typedef component-type-id {
        type uint8 {
          range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 | 80 | 81 | 96 | 112 | 12
8";
        }
        description "Specifies the Component Type";
    }

    grouping packet-filter {
      leaf direction {
          type threegpp:packet-filter-direction;
          description "Filter Direction";
      }
      leaf identifier {
          type uint8 {
            range "1..15";
          }
          description "Filter Identifier";
      }
      leaf evaluation-precedence {
          type uint8;
          description "Evaluation Precedence";
      }
      list contents {
        key component-type-identifier;
        description "Filter Contents";
        leaf component-type-identifier {
```

```
            type threegpp:component-type-id;
            description "Component Type";
          }
          choice value {
            case ipv4-local {
              leaf ipv4-local {
                type inet:ipv4-address;
                description "IPv4 Local Address";
              }
            }
            case ipv6-prefix-local {
              leaf ipv6-prefix-local {
                type inet:ipv6-prefix;
                description "IPv6 Local Prefix";
              }
            }
            case ipv4-ipv6-remote {
              leaf ipv4-ipv6-remote {
                type inet:ip-address;
                description "Ipv4 Ipv6 remote address";
              }
            }
            case ipv6-prefix-remote {
              leaf ipv6-prefix-remote {
                type inet:ipv6-prefix;
                description "IPv6 Remote Prefix";
              }
            }
            case next-header {
              leaf next-header {
                type uint8;
                description "Next Header";
              }
            }
            case local-port {
              leaf local-port {
                type inet:port-number;
                description "Local Port";
              }
            }
            case local-port-range {
              leaf local-port-lo {
                type inet:port-number;
                description "Local Port Min Value";
              }
              leaf local-port-hi {
                type inet:port-number;
                description "Local Port Max Value";
```

```
            }
          }
          case remote-port {
            leaf remote-port {
              type inet:port-number;
              description "Remote Port";
            }
          }
          case remote-port-range {
            leaf remote-port-lo {
              type inet:port-number;
              description "Remote Por Min Value";
            }
            leaf remote-port-hi {
              type inet:port-number;
              description "Remote Port Max Value";
            }
          }
          case ipsec-index {
            leaf ipsec-index {
              type traffic-selectors:ipsec-spi;
              description "IPSec Index";
            }
          }
          case traffic-class {
            leaf traffic-class {
              type inet:dscp;
              description "Traffic Class";
            }
          }
          case traffic-class-range {
              leaf traffic-class-lo {
                type inet:dscp;
                description "Traffic Class Min Value";
              }
              leaf traffic-class-hi {
                type inet:dscp;
                description "Traffic Class Max Value";
              }
          }
          case flow-label-type {
            leaf-list flow-label {
              type inet:ipv6-flow-label;
              description "Flow Label";
            }
          }
          description "Component Value";
        }
```

```
      }
      description "Packet Filter";
    }

    grouping tft {
      list packet-filters {
          key identifier;
          uses threegpp:packet-filter;
          description "List of Packet Filters";
      }
      description "Packet Filter List";
    }

    typedef imsi-type {
        type uint64;
        description "International Mobile Subscriber Identity (IMSI) Value Type"
;
    }

    typedef threegpp-instr {
      type bits {
        bit assign-ip {
          position 0;
          description "Assign IP Address/Prefix";
        }
        bit assign-fteid-ip {
          position 1;
          description "Assign FTEID-IP";
        }
        bit assign-fteid-teid {
          position 2;
          description "Assign FTEID-TEID";
        }
        bit session {
          position 3;
          description "Commands apply to the Session Level";
        }
        bit uplink {
          position 4;
          description "Commands apply to the Uplink";
        }
        bit downlink {
          position 5;
          description "Commands apply to the Downlink";
        }
        bit assign-dpn {
          position 6;
          description "Assign DPN";
        }
```

```
      }
      description "Instruction Set for 3GPP R11";
    }

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

    // Contexts Update - Contexts / UL / mob-profile
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:ul/fpc:mo
bility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Context UL Tunnel";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Create Context UL Tunnel";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:profile-parame
ters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Bundles Create Context UL Tunnel";
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:create-or-update-succ
ess/fpc:contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Create Context UL Tunnel Response";
    }
    augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:result-type/fpc:c
reate-or-update-success/fpc:contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:p
rofile-parameters" {
      case threegpp-tunnel {
```

```
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
        }
        description "Bundles Create Context UL Tunnel Response";
      }

      // Contexts Update - Contexts / DL / mob-profile
      augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:dl/fpc:mo
bility-tunnel-parameters/fpc:profile-parameters" {
        case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
        }
        description "Context DL Tunnel";
      }
      augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
        case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
        }
        description "Bundles Create Context DL Tunnel";
      }
      augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:profile-parame
ters" {
        case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
        }
        description "Bundles Create Context DL Tunnel";
      }
      augment "/fpc:configure/fpc:output/fpc:result-type/fpc:create-or-update-succ
ess/fpc:contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
        case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
        }
        description "Create Context DL Tunnel Response";
      }
      augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:result-type/fpc:c
reate-or-update-success/fpc:contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:p
rofile-parameters" {
        case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
        }
        description "Bundles Create Context DL Tunnel Response";
      }
```

```
    // Contexts Update -  Contexts / dpns / mobility-tunnel-parameters
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:dpns/fpc:
mobility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Context 3GPP TFT and Tunnel Information";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Configure 3GPP TFT and Tunnel Information";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:profile-para
meters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Configure Bundles 3GPP TFT and Tunnel Information";
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:create-or-update-succ
ess/fpc:contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:profile-parameters"
 {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Configure 3GPP TFT and Tunnel Information Response";
    }
    augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:result-type/fpc:c
reate-or-update-success/fpc:contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc
:profile-parameters" {
      case threegpp-tunnel {
          uses threegpp:threeGPP-tunnel;
          uses threegpp:tft;
          description "3GPP TFT and Tunnel Information";
      }
      description "Configure Bundles 3GPP TFT and Tunnel Information Response";
    }


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

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

    // QoS Updates -  Context / DL / QoS Profile
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:dl/fpc:qo
s-profile-parameters/fpc:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
          description "3GPP QoS Values";
      }
      description "Context DL 3GPP QoS Values";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
      case threegpp-qos {
          uses threegpp:threeGPP-QoS;
          description "3GPP QoS Values";
      }
      description "Configure Context DL 3GPP QoS Values";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
```

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

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

    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts" {
      uses threegpp:threegpp-properties;
      description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts" {
      uses threegpp:threegpp-properties;
      description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts" {
       uses threegpp:threegpp-properties;
       description "3GPP Mobility Session Properties";
```

```
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:create-or-update-succ
ess/fpc:contexts" {
      uses threegpp:threegpp-properties;
      description "3GPP Mobility Session Properties";
    }
    augment "/fpc:configure-bundles/fpc:output/fpc:bundles/fpc:result-type/fpc:c
reate-or-update-success/fpc:contexts" {
       uses threegpp:threegpp-properties;
       description "3GPP Mobility Session Properties";
    }

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

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


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

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

A.2.5. FPC / PMIP Integration YANG Model

This module defines the integration between FPC and PMIP models.

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

```
<CODE BEGINS> file "ietf-dmm-fpc-pmip@2016-01-19.yang"
module ietf-dmm-fpc-pmip {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-pmip";
    prefix fpc-pmip;

    import ietf-dmm-fpc { prefix fpc; }
    import ietf-pmip-qos { prefix qos-pmip; }
    import ietf-traffic-selector-types { prefix traffic-selectors; }

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

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

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

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

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

```
        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

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

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

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

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

    identity pmip-tunnel-type {
        description "PMIP Tunnel Type";
    }
    identity grev1 {
        base "fpc-pmip:pmip-tunnel-type";
```

```
        description "GRE v1";
    }
    identity grev2 {
        base "fpc-pmip:pmip-tunnel-type";
        description "GRE v2";
    }
    identity ipinip {
        base "fpc-pmip:pmip-tunnel-type";
        description "IP in IP";
    }
    grouping pmip-mobility {
        leaf type {
            type identityref {
                base "fpc-pmip:pmip-tunnel-type";
            }
            description "PMIP Mobility";
        }
        choice value {
            case gre {
                leaf key {
                    type uint32;
                    description "GRE_KEY";
                }
                description "GRE Value";
            }
            description "PMIP Mobility value";
        }
        description "PMIP Mobility Value";
    }

    typedef pmip-instr {
      type bits {
        bit assign-ip {
          position 0;
          description "Assign IP";
        }
        bit assign-dpn {
          position 1;
          description "Assign DPN";
        }
        bit session {
          position 2;
          description "Session Level";
        }
        bit uplink {
          position 3;
          description "Uplink";
        }
```

```
        bit downlink {
          position 4;
          description "Downlink";
        }
      }
      description "Instruction Set for PMIP";
    }

    // Descriptors update - goes to Entities, Configure and Configure Bundles
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-policy/fpc:descriptors/fpc:descript
or-value" {
      case pmip-selector {
          uses traffic-selectors:traffic-selector;
          description "PMIP Selector";
      }
      description "Policy Descriptor";
    }

    // Contexts Update - Contexts / UL / mob-profile, Contexts / DL / mob-profil
e and Contexts / dpns / mobility-tunnel-parameters
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:ul/fpc:mo
bility-tunnel-parameters/fpc:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "Context UL Mobility";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "CONF Context UL Mobility";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:ul/fpc:mobility-tunnel-parameters/fpc:profile-parame
ters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "CONF_BUNDLES Context UL Mobility";
    }

    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:dl/fpc:mo
bility-tunnel-parameters/fpc:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
```

```
      }
      description "Context DL Mobility";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "CONF Context DL Mobility";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:dl/fpc:mobility-tunnel-parameters/fpc:profile-parame
ters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "CONF_BUNDLES Context DL Mobility";
    }

    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:dpns/fpc:
mobility-tunnel-parameters/fpc:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "Context DPN Mobility";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:profile-parameters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "CONF Context DPN Mobility";
    }
    augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:dpns/fpc:mobility-tunnel-parameters/fpc:profile-para
meters" {
      case pmip-tunnel {
        uses fpc-pmip:pmip-mobility;
        uses traffic-selectors:traffic-selector;
        description "PMIP Tunnel Information";
      }
      description "CONF_BUNDLES Context DPN Mobility";
    }

    // QoS Updates - Context / UL / qosprofile, Context / DL / QoS Profile
    augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:ul/fpc:qo
s-profile-parameters/fpc:value" {
      case qos-pmip {
```

```
          uses qos-pmip:qosattribute;
          description "PMIP QoS Information";
        }
        description "Context UL QoS";
      }
      augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
        case qos-pmip {
          uses qos-pmip:qosattribute;
          description "PMIP QoS Information";
        }
        description "CONF Context UL QoS";
      }
      augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:ul/fpc:qos-profile-parameters/fpc:value" {
        case qos-pmip {
          uses qos-pmip:qosattribute;
          description "PMIP QoS Information";
        }
        description "CONF_BUNDLES Context UL QoS";
      }

      augment "/fpc:tenants/fpc:tenant/fpc:fpc-mobility/fpc:contexts/fpc:dl/fpc:qo
s-profile-parameters/fpc:value" {
        case qos-pmip {
          uses qos-pmip:qosattribute;
          description "PMIP QoS Information";
        }
        description "Context DL QoS";
      }
      augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
        case qos-pmip {
          uses qos-pmip:qosattribute;
          description "PMIP QoS Information";
        }
        description "CONF Context DL QoS";
      }
      augment "/fpc:configure-bundles/fpc:input/fpc:bundles/fpc:op_body/fpc:create
_or_update/fpc:contexts/fpc:dl/fpc:qos-profile-parameters/fpc:value" {
        case qos-pmip {
          uses qos-pmip:qosattribute;
          description "PMIP QoS Information";
        }
        description "CONF_BUNDLES Context DL QoS";
      }

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

```
    }

    // Instructions Update - OP BODY, Context, Port
    augment "/fpc:configure/fpc:input/fpc:instructions/fpc:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
        description "PMIP Commandset";
      }
      description "CONF Instructions";
    }
    augment "/fpc:configure/fpc:input/fpc:op_body/fpc:create_or_update/fpc:conte
xts/fpc:instructions/fpc:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
        description "PMIP Commandset";
      }
      description "CONF Context Instructions";
    }
    augment "/fpc:configure/fpc:output/fpc:result-type/fpc:create-or-update-succ
ess/fpc:contexts/fpc:instructions/fpc:instr-type" {
      case pmip-instr {
        uses fpc-pmip:pmip-commandset;
        description "PMIP Commandset";
      }
      description "CONF Result Context Instructions";
    }

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

A.2.6. FPC Policy Extension YANG Model

This module defines extensions to FPC policy structures.

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

```
<CODE BEGINS> file "ietf-dmm-fpc-policyext@2016-08-03.yang"
module ietf-dmm-fpc-policyext {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-policyext";
    prefix fpcpolicyext;

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

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

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

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

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

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

        Editor:   Lyle Bertz
                  <mailto:lyleb551144@gmail.com>";

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

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

     This document is subject to BCP 78 and the IETF Trust's Legal
     Provisions Relating to IETF Documents
     (http://trustee.ietf.org/license-info) in effect on the date of
     publication of this document. Please review these documents
     carefully, as they describe your rights and restrictions with
     respect to this document. Code Components extracted from this
```

```
     document must include Simplified BSD License text as described
     in Section 4.e of the Trust Legal Provisions and are provided
     without warranty as described in the Simplified BSD License.";

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

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

    identity nat-service {
        base "service-function";
        description "NAT Service";
    }
    grouping simple-napt {
      leaf source-port {
        type inet:port-number;
        description "Source Port";
      }
      leaf outbound-napt-address {
        type inet:ip-address;
        description "Outbound NAPT Address";
      }
      leaf destination-port {
        type inet:port-number;
        description "Destination Port";
      }
      description "Simple NAPT Configuration";
    }

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

```
    }
    grouping copy-forward {
      container destination {
        choice value {
          case port-ref {
            leaf port-ref {
              type fpc:fpc-port-id;
              description "Port";
            }
            description "Port Forward Case";
          }
          case context-ref {
            leaf context-ref {
              type fpc:fpc-context-id;
              description "Context";
            }
            description "Context Forward Case";
          }
          description "Copy Forward Value";
        }
        description "destination";
      }
      description "Copy Then Forward to Port/Context Action";
    }

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

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

```
            description "Rule of source IP";
        }
        description
        "Traffic descriptor group collects parameters to
         identify target traffic flow.  It represents
         source/destination as IP prefixes";
    }

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

A.3. FPC nformation Model YANG Tree

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

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

```
|     |     +--rw action-id      fpc:fpc-action-id-type
|     |     +--rw action-type    identityref
|     |     +--rw (action-value)?
|     |        +--:(drop)
|     |           +--rw drop?          empty
|     +--ro fpc-mobility
|     |  +--ro contexts* [context-id]
|     |  |  +--ro context-id                fpc:fpc-context-id
|     |  |  +--ro ports*                    fpc:fpc-port-id
|     |  |  +--ro dpn-group?                fpc:fpc-dpn-group-id
|     |  |  +--ro delegating-ip-prefixes*   inet:ip-prefix
|     |  |  +--ro ul {fpc:fpc-basic-agent}?
|     |  |  |  +--ro tunnel-local-address?         inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
|     |  |  |  +--ro mtu-size?                     uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?   empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?   identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip)
|     |  |  |  |     |  +--ro ip?             inet:ip-address
|     |  |  |  |     +--:(servicepath)
|     |  |  |  |        +--ro servicepath?    fpc: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      fpc:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?    empty
|     |  |  +--ro dl {fpc:fpc-basic-agent}?
|     |  |  |  +--ro tunnel-local-address?         inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
|     |  |  |  +--ro mtu-size?                     uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?   empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?   identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip)
```

```
|     |  |  |  |     |  +--ro ip?             inet:ip-address
|     |  |  |  |     +--:(servicepath)
|     |  |  |  |        +--ro servicepath?    fpc: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      fpc:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?    empty
|     |  |  +--ro dpns* [dpn-id direction] {fpc:fpc-multi-dpn}?
|     |  |  |  +--ro dpn-id                        fpc:fpc-dpn-id
|     |  |  |  +--ro direction                     fpc:fpc-direction
|     |  |  |  +--ro tunnel-local-address?         inet:ip-address
|     |  |  |  +--ro tunnel-remote-address?        inet:ip-address
|     |  |  |  +--ro mtu-size?                     uint32
|     |  |  |  +--ro mobility-tunnel-parameters
|     |  |  |  |  +--ro (profile-parameters)?
|     |  |  |  |     +--:(nothing)
|     |  |  |  |        +--ro none?   empty
|     |  |  |  +--ro nexthop
|     |  |  |  |  +--ro nexthop-type?   identityref
|     |  |  |  |  +--ro (nexthop-value)?
|     |  |  |  |     +--:(ip)
|     |  |  |  |     |  +--ro ip?             inet:ip-address
|     |  |  |  |     +--:(servicepath)
|     |  |  |  |        +--ro servicepath?    fpc: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      fpc:fpc-identity
|     |  |  |     +--ro vendor-type    identityref
|     |  |  |     +--ro (value)?
|     |  |  |        +--:(empty-type)
|     |  |  |           +--ro empty-type?    empty
|     |  |  +--ro parent-context?           fpc:fpc-context-id
|     |  +--ro ports* [port-id]
|     |  |  +--ro port-id          fpc:fpc-port-id
|     |  |  +--ro policy-groups*   fpc:fpc-policy-group-id
|     |  +--ro monitors*
|     |     +--ro monitor-id?         fpc:fpc-identity
|     |     +--ro target?             fpc-identity
|     |     +--ro (event-config-value)?
```

```
   |     |        +--:(periodic-config)
   |     |        |  +--ro period?             uint32
   |     |        +--:(threshold-config)
   |     |        |  +--ro lo-thresh?          uint32
   |     |        |  +--ro hi-thresh?          uint32
   |     |        +--:(scheduled-config)
   |     |        |  +--ro report-time?        uint32
   |     |        +--:(events-config-ident)
   |     |        |  +--ro event-identities*   identityref
   |     |        +--:(events-config)
   |     |           +--ro event-ids*          uint32
   |     +--rw fpc-topology
   |        +--rw domains* [domain-id]
   |        |  +--rw domain-id          fpc:fpc-domain-id
   |        |  +--rw domain-name?       string
   |        |  +--rw domain-type?       string
   |        |  +--rw basename?          fpc:fpc-identity {fpc:fpc-basename-regis
try}?
   |        |  +--rw base-state?        string {fpc:fpc-basename-registry}?
   |        |  +--rw base-checkpoint?   string {fpc:fpc-basename-registry}?
   |        +--rw dpn-group-peers* [remote-dpn-group-id] {fpc:fpc-basic-agent}?
   |        |  +--rw remote-dpn-group-id        fpc:fpc-dpn-group-id
   |        |  +--rw remote-mobility-profile?   identityref
   |        |  +--rw remote-data-plane-role?    identityref
   |        |  +--rw remote-endpoint-address?   inet:ip-address
   |        |  +--rw local-endpoint-address?    inet:ip-address
   |        |  +--rw mtu-size?                  uint32
   |        +--rw dpn-id?              fpc:fpc-dpn-id {fpc:fpc-basic-agent}?
   |        +--rw control-protocols*   identityref {fpc:fpc-basic-agent}?
   |        +--rw dpn-groups* [dpn-group-id] {fpc:fpc-multi-dpn}?
   |        |  +--rw dpn-group-id        fpc:fpc-dpn-group-id
   |        |  +--rw data-plane-role?    identityref
   |        |  +--rw access-type?        identityref
   |        |  +--rw mobility-profile?   identityref
   |        |  +--rw dpn-group-peers* [remote-dpn-group-id]
   |        |  |  +--rw remote-dpn-group-id        fpc:fpc-dpn-group-id
   |        |  |  +--rw remote-mobility-profile?   identityref
   |        |  |  +--rw remote-data-plane-role?    identityref
   |        |  |  +--rw remote-endpoint-address?   inet:ip-address
   |        |  |  +--rw local-endpoint-address?    inet:ip-address
   |        |  |  +--rw mtu-size?                  uint32
   |        |  +--rw domains* [domain-id]
   |        |     +--rw domain-id          fpc:fpc-domain-id
   |        |     +--rw domain-name?       string
   |        |     +--rw domain-type?       string
   |        |     +--rw basename?          fpc:fpc-identity {fpc:fpc-basename-re
gistry}?
   |        |     +--rw base-state?        string {fpc:fpc-basename-registry}?
   |        |     +--rw base-checkpoint?   string {fpc:fpc-basename-registry}?
   |        +--rw dpns* [dpn-id] {fpc:fpc-multi-dpn}?
```

```
   |           +--rw dpn-id            fpc:fpc-dpn-id
   |           +--rw dpn-name?         string
   |           +--rw dpn-groups*       fpc:fpc-dpn-group-id
   |           +--rw node-reference?   instance-identifier
   +--rw fpc-agent-info
      +--rw supported-features*      string
      +--rw supported-events* [event]
      |  +--rw event       identityref
      |  +--rw event-id?   fpc:event-type-id
      +--rw supported-error-types* [error-type]
         +--rw error-type       identityref
         +--rw error-type-id?   fpc:error-type-id
rpcs: ...
```

Figure 28: YANG FPC Agent Tree

Authors' Addresses

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

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

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: 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

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

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

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

Abstract

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

Status of This Memo

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

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

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

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

Table of Contents

1. Introduction

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

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

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

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

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

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

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

2. Terminology

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

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

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

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

3. FPC Design Objectives and Deployment

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

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

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

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

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

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

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

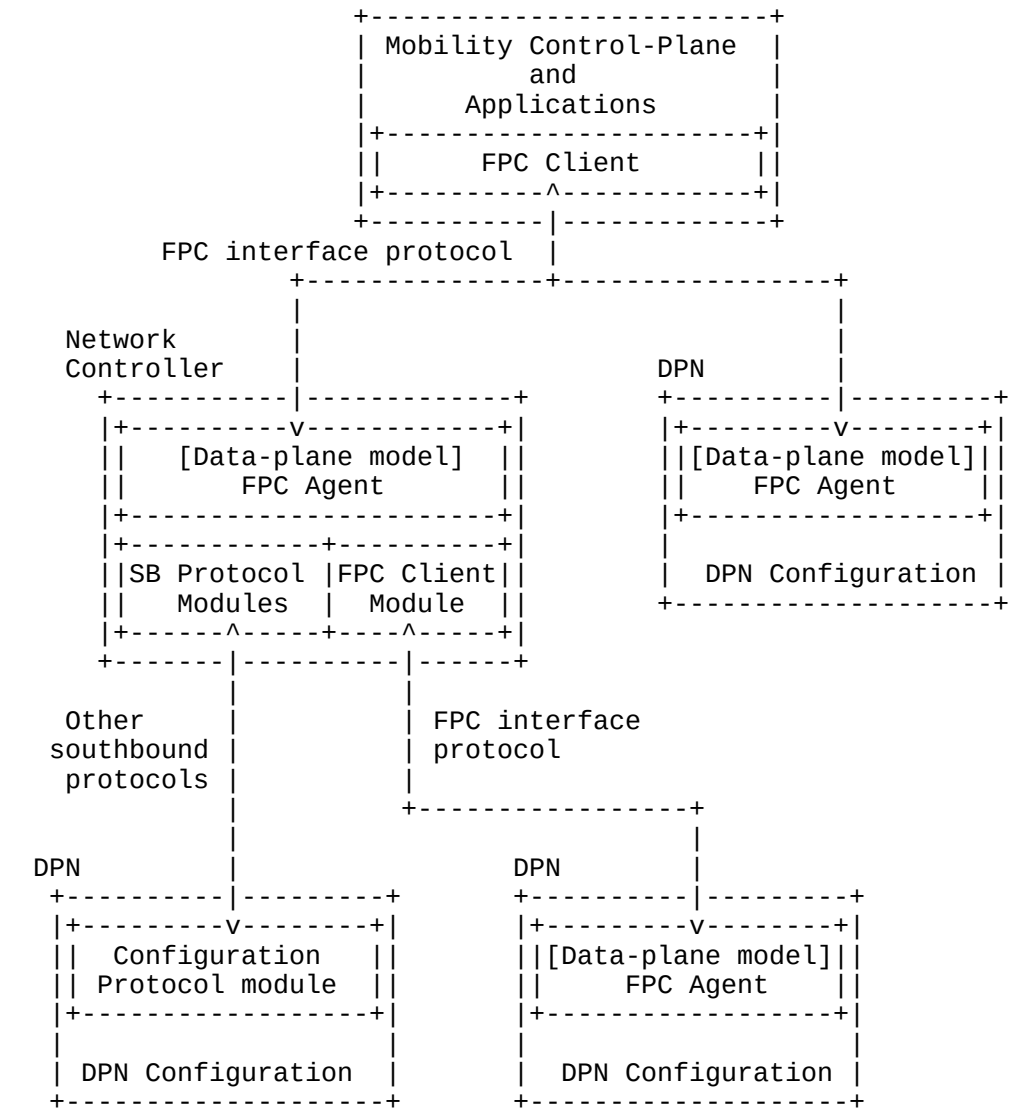


Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

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

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

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

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

4. FPC Mobility Information Model

The FPC information model includes the following components:

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

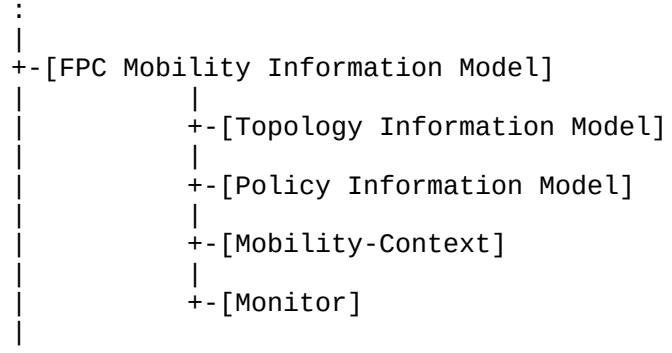


Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

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

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

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

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

Figure 3: Model Notation - An Example

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

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

Set: A nonempty collection of entities without duplications

Name: A human-readable string

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

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

G-Key: A key unique within a Tenant

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

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

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

Figure 4

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

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

EntityA definition references an entityB model element.

EntityB model elements are indexed by entityB-Key.

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

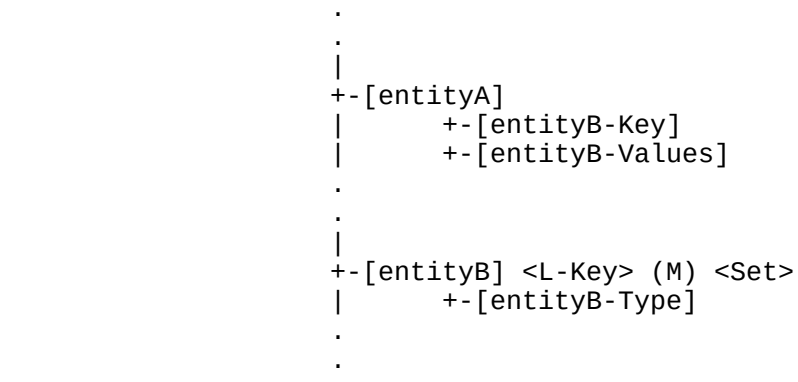


Figure 5: Indexed sets of entities

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

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

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

4.2. Templates and Attributes

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

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

Figure 6: Template entities

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

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

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

Version: Provides a version tag for the Template.

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

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

4.3. Attribute-Expressions

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

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

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

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

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

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

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

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

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

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

4.4. Attribute Value Types

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

4.5. Namespace and Format

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

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

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

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

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

4.6. Configuring Attribute Values

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

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

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

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

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

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

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

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

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

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

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

Entity blocks describe the format of the policy configurations.

4.7. Entity Configuration Blocks

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

assign new values if the Attribute is not static

make attributes static if they were not

make an attribute mandatory

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

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

Figure 7: Entity Configuration Block

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

Descriptor-Configuration

Action-Configuration

Rule-Configuration

Interface-Configuration

Service-Group-Configuration

Domain-Policy-Configuration

DPN-Policy-Configuration

Policy-Configuration

MN-Policy-Configuration

Service-Data-Flow-Policy-Configuration

4.8. Information Model Checkpoint

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

Checkpoints are defined for the following information model components:

Service-Group

DPN Information Model

Domain Information Model

Policy Information Model

4.9. Information Model Components

4.9.1. Topology Information Model

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

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

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

Figure 8: Topology Structure

4.9.2. Service-Group

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

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

Figure 9: Service Group

Each Service-Group element contains the following information:

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

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

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

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

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

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

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

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

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

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

4.9.3. Domain Information Model

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

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

Figure 10: Domain Information Model

Each Domain entry contains the following information:

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

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

4.9.4. DPN Information Model

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

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

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

DPN-Name: A human-readable display string.

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

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

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

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

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

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

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

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

4.9.5. Policy Information Model

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

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

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

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

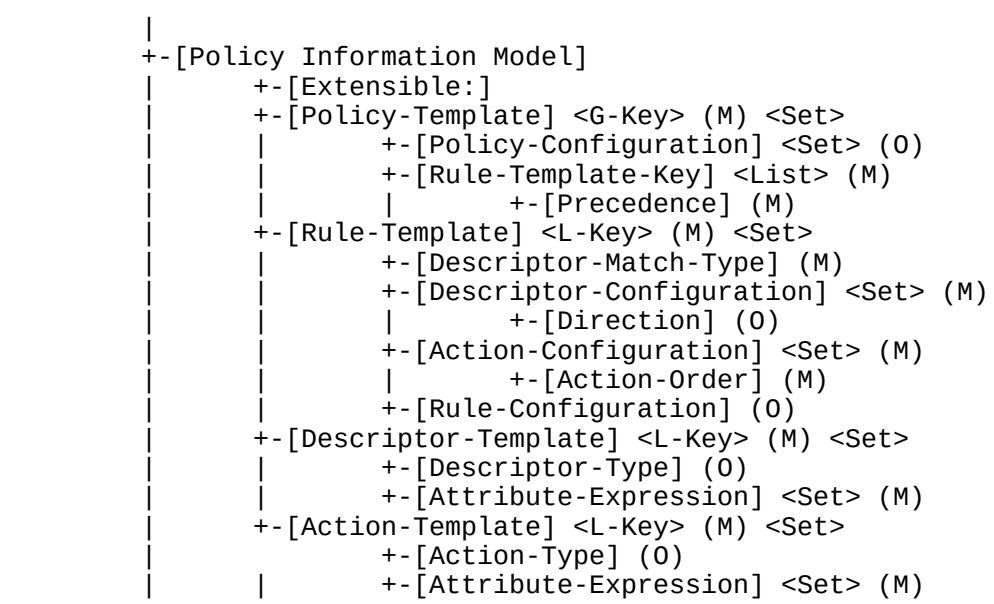


Figure 12: Policy Information Model

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.9.6. Mobility-Context Information Model

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

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

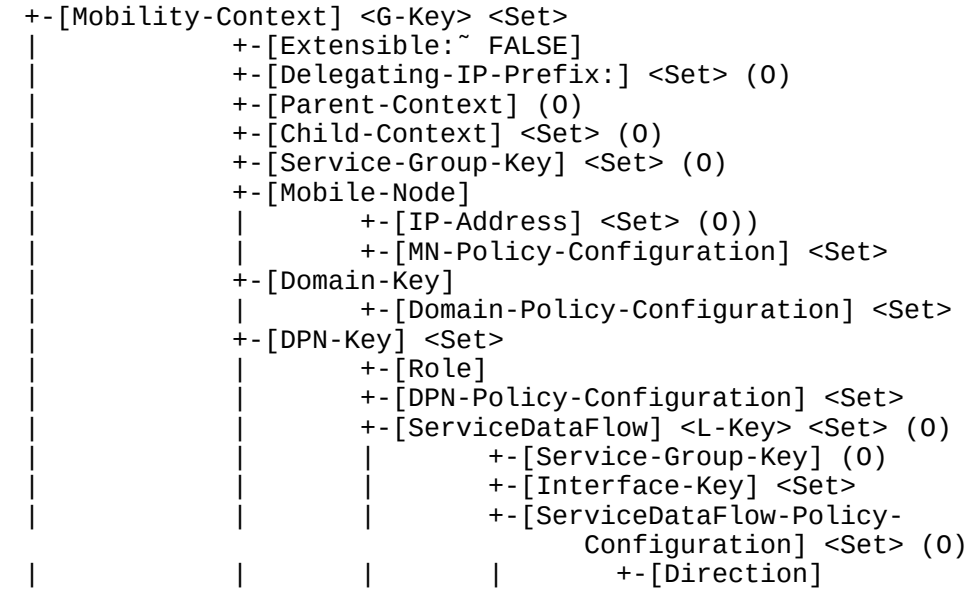


Figure 13: Mobility-Context Information Model

The Mobility-Context Substructure holds the following entries:

Mobility-Context-Key: Identifies a Mobility-Context

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

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

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

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

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

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

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

Domain-Key: Enables access to a Domain instance.

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

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

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

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

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

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

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

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

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

4.9.7. Monitor Information Model

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

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

```
                          |
                          +-[Monitor] <G-Key> <List>
                          |         +-[Extensible:]
                          |         +-[Target:]
                          |         +-[Deferrable]
                          |         +-[Configuration]
```

Figure 14: Monitor Substructure

Monitor-Key: Identifies the Monitor.

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

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

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

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

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

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

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

5. Protocol

5.1. Protocol Messages and Semantics

Four Client to Agent messages are supported.

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

Table 1: Client to Agent Messages

Each message contains a header with the following information:

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

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

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

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

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

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

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

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

Error-Tag: An error tag.

Error-App-Tag: Application specific error tag.

Error-Message: A message describing the error.

Error-Info: Any data required for the response.

```
                      |
                      +-[Errors] <List>
                      |         +-[(Enumeration) Error-Type ]
                      |         +-[(String) Error-Tag ]
                      |         +-[(String) Error-App-Tag ] (O)
                      |         +-[(String) Error-Message ] (O)
                      |         +-[Error-Info] (O)
```

Figure 15: Error Information Model

Two Agent to Client notifications are supported.

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

Table 2: Agent to Client Messages (notifications)

5.1.1. Configure Message

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

5.1.1.1. Edit Operation Types

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

Table 3: Configure Edit Operations

5.1.1.2. Edit Operation

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

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

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

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

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

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

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

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

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

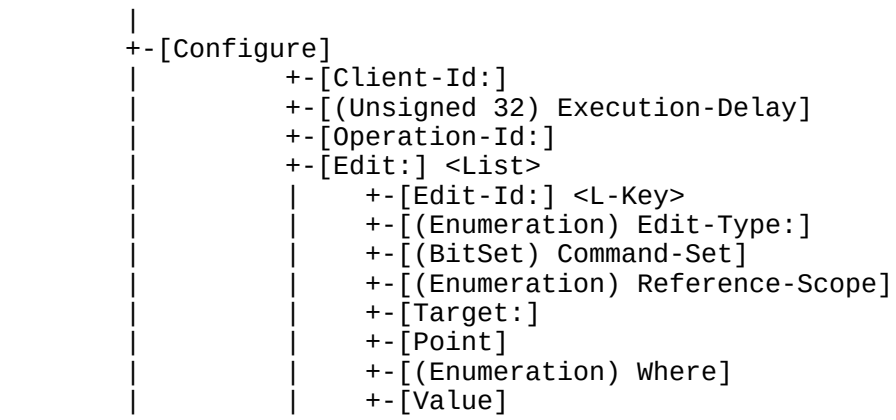


Figure 16: Configure Request

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

1. Action Templates

2. Descriptor Templates

3. Rule Templates

4. Policy Templates

5. DPN Templates

6. Mobility Contexts

5.1.1.3. Reference Scope

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

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

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

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

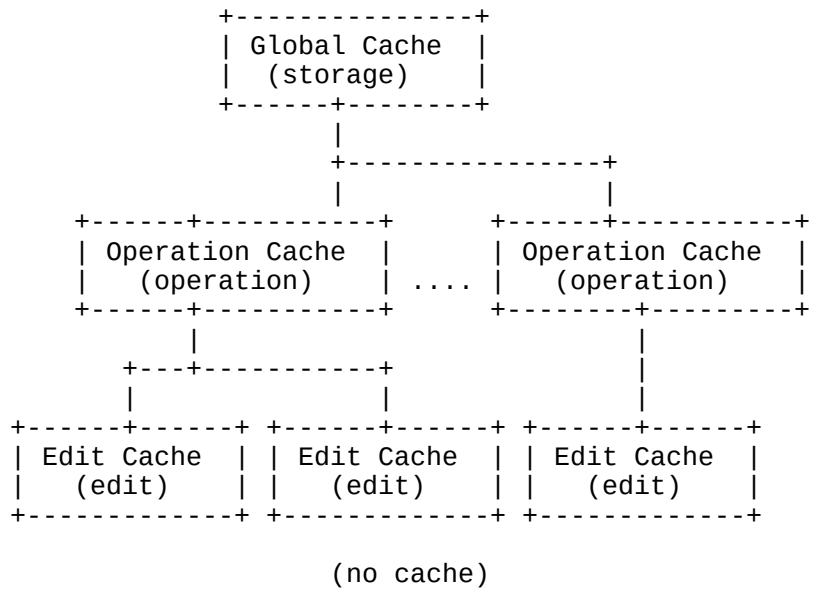


Figure 17: Example Hierarchical Cache

5.1.1.4. Operation Response

5.1.1.4.1. Immediate Response

The Response MUST include the following:

Operation Identifier of the corresponding request.

Global Status for the operation (see Table 1).

A list of Edit results (described below).

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

Edit-Id: Edit Identifier.

Edit-Status: OK.

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

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

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

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

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

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

Figure 18: Configure Operation Response

5.1.1.4.2. Asynchronous Notification

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

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

5.1.1.5. Reserved Identities

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

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

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

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

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

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

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

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

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

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

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

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

5.1.2. Monitor Messages

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

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

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

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

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

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

The Monitor messages use the following data:

Monitor-Key: Monitor Key.

Monitor: A Monitor configuration (see Section 4.9.7).

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

```
|
+-[Register-Monitor]
|         +-[Client-Id:]
|         +-[(Unsigned 32) Execution-Delay]
|         +-[Operation-Id:]
|         +-[Monitor:] <List>
|         |         +-[Extensible:]
|         |         +-[Monitor-Key:] <U-Key>
|         |         +-[Target:]
|         |         +-[Deferrable]
|         |         +-[Configuration:]

|
+-[Deregister-Monitor]
|         +-[Client-Id:]
|         +-[(Unsigned 32) Execution-Delay]
|         +-[Operation-Id:]
|         +-[Monitor:] <List>
|         |         +-[Monitor-Key:] <U-Key>
|         |         +-[(Boolean) Send-Data ~ False]

|
+-[Probe]
|         +-[Client-Id:]
|         +-[(Unsigned 32) Execution-Delay]
|         +-[Operation-Id:]
|         +-[Monitor-Key:] <List>
```

Figure 19: Monitor Messages

5.1.2.1. Asynchronous Notification

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

Triggers include but are not limited to

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

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

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

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

Figure 20: Monitor Messages

5.2. Protocol Operation

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

5.2.1. DPN Selection

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

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

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

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

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

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

Figure 21: Monitor Messages

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

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

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

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

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

5.2.2. Policy Creation and Installation

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

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

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

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

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

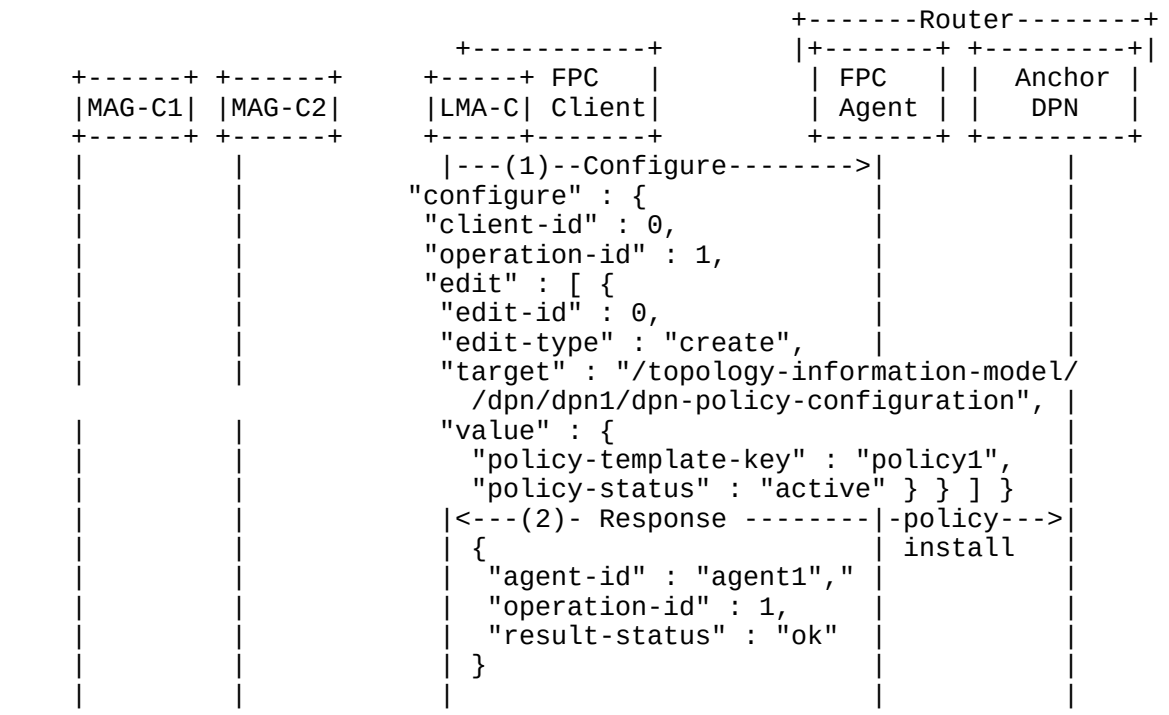


Figure 23: Example Policy Installation (focus on FPC reference point)

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

5.2.3. Simple RPC Operation

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

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

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

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

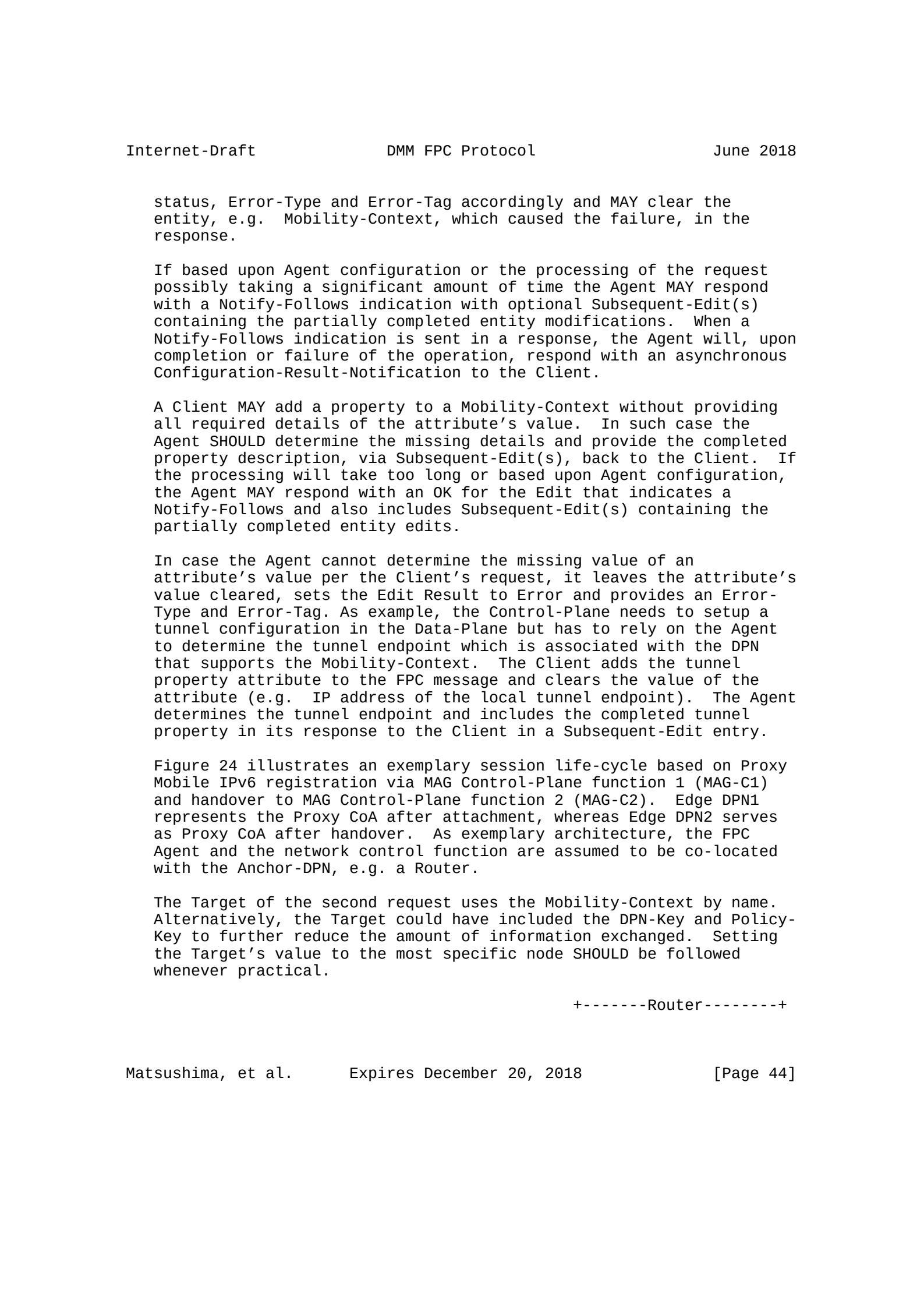
If based upon Agent configuration or the processing of the request possibly taking a significant amount of time the Agent MAY respond with a Notify-Follows indication with optional Subsequent-Edit(s) containing the partially completed entity modifications. When a Notify-Follows indication is sent in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

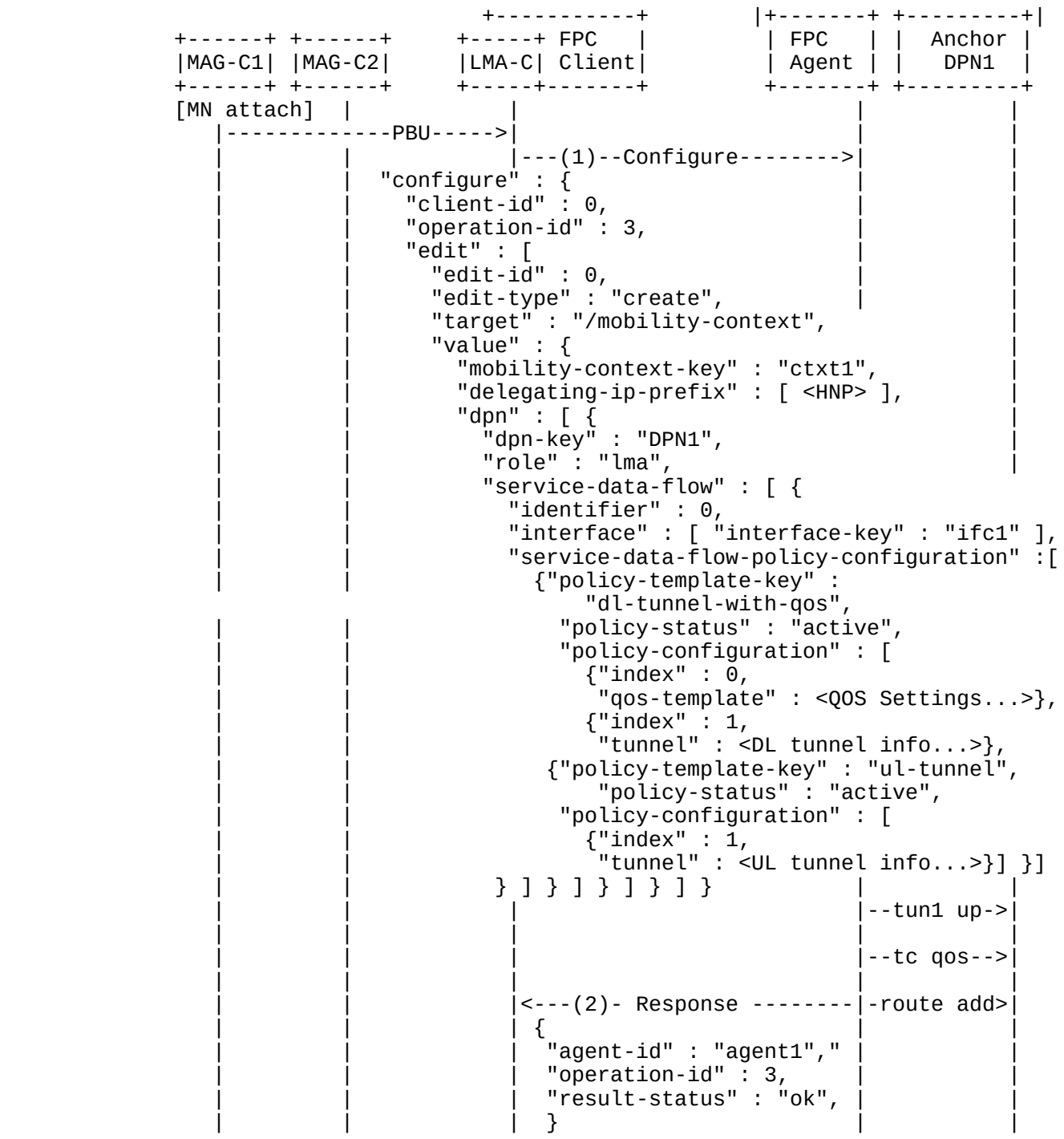
A Client MAY add a property to a Mobility-Context without providing all required details of the attribute's value. In such case the Agent SHOULD determine the missing details and provide the completed property description, via Subsequent-Edit(s), back to the Client. If the processing will take too long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

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

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

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




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

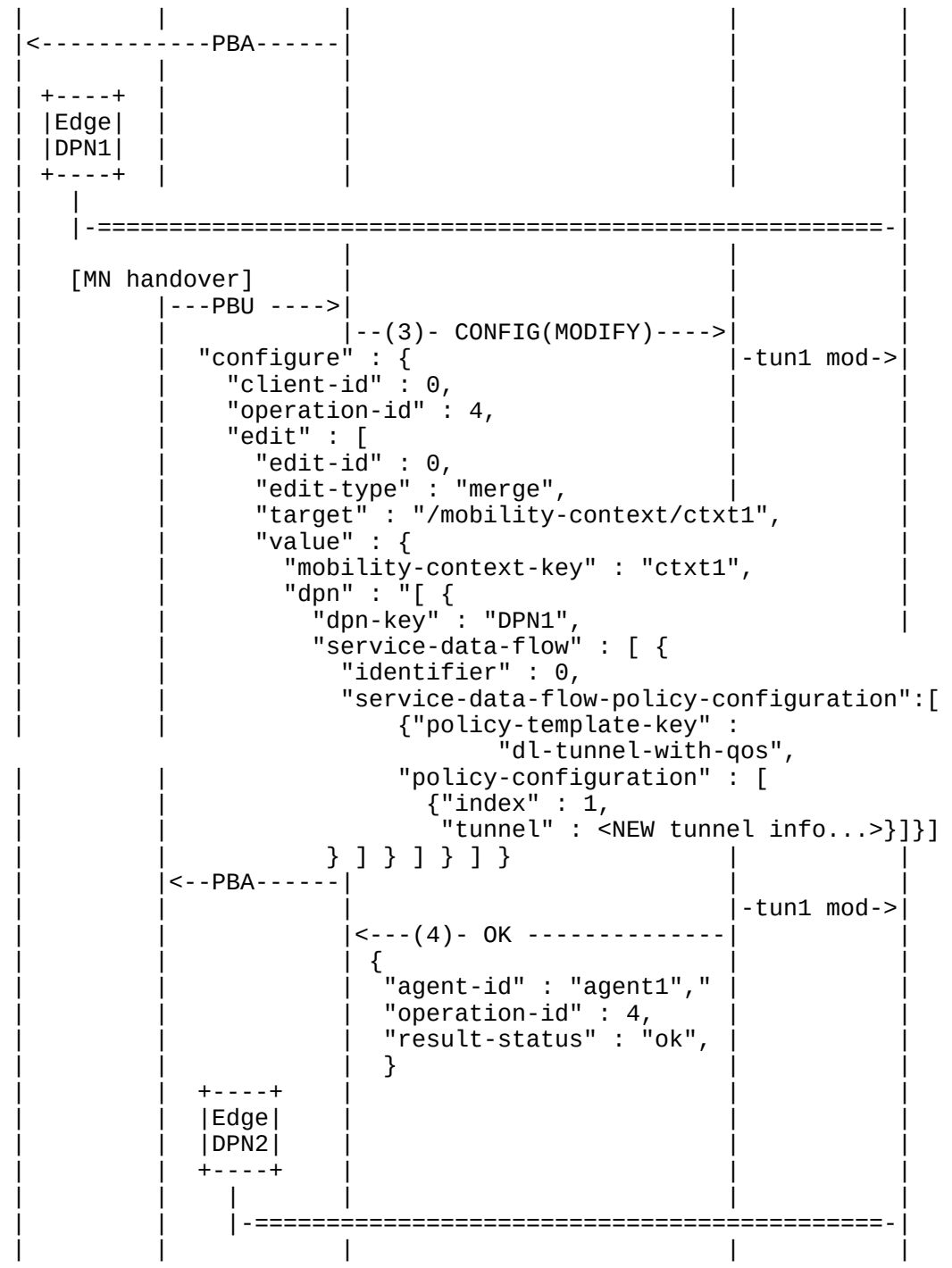

|<------------PBA------|
| +----+ |
| |Edge| |
| |DPN1| |
| +----+ |
[MN handover]
|---PBU ---->|
|--(3)- CONFIG(MODIFY)---->|
"configure" : {
"client-id" : 0,
"operation-id" : 4,
"edit" : [
"edit-id" : 0,
"edit-type" : "merge",
"target" : "/mobility-context/ctxt1",
"value" : {
"mobility-context-key"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

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

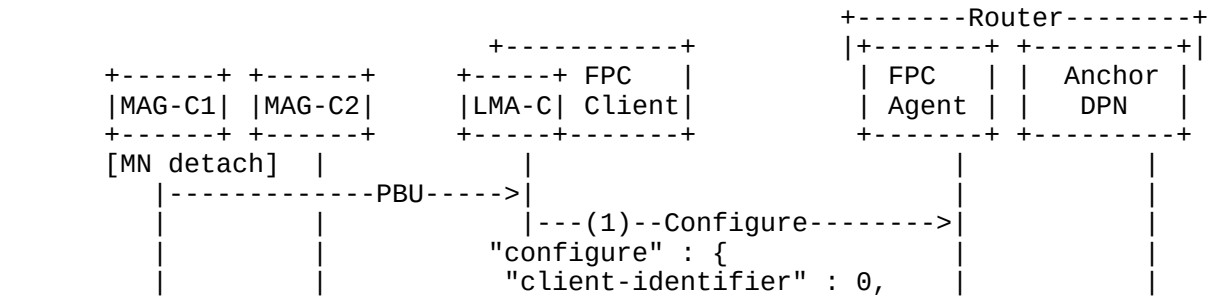
After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node's (MN) traffic. The LMA-C adds a new logical Mobility-Context to the DPN to treat the MN's traffic (1) and includes a Mobility-Context-Key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

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

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

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

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



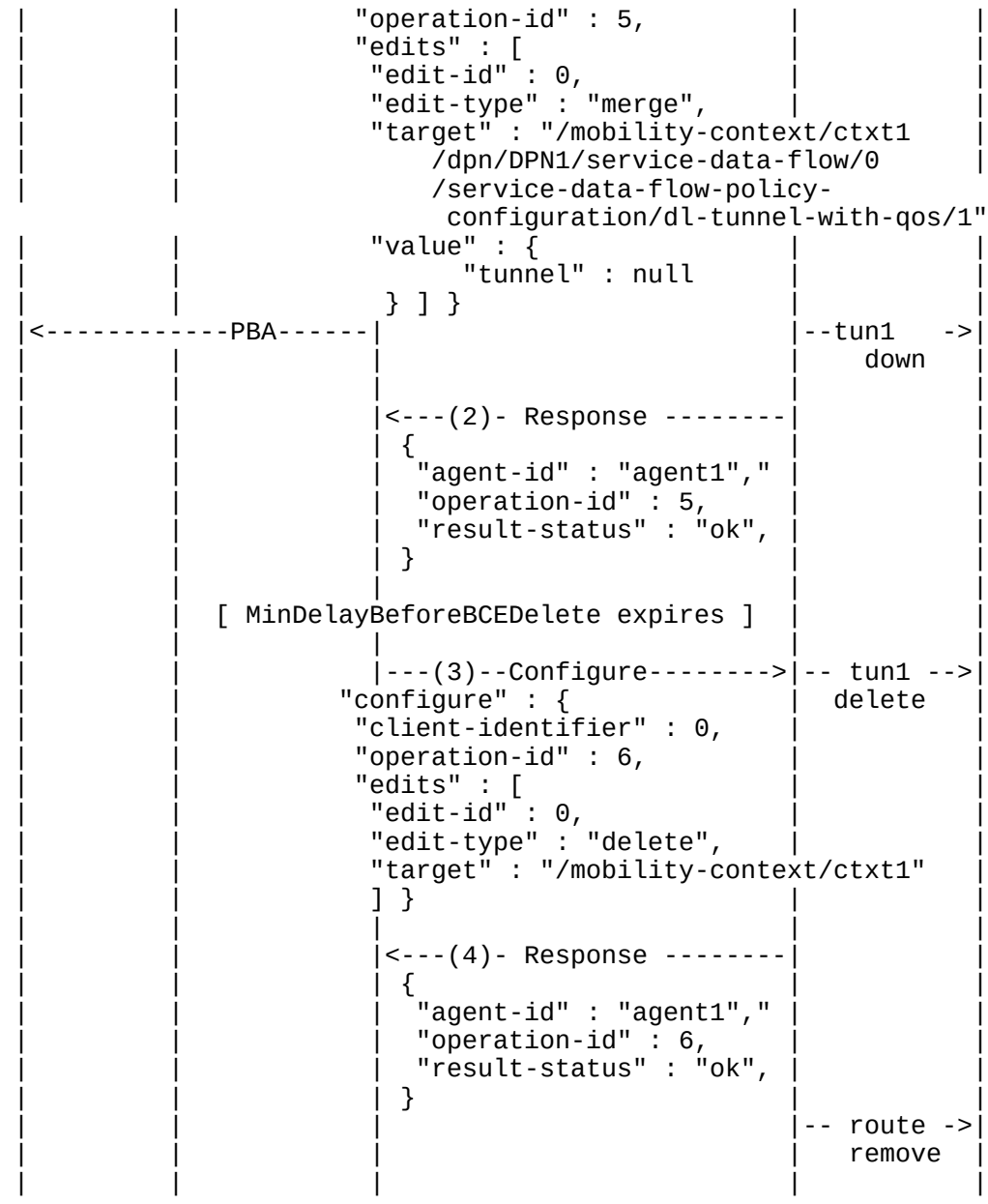


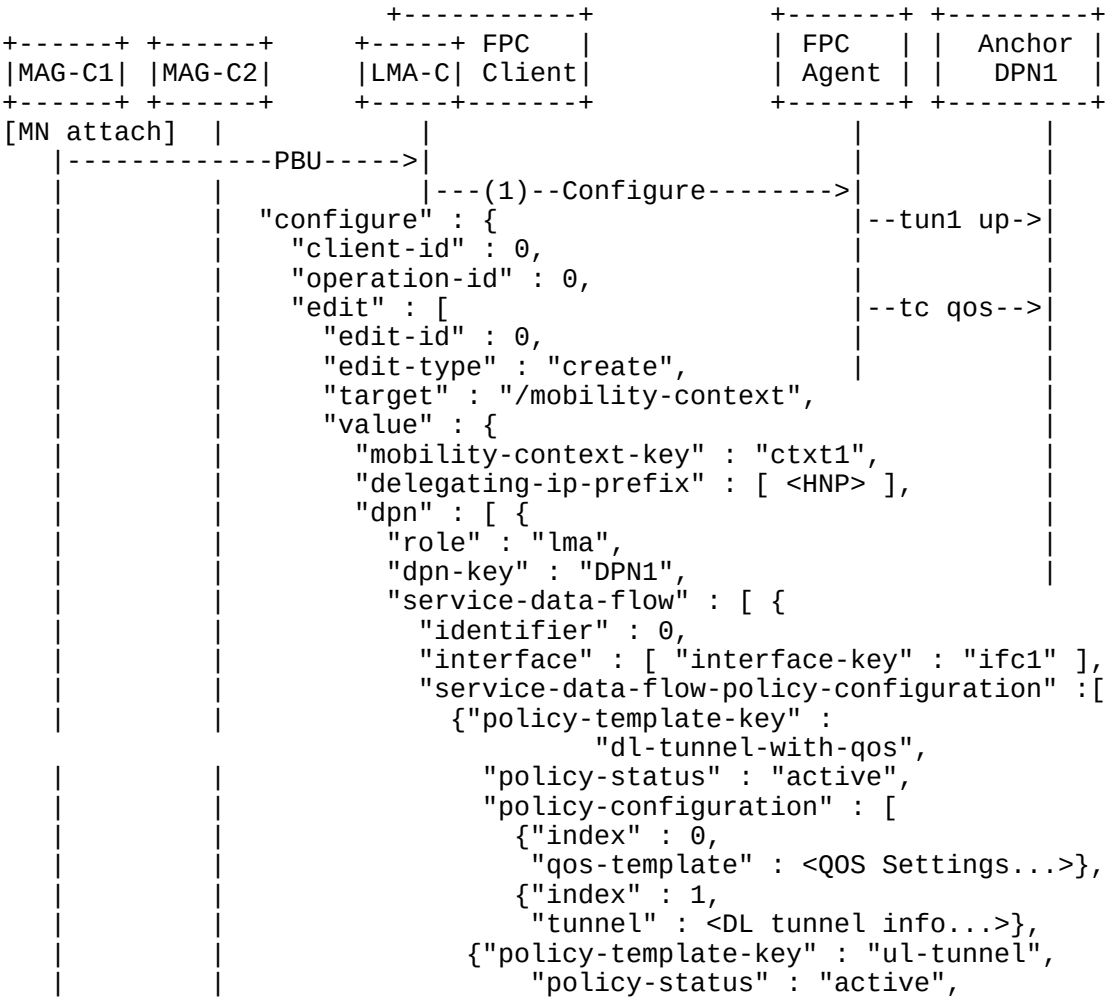
Figure 25: Single Agent with Deletion (focus on FPC reference point)

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

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

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

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




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

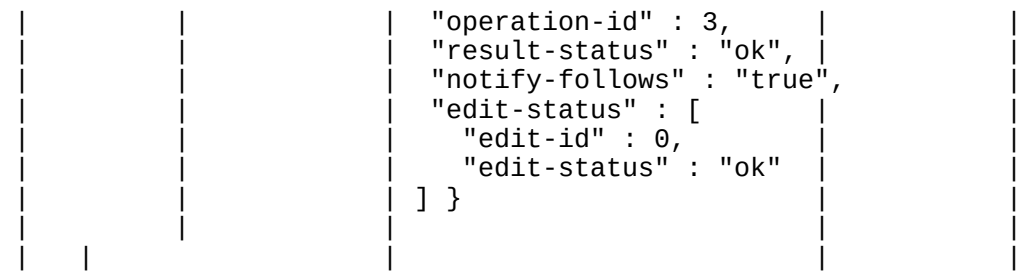


Figure 26: Exemplary Message Sequence for Multi-DPN Agent

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

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

5.2.4. Policy and Mobility on the Agent

A Client may build Policy and Topology using Configure messages.

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

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

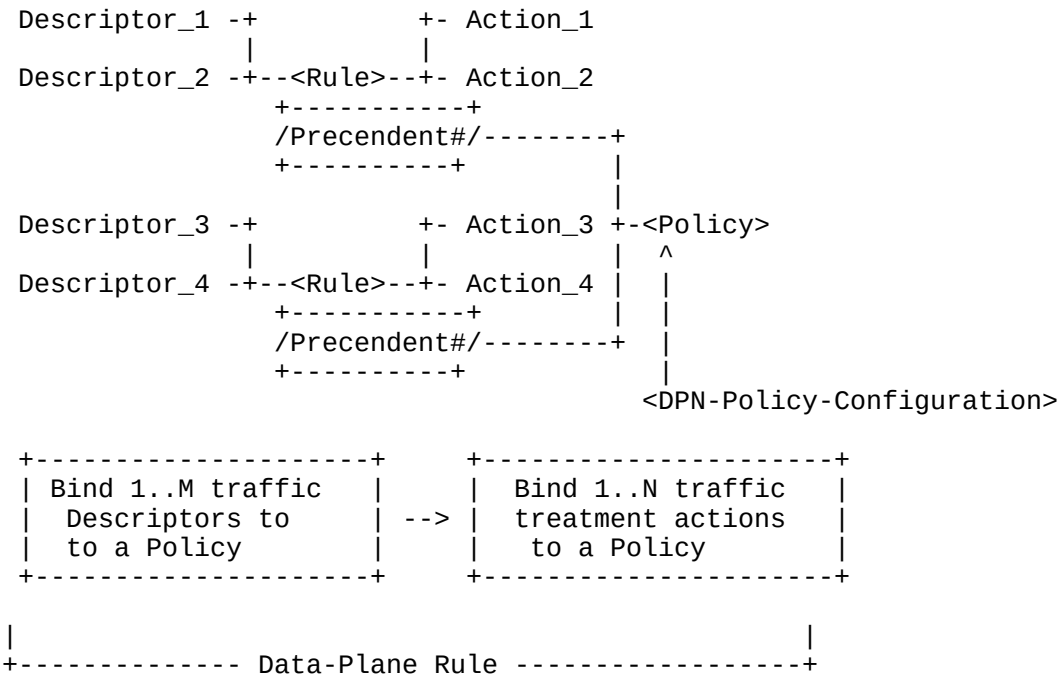


Figure 27: Structure of Configurable Policies

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

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

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

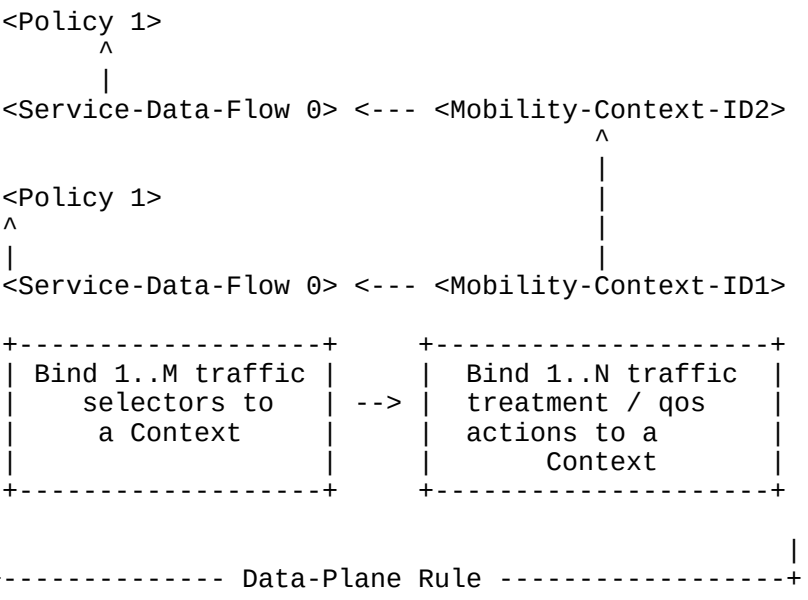


Figure 28: Mobility Context Hierarchy

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

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

5.2.5. Monitor Example

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

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

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

6. Templates and Command Sets

Configuration templates are shown below.

6.1. Monitor Configuration Templates

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

A scheduled configuration specifies a time for reporting.

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

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

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

Figure 30: Monitor Configuration Templates

6.2. Descriptor Templates

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

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

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

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

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

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

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

Figure 31: Descriptor Templates

6.3. Tunnel Templates

The Network Service Header is specified in [RFC8300].

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

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

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

For GRE, the gre-key MUST be present.

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

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

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

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

Tunnels can be used as Next Hop and Descriptor values.

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

Figure 32: Tunnel Templates

6.4. Action Templates

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

Drop action has no values.

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

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

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

Figure 33: Action Templates

6.5. Quality of Service Action Templates

PMIP QoS is specified in [RFC7222].

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

Figure 34: QoS Templates

6.6. PMIP Command-Set

The following Command Set values are supported for IETF PMIP.

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

6.7. 3GPP Specific Templates and Command-Set

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

APN-AMBR: Access Point Name Aggregate Maximum Bit Rate

UE-AMBR: User Equipment Aggregate Maximum Bit Rate

QCI: QoS Class Identifier

EBI: EPS Bearer Identity

LBI: Linked Bearer Identity

IMSI: International Mobile Subscriber Identity

TFT: Traffic Flow Template (TFT)

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

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

Figure 35: 3GPP Mobility Templates

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

Figure 36: 3GPP Packet Filter Template (Descriptor)

The following Command Set values are supported for 3GPP.

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

7. Implementation Status

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

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

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

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

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

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

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

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

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

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

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

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

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

fpc currently supports the following:

1 proprietary DPN API

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

CONFIG and CONF_BUNDLE (all operations)

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

Immediate Response is always an OK_NOTIFY_FOLLOWS.

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

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

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

Figure 37: fpc pseudo code

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

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

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

8. Security Considerations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

General usage of FPC MUST consider the following:

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

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

9. IANA Considerations

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

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

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

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

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

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

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

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

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

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

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

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

10. Work Team Participants

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

11. References

11.1. Normative References

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

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

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

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

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

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

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

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

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

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

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

11.2. Informative References

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

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

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

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

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

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

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

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

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

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

Appendix A. YANG Data Model for the FPC protocol

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

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

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

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

Mapping of templates to YANG are performed as follows:

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

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

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

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

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

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

Five modules are defined:

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

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

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

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

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

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

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

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

- qosoption (ietf-qos-pmip module)

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

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

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

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

A.1. FPC YANG Model

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

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

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

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

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

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

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

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

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

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

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

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

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

       revision 2018-05-17 {
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

               description "Edit list";
```

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

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

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

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

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

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

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

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

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

```
            }
            description "Monitor Response";
        }

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

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

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

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

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

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

A.2. FPC YANG Settings and Extensions Model

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A.3. PMIP QoS Model

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

This module references [RFC6991].

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

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

    prefix "qos-pmip";

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

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

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

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

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

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

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

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

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

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

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

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

       // Type Definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A.4. Traffic Selectors YANG Model

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

This module references [RFC6991].

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

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

 prefix "traffic-selectors";

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

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

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

 WG Chair: Dapeng Liu
 <mailto:maxpassion@gmail.com>

 WG Chair: Sri Gundavelli
 <mailto:sgundave@cisco.com>

 Editor: Satoru Matsushima
 <mailto:satoru.matsushima@g.softbank.co.jp>

 Editor: Lyle Bertz
 <mailto:lylebe551144@gmail.com>";
```

```
    description
    "This module contains a collection of YANG definitions for
    traffic selectors for flow bindings.

    Copyright (c) 2016 IETF Trust and the persons identified as the
    document authors. All rights reserved.

    This document is subject to BCP 78 and the IETF Trust's Legal
    Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info) in effect on the date of
    publication of this document. Please review these documents
    carefully, as they describe your rights and restrictions with
    respect to this document. Code Components extracted from this
    document must include Simplified BSD License text as described
    in Section 4.e of the Trust Legal Provisions and are provided
    without warranty as described in the Simplified BSD License.";

     revision 2018-05-17 {
       description
        "Initial Revision.";
       reference
        "RFC 6088: Traffic Selectors for Flow Bindings";
     }

    // Identities
      identity traffic-selector-format {
        description
        "The base type for Traffic-Selector Formats";
      }

      identity ipv4-binary-selector-format {
        base traffic-selector-format;
        description
          "IPv4 Binary Traffic Selector Format";
      }

      identity ipv6-binary-selector-format {
        base traffic-selector-format;
        description
          "IPv6 Binary Traffic Selector Format";
      }

      // Type definitions and groupings
      typedef ipsec-spi {
        type uint32;
        description
         "The first 32-bit IPsec Security Parameter Index (SPI)
         value on data. This field is defined in [RFC4303].";
```

```
          reference
          "RFC 4303: IP Encapsulating Security
          Payload (ESP)";
        }

        grouping traffic-selector-base {
          description "A grouping of the commen leaves between the
            v4 and v6 Traffic Selectors";
          container ipsec-spi-range {
            presence "Enables setting ipsec spi range";
            description
            "Inclusive range representing IPSec Security Parameter
            Indices to be used. When only start-spi is present, it
            represents a single spi.";
        leaf start-spi {
            type ipsec-spi;
            mandatory true;
            description
              "The first 32-bit IPsec SPI value on data.";
            }
        leaf end-spi {
              type ipsec-spi;
              must ". >= ../start-spi" {
                error-message
                  "The end-spi must be greater than or equal
                   to start-spi";
            }
           description
             "If more than one contiguous SPI value needs to be matched,
             then this field indicates the end value of a range.";
            }
         }
         container source-port-range {
           presence "Enables setting source port range";
           description
            "Inclusive range representing source ports to be used.
             When only start-port is present, it represents a single
          port. These value(s) are from the range of port numbers
             defined by IANA (http://www.iana.org).";
           leaf start-port {
              type inet:port-number;
              mandatory true;
              description
              "The first 16-bit source port number to be matched";
           }
           leaf end-port {
              type inet:port-number;
              must ". >= ../start-port" {
```

```
            error-message
             "The end-port must be greater than or equal to start-port";
           }
           description
            "The last 16-bit source port number to be matched";
          }
        }
        container destination-port-range {
          presence "Enables setting destination port range";
          description
           "Inclusive range representing destination ports to be used.
           When only start-port is present, it represents a single
           port.";
            leaf start-port {
              type inet:port-number;
              mandatory true;
              description
              "The first 16-bit destination port number to be matched";
           }
           leaf end-port {
             type inet:port-number;
             must ". >= ../start-port" {
             error-message
               "The end-port must be greater than or equal to
              start-port";
            }
            description
       "The last 16-bit destination port number to be matched";
        }
      }
    }

    grouping ipv4-binary-traffic-selector {
      container source-address-range-v4 {
         presence "Enables setting source IPv4 address range";
         description
          "Inclusive range representing IPv4 addresses to be used. When
          only start-address is present, it represents a single
          address.";
         leaf start-address {
           type inet:ipv4-address;
           mandatory true;
          description
           "The first source address to be matched";
         }
         leaf end-address {
           type inet:ipv4-address;
           description
```

```
            "The last source address to be matched";
          }
        }
        container destination-address-range-v4 {
           presence "Enables setting destination IPv4 address range";
           description
             "Inclusive range representing IPv4 addresses to be used.
             When only start-address is present, it represents a
             single address.";
           leaf start-address {
             type inet:ipv4-address;
             mandatory true;
             description
              "The first destination address to be matched";
           }
           leaf end-address {
             type inet:ipv4-address;
             description
              "The last destination address to be matched";
           }
        }
        container ds-range {
           presence "Enables setting dscp range";
           description
            "Inclusive range representing DiffServ Codepoints to be used.
            When only start-ds is present, it represents a single
            Codepoint.";
           leaf start-ds {
             type inet:dscp;
             mandatory true;
             description
              "The first differential service value to be matched";
         }
         leaf end-ds {
           type inet:dscp;
           must ". >= ../start-ds" {
             error-message
               "The end-ds must be greater than or equal to start-ds";
           }
           description
             "The last differential service value to be matched";
         }
        }
        container protocol-range {
          presence "Enables setting protocol range";
          description
            "Inclusive range representing IP protocol(s) to be used. When
             only start-protocol is present, it represents a single
```

```
          protocol.";
        leaf start-protocol {
          type uint8;
          mandatory true;
          description
            "The first 8-bit protocol value to be matched.";
        }
        leaf end-protocol {
          type uint8;
          must ". >= ../start-protocol" {
            error-message
              "The end-protocol must be greater than or equal to
             start-protocol";
          }
        description
          "The last 8-bit protocol value to be matched.";
        }
      }
      description "ipv4 binary traffic selector";
    }
     grouping ipv6-binary-traffic-selector {
      container source-address-range-v6 {
        presence "Enables setting source IPv6 address range";
         description
          "Inclusive range representing IPv6 addresses to be used.
          When only start-address is present, it represents a
          single address.";
         leaf start-address {
           type inet:ipv6-address;
           mandatory true;
           description
           "The first source address, from the
           range of 128-bit IPv6 addresses to be matched";
         }
         leaf end-address {
           type inet:ipv6-address;
           description
               "The last source address, from the
               range of 128-bit IPv6 addresses to be matched";
         }
      }
      container destination-address-range-v6 {
        presence "Enables setting destination IPv6 address range";
        description
          "Inclusive range representing IPv6 addresses to be used.
           When only start-address is present, it represents a
           single address.";
        leaf start-address {
```

```
          type inet:ipv6-address;
          mandatory true;
          description
              "The first destination address, from the
              range of 128-bit IPv6 addresses to be matched";
        }
        leaf end-address {
          type inet:ipv6-address;
          description
              "The last destination address, from the
              range of 128-bit IPv6 addresses to be matched";
      }
    }
    container flow-label-range {
      presence "Enables setting Flow Label range";
      description
        "Inclusive range representing IPv4 addresses to be used. When
         only start-flow-label is present, it represents a single
         flow label.";
      leaf start-flow-label {
        type inet:ipv6-flow-label;
        description
          "The first flow label value to be matched";
      }
      leaf end-flow-label {
        type inet:ipv6-flow-label;
        must ". >= ../start-flow-label" {
          error-message
            "The end-flow-lable must be greater than or equal to
             start-flow-label";
        }
        description
           "The first flow label value to be matched";
      }
     }
    container traffic-class-range {
      presence "Enables setting the traffic class range";
      description
       "Inclusive range representing IPv4 addresses to be used. When
        only start-traffic-class is present, it represents a single
        traffic class.";
      leaf start-traffic-class {
        type inet:dscp;
        description
         "The first traffic class value to be matched";
        reference
         "RFC 3260: New Terminology and Clarifications for Diffserv
          RFC 3168: The Addition of Explicit Congestion Notification
```

```
           (ECN) to IP";
       }
       leaf end-traffic-class {
         type inet:dscp;
         must ". >= ../start-traffic-class" {
           error-message
             "The end-traffic-class must be greater than or equal to
              start-traffic-class";
         }
         description
           "The last traffic class value to be matched";
       }
     }
     container next-header-range {
       presence "Enables setting Next Header range";
       description
        "Inclusive range representing Next Headers to be used. When
         only start-next-header is present, it represents a
         single Next Header.";
       leaf start-next-header {
         type uint8;
         description
          "The first 8-bit next header value to be matched.";
       }
       leaf end-next-header {
         type uint8;
         must ". >= ../start-next-header" {
           error-message
             "The end-next-header must be greater than or equal to
              start-next-header";
         }
         description
           "The last 8-bit next header value to be matched.";
       }
     }
     description "ipv6 binary traffic selector";
   }

     grouping traffic-selector {
       leaf ts-format {
          type identityref {
            base traffic-selector-format;
          }
          description "Traffic Selector Format";
       }
       uses traffic-selectors:traffic-selector-base;
       uses traffic-selectors:ipv4-binary-traffic-selector;
       uses traffic-selectors:ipv6-binary-traffic-selector;
```

```
       description
        "The traffic selector includes the parameters used to match
          packets for a specific flow binding.";
       reference
        "RFC 6089: Flow Bindings in Mobile IPv6 and Network
          Mobility (NEMO) Basic Support";
     }
   }
   <CODE ENDS>
```

A.5. RFC 5777 Classifier YANG Model

This module defines the RFC 5777 Classifer.

This module references [RFC5777].

```
<CODE BEGINS> file "ietf-diam-trafficclassifier@2018-05-17.yang"
module ietf-diam-trafficclassifier {
 yang-version 1.1;

 namespace
 "urn:ietf:params:xml:ns:yang:ietf-diam-trafficclassifier";

 prefix "diamclassifier";

 import ietf-inet-types {
   prefix inet;
   revision-date 2013-07-15;
 }
 import ietf-yang-types { prefix yang-types; }

 organization "IETF Distributed Mobility Management (DMM)
 Working Group";

 contact
 "WG Web: <http://tools.ietf.org/wg/netmod/>
 WG List: <mailto:netmod@ietf.org>

 WG Chair: Dapeng Liu
 <mailto:maxpassion@gmail.com>

 WG Chair: Sri Gundavelli
 <mailto:sgundave@cisco.com>

 Editor: Satoru Matsushima
 <mailto:satoru.matsushima@g.softbank.co.jp>

 Editor: Lyle Bertz
```

```
    <mailto:lylebe551144@gmail.com>";

    description
    "This module contains a collection of YANG definitions for
    traffic classification and QoS Attributes for Diameter.

    Copyright (c) 2018 IETF Trust and the persons identified as the
    document authors. All rights reserved.

    This document is subject to BCP 78 and the IETF Trust's Legal
    Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info) in effect on the date of
    publication of this document. Please review these documents
    carefully, as they describe your rights and restrictions with
    respect to this document. Code Components extracted from this
    document must include Simplified BSD License text as described
    in Section 4.e of the Trust Legal Provisions and are provided
    without warranty as described in the Simplified BSD License.";

     revision 2018-05-17 {
       description
           "Initial";
       reference
       "RFC 5777: Traffic Classification and Quality of Service (QoS)
           Attributes for Diameter";
     }

       typedef eui64-address-type {
           type string {
               length "6";
           }
           description
               "specifies a single layer 2 address in EUI-64 format.
                The value is an 8-octet encoding of the address as
                it would appear in the frame header.";
       }
       typedef direction-type {
           type enumeration {
               enum IN {
                   value 0;
                   description
                       "Applies to flows from the managed terminal.";
               }
               enum OUT {
                   value 1;
                   description
                    "Applies to flows to the managed terminal.";
               }
```

```
                enum BOTH {
                    value 2;
                    description
                        "Applies to flows both to and from the managed
                            terminal.";
                }
            }
            description
                "Specifies in which direction to apply the classifier.";
        }
        typedef negated-flag-type {
            type enumeration {
                enum False { value 0;
                    description "false"; }
                enum True { value 1;
                    description "True"; }
            }
            description
                "When set to True, the meaning of the match is
                 inverted and the classifier will match addresses
                 other than those specified by the From-Spec or
                 To-Spec AVP.

                 Note that the negation does not impact the port
                    comparisons.";
        }
        grouping index {
            leaf index {
                type uint16;
                mandatory true;
                description "Identifier used for referencing";
            }
            description "Index Value";
        }
        grouping to-from-spec-value {
            leaf-list ip-address {
                type inet:ip-address;
                description "IP address";
            }
            list ip-address-range {
                key index;
                uses diamclassifier:index;
                leaf ip-address-start {
                    type inet:ip-address;
                    description "IP Address Start";
                }
                leaf ip-address-end {
                    type inet:ip-address;
```

```
                description "IP Address End";
            }
            description "IP Address Range";
        }
        leaf-list ip-address-mask {
            type inet:ip-prefix;
            description "IP Address Mask";
        }
        leaf-list mac-address {
            type yang-types:mac-address;
            description "MAC address";
        }
        list mac-address-mask {
            key mac-address;
            leaf mac-address {
                type yang-types:mac-address;
                mandatory true;
                description "MAC address";
            }
            leaf macaddress-mask-pattern {
                type yang-types:mac-address;
                mandatory true;
                description
                 "The value specifies the bit positions of a
                  MAC address that are taken for matching.";
            }
            description "MAC Address Mask";
        }
        leaf-list eui64-address {
            type diamclassifier:eui64-address-type;
            description "EUI64 Address";
        }
        list eui64-address-mask {
            key eui64-address;
            leaf eui64-address {
                type diamclassifier:eui64-address-type;
                mandatory true;
                description "eui64 address";
            }
            leaf eui64-address-mask-pattern {
                type diamclassifier:eui64-address-type;
                mandatory true;
                description
                 "The value is 8 octets specifying the bit
                  positions of a EUI64 address that are taken
                 for matching.";
            }
            description "EUI64 Address Mask";
```

```
            }
            leaf-list port {
                type inet:port-number;
                description "Port Number";
            }
            list port-range {
                key index;
                uses diamclassifier:index;
                leaf ip-address-start {
                    type inet:port-number;
                    description "Port Start";
                }
                leaf ip-address-end {
                    type inet:port-number;
                    description "Port End";
                }
                description "Port Range";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            leaf use-assigned-address {
                type boolean;
                description "Use Assigned Address";
            }
            description
                "Basic traffic description value";
        }

        grouping option-type-group {
            leaf option-type {
                type uint8;
                mandatory true;
                description "Option Type";
            }
            leaf-list ip-option-value {
                type string;
                description "Option Value";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            description "Common X Option Pattern";
        }
        typedef vlan-id {
            type uint32 {
```

```
                range "0..4095";
            }
            description "VLAN ID";
        }

      grouping classifier {
        leaf protocol {
            type uint8;
            description "Protocol";
        }
        leaf direction {
            type diamclassifier:direction-type;
            description "Direction";
        }
        list from-spec {
            key index;
            uses diamclassifier:index;
            uses diamclassifier:to-from-spec-value;
            description "from specification";
        }
        list to-spec {
            key index;
            uses diamclassifier:index;
            uses diamclassifier:to-from-spec-value;
            description "to specification";
        }
        leaf-list disffserv-code-point {
            type inet:dscp;
            description "DSCP";
        }
        leaf fragmentation-flag {
        type enumeration {
                enum DF {
                    value 0;
                    description "Don’t Fragment";
                }
                enum MF {
                    value 1;
                    description "More Fragments";
                }
            }
            description "Fragmenttation Flag";
        }
        list ip-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "IP Option Value";
        }
```

```
        list tcp-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "TCP Option Value";
        }
        list tcp-flag {
            key tcp-flag-type;
            leaf tcp-flag-type {
                type uint32;
                mandatory true;
                description "TCP Flag Type";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            description "TCP Flags";
        }
        list icmp-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "ICMP Option Value";
        }
        list eth-option {
            key index;
            uses diamclassifier:index;
            container eth-proto-type {
                leaf-list eth-ether-type {
                    type string {
                        length "2";
                    }
                    description "value of ethertype field";
                }
                leaf-list eth-sap {
                    type string {
                        length "2";
                    }
                    description "802.2 SAP";
                }
                description "Ether Proto Type";
            }
            list vlan-id-range {
                key index;
                uses diamclassifier:index;
                leaf-list s-vlan-id-start {
                    type diamclassifier:vlan-id;
                    description "S-VID  VLAN ID Start";
                }
```

```
                leaf-list s-vlan-id-end {
                    type diamclassifier:vlan-id;
                    description "S-VID VLAN ID End";
                }
                leaf-list c-vlan-id-start {
                    type diamclassifier:vlan-id;
                    description "C-VID  VLAN ID Start";
                }
                leaf-list c-vlan-id-end {
                    type diamclassifier:vlan-id;
                    description "C-VID  VLAN ID End";
                }
                description "VLAN ID Range";
            }
            list user-priority-range {
                key index;
                uses diamclassifier:index;
                leaf-list low-user-priority {
                    type uint32 {
                        range "0..7";
                    }
                    description "Low User Priority";
                }
                leaf-list high-user-priority {
                    type uint32 {
                        range "0..7";
                    }
                    description "High User Priority";
                }
                description "User priority range";
            }
            description "Ether Option";
        }
        description "RFC 5777 Classifier";
    }
}
<CODE ENDS>
```

Appendix B. FPC YANG Tree Structure

This section only shows the structure for FPC YANG model. NOTE, it does NOT show the settings, Action values or Descriptor Value.

```
descriptor_value:
+--rw (descriptor-value)
  +--:(all-traffic)
  |  +--rw all-traffic?                    empty
  +--:(no-traffic)
```

```
     |  +--rw no-traffic?                    empty
     +--:(prefix-descriptor)
     |  +--rw destination-ip?                inet:ip-prefix
     |  +--rw source-ip?                     inet:ip-prefix
     +--:(pmip-selector)
     |  +--rw ts-format?                     identityref
     |  +--rw ipsec-spi-range!
     |  |  +--rw start-spi    ipsec-spi
     |  |  +--rw end-spi?     ipsec-spi
     |  +--rw source-port-range!
     |  |  +--rw start-port    inet:port-number
     |  |  +--rw end-port?     inet:port-number
     |  +--rw destination-port-range!
     |  |  +--rw start-port    inet:port-number
     |  |  +--rw end-port?     inet:port-number
     |  +--rw source-address-range-v4!
     |  |  +--rw start-address    inet:ipv4-address
     |  |  +--rw end-address?     inet:ipv4-address
     |  +--rw destination-address-range-v4!
     |  |  +--rw start-address    inet:ipv4-address
     |  |  +--rw end-address?     inet:ipv4-address
     |  +--rw ds-range!
     |  |  +--rw start-ds    inet:dscp
     |  |  +--rw end-ds?     inet:dscp
     |  +--rw protocol-range!
     |  |  +--rw start-protocol    uint8
     |  |  +--rw end-protocol?     uint8
     |  +--rw source-address-range-v6!
     |  |  +--rw start-address    inet:ipv6-address
     |  |  +--rw end-address?     inet:ipv6-address
     |  +--rw destination-address-range-v6!
     |  |  +--rw start-address    inet:ipv6-address
     |  |  +--rw end-address?     inet:ipv6-address
     |  +--rw flow-label-range!
     |  |  +--rw start-flow-label?   inet:ipv6-flow-label
     |  |  +--rw end-flow-label?     inet:ipv6-flow-label
     |  +--rw traffic-class-range!
     |  |  +--rw start-traffic-class?   inet:dscp
     |  |  +--rw end-traffic-class?     inet:dscp
     |  +--rw next-header-range!
     |     +--rw start-next-header?   uint8
     |     +--rw end-next-header?     uint8
     +--:(rfc5777-classifier-template)
     |  +--rw rfc5777-classifier-template
     |     +--rw protocol?               uint8
     |     +--rw direction?              diamclassifier:direction-type
     |     +--rw from-spec* [index]
     |     |  +--rw index                   uint16
```

```
|     |  +--rw ip-address*             inet:ip-address
|     |  +--rw ip-address-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:ip-address
|     |  |  +--rw ip-address-end?      inet:ip-address
|     |  +--rw ip-address-mask*         inet:ip-prefix
|     |  +--rw mac-address*             yang-types:mac-address
|     |  +--rw mac-address-mask* [mac-address]
|     |  |  +--rw mac-address              yang-types:mac-address
|     |  |  +--rw macaddress-mask-pattern  yang-types:mac-address
|     |  +--rw eui64-address*
                     diamclassifier:eui64-address-type
|     |  +--rw eui64-address-mask* [eui64-address]
|     |  |  +--rw eui64-address
                     diamclassifier:eui64-address-type
|     |  |  +--rw eui64-address-mask-pattern
                     diamclassifier:eui64-address-type
|     |  +--rw port*                    inet:port-number
|     |  +--rw port-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:port-number
|     |  |  +--rw ip-address-end?      inet:port-number
|     |  +--rw negated?
                     diamclassifier:negated-flag-type
|     |  +--rw use-assigned-address?   boolean
|     +--rw to-spec* [index]
|     |  +--rw index                    uint16
|     |  +--rw ip-address*              inet:ip-address
|     |  +--rw ip-address-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:ip-address
|     |  |  +--rw ip-address-end?      inet:ip-address
|     |  +--rw ip-address-mask*         inet:ip-prefix
|     |  +--rw mac-address*             yang-types:mac-address
|     |  +--rw mac-address-mask* [mac-address]
|     |  |  +--rw mac-address                yang-types:mac-address
|     |  |  +--rw macaddress-mask-pattern    yang-types:mac-address
|     |  +--rw eui64-address*
                     diamclassifier:eui64-address-type
|     |  +--rw eui64-address-mask* [eui64-address]
|     |  |  +--rw eui64-address
                     diamclassifier:eui64-address-type
|     |  |  +--rw eui64-address-mask-pattern
                     diamclassifier:eui64-address-type
|     |  +--rw port*                    inet:port-number
|     |  +--rw port-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:port-number
```

```
|     |  |  +--rw ip-address-end?     inet:port-number
|     |  +--rw negated?
                    diamclassifier:negated-flag-type
|     |  +--rw use-assigned-address?   boolean
|     +--rw disffserv-code-point*   inet:dscp
|     +--rw fragmentation-flag?     enumeration
|     +--rw ip-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw tcp-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw tcp-flag* [tcp-flag-type]
|     |  +--rw tcp-flag-type    uint32
|     |  +--rw negated?         diamclassifier:negated-flag-type
|     +--rw icmp-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw eth-option* [index]
|        +--rw index                 uint16
|        +--rw eth-proto-type
|        |  +--rw eth-ether-type*   string
|        |  +--rw eth-sap*          string
|        +--rw vlan-id-range* [index]
|        |  +--rw index              uint16
|        |  +--rw s-vlan-id-start*   diamclassifier:vlan-id
|        |  +--rw s-vlan-id-end*     diamclassifier:vlan-id
|        |  +--rw c-vlan-id-start*   diamclassifier:vlan-id
|        |  +--rw c-vlan-id-end*     diamclassifier:vlan-id
|        +--rw user-priority-range* [index]
|           +--rw index                uint16
|           +--rw low-user-priority*    uint32
|           +--rw high-user-priority*   uint32
+--:(packet-filter)
|  +--rw packet-filter
|     +--rw direction?            fpcbase:packet-filter-direction
|     +--rw identifier?             uint8
|     +--rw evaluation-precedence?   uint8
|     +--rw contents* [component-type-identifier]
|        +--rw component-type-identifier fpcbase:component-type-id
|        +--rw (value)?
|           +--:(ipv4-local)
|           |  +--rw ipv4-local?                 inet:ipv4-address
|           +--:(ipv6-prefix-local)
|           |  +--rw ipv6-prefix-local?          inet:ipv6-prefix
```

```
     |           +--:(ipv4-ipv6-remote)
     |           |  +--rw ipv4-ipv6-remote?           inet:ip-address
     |           +--:(ipv6-prefix-remote)
     |           |  +--rw ipv6-prefix-remote?         inet:ipv6-prefix
     |           +--:(next-header)
     |           |  +--rw next-header?                uint8
     |           +--:(local-port)
     |           |  +--rw local-port?                 inet:port-number
     |           +--:(local-port-range)
     |           |  +--rw local-port-lo?              inet:port-number
     |           |  +--rw local-port-hi?              inet:port-number
     |           +--:(remote-port)
     |           |  +--rw remote-port?                inet:port-number
     |           +--:(remote-port-range)
     |           |  +--rw remote-port-lo?             inet:port-number
     |           |  +--rw remote-port-hi?             inet:port-number
     |           +--:(ipsec-index)
     |           |  +--rw ipsec-index?     traffic-selectors:ipsec-spi
     |           +--:(traffic-class)
     |           |  +--rw traffic-class?              inet:dscp
     |           +--:(traffic-class-range)
     |           |  +--rw traffic-class-lo?           inet:dscp
     |           |  +--rw traffic-class-hi?           inet:dscp
     |           +--:(flow-label)
     |              +--rw flow-label*    inet:ipv6-flow-label
     +--:(tunnel-info)
        +--rw tunnel-info
           +--rw tunnel-local-address?    inet:ip-address
           +--rw tunnel-remote-address?   inet:ip-address
           +--rw mtu-size?                uint32
           +--rw tunnel?                  identityref
           +--rw payload-type?            enumeration
           +--rw gre-key?                 uint32
           +--rw gtp-tunnel-info
           |  +--rw local-tunnel-identifier?    uint32
           |  +--rw remote-tunnel-identifier?   uint32
           |  +--rw sequence-numbers-enabled?   boolean
           +--rw ebi?                     fpcbase:ebi-type
           +--rw lbi?                     fpcbase:ebi-type

   action_value:
   +--:(action-value)
   |  +--rw (action-value)
   |     +--:(drop)
   |     |  +--rw drop?                          empty
   |     +--:(rewrite)
   |     |  +--rw rewrite
   |     |     +--rw (rewrite-value)?
```

```
   |     |        +--:(prefix-descriptor)
   |     |        |  +--rw destination-ip?              inet:ip-prefix
   |     |        |  +--rw source-ip?                   inet:ip-prefix
   |     |        +--:(pmip-selector)
   |     |        |  +--rw ts-format?                     identityref
   |     |        |  +--rw ipsec-spi-range!
   |     |        |  |  +--rw start-spi    ipsec-spi
   |     |        |  |  +--rw end-spi?     ipsec-spi
   |     |        |  +--rw source-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw destination-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw source-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw destination-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw ds-range!
   |     |        |  |  +--rw start-ds    inet:dscp
   |     |        |  |  +--rw end-ds?     inet:dscp
   |     |        |  +--rw protocol-range!
   |     |        |  |  +--rw start-protocol    uint8
   |     |        |  |  +--rw end-protocol?     uint8
   |     |        |  +--rw source-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw destination-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw flow-label-range!
   |     |        |  |  +--rw start-flow-label?   inet:ipv6-flow-label
   |     |        |  |  +--rw end-flow-label?     inet:ipv6-flow-label
   |     |        |  +--rw traffic-class-range!
   |     |        |  |  +--rw start-traffic-class?   inet:dscp
   |     |        |  |  +--rw end-traffic-class?     inet:dscp
   |     |        |  +--rw next-header-range!
   |     |        |     +--rw start-next-header?   uint8
   |     |        |     +--rw end-next-header?     uint8
   |     |        +--:(rfc5777-classifier-template)
   |     |           +--rw rfc5777-classifier-template
   |     |              +--rw protocol?               uint8
   |     |              +--rw direction?
                         diamclassifier:direction-type
   |     |              +--rw from-spec* [index]
   |     |              |  +--rw index                  uint16
```

```
   |     |              |  +--rw ip-address*             inet:ip-address
   |     |              |  +--rw ip-address-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:ip-address
   |     |              |  |  +--rw ip-address-end?     inet:ip-address
   |     |              |  +--rw ip-address-mask*        inet:ip-prefix
   |     |              |  +--rw mac-address*    yang-types:mac-address
   |     |              |  +--rw mac-address-mask* [mac-address]
   |     |              |  |  +--rw mac-address
                                  yang-types:mac-address
   |     |              |  |  +--rw macaddress-mask-pattern
                                  yang-types:mac-address
   |     |              |  +--rw eui64-address*
                                  diamclassifier:eui64-address-type
   |     |              |  +--rw eui64-address-mask* [eui64-address]
   |     |              |  |  +--rw eui64-address
                                  diamclassifier:eui64-address-type
   |     |              |  |  +--rw eui64-address-mask-pattern
                                  diamclassifier:eui64-address-type
   |     |              |  +--rw port*                inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
                                  diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw to-spec* [index]
   |     |              |  +--rw index                   uint16
   |     |              |  +--rw ip-address*             inet:ip-address
   |     |              |  +--rw ip-address-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:ip-address
   |     |              |  |  +--rw ip-address-end?     inet:ip-address
   |     |              |  +--rw ip-address-mask*        inet:ip-prefix
   |     |              |  +--rw mac-address*
                                  yang-types:mac-address
   |     |              |  +--rw mac-address-mask* [mac-address]
   |     |              |  |  +--rw mac-address
                                  yang-types:mac-address
   |     |              |  |  +--rw macaddress-mask-pattern
                                  yang-types:mac-address
   |     |              |  +--rw eui64-address*
                                  diamclassifier:eui64-address-type
   |     |              |  +--rw eui64-address-mask* [eui64-address]
   |     |              |  |  +--rw eui64-address
                                  diamclassifier:eui64-address-type
   |     |              |  |  +--rw eui64-address-mask-pattern
```

```
                                    diamclassifier:eui64-address-type
   |     |              |  +--rw port*                  inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index                uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw disffserv-code-point*   inet:dscp
   |     |              +--rw fragmentation-flag?     enumeration
   |     |              +--rw ip-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw tcp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw tcp-flag* [tcp-flag-type]
   |     |              |  +--rw tcp-flag-type    uint32
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw icmp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw eth-option* [index]
   |     |                 +--rw index                  uint16
   |     |                 +--rw eth-proto-type
   |     |                 |  +--rw eth-ether-type*   string
   |     |                 |  +--rw eth-sap*          string
   |     |                 +--rw vlan-id-range* [index]
   |     |                 |  +--rw index             uint16
   |     |                 |  +--rw s-vlan-id-start*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw s-vlan-id-end*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-start*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-end*
                                    diamclassifier:vlan-id
   |     |                 +--rw user-priority-range* [index]
   |     |                    +--rw index                uint16
   |     |                    +--rw low-user-priority*   uint32
```

```
|     |                    +--rw high-user-priority*   uint32
|     +--:(copy-forward-nexthop)
|     |  +--rw copy-forward-nexthop
|     |     +--rw (next-hop-value)?
|     |        +--:(ip-address)
|     |        |  +--rw ip-address?           inet:ip-address
|     |        +--:(mac-address)
|     |        |  +--rw mac-address?          ytypes:mac-address
|     |        +--:(service-path)
|     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
|     |        +--:(mpls-path)
|     |        |  +--rw mpls-path?            fpcbase:fpc-mpls-label
|     |        +--:(nsh)
|     |        |  +--rw nsh?                  string
|     |        +--:(interface)
|     |        |  +--rw interface?            uint16
|     |        +--:(segment-identifier)
|     |        |  +--rw segment-identifier?   fpcbase:segment-id
|     |        +--:(mpls-label-stack)
|     |        |  +--rw mpls-label-stack*     fpcbase:fpc-mpls-label
|     |        +--:(mpls-sr-stack)
|     |        |  +--rw mpls-sr-stack*        fpcbase:fpc-mpls-label
|     |        +--:(srv6-stack)
|     |        |  +--rw srv6-stack*           fpcbase:segment-id
|     |        +--:(tunnel-info)
|     |           +--rw tunnel-info
|     |              +--rw tunnel-local-address?    inet:ip-address
|     |              +--rw tunnel-remote-address?   inet:ip-address
|     |              +--rw mtu-size?                uint32
|     |              +--rw tunnel?                  identityref
|     |              +--rw payload-type?            enumeration
|     |              +--rw gre-key?                 uint32
|     |              +--rw gtp-tunnel-info
|     |              |  +--rw local-tunnel-identifier?    uint32
|     |              |  +--rw remote-tunnel-identifier?   uint32
|     |              |  +--rw sequence-numbers-enabled?   boolean
|     |              +--rw ebi?                     fpcbase:ebi-type
|     |              +--rw lbi?                     fpcbase:ebi-type
|     +--:(nexthop)
|     |  +--rw nexthop
|     |     +--rw (next-hop-value)?
|     |        +--:(ip-address)
|     |        |  +--rw ip-address?           inet:ip-address
|     |        +--:(mac-address)
|     |        |  +--rw mac-address?          ytypes:mac-address
|     |        +--:(service-path)
|     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
|     |        +--:(mpls-path)
```

```
|     |        |  +--rw mpls-path?           fpcbase:fpc-mpls-label
|     |        +--:(nsh)
|     |        |  +--rw nsh?                string
|     |        +--:(interface)
|     |        |  +--rw interface?          uint16
|     |        +--:(segment-identifier)
|     |        |  +--rw segment-identifier? fpcbase:segment-id
|     |        +--:(mpls-label-stack)
|     |        |  +--rw mpls-label-stack*   fpcbase:fpc-mpls-label
|     |        +--:(mpls-sr-stack)
|     |        |  +--rw mpls-sr-stack*      fpcbase:fpc-mpls-label
|     |        +--:(srv6-stack)
|     |        |  +--rw srv6-stack*         fpcbase:segment-id
|     |        +--:(tunnel-info)
|     |           +--rw tunnel-info
|     |              +--rw tunnel-local-address?    inet:ip-address
|     |              +--rw tunnel-remote-address?   inet:ip-address
|     |              +--rw mtu-size?                uint32
|     |              +--rw tunnel?                  identityref
|     |              +--rw payload-type?            enumeration
|     |              +--rw gre-key?                 uint32
|     |              +--rw gtp-tunnel-info
|     |              |  +--rw local-tunnel-identifier?    uint32
|     |              |  +--rw remote-tunnel-identifier?   uint32
|     |              |  +--rw sequence-numbers-enabled?   boolean
|     |              +--rw ebi?                     fpcbase:ebi-type
|     |              +--rw lbi?                     fpcbase:ebi-type
|     +--:(qos)
|        +--rw trafficclass?                  inet:dscp
|        +--rw per-mn-agg-max-dl?
                 qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value
|        +--rw per-mn-agg-max-ul?
                 qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value
|        +--rw per-session-agg-max-dl
|        |  +--rw max-rate        uint32
|        |  +--rw service-flag    boolean
|        |  +--rw exclude-flag    boolean
|        +--rw per-session-agg-max-ul
|        |  +--rw max-rate        uint32
|        |  +--rw service-flag    boolean
|        |  +--rw exclude-flag    boolean
|        +--rw priority-level                 uint8
|        +--rw preemption-capability           enumeration
|        +--rw preemption-vulnerability        enumeration
|        +--rw agg-max-dl?
                 qos-pmip:Aggregate-Max-DL-Bit-Rate-Value
|        +--rw agg-max-ul?
                 qos-pmip:Aggregate-Max-UL-Bit-Rate-Value
```

```
|        +--rw gbr-dl?
                 qos-pmip:Guaranteed-DL-Bit-Rate-Value
|        +--rw gbr-ul?
                 qos-pmip:Guaranteed-UL-Bit-Rate-Value
|        +--rw qci?
                 fpcbase:fpc-qos-class-identifier
|        +--rw ue-agg-max-bitrate?            uint32
|        +--rw apn-ambr?                      uint32

policy-configuration-value:
|  |  |     +--rw (policy-configuration-value)?
|  |  |        +--:(descriptor-value)
|  |  |        |  ...
|  |  |        +--:(action-value)
|  |  |        |  ...
|  |  |        +--:(setting-value)
|  |  |           +--rw setting?                        <anydata>

policy-configuration:
|  |  |  +--rw policy-configuration* [index]
|  |  |     +--rw index                              uint16
|  |  |     +--rw extensible?             boolean
|  |  |     +--rw static-attributes*      string
|  |  |     +--rw mandatory-attributes*   string
|  |  |     +--rw entity-state?           enumeration
|  |  |     +--rw version?                uint32
|  |  |     +--rw (policy-configuration-value)?
|  |  |        ...

module: ietf-dmm-fpc
+--rw tenant* [tenant-key]
  +--rw tenant-key                    fpc:fpc-identity
  +--rw topology-information-model
  |  +--rw service-group* [service-group-key role-key]
  |  |  +--rw service-group-key     fpc:fpc-identity
  |  |  +--rw service-group-name?   string
  |  |  +--rw role-key              identityref
  |  |  +--rw role-name?            string
  |  |  +--rw protocol*             identityref
  |  |  +--rw feature*              identityref
  |  |  +--rw service-group-configuration* [index]
  |  |  |  +--rw index                           uint16
  |  |  |  +--rw (policy-configuration-value)?
  |  |  |        |  ...
  |  |  +--rw dpn* [dpn-key]
  |  |     +--rw dpn-key                fpc:fpc-identity
  |  |     +--rw referenced-interface* [interface-key]
  |  |        +--rw interface-key             fpc:fpc-identity
```

```
|  |        +--rw peer-service-group-key*   fpc:fpc-identity
|  +--rw dpn* [dpn-key]
|  |  +--rw dpn-key                          fpc:fpc-identity
|  |  +--rw dpn-name?                        string
|  |  +--rw dpn-resource-mapping-reference?  string
|  |  +--rw domain-key                       fpc:fpc-identity
|  |  +--rw service-group-key*               fpc:fpc-identity
|  |  +--rw interface* [interface-key]
|  |  |  +--rw interface-key        fpc:fpc-identity
|  |  |  +--rw interface-name?      string
|  |  |  +--rw role?                identityref
|  |  |  +--rw protocol*            identityref
|  |  |  +--rw interface-configuration* [index]
|  |  |     +--rw (policy-configuration-value)?
|  |  |           |  ...
|  |  +--rw dpn-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        +--rw index    uint16
|  |          +--rw (policy-configuration-value)?
|  |               |  ...
|  +--rw domain* [domain-key]
|  |  +--rw domain-key        fpc:fpc-identity
|  |  +--rw domain-name?      string
|  |  +--rw domain-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |             |         ...
|  +--rw dpn-checkpoint
|  |  +--rw basename?          fpc:fpc-identity
|  |  +--rw base-checkpoint?   string
|  +--rw service-group-checkpoint
|  |  +--rw basename?          fpc:fpc-identity
|  |  +--rw base-checkpoint?   string
|  +--rw dpn-checkpoint
|  |  +--rw basename?                   fpc:fpc-identity
|  |  +--rw base-checkpoint?            string
+--rw policy-information-model
|  +--rw action-template* [action-template-key]
|  |  +--rw action-template-key        fpc:fpc-identity
|  |  +--rw (action-value)
|  |  |                   ...
|  |  +--rw extensible?                boolean
|  |  +--rw static-attributes*         string
|  |  +--rw mandatory-attributes*      string
|  |  +--rw entity-state?              enumeration
|  |  +--rw version?                   uint32
|  +--rw descriptor-template* [descriptor-template-key]
```

```
|  |  +--rw descriptor-template-key         fpc:fpc-identity
|  |  +--rw (descriptor-value)
|  |  |                    ...
|  |  +--rw extensible?                     boolean
|  |  +--rw static-attributes*              string
|  |  +--rw mandatory-attributes*           string
|  |  +--rw entity-state?                   enumeration
|  |  +--rw version?                        uint32
|  +--rw rule-template* [rule-template-key]
|  |  +--rw rule-template-key           fpc:fpc-identity
|  |  +--rw descriptor-match-type       enumeration
|  |  +--rw descriptor-configuration* [descriptor-template-key]
|  |  |  +--rw descriptor-template-key    fpc:fpc-identity
|  |  |  +--rw direction?                 rfc5777:direction-type
|  |  |  +--rw setting?                        <anydata>
|  |  |  +--rw attribute-expression* [index]
|  |  |     +--rw index                           uint16
|  |  |     +--rw (descriptor-value)
|  |  |     |  ...
|  |  +--rw action-configuration* [action-order]
|  |  |  +--rw action-order            uint32
|  |  |  +--rw action-template-key     fpc:fpc-identity
|  |  |  +--rw setting?                        <anydata>
|  |  |  +--rw attribute-expression* [index]
|  |  |     +--rw index                     uint16
|  |  |     +--rw (action-value)
|  |  |     |  ...
|  |  +--rw extensible?                 boolean
|  |  +--rw static-attributes*          string
|  |  +--rw mandatory-attributes*       string
|  |  +--rw entity-state?               enumeration
|  |  +--rw version?                    uint32
|  |  +--rw rule-configuration* [index]
|  |     +--rw index    uint16
|  |         +--rw (policy-configuration-value)?
|  |           |  ...
|  +--rw policy-template* [policy-template-key]
|  |  +--rw policy-template-key     fpc:fpc-identity
|  |  +--rw rule-template* [precedence]
|  |  |  +--rw precedence           uint32
|  |  |  +--rw rule-template-key    fpc:fpc-identity
|  |  +--rw extensible?             boolean
|  |  +--rw static-attributes*      string
|  |  +--rw mandatory-attributes*   string
|  |  +--rw entity-state?           enumeration
|  |  +--rw version?                uint32
|  |  +--rw policy-configuration* [index]
|  |      ...
```

```
|  +--rw basename?              fpc:fpc-identity
|  +--rw base-checkpoint?       string
+--rw mobility-context* [mobility-context-key]
|  +--rw mobility-context-key    fpc:fpc-identity
|  +--rw delegating-ip-prefix*   inet:ip-prefix
|  +--rw parent-context?         fpc:fpc-identity
|  +--rw child-context*          fpc:fpc-identity
|  +--rw mobile-node
|  |  +--rw ip-address*    inet:ip-address
|  |  +--rw imsi?          fpcbase:imsi-type
|  |  +--rw mn-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        ...
|  +--rw domain
|  |  +--rw domain-key?        fpc:fpc-identity
|  |  +--rw domain-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        ...
|   +--rw dpn* [dpn-key]
|      +--rw dpn-key              fpc:fpc-identity
|      +--rw dpn-policy-configuration* [policy-template-key]
|      |  +--rw policy-template-key     fpc:fpc-identity
|      |  +--rw policy-configuration* [index]
|      |     ...
|      +--rw role?                identityref
|      +--rw service-data-flow* [identifier]
|         +--rw identifier           uint32
|         +--rw service-group-key?   fpc:fpc-identity
|         +--rw interface* [interface-key]
|         |  +--rw interface-key    fpc:fpc-identity
|         +--rw service-data-flow-policy-
                configuration* [policy-template-key]
|            +--rw policy-template-key     fpc:fpc-identity
|            +--rw policy-configuration* [index]
|               ...
+--rw monitor* [monitor-key]
  +--rw extensible?             boolean
  +--rw static-attributes*      string
  +--rw mandatory-attributes*   string
  +--rw entity-state?           enumeration
  +--rw version?                uint32
  +--rw monitor-key             fpc:fpc-identity
  +--rw target?                 string
  +--rw deferrable?             boolean
  +--rw (configuration)
     +--:(period)
```

```
          |  +--rw period?           uint32
          +--:(threshold-config)
          |  +--rw low?              uint32
          |  +--rw hi?               uint32
          +--:(schedule)
          |  +--rw schedule?         uint32
          +--:(event-identities)
          |  +--rw event-identities*       identityref
          +--:(event-ids)
             +--rw event-ids*        uint32

     rpcs:
       +---x configure
       |  +---w input
       |  |  +---w client-id          fpc:client-identifier
       |  |  +---w execution-delay?   uint32
       |  |  +---w yang-patch
       |  |     +---w patch-id    string
       |  |     +---w comment?    string
       |  |     +---w edit* [edit-id]
       |  |        +---w edit-id         string
       |  |        +---w operation       enumeration
       |  |        +---w target          target-resource-offset
       |  |        +---w point?          target-resource-offset
       |  |        +---w where?          enumeration
       |  |        +---w value?          <anydata>
       |  |        +---w reference-scope?   fpc:ref-scope
       |  |        +---w command-set
       |  |           +---w (instr-type)?
       |  |              +--:(instr-3gpp-mob)
       |  |              |  +---w instr-3gpp-mob? fpcbase:threegpp-instr
       |  |              +--:(instr-pmip)
       |  |                 +---w instr-pmip?       pmip-commandset
       |  +--ro output
       |     +--ro yang-patch-status
       |        +--ro patch-id       string
       |        +--ro (global-status)?
       |        |  +--:(global-errors)
       |        |  |  +--ro errors
       |        |  |     +--ro error*
       |        |  |        +--ro error-type       enumeration
       |        |  |        +--ro error-tag        string
       |        |  |        +--ro error-app-tag?   string
       |        |  |        +--ro error-path?      instance-identifier
       |        |  |        +--ro error-message?   string
       |        |  |        +--ro error-info?      <anydata>
       |        |  +--:(ok)
       |        |     +--ro ok?            empty
```

```
|        +--ro edit-status
|           +--ro edit* [edit-id]
|              +--ro edit-id            string
|              +--ro (edit-status-choice)?
|                 +--:(ok)
|                 |  +--ro ok?                empty
|                 |  +--ro notify-follows?    boolean
|                 |  +--ro subsequent-edit* [edit-id]
|                 |     +--ro edit-id      string
|                 |     +--ro operation    enumeration
|                 |     +--ro target
                              ypatch:target-resource-offset
|                 |     +--ro point?
                                    ypatch:target-resource-offset
|                 |     +--ro where?       enumeration
|                 |     +--ro value?       <anydata>
|                 +--:(errors)
|                    +--ro errors
|                       +--ro error*
|                          +--ro error-type       enumeration
|                          +--ro error-tag        string
|                          +--ro error-app-tag?   string
|                          +--ro error-path?
                                    instance-identifier
|                          +--ro error-message?   string
|                          +--ro error-info?      <anydata>
+---x register_monitor
|  +---w input
|  |  +---w client-id          fpc:client-identifier
|  |  +---w execution-delay?   uint32
|  |  +---w operation-id              uint64
|  |  +---w monitor* [monitor-key]
|  |     +---w extensible?             boolean
|  |     +---w static-attributes*      string
|  |     +---w mandatory-attributes*   string
|  |     +---w entity-state?           enumeration
|  |     +---w version?                uint32
|  |     +---w monitor-key             fpc:fpc-identity
|  |     +---w target?                 string
|  |     +---w deferrable?             boolean
|  |     +---w (configuration)
|  |        +--:(period)
|  |        |  +---w period?                uint32
|  |        +--:(threshold-config)
|  |        |  +---w low?                   uint32
|  |        |  +---w hi?                    uint32
|  |        +--:(schedule)
|  |        |  +---w schedule?              uint32
```

```
|  |        +--:(event-identities)
|  |        |  +---w event-identities*       identityref
|  |        +--:(event-ids)
|  |           +---w event-ids*              uint32
|  +--ro output
|     +--ro operation-id     uint64
|     +--ro (edit-status-choice)?
|        +--:(ok)
|        |  +--ro ok?       empty
|        +--:(errors)
|           +--ro errors
|              +--ro error*
|                 +--ro error-type       enumeration
|                 +--ro error-tag        string
|                 +--ro error-app-tag?   string
|                 +--ro error-path?      instance-identifier
|                 +--ro error-message?   string
|                 +--ro error-info?      <anydata>
+---x deregister_monitor
|  +---w input
|  |  +---w client-id          fpc:client-identifier
|  |  +---w execution-delay?   uint32
|  |  +---w operation-id              uint64
|  |  +---w monitor* [monitor-key]
|  |     +---w monitor-key    fpc:fpc-identity
|  |     +---w send_data?     boolean
|  +--ro output
|     +--ro operation-id     uint64
|     +--ro (edit-status-choice)?
|        +--:(ok)
|        |  +--ro ok?       empty
|        +--:(errors)
|           +--ro errors
|              +--ro error*
|                 +--ro error-type       enumeration
|                 +--ro error-tag        string
|                 +--ro error-app-tag?   string
|                 +--ro error-path?      instance-identifier
|                 +--ro error-message?   string
|                 +--ro error-info?      <anydata>
+---x probe
   +---w input
   |  +---w client-id          fpc:client-identifier
   |  +---w execution-delay?   uint32
   |  +---w operation-id              uint64
   |  +---w monitor* [monitor-key]
   |     +---w monitor-key    fpc:fpc-identity
   +--ro output
```

```
            +--ro operation-id     uint64
            +--ro (edit-status-choice)?
               +--:(ok)
               |  +--ro ok?       empty
               +--:(errors)
                  +--ro errors
                     +--ro error*
                        +--ro error-type       enumeration
                        +--ro error-tag        string
                        +--ro error-app-tag?   string
                        +--ro error-path?      instance-identifier
                        +--ro error-message?   string
                        +--ro error-info?      <anydata>

     notifications:
       +---n config-result-notification
       |  +--ro yang-patch-status
       |  |  +--ro patch-id       string
       |  |  +--ro (global-status)?
       |  |  |  +--:(global-errors)
       |  |  |  |  +--ro errors
       |  |  |  |     +--ro error*
       |  |  |  |        +--ro error-type       enumeration
       |  |  |  |        +--ro error-tag        string
       |  |  |  |        +--ro error-app-tag?   string
       |  |  |  |        +--ro error-path?      instance-identifier
       |  |  |  |        +--ro error-message?   string
       |  |  |  |        +--ro error-info?      <anydata>
       |  |  |  +--:(ok)
       |  |  |     +--ro ok?            empty
       |  |  +--ro edit-status
       |  |     +--ro edit* [edit-id]
       |  |        +--ro edit-id    string
       |  |        +--ro (edit-status-choice)?
       |  |           +--:(ok)
       |  |           |  +--ro ok?        empty
       |  |           +--:(errors)
       |  |              +--ro errors
       |  |                 +--ro error*
       |  |                    +--ro error-type       enumeration
       |  |                    +--ro error-tag        string
       |  |                    +--ro error-app-tag?   string
       |  |                    +--ro error-path?
       |  |                            instance-identifier
       |  |                    +--ro error-message?   string
       |  |                    +--ro error-info?      <anydata>
       |  +--ro subsequent-edit* [edit-id]
       |     +--ro edit-id      string
```

```
|     +--ro operation    enumeration
|     +--ro target       ypatch:target-resource-offset
|     +--ro point?       ypatch:target-resource-offset
|     +--ro where?       enumeration
|     +--ro value?       <anydata>
+---n notify
   +--ro notification-id?   uint32
   +--ro timestamp?         uint32
   +--ro report* [monitor-key]
      +--ro monitor-key                fpc:fpc-identity
      +--ro trigger?                   identityref
      +--ro (value)?
         +--:(dpn-candidate-available)
         |  +--ro node-id?                    inet:uri
         |  +--ro supported-interface-list* [role-key]
         |     +--ro role-key    identityref
         +--:(dpn-unavailable)
         |  +--ro dpn-id?                     fpc:fpc-identity
         +--:(report-value)
            +--ro report-value?             <anydata>
```

Figure 38: YANG FPC Agent Tree

Appendix C. Change Log

C.1. Changes since Version 09

The following changes have been made since version 09

Migration to a Template based framework. This affects all elements. The framework has a template definition language.

Basename is split into two aspects. The first is version which applies to Templates. The second is checkpointing which applies to specific sections only.

Rule was inside Policy and now is Rule-Template and stands as a peer structure to Policy.

Types, e.g. Descriptor Types, Action Types, etc., are now templates that have no values filled in.

The embedded rule has been replaced by a template that has no predefined variables. All rules, pre-configured or embedded, are realized as Policy instantiations.

The Unassigned DPN is used to track requests vs. those that are installed, i.e. Agent assignment of Policy is supported.

The Topology system supports selection information by ServiceGroup or ServiceEndpoint.

DPN Peer Groups and DPN Groups are now PeerServiceGroup and ServiceGroup.

Bulk Configuration and Configuration now follow a style similar to YANG Patch. Agents MAY response back with edits it made to complete the Client edit request.

RFC 5777 Classifiers have been added.

All operations have a common error format.

C.2. Changes since Version 10

The following changes have been made since version 10

Sevice-Endpoints eliminated. Service-Group and DPN interfaces changed to hold information previously held by Service-Endpoint as noted in ML during IETF 101.

Service-Group resides under the Topology-Information-Mode

The Domain now has a checkpoint and the Topology Information Model checkpoint was removed to avoid any overlaps in checkpoints.

Scrubbed YANG for NMDA compliance and Guidelines (RFC 6087bis).

Monitor lifecycle, policy and policy installation examples added.

Authors' Addresses

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

Email: satoru.matsushima@g.softbank.co.jp

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: lylebe551144@gmail.com

Marco Liebsch
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Phone: +49 6221 4342146
Email: liebsch@neclab.eu

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2017

Z. Yan
CNNIC
J. Lee
Sangmyung University
X. Lee
CNNIC

July 1, 2016

Home Network Prefix Renumbering in PMIPv6
draft-ietf-dmm-hnprenum-03

Abstract

In the basic Proxy Mobile IPv6 (PMIPv6) specification, a Mobile Node (MN) is assigned with a Home Network Prefix (HNP) during its initial attachment and the MN configures its Home Address (HoA) with the HNP. During the movement of the MN, the HNP is remained unchanged to keep ongoing communications associated with the HoA. However, the current PMIPv6 specification does not specify related operations when an HNP renumbering is happened. In this document, a solution to support the HNP renumbering is proposed, as an update of the PMIPv6 specification.

Requirements Language

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]

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 January 2, 2017.

Table of Contents

1. Introduction

Network managers currently prefer Provider Independent (PI) addressing for IPv6 to attempt to minimize the need for future possible renumbering. However, a widespread use of PI addresses may cause Border Gateway Protocol (BGP) scaling problems. It is thus desirable to develop tools and practices that make IPv6 renumbering a simpler process to reduce demand for IPv6 PI space [RFC6879]. In this document, we aim to solve the HNP renumbering problem when the HNP in PMIPv6 [RFC5213] is not the type of PI.

2. Usage Scenarios

There are a number of reasons why the HNP renumbering support in PMIPv6 is useful and some scenarios are identified below:

- Scenario 1: the HNP set used by a PMIPv6 service provider is assigned by a different Internet Service Provider (ISP), and then

the HNP renumbering may happen if the PMIPv6 service provider switches to a different ISP.

- o Scenario 2: multiple Local Mobility Anchors (LMAs) may be deployed by the same PMIPv6 service provider, and then each LMA may serve for a specific HNP set. In this case, the HNP of an MN may change if the current serving LMA switches to another LMA but without inheriting the assigned HNP set [RFC6463].

- o Scenario 3: the PMIPv6 HNP renumbering may be caused by the re-building of the network architecture as the companies split, merge, grow, relocate, or reorganize. For example, the PMIPv6 service provider may reorganize its network topology.

In the scenario 1, we assume that only the HNP is renumbered while the serving LMA remains unchanged and this is the basic scenario considered in this document. In the scenario 2 and scenario 3, more complex results may be caused, for example, the HNP renumbering may happen due to the switchover of a serving LMA.

In the Mobile IPv6 (MIPv6) protocol, when a home network prefix changes, the Home Agent (HA) will actively notify the new prefix to its MN and then the renumbering of the Home Network Address (HoA) can be well supported [RFC6275]. In the basic PMIPv6, the PMIPv6 binding is triggered by a Mobile Access Gateway (MAG), which detects the attachment of the MN. A scheme is also needed for the LMA to immediately initiate the PMIPv6 binding state refreshment during the HNP renumbering process. Although this issue is also mentioned in Section 6.12 of [RFC5213], the related solution has not been specified.

3. PMIPv6 Extensions

When the HNP renumbering happens in PMIPv6, the LMA has to notify a new HNP to an MAG and then the MAG has to announce the new HNP to the attached MN accordingly. Also, the LMA and the MAG must update the routing states for the HNP and the related addresses. To support this procedure, [RFC7077] can be adopted which specifies an asynchronous update from the LMA to the MAG about specific session parameters. This document considers the following two cases:

(1) HNP is renumbered under the same LMA

In this case, the LMA remains unchanged as in the scenario 1 and scenario 3. The operation steps are shown in Figure 1.

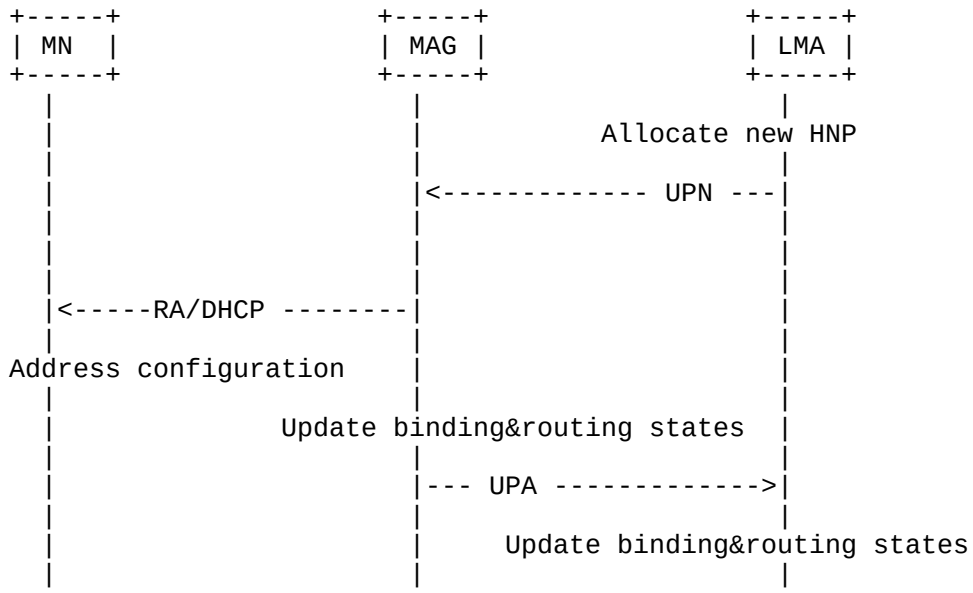


Figure 1: Signaling call flow of the HNP renumbering

- When a PMIPv6 service provider renumbers the HNP set under the same LMA, the serving LMA will initiate the HNP renumbering operation. The LMA allocates a new HNP for the related MN.

- The LMA sends the Update Notification (UPN) message to the MAG to update the HNP information. If the Dynamic Host Configuration Protocol (DHCP) is used to allocate the address, the new HNP should be also notified to the DHCP infrastructure.

- Once the MAG receives this UPN message, it recognizes that the related MN has the new HNP. Then the MAG should notify the MN about the new HNP with a Router Advertisement (RA) message or allocate a new address within the new HNP through a DHCP procedure.

- After the MN obtains the HNP information through the RA message, it deletes the old HoA and configures a new HoA with the newly allocated HNP.

- When the new HNP is announced or the new address is configured to the MN successfully, the MAG updates the related binding and routing states. Then the MAG sends back the Update Notification Acknowledgement (UPA) message to the LMA for the notification of successful update of the HNP, related binding state, and routing

state. Then the LMA updates the routing and binding information corresponding to the MN to replace the old HNP with the new one.

(2) HNP renumbering caused by the LMA switchover

Since the HNP is assigned by the LMA, the HNP renumbering may be caused by the LMA switchover, as in the scenario 2 and scenario 3.

The information of LMA is the basic configuration information of MAG. When the LMA changes, the related profile should be updated by the service provider. In this way, the MAG initiates the registration to the new LMA as specified in [RFC5213]. When the HNP renumbering is caused in this case, the new HNP information is sent by the LMA during the new binding procedure. Accordingly, the MAG withdraws the old HNP of the MN and announces the new HNP to the MN as like the case of the HNP is renumbered under the same LMA.

4. Session Connectivity

The HNP renumbering may cause the disconnection of the ongoing communications of the MN. Basically, there are two modes to manage the session connectivity during the HNP renumbering.

(1) Soft-mode

The LMA will temporarily maintain the state of the old HNP during the HNP renumbering (after the UPA reception) in order to redirect the packets to the MN before the MN reconnects the ongoing session and notifies its new HoA to the Correspondent Node (CN). This mode is aiming to reduce the packet loss during the HNP renumbering but the binding state corresponding to the old HNP should be marked for example as transient binding [RFC6058]. This temporary binding should only be used for the downwards packet transmission and the LMA should stop broadcasting the routing information about the old HNP if the old HNP is no longer anchored at this LMA.

(2) Hard-mode

If the HNP renumbering happens with the switchover of the LMA, the hard-mode is recommended to keep the protocol simple. In this mode, the LMA deletes the binding state of the old HNP after it receives the UPA message from the MAG and the LMA silently discards the packets destined to the old HNP.

5. Message Format

(1) UPN message

In the UPN message sent from the LMA to the MAG, the notification reason is set to 2 (UPDATE-SESSION-PARAMETERS). Besides, the HNP Option [RFC5213] containing the new HNP and the Mobile Node Identifier Option [RFC4283] carrying identifier of MN are contained as Mobility Options of UPN. The order of HNP Option and Mobile Node Identifier Option in the UPN message is not mandated in this draft.

(2) UPA message

The MAG sends this message in order to acknowledge that it has received an UPN message with the (A) flag set and to indicate the status after processing the message. When the MAG did not successfully renumber the HNP which is required in the UPN message, the Status Code of 128 is set in the UPA message and the following operation of LMA is PMIPv6 service provider specific.

(3) RA Message

When the RA message is used by the MAG to advise the new HNP, two Prefix Information Options are contained in the RA message [RFC4861]. In the first Prefix Information Option, the old HNP is carried but both the related Valid Lifetime and Preferred Lifetime are set to 0. In the second Prefix Information Option, the new HNP is carried with the Valid Lifetime and Preferred Lifetime set to larger than 0.

(4) DHCP Message

When the DHCP is used in PMIPv6 to configure the addresses for the MN, new IPv6 address(es) (e.g., HoA) will be generated based on the new HNP and the related DHCP procedure is also triggered by the reception of UPN message [RFC3315].

6. Other Issues

In order to maintain the reachability of the MN, the Domain Name System (DNS) resource record corresponding to this MN may need to be updated when the HNP of MN changes [RFC3007]. However, this is beyond the scope of this document.

7. Security Considerations

The protection of UPN and UPA messages in this document follows [RFC5213] and [RFC7077]. This extension causes no further security problem.

8. IANA Considerations

This document presents no IANA considerations.

9. References

9.1. 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>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <http://www.rfc-editor.org/info/rfc3007>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4283] Patel, A., Leung, K., Khalil, M., Akhtar, H., and K. Chowdhury, "Mobile Node Identifier Option for Mobile IPv6 (MIPv6)", RFC 4283, DOI 10.17487/RFC4283, November 2005, <http://www.rfc-editor.org/info/rfc4283>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[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>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[RFC6463] Korhonen, J., Ed., Gundavelli, S., Yokota, H., and X. Cui, "Runtime Local Mobility Anchor (LMA) Assignment Support for Proxy Mobile IPv6", RFC 6463, DOI 10.17487/RFC6463, February 2012, <http://www.rfc-editor.org/info/rfc6463>.

[RFC7077] Krishnan, S., Gundavelli, S., Liebsch, M., Yokota, H., and J. Korhonen, "Update Notifications for Proxy Mobile IPv6", RFC 7077, DOI 10.17487/RFC7077, November 2013, <http://www.rfc-editor.org/info/rfc7077>.

9.2. Informative References

[RFC6058] Liebsch, M., Ed., Muhanna, A., and O. Blume, "Transient Binding for Proxy Mobile IPv6", RFC 6058, DOI 10.17487/RFC6058, March 2011, <http://www.rfc-editor.org/info/rfc6058>.

[RFC6879] Jiang, S., Liu, B., and B. Carpenter, "IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods", RFC 6879, DOI 10.17487/RFC6879, February 2013, <http://www.rfc-editor.org/info/rfc6879>.

Authors' Addresses

Zhiwei Yan
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing 100190
China

EMail: yan@cnnic.cn

Jong-Hyouk Lee
Sangmyung University
31, Sangmyeongdae-gil, Dongnam-gu
Cheonan 31066
Republic of Korea

EMail: jonghyouk@smu.ac.kr

Xiaodong Lee
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing 100190
China

EMail: xl@cnnic.cn

DMM WG
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2017

D. Patki
S. Gundavelli
Cisco
J. Lee
Sangmyung University
Q. Fu
China Mobile
L. Bertz
Sprint

July 1, 2016

LMA Controlled MAG Session Parameters
draft-ietf-dmm-lma-controlled-mag-params-02.txt

Abstract

This specification defines a new extension, LMA-Controlled-MAG-Session-Params to Proxy Mobile IPv6. This option can be used by the LMA in PMIPv6 signaling for notifying the MAG to conform to various parameters contained in this extension.

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 January 2, 2017.

Copyright Notice

Table of Contents

1. Introduction

A large PMIPv6 deployment, such as residential deployment, can have tens of thousands of MAGs spread across geographical locations. While it can be operationally challenging to manage such a large number of MAGs, it can also be very difficult to ensure configuration consistency across all the MAGs if they are not centrally managed. Configuring aggressive values of parameters such as re-registration timeout and heartbeat interval can potentially create considerable signaling load on the LMA. This document provides a new option to enable the LMA to control various parameters on the MAG such as the re-registration frequency [RFC5213] and heartbeat frequency [RFC5847]. With this option, the configuration of these tunable parameters done centrally on the LMA enables Service Providers to have better control on the behavior of the MAGs with deterministic singaling load on the LMA.

2. Conventions and Terminology

2.1. Conventions

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].

2.2. Terminology

All the terms used in this document are to be interpreted as defined in [RFC5213], [RFC5847] and [RFC7563].

3. Protocol Extension

The LMA Controlled MAG Parameters (LCMP) option is a mobility header option used to exchange information related to the parameters that a local mobility anchor enforces on a mobile access gateway. The option can be included in Proxy Binding Acknowledgement (PBA) message only, and there MUST NOT be more than a single instance of this mobility option in a mobility message. This mobility option MUST contain one or more LMA Controlled MAG Parameters sub-options. The suboptions are defined in Section 3.1. The alignment of this option MUST be 4n [RFC2460].

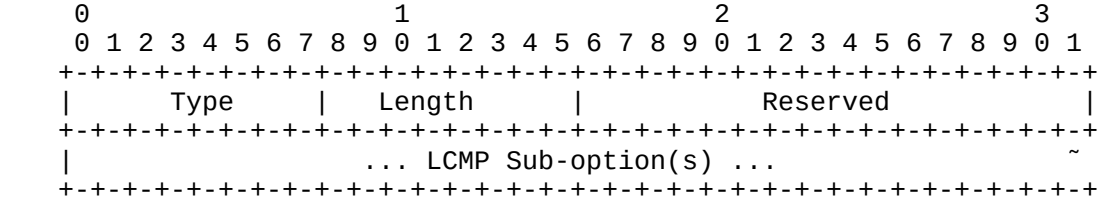


Figure 1: LMA Controlled MAG Parameters Option

Type

MUST be set to the value of IANA-1, indicating that it is a LMA-Controlled-MAG-Parameters option.

Length

8-bit unsigned integer indicating the length in octets of the option, excluding the Type and Length fields.

Reserved

MUST be set to zero when sending and ignored when received.

LCMP Sub-option(s)

LCMP Sub-options are described in the below sections. The sub-options are optional and can be present in any order.

3.1. Format of the LCMP Sub-Options

The LMA Controlled MAG Parameters sub-options are used for carrying information elements related to various parameters that need to be configured on the MAG. These sub-options can be included in the LMA Controlled MAG Parameters option defined in Section 3. The alignment of the sub-option MUST be 4n. The format of this sub-option is as follows.

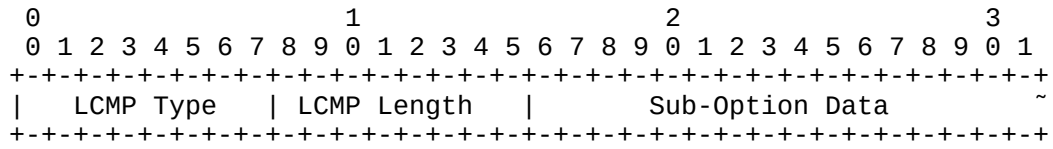


Figure 2: LMA Controlled MAG Parameters Sub-Option

Type

8-bit unsigned integer indicating the type of the LMA Controlled MAG Parameters sub-option. This specification defines the following types:

0 - Reserved

1 - Binding Refresh Control Sub-Option

2 - Heartbeat Control Sub-Option

Length

8-bit unsigned integer indicating the number of octets needed to encode the Option Data, excluding the LCMP Type and LCMP Length fields of the sub-option.

3.1.1. Binding Re-registration Control Sub-Option

The Binding Re-registration Control Sub-Option is a mobility sub-option carried in the LMA Controlled MAG Parameters mobility option defined in Section 3.1. This sub-option carries re-registration related timer values. There MUST be no more than a single instance of this sub-option in LMA Controlled MAG Parameters option. The format of this sub-option is defined below.

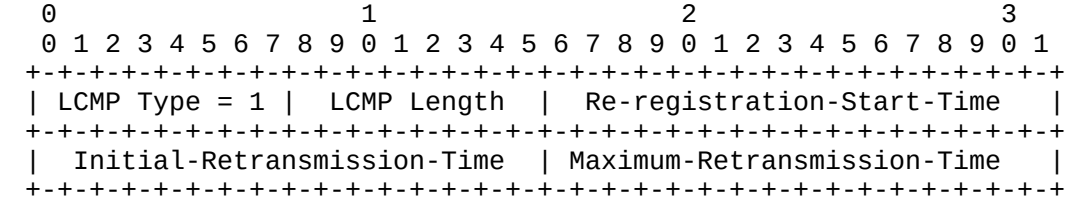


Figure 3: Binding Re-registration Control Sub-Option

Re-registration-Start-Time

16-bit unsigned integer indicating the number of time units before the expiry of the PMIPv6 binding lifetime when the registration refresh process needs to be activated. One time unit is 4 seconds.

Initial-Retransmission-Time

16-bit unsigned integer indicating minimum delay in seconds before the first PBU retransmission of the exponential back-off process.

Maximum-Retransmission-Time

16-bit unsigned integer indicating maximum delay in seconds before the last PBU retransmission message of the exponential back-off process.

3.1.2. Heartbeat Control Sub-Option

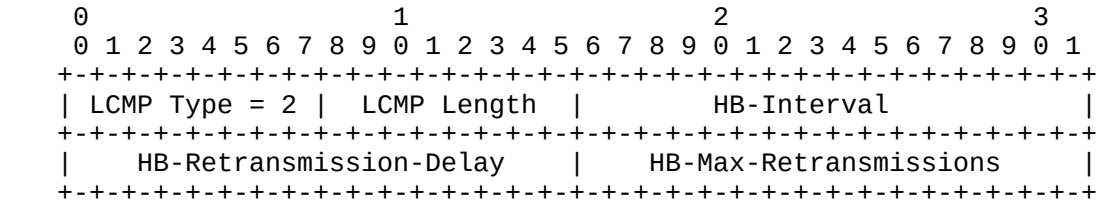


Figure 4: Heartbeat Control Sub-Option

HB-Interval

16-bit unsigned integer indicating heartbeat interval, i.e. time delay in seconds after a successful heartbeat exchange (request followed by response) when the next heartbeat exchange can be triggered.

HB-Retransmission-Delay

16-bit unsigned integer indicating minimum time delay in seconds before a heartbeat message is retransmitted.

HB-Max-Retransmissions

16-bit unsigned integer indicating maximum number of heartbeat retransmissions.

4. Protocol Configuration Variables

4.1. Local Mobility Anchor - Configuration Variables

The local mobility anchor MUST allow the following variables to be configured by the system management. The configured values for these protocol variables MUST survive server reboots and service restarts.

EnableLCMPSubOptReregControl

This flag indicates the operational state of the Binding Re-registration Control sub-option support. The default value for this flag is set to (0), indicating that support for the Binding Re-registration Control sub-option is disabled.

When this flag on the local mobility anchor is set to a value of (1), the local mobility anchor SHOULD include this sub-option in the Proxy Binding Acknowledge messages that it sends to the mobile access gateway; otherwise, it MUST NOT include the sub-option. There can be situations where the local mobility anchor is unable

to obtain the Binding Re-registration Control information and may not be able to construct this sub-option.

EnableLCMPSubOptHeartbeatControl

This flag indicates the operational state of the Heartbeat Control sub-option support. The default value for this flag is set to (0), indicating that support for the Heartbeat Control sub-option is disabled.

When this flag on the local mobility anchor is set to a value of (1), the local mobility anchor SHOULD include this sub-option in the Proxy Binding Acknowledge messages that it sends to the mobile access gateway; otherwise, it MUST NOT include the sub-option. There can be situations where the local mobility anchor is unable to obtain the Heartbeat Control information and may not be able to construct this sub-option.

The following variables MAY be defined at various granularity such as per binding, per peering MAG, per cluster of MAGs or any other custom grouping. Regardless of the granularity of this configuration, the local mobility anchor should be able to determine the value of these variables on an individual binding basis by way of configuration hierarchy.

LCMPReregistrationStartTime

This variable is used to set the minimum time interval in number of seconds before the expiry of the PMIPv6 binding lifetime when the registration refresh process SHOULD be activated. The default value is 10 units, where each unit is 4 seconds.

LCMPInitialRetransmissionTime

This variable is used to set the minimum delay in seconds before the first PBU retransmission of the exponential back-off process. This variable is same as INITIAL_BINDACK_TIMEOUT mentioned in Section 6.9.4 of [RFC5213]. The default value is 1 second.

LCMPMaximumRetransmissionTime

This variable is used to set the maximum delay in seconds before the last PBU retransmission message of the exponential back-off process. This variable is same as MAX_BINDACK_TIMEOUT mentioned in Section 6.9.4 of [RFC5213]. The default value is 32 seconds.

LCMPHeartbeatInterval

This variable is used to set the time delay in seconds after a successful heartbeat exchange (request followed by response) when the next heartbeat exchange can be triggered. The default value is 60 seconds. It SHOULD NOT be set to less than 30 seconds or more than 3600 seconds. The value of this variable MAY be derived from the variable HEARTBEAT_INTERVAL defined in Section 5 of [RFC5847] if defined on the local mobility anchor.

LCMPHeartbeatRetransmissionDelay

This variable is used to set the minimum time delay in seconds before a heartbeat message is retransmitted. The value of this variable SHOULD be less than LCMP_HEARTBEAT_INTERVAL. The default value is 5 seconds.

LCMPHeartbeatMaxRetransmissions

This variable is used to set the maximum number of heartbeat retransmissions. The default value for this variable is 3. The value of this variable MAY be derived from the variable MISSING_HEARTBEATS_ALLOWED defined in Section 5 of [RFC5847] if defined on the local mobility anchor.

5. Protocol Considerations

The following considerations apply to the local mobility anchor and the mobile access gateway.

The conceptual Binding Cache Entry data structure maintained by the local mobility anchor, described in Section 5.1 of [RFC5213] and the conceptual Binding Update List entry data structure maintained by the mobile access gateway, described in Section 6.1 of [RFC5213], MUST be extended to store the LMA Controlled MAG Parameters option related information elements associated with the current session. Specifically the following parameters MUST be defined:

- LCMPReregistrationStartTime
- LCMPInitialRetransmissionTime
- LCMPMaximumRetransmissionTime
- LCMPHeartbeatInterval
- LCMPHeartbeatRetransmissionDelay
- LCMPHeartbeatMaxRetransmissions

5.1. Local Mobility Anchor Considerations

- On receiving a Proxy Binding Update message [RFC5213] from a mobile access gateway, the local mobility anchor should check if EnableLCMPSubOptReregControl is set to (1). If yes, and if all of LCMPReregistrationStartTime, LCMPInitialRetransmissionTime and LCMPMaximumRetransmissionTime are set to NON_ZERO values, then in SHOULD include Binding Re-registration Control Sub-Option in the LMA Controlled MAG Parameters mobility option which is in turn included in the Proxy Binding Acknowledge message.

- If EnableLCMPSubOptReregControl is set to (1) and if any of LCMPReregistrationStartTime, LCMPInitialRetransmissionTime and LCMPMaximumRetransmissionTime is set to ZERO value, then the local mobility anchor should report a configuration error.

- The local mobility anchor should also check if EnableLCMPSubOptHeartbeatControl is set to (1). If yes, and if all of LCMPHeartbeatInterval, LCMPHeartbeatRetransmissionDelay and LCMPHeartbeatMaxRetransmissions are set to NON_ZERO values, then in SHOULD include Heartbeat Control Sub-Option in the LMA Controlled MAG Parameters mobility option which is in turn included in the Proxy Binding Acknowledge message.

- If EnableLCMPSubOptHeartbeatControl is set to (1) and if any of LCMPHeartbeatInterval, LCMPHeartbeatRetransmissionDelay and LCMPHeartbeatMaxRetransmissions is set to ZERO value, then the local mobility anchor should report a configuration error.

5.2. Mobile Access Gateway Considerations

- On Receiving Proxy Binding Acknowledge message [RFC5213] from the local mobility anchor with LMA Controlled MAG Parameters mobility option, the mobile access gateway MUST overwrite the binding re-registration related timer parameters with the parameters received in Binding Re-registration Control Sub-Option, if present in the LMA Controlled MAG Parameters mobility option. Similarly, the mobile access gateway MUST overwrite the heartbeat related timer parameters with the parameters received in Heartbeat Control Sub-Option, if present in the LMA Controlled MAG Parameters mobility option.

- If any of the parameters in the Binding Re-registration Control Sub-Option is ZERO, then the sub-option MUST be ignored and an error message SHOULD be logged.

- If any of the parameters in the Heartbeat Control Sub-Option except HB-Retransmission-Delay is ZERO, then the sub-option MUST be ignored and error message SHOULD be logged.

6. IANA Considerations

This document requires the following IANA actions.

- Action 1: This specification defines a new mobility header option, the LMA Controlled MAG Parameters. This mobility option is described in Section 3. The type value (IANA-1) for this option needs to be assigned from the same numbering space as allocated for the other mobility options, as defined in [RFC6275].

- Action 2: This specification defines a new mobility sub-option format, the LMA Controlled MAG Parameters sub-option. The format of this mobility sub-option is described in Section 3.1. This sub-option can be carried in the LMA Controlled MAG Parameters option. The type value for this sub-option needs to be managed by IANA, under the registry "LMA Controlled MAG Parameters Sub-Option Type Values". This specification reserves the following type values. Approval of new LMA Controlled MAG Parameters sub-option type values are to be made through IANA Expert Review.

0	Reserved
1	Binding Re-registration Control Sub-Option
2	Heartbeat Control Sub-Option

7. Security Considerations

The LMA Controlled MAG Parameters option defined in this specification is for use in Proxy Binding Acknowledgement message. This option is carried like any other mobility header option as specified in [RFC6275] and does not require any special security considerations.

8. References

8.1. 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>.

[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>.

[RFC5847] Devarapalli, V., Ed., Koodli, R., Ed., Lim, H., Kant, N., Krishnan, S., and J. Laganier, "Heartbeat Mechanism for Proxy Mobile IPv6", RFC 5847, DOI 10.17487/RFC5847, June 2010, <http://www.rfc-editor.org/info/rfc5847>.

[RFC7563] Pazhyannur, R., Speicher, S., Gundavelli, S., Korhonen, J., and J. Kaippallimalil, "Extensions to the Proxy Mobile IPv6 (PMIPv6) Access Network Identifier Option", RFC 7563, DOI 10.17487/RFC7563, June 2015, <http://www.rfc-editor.org/info/rfc7563>.

8.2. Informative References

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

Authors' Addresses

Dhananjay Patki
Cisco
Cessna Business Park SEZ, Kadubeesanahalli
Bangalore, Karnataka 560087
India

Email: dhpatki@cisco.com

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Jong-Hyouk Lee
Sangmyung University
31, Sangmyeongdae-gil, Dongnam-gu
Cheonan 330-720
Republic of Korea

Email: jonghyouk@smu.ac.kr

Qiao Fu
China Mobile
Xuanwumenxi Ave. No.32
Beijing
China

Email: fuqiao1@outlook.com

Lyle T Bertz
Sprint
Kansas
USA

Email: Lyle.T.Bertz@sprint.com

DMM WG
Internet-Draft
Intended status: Standards Track
Expires: January 26, 2017

P. Seite
Orange
A. Yegin
Samsung
S. Gundavelli
Cisco

July 25, 2016

MAG Multipath Binding Option
draft-ietf-dmm-mag-multihoming-02.txt

Abstract

The document [RFC4908] proposes to rely on multiple Care-of Addresses (CoAs) capabilities of Mobile IP [RFC6275] an Network Mobility (NEMO; [RFC3963]) to enable Multihoming technology for Small-Scale Fixed Networks. In the continuation of [RFC4908], this document specifies a multiple proxy Care-of Addresses (pCoAs) extension for Proxy Mobile IPv6 [RFC5213]. This extension allows a multihomed Mobile Access Gateway (MAG) to register more than one proxy care-of-address to the Local Mobility Anchor (LMA).

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 January 26, 2017.

Copyright Notice

Table of Contents

1. Introduction

Using several links, the multihoming technology can improve connectivity availability and quality of communications; the goals and benefits of multihoming are as follows:

- Redundancy/Fault-Recovery
- Load balancing
- Load sharing
- Preferences settings

According to [RFC4908], users of Small-Scale Networks can take benefit of multihoming using mobile IP [RFC6275] and Network Mobility (NEMO) [RFC3963] architecture in a mobile and fixed networking environment. This document is introducing the concept of multiple Care-of Addresses (CoAs) [RFC5648] that have been specified since then.

In the continuation of [RFC4908], a Proxy Mobile IPv6 [RFC5213] based multihomed achitecture could be defined. The motivation to update [RFC4908] with proxy Mobile IPv6 is to leverage on latest mobility working group achievments, namely:

- using GRE as mobile tuneling, possibly with its key extension [RFC5845] (a possible reason to use GRE is given on Section 3.2).
- using UDP encapsulation [RFC5844] in order to support NAT traversal in IPv4 networking environment.
- Prefix Delegation mechanism [RFC7148].
- Using the vendor specific mobility option [RFC5094], for example to allow the MAG and LMA to exchange information (e.g. WAN interface QoS metrics) allowing to make appropriate traffic steering decision.

Proxy Mobile IPv6 (PMIPv6) relies on two mobility entities: the mobile access gateway (MAG), which acts as the default gateway for the end-node and the local mobility anchor (LMA), which acts as the topological anchor point. Point-to-point links are established, using IP-in-IP tunnels, between MAG and LMA. Then, the MAG and LMA are distributing traffic over these tunnels. All PMIPv6 operations are performed on behalf of the end-node and its corespondent node, it thus makes PMIPv6 well adapted to multihomed architecture as considered in [RFC4908]. Taking the LTE and WLAN networking environments as an example, the PMIPv6 based multihomed architecture is depicted on Figure 1. Flow-1,2 and 3 are distributed either on Tunnel-1 (over LTE) or Tunnel-2 (over WLAN), while Flow-4 is spread on both Tunnel-1 and 2.

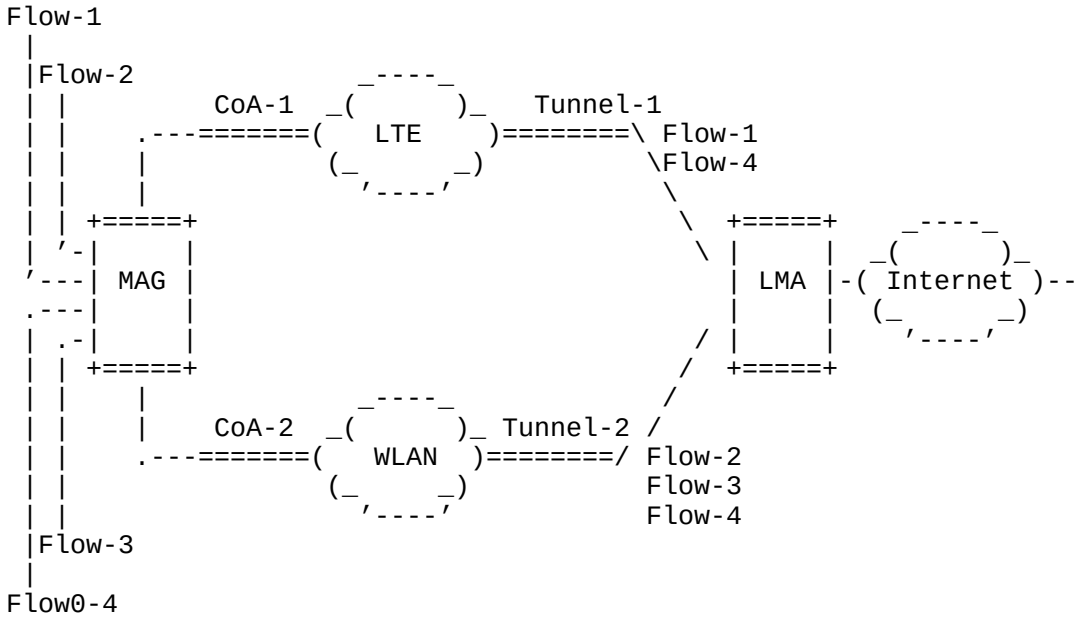


Figure 1: Multihomed MAG using Proxy Mobile IPv6

The current version of Proxy Mobile IPv6 does not allow a MAG to register more than one proxy Care-of-Adresse to the LMA. In other words, only one MAG/LMA link, i.e. IP-in-IP tunnel, can be used at the same time. This document overcomes this limitation by defining the multiple proxy Care-of Addresses (pCoAs) extension for Proxy Mobile IPv6.

2. Conventions and Terminology

2.1. Conventions

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 RFC 2119 [RFC2119].

2.2. Terminology

All mobility related terms used in this document are to be interpreted as defined in [RFC5213], [RFC5844] and [RFC7148]. Additionally, this document uses the following terms:

IP-in-IP

IP-within-IP encapsulation [RFC2473], [RFC4213]

3. Overview

3.1. Example Call Flow

Figure 2 is the callflow detailing multi-access support with PMIPv6. The MAG in this example scenario is equipped with both WLAN and LTE interfaces and is also configured with the multihoming functionality. The steps of the callflow are as follows:

Steps (1) and (2): the MAG attaches to both WLAN and LTE networks; the MAG obtains respectively two different proxy care-of-addresses (pCoA).

Step (3): The MAG sends, over the WLAN access, a Proxy Binding Update (PBU) message, with the new MAG Multipath Binding (MMB) and MAG Identifier (MAG-NAI) options to the LMA. A logical-NAI (MAG-NAI) with ALWAYS-ON configuration is enabled on the MAG. The mobility session that is created (i.e. create a Binding Cache Entry) on the LMA is for the logical-NAI. The LMA and allocates a Home Network Prefix (HNP), that shall be delegated to mobile nodes, to the MAG.

Step (4): the LMA sends back a Proxy Binding Acknowledgement (PBA) including the HNP allocated to the MAG.

Step (5): IP tunnel (IP-in-IP, GRE ...) is created over the WLAN access.

Steps (6) to (8): The MAG repeats steps (3) to (5) on the LTE access. The MAG includes the HNP, received on step (4) in the PBU. The LMA update its binding cache by creating a new mobility session for this MAG.

Steps (9) and (10): The IP hosts MN_1 and MN_2 are assigned IP addresses from the mobile network prefix delegated by the MAG.

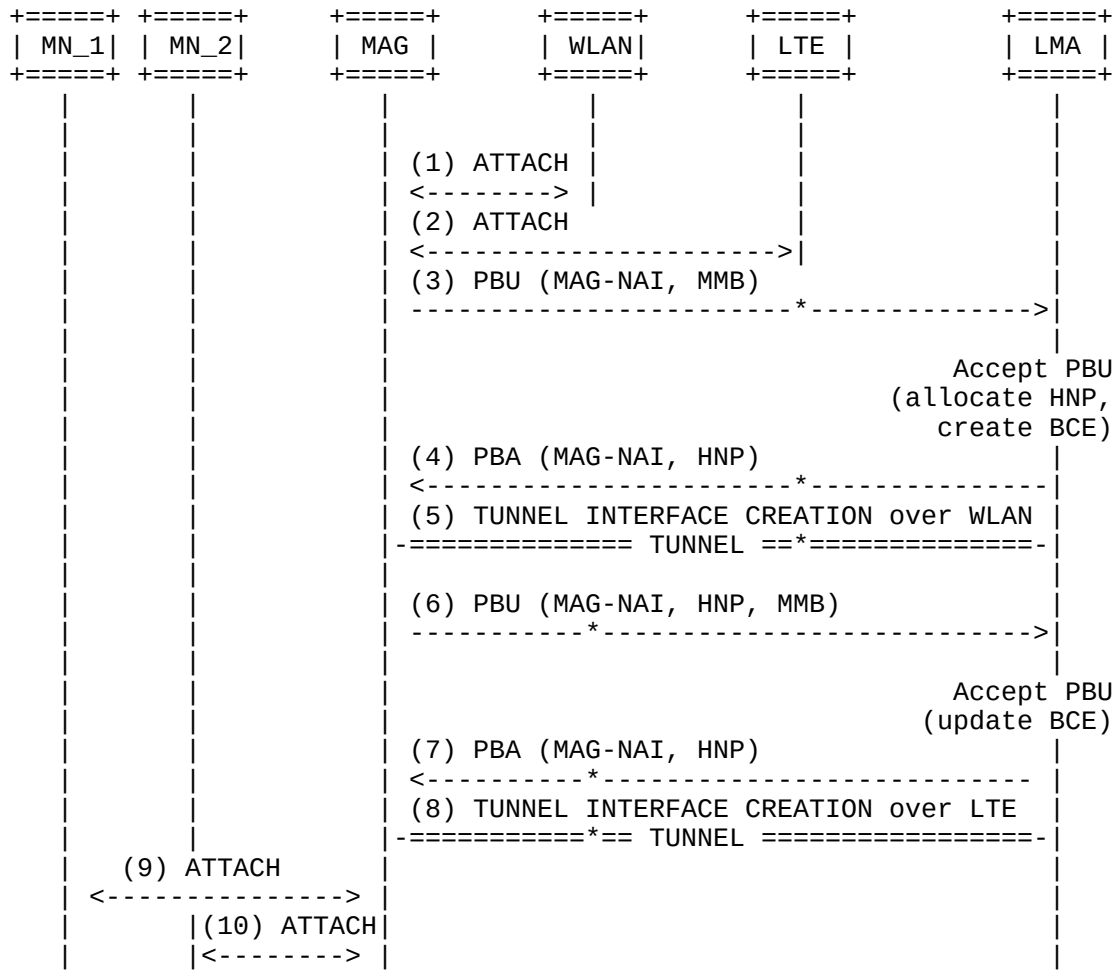


Figure 2: Functional Separation of the Control and User Plane

3.2. Traffic distribution schemes

When receiving packets from the MN, the MAG distributes packets over tunnels that have been established. Traffic distribution can be managed either on a per-flow or on a per-packet basis:

- Per-flow traffic management: each IP flow (both upstream and downstream) is mapped to a given tunnel, corresponding to a given WAN interface. Flow binding extension [RFC6089] is used to exchange, and synchronize, IP flow management policies (i.e. rules associating traffic selectors [RFC6088] to a tunnel).

- o Per-packet management: the LMA and the MAG distribute packets, belonging to a same IP flow, over more than one bindings (i.e. more than one WAN interface). When operating at the IP packet level, different packets distribution algorithms are possible. For example, the algorithm may give precedence to one given access: the MAG overflows traffic from the primary access, e.g. WLAN, to the second one, only when load on primary access reaches a given threshold. The distribution algorithm is left to implementer but whatever the algorithm is, packets distribution likely introduces packet latency and out-of-order delivery. LMA and MAG shall thus be able to make reordering before packets delivery. Sequence number can be can be used for that purpose, for example using GRE with sequence number option [RFC5845]. However, more detailed considerations on reordering and IP packet distribution scheme (e.g. definition of packets distribution algorithm) are out the scope of this document.

Because latency introduced by per-packet can cause injury to some application, per-flow and per-packet distribution schemes could be used in conjunction. For example, high throughput services (e.g. video streaming) may benefit from per-packet distribution scheme, while latency sensitive applications (e.g. VoIP) are not be spread over different WAN paths. IP flow mobility extensions, [RFC6089] and [RFC6088], can be used to provision the MAG with such flow policies.

4. Protocol Extensions

4.1. MAG Multipath-Binding Option

The MAG Multipath-Binding option is a new mobility header option defined for use with Proxy Binding Update and Proxy Binding Acknowledgement messages exchanged between the local mobility anchor and the mobile access gateway.

This mobility header option is used for requesting multipath support. It indicates that the mobile access gateway is requesting the local mobility anchor to register the current care-of address associated with the request as one of the many care-addresses through which the mobile access gateway can be reached. It is also for carrying the information related to the access network associated with the care-of address.

The MAG Multipath-Binding option has an alignment requirement of 8n+2. Its format is as shown in Figure 3:

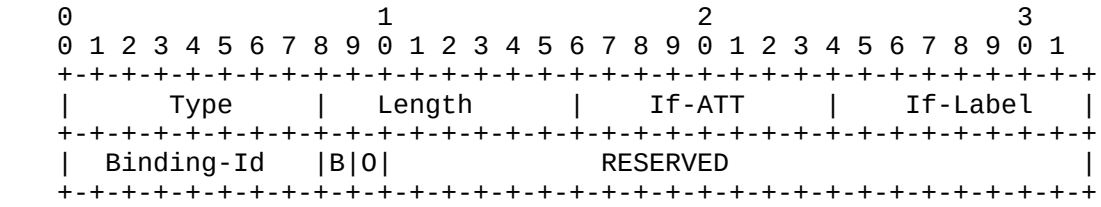


Figure 3: MAG Multipath Binding Option

Type

<IANA-1> To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields.

Interface Access-Technology Type (If-ATT)

This 8-bit field identifies the Access-Technology type of the interface through which the mobile node is connected. The permitted values for this are from the Access Technology Type registry defined in [RFC5213].

Interface Label (If-Label)

This 8-bit field represents the interface label represented as an unsigned integer. The MAG identifies the label for each of the interfaces through which it registers a pCoA with the LMA. When using static traffic flow policies on the mobile node and the home agent, the label can be used for generating forwarding policies. For example, the operator may have policy which binds traffic for Application "X" needs to interface with Label "Y". When a registration through an interface matching Label "Y" gets activated, the home agent and the mobile node can dynamically generate a forwarding policy for forwarding traffic for Application "X" through mobile IP tunnel matching Label "Y". Both the home agent and the mobile node can route the Application-X traffic through that interface. The permitted values for If-Label are 1 through 255.

Binding-Identifier (BID)

This 8-bit field is used for carrying the binding identifier. It uniquely identifies a specific binding of the mobile node, to which this request can be associated. Each binding identifier is

represented as an unsigned integer. The permitted values are 1 through 254. The BID value of 0 and 255 are reserved. The mobile access gateway assigns a unique value for each of its interfaces and includes them in the message.

Bulk Re-registration Flag (B)

This flag, if set to a value of (1), is to notify the local mobility anchor to consider this request as a request to update the binding lifetime of all the mobile node's bindings, upon accepting this specific request. This flag MUST NOT be set to a value of (1), if the value of the Registration Overwrite Flag (O) is set to a value of (1).

Binding Overwrite (O)

This flag, if set to a value of (1), notifies the local mobility anchor that upon accepting this request, it should replace all of the mobile node's existing bindings with this binding. This flag MUST NOT be set to a value of (1), if the value of the Bulk Re-registration Flag (B) is set to a value of (1). This flag MUST be set to a value of (0), in de-registration requests.

Reserved

This field is unused in this specification. The value MUST be set to zero (0) by the sender and MUST be ignored by the receiver.

4.2. MAG Identifier Option

The MAG Identifier option is a new mobility header option defined for use with Proxy Binding Update and Proxy Binding Acknowledgement messages exchanged between the local mobility anchor and the mobile access gateway. This mobility header option is used for conveying the MAG's identity.

This option does not have any alignment requirements.

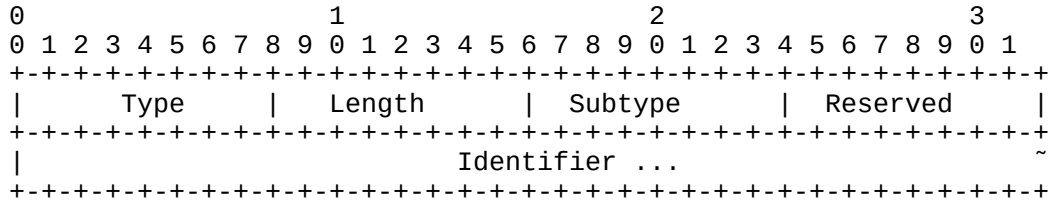


Figure 4: MAG Identifier Option

Type

<IANA-2> To be assigned by IANA.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields.

Subtype

One byte unsigned integer used for identifying the type of the Identifier field. Accepted values for this field are the registered type values from the Mobile Node Identifier Option Subtypes registry.

Reserved

This field is unused in this specification. The value MUST be set to zero (0) by the sender and MUST be ignored by the receiver.

Identifier

A variable length identifier of type indicated in the Subtype field.

4.3. New Status Code for Proxy Binding Acknowledgement

This document defines the following new Status Code value for use in Proxy Binding Acknowledgement message.

CANNOT_SUPPORT_MULTIPATH_BINDING (Cannot Support Multipath Binding): <IANA-4>

5. IANA Considerations

This document requires the following IANA actions.

- Action-1: This specification defines a new mobility option, the MAG Multipath-Binding option. The format of this option is described in Section 4.1. The type value <IANA-1> for this mobility option needs to be allocated from the Mobility Options registry at <http://www.iana.org/assignments/mobility-parameters>. RFC Editor: Please replace <IANA-1> in Section 4.1 with the assigned value and update this section accordingly.

- Action-2: This specification defines a new mobility option, the MAG Identifier option. The format of this option is described in

Section 4.2. The type value <IANA-2> for this mobility option needs to be allocated from the Mobility Options registry at <http://www.iana.org/assignments/mobility-parameters>. RFC Editor: Please replace <IANA-2> in Section 4.2 with the assigned value and update this section accordingly.

- Action-3: This document defines a new status value, CANNOT_SUPPORT_MULTIPATH_BINDING (<IANA-3>) for use in Proxy Binding Acknowledgement message, as described in Section 4.3. This value is to be assigned from the "Status Codes" registry at <http://www.iana.org/assignments/mobility-parameters>. The allocated value has to be greater than 127. RFC Editor: Please replace <IANA-4> in Section 4.3 with the assigned value and update this section accordingly.

6. Security Considerations

This specification allows a mobile access gateway to establish multiple Proxy Mobile IPv6 tunnels with a local mobility anchor, by registering a care-of address for each of its connected access networks. This essentially allows the mobile node’s IP traffic to be routed through any of the tunnel paths and either based on a static or a dynamically negotiated flow policy. This new capability has no impact on the protocol security. Furthermore, this specification defines two new mobility header options, MAG Multipath-Binding option and the MAG Identifier option. These options are carried like any other mobility header option as specified in [RFC5213]. Therefore, it inherits security guidelines from [RFC5213]. Thus, this specification does not weaken the security of Proxy Mobile IPv6 Protocol, and does not introduce any new security vulnerabilities.

7. Acknowledgements

The authors of this draft would like to acknowledge the discussions and feedback on this topic from the members of the DMM working group.

8. References

8.1. 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>.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <http://www.rfc-editor.org/info/rfc3963>.

[RFC5094] Devarapalli, V., Patel, A., and K. Leung, "Mobile IPv6 Vendor Specific Option", RFC 5094, DOI 10.17487/RFC5094, December 2007, <http://www.rfc-editor.org/info/rfc5094>.

[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>.

[RFC5648] Wakikawa, R., Ed., Devarapalli, V., Tsirtsis, G., Ernst, T., and K. Nagami, "Multiple Care-of Addresses Registration", RFC 5648, DOI 10.17487/RFC5648, October 2009, <http://www.rfc-editor.org/info/rfc5648>.

[RFC5844] Wakikawa, R. and S. Gundavelli, "IPv4 Support for Proxy Mobile IPv6", RFC 5844, DOI 10.17487/RFC5844, May 2010, <http://www.rfc-editor.org/info/rfc5844>.

[RFC5845] Muhanna, A., Khalil, M., Gundavelli, S., and K. Leung, "Generic Routing Encapsulation (GRE) Key Option for Proxy Mobile IPv6", RFC 5845, DOI 10.17487/RFC5845, June 2010, <http://www.rfc-editor.org/info/rfc5845>.

[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>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[RFC7148] Zhou, X., Korhonen, J., Williams, C., Gundavelli, S., and CJ. Bernardos, "Prefix Delegation Support for Proxy Mobile IPv6", RFC 7148, DOI 10.17487/RFC7148, March 2014, <http://www.rfc-editor.org/info/rfc7148>.

8.2. Informative References

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <http://www.rfc-editor.org/info/rfc2473>.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <http://www.rfc-editor.org/info/rfc4213>.

[RFC4908] Nagami, K., Uda, S., Ogashiwa, N., Esaki, H., Wakikawa, R., and H. Ohnishi, "Multi-homing for small scale fixed network Using Mobile IP and NEMO", RFC 4908, DOI 10.17487/RFC4908, June 2007, <http://www.rfc-editor.org/info/rfc4908>.

Authors' Addresses

Pierrick Seite
Orange
4, rue du Clos Courtel, BP 91226
Cesson-Sevigne 35512
France

Email: pierrick.seite@orange.com

Alper Yegin
Samsung
Istanbul
Turkey

Email: alper.yegin@partner.samsung.com

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

DMM Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2017

A. Yegin
Actility
D. Moses
Intel
K. Kweon
J. Lee
J. Park
Samsung

July 6, 2016

On Demand Mobility Management
draft-ietf-dmm-ondemand-mobility-07

Abstract

Applications differ with respect to whether they need IP session continuity and/or IP address reachability. The network providing the same type of service to any mobile host and any application running on the host yields inefficiencies. This document describes a solution for taking the application needs into account in selectively providing IP session continuity and IP address reachability on a per-socket basis.

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 January 7, 2017.

Table of Contents

1. Introduction

In the context of Mobile IP [RFC5563][RFC6275][RFC5213][RFC5944], following two attributes are defined for the IP service provided to the mobile hosts:

IP session continuity: The ability to maintain an ongoing IP session by keeping the same local end-point IP address throughout the session despite the mobile host chaging its point of attachment within the IP network topology. The IP address of the host may change between two independent IP sessions, but that does not jeopardize the IP session continuity. IP session continuity is essential for mobile hosts to maintain ongoing flows without any interruption.

IP address reachability: The ability to maintain the same IP address for an extended period of time. The IP address stays the same across independent IP sessions, and even in the absence of any IP session. The IP address may be published in a long-term registry (e.g., DNS),

and it is made available for serving incoming (e.g., TCP) connections. IP address reachability is essential for mobile hosts to use specific/published IP addresses.

Mobile IP is designed to provide both IP session continuity and IP address reachability to mobile hosts. Architectures utilizing these protocols (e.g., 3GPP, 3GPP2, WIMAX) ensure that any mobile host attached to the compliant networks can enjoy these benefits. Any application running on these mobile hosts is subjected to the same treatment with respect to the IP session continuity and IP address reachability.

It should be noted that in reality not every application may need those benefits. IP address reachability is required for applications running as servers (e.g., a web server running on the mobile host). But, a typical client application (e.g., web browser) does not necessarily require IP address reachability. Similarly, IP session continuity is not required for all types of applications either. Applications performing brief communication (e.g., DNS client) can survive without having IP session continuity support.

Achieving IP session continuity and IP address reachability by using Mobile IP incurs some cost. Mobile IP protocol forces the mobile host's IP traffic to traverse a centrally-located router (Home Agent, HA), which incurs additional transmission latency and use of additional network resources, adds to the network CAPEX and OPEX, and decreases the reliability of the network due to the introduction of a single point of failure [I-D.ietf-dmm-requirements]. Therefore, IP session continuity and IP address reachability should be be provided only when needed.

Furthermore, when an application needs session continuity, it may be able to satisfy that need by using a solution above the IP layer, such as MPTCP [RFC6824], SIP mobility [RFC3261], or an application-layer mobility solution. Those higher-layer solutions are not subject to the same issues that arise with the use of Mobile IP since they can utilize the most direct data path between the end-points. But, if Mobile IP is being applied to the mobile host, those higher-layer protocols are rendered useless because their operation is inhibited by the Mobile IP. Since Mobile IP ensures the IP address of the mobile host remains fixed (despite the location and movement of the mobile host), the higher-layer protocols never detect the IP-layer change and never engage in mobility management.

This document proposes a solution for the applications running on the mobile host to indicate whether they need IP session continuity or IP address reachability. The network protocol stack on the mobile host, in conjunction with the network infrastructure, would provide the

required type of IP service. It is for the benefit of both the users and the network operators not to engage an extra level of service unless it is absolutely necessary. So it is expected that applications and networks compliant with this specification would utilize this solution to use network resources more efficiently.

2. Notational Conventions

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. Solution

3.1. Types of IP Addresses

Three types of IP addresses are defined with respect to the mobility management.

- Fixed IP Address

A Fixed IP address is an address with a guarantee to be valid for a very long time, regardless of whether it is being used in any packet to/from the mobile host, or whether or not the mobile host is connected to the network, or whether it moves from one point-of-attachment to another (with a different subnet or IP prefix) while it is connected.

Fixed IP address are required by applications that need both IP session continuity and IP address reachability.

- Session-lasting IP Address

A session-lasting IP address is an address with a guarantee to be valid through-out the IP session(s) for which it was requested. It is guaranteed to be valid even after the mobile host had moved from one point-of-attachment to another (with a different subnet or IP prefix).

Session-lasting IP addresses are required by applications that need IP session continuity but do not need IP address reachability.

- Non-persistent IP Address

This type of IP address provides neither IP session continuity nor IP address reachability. The IP address is obtained from the serving IP gateway and it is not maintained across gateway changes. In other words, the IP address may be released and replaced by a new IP

address when the IP gateway changes due to the movement of the mobile host.

Applications running as servers at a published IP address require a Fixed IP Address. Long-standing applications (e.g., an SSH session) may also require this type of address. Enterprise applications that connect to an enterprise network via virtual LAN require a Fixed IP Address.

Applications with short-lived transient IP sessions can use Session-lasting IP Addresses. For example: Web browsers.

Applications with very short IP sessions, such as DNS client and instant messengers, can utilize Non-persistent IP Addresses. Even though they could very well use a Fixed of Session-lasting IP Addresses, the transmission latency would be minimized when a Non-persistent IP Address is used.

The network creates the desired guarantee (Fixed, Session-lasting or Non-persistent) by either assigning an IP address (as part of a stateful IP address generation), or by assigning the address prefix (as part of a stateless address generation process).

3.2. Granularity of Selection

The IP address type selection is made on a per-socket granularity. Different parts of the same application may have different needs. For example, control-plane of an application may require a Fixed IP Address in order to stay reachable, whereas data-plane of the same application may be satisfied with a Session-lasting IP Address.

3.3. On Demand Nature

At any point in time, a mobile host may have a combination of IP addresses configured. Zero or more Non-persistent, zero or more Session-lasting, and zero or more Fixed IP addresses may be configured on the IP stack of the host. The combination may be as a result of the host policy, application demand, or a mix of the two.

When an application requires a specific type of IP address and such address is not already configured on the host, the IP stack shall attempt to configure one. For example, a host may not always have a Session-lasting IP address available. In case an application requests one, the IP stack shall make an attempt to configure one by issuing a request to the network. If the operation fails, the IP stack shall fail the associated socket request. If successful, a Session-lasting IP Address gets configured on the mobile host. If another socket requests a Session-lasting IP address at a later time,

the same IP address may be served to that socket as well. When the last socket using the requested IP address is closed, the IP address may be released or kept for future applications that may be launched and require a Session-lasting IP address.

In some cases it might be preferable for the mobile host to request a new Session-lasting IP address for a new opening of an IP session (even though one was already assigned to the mobile host by the network and might be in use in a different, already active IP session). It is out of the scope of this specification to define criteria for selecting to use available addresses or choose to request new ones. It supports both alternatives (and any combination).

It is outside of the scope of this specification to define how the host requests a specific type of address (Fixed, Session-lasting or Non-persistent) and how the network indicates the type of address in its advertisement of addresses (or in its reply to an address request).

The following are matters of policy, which may be dictated by the host itself, the network operator, or the system architecture standard:

- The initial set of IP addresses configured on the host at the boot time.

- Permission to grant various types of IP addresses to a requesting application.

- Determination of a default address type when an application does not make any explicit indication, whether it already supports the required API or it is just a legacy application.

3.4. Conveying the Selection

The selection of the address type is conveyed from the applications to the IP stack in a way to influence the source address selection algorithm [RFC6724].

The current source address selection algorithm operates on the available set of IP addresses when selecting an address. According to the proposed solution, if the requested type IP address is not available at the time of the request, the IP stack shall make an attempt to configure one such IP address. The selected IP address shall be compliant with the requested IP address type, whether it is selected among available addresses or dynamically configured. In the absence of a matching type (because it is not available and not

configurable on demand), the source address selection algorithm shall return an empty set.

A Socket API-based interface for enabling applications to influence the source address selection algorithm is described in [RFC5014]. That specification defines IPV6_ADDR_PREFERENCES option at the IPPROTO_IPV6 level. That option can be used with setsockopt() and getsockopt() calls to set and get address selection preferences.

Furthermore, that RFC also specifies two flags that relate to IP mobility management: IPV6_PREFER_SRC_HOME and IPV6_PREFER_SRC_COA. These flags are used for influencing the source address selection to prefer either a Home Address or a Care-of Address.

Unfortunately, these flags do not satisfy the aforementioned needs due to the following reasons, therefore new flags are proposed in this document:

- Current flags indicate a "preference" whereas there is a need for indicating "requirement". Source address selection algorithm does not have to produce an IP address compliant with the "preference" , but it has to produce an IP address compliant with the "requirement".

- Current flags influence the selection made among available IP addresses. The new flags force the IP stack to configure a compliant IP address if none is available at the time of the request.

- The Home vs. Care-of Address distinction is not sufficient to capture the three different types of IP addresses described in Section 2.1.

The following new flags are defined in this document and they shall be used with Socket API in compliance with the [RFC5014]:

IPV6_REQUIRE_FIXED_IP /* Require a Fixed IP address as source */

IPV6_REQUIRE_SESSION_LASTING_IP /* Require a Session-lasting IP address as source */

IPV6_REQUIRE_NON-PERSISTENT_IP /* Require a Non-persistent IP address as source */

Only one of these flags may be set on the same socket. If an application attempts to set more than one flag, the most recent setting will be the one in effect.

When any of these new flags is used, then the IPV6_PREFER_SRC_HOME and IPV6_PREFER_SRC_COA flags, if used, shall be ignored.

These new flags are used with setsockopt()/getsockopt(), getaddrinfo(), and inet6_is_srcaddr() functions [RFC5014]. Similar with the setsockopt()/getsockopt() calls, getaddrinfo() call shall also trigger configuration of the required type IP address, if one is not already available. When the new flags are used with getaddrinfo() and the triggered configuration fails, the getaddrinfo() call shall ignore that failure (i.e., not return an error code to indicate that failure). Only the setsockopt() shall return an error when configuration of the requested type IP address fails.

The following new error codes are also defined in the document and will be used in the Socket API in compliance with [RFC5014].

EAI_REQUIREDIPNOTSUPPORTED /* The network does not support the ability to request that specific IP address type */

EAI_REQUIREDIPFAILED /* The network could not assign that specific IP address type */

4. Backwards Compatibility Considerations

Backwards compatibility support is required by the following 3 types of entities:

- The Applications on the mobile host

- The IP stack in the mobile host

- The network infrastructure

4.1. Applications

Legacy applications that do not support the new flags will use the legacy API to the IP stack and will not enjoy On-Demand Mobility feature.

Applications using the new flags must be aware that they may be executed in environments that do not support On-Demand Mobility feature. Such environments may include legacy IP stack in the mobile host, legacy network infrastructure, or both. In either case, the API will return an error code and the invoking applications must respond with using legacy calls without On-Demand Mobility feature.

4.2. IP Stack in the Mobile Host

New IP stacks must continue to support all legacy operations. If an application does not use On-Demand Mobility feature, the IP stack must respond in a legacy manner.

If the network infrastructure supports On-Demand Mobility feature, the IP stack should follow the application request: If the application requests a specific address type, the stack should forward this request to the network. If the application does not request an address type, the IP stack must not request an address type and leave it to the network's default behavior to choose the type of the allocated IP address. If an IP address was already allocated to the host, the IP stack uses it and may not request a new one from the network.

4.3. Network Infrastructure

The network infrastructure may or may not support the On-Demand Mobility feature. How the IP stack on the host and the network infrastructure behave in case of a compatibility issue is outside the scope of this API specification.

5. Summary of New Definitions

The following list summarizes the new constants definitions discussed in this memo:

<netdb.h>	IPV6_REQUIRE_FIXED_IP
<netdb.h>	IPV6_REQUIRE_SESSION_LASTING_IP
<netdb.h>	IPV6_REQUIRE_NON_PERSISTENT_IP
<netdb.h>	EAI_REQUIREDIPNOTSUPPORTED
<netdb.h>	EAI_REQUIREDIPFAILED
<netinet/in.h>	IPV6_REQUIRE_FIXED_IP
<netinet/in.h>	IPV6_REQUIRE_SESSION_LASTING_IP
<netinet/in.h>	IPV6_REQUIRE_NON_PERSISTENT_IP
<netinet/in.h>	EAI_REQUIREDIPNOTSUPPORTED
<netinet/in.h>	EAI_REQUIREDIPFAILED

6. Security Considerations

The setting of certain IP address type on a given socket may be restricted to privileged applications. For example, a Fixed IP Address may be provided as a premium service and only certain

applications may be allowed to use them. Setting and enforcement of such privileges are outside the scope of this document.

7. IANA Considerations

This document has no IANA considerations.

8. Acknowledgements

We would like to thank Alexandru Petrescu, John Kaippallimalil, Jouni Korhonen, Seil Jeon, and Sri Gundavelli for their valuable comments and suggestions on this work.

9. References

9.1. 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>.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <http://www.rfc-editor.org/info/rfc5014>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

9.2. Informative References

[I-D.ietf-dmm-requirements] Chan, A., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", draft-ietf-dmm-requirements-17 (work in progress), June 2014.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[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>.

[RFC5563] Leung, K., Dommety, G., Yegani, P., and K. Chowdhury, "WiMAX Forum / 3GPP2 Proxy Mobile IPv4", RFC 5563, DOI 10.17487/RFC5563, February 2010, <http://www.rfc-editor.org/info/rfc5563>.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <http://www.rfc-editor.org/info/rfc5944>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

Authors' Addresses

Alper Yegin
Actility
Istanbul
Turkey

Email: alper.yegin@actility.com

Danny Moses
Intel Corporation
Petah Tikva
Israel

Email: danny.moses@intel.com

Kisuk Kweon
Samsung
Suwon
South Korea

Email: kisuk.kweon@samsung.com

Jinsung Lee
Samsung
Suwon
South Korea

Email: js81.lee@samsung.com

Jungshin Park
Samsung
Suwon
South Korea

Email: shin02.park@samsung.com

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 31, 2017

D. Moses
Intel
A. Yegin

September 27, 2016

DHCPv6 Extension for On Demand Mobility exposure
draft-moses-dmm-dhcp-ondemand-mobility-04

Abstract

Applications differ with respect to whether or not they need IP session continuity and/or IP address reachability. Networks providing the same type of service to any mobile host and any application running on the host yields inefficiencies. This document describes extensions to the DHCPv6 protocol to enable mobile hosts to indicate the required mobility service type associated with a requested IP address, and networks to indicate the type of mobility service associated with the allocated IP address in return.

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 March 31, 2017.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-dmm-ondemand-mobility] defines different types of mobility-associated services provided by access networks to mobile hosts with regards to maintaining IPv6 address continuity after an event of the host moving to different locations with different points of attachments within the IP network topology. It further specifies means for applications to convey to the IP stack in the mobile host, their requirements regarding these services.

This document defines extensions to the DHCPv6 protocol ([RFC3315]) in the form of a new DHCP option that specifies the type of mobility services associated with an IPv6 address. The IP stack in a mobile host uses the DHCP client to communicate the type of mobility service it wishes to receive from the network. The DHCP server in the network uses this option to convey the type of service that is guaranteed with the assigned IPv6 address in return.

This new option also extends the ability of mobile routers to specify desired mobility service in a request for IPv6 proxies (as specified in [RFC3633]), and delegating routers to convey the type of mobility service that is committed with the allocated IPv6 proxies in return.

In a distributed mobility management environment, there are multiple Mobility Anchors (as specified in [TBD reference to the Distributed Mobility Management architecture RFC]). In some use-cases, mobile hosts may wish to indicate to the network, preference of the serving Mobility Anchor. This document specifies a new DHCPv6 option that is used by DHCPv6 clients to convey this preference.

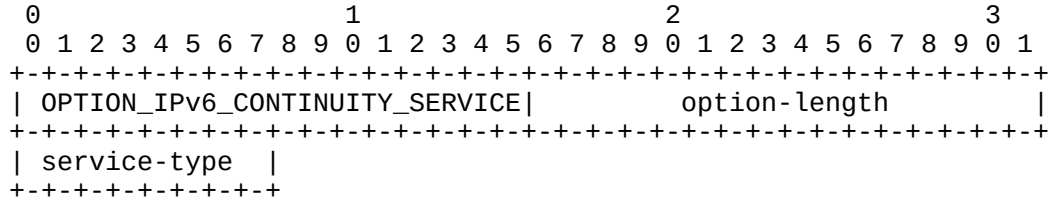
2. Notational Conventions

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. IPv6 Continuity Service Option

The IPv6 Continuity Service option is used to specify the type of continuity service associated with a source IPv6 address or IPv6 prefix. The IPv6 Continuity Service option must be encapsulated in the IAaddr-options field of the IA Address option when associated with an IPv6 address, and in the IAprefix-options field of the IA_PD prefix option when associated with an IPv6 prefix.

The format of the IPv6 Continuity Service option is:



option-code OPTION_IPv6_CONTINUITY_SERVICE (TBD)

option-len 1

service-type one of the following values:

Non-Persistent - a non-persistent IP address or prefix (1)

Session-Lasting - a session-lasting IP address or prefix (2)

Fixed - a fixed IP address or prefix (3)

Anytype - Anyone of the above (0)

The definition of these service types is available in [I-D.ietf-dmm-ondemand-mobility].

All other values (4-255) are reserved for future usage and should not be used. If the OPTION_IPv6_CONTINUITY_SERVICE option is received and its service-type is equal to one of the reserved values, the option should be ignored.

This option can appear in one of two contexts: (1) As part of a request to assign a source IPv6 address of the specified mobility service type, and (2) As part of a request to assign an IPv6 prefix of the specified mobility service type.

3.1. Source IPv6 Address Type Specification

In this context, the IPv6 Continuity Service option is encapsulated in the IAaddr-options field of the IA Address option.

When in a message sent from a client to a server, the value of the IPv6 Continuity Service option indicates the type of continuity service required for the IPv6 address requested by the client.

When in a message sent from a server to a client, the value of the IPv6 Continuity Service option indicates the type of IP continuity service committed by the network for the associated IPv6 address. The value 'AnyType' can only appear in the message sent from the client to the server to indicate that the client has no specific preference. However, it cannot appear in a message sent from the server.

Once an IPv6 address type was requested and provided, any subsequent messages involving this address (lease renewal - for example) must include the IPv6 Continuity Service option with the same service type that was assigned by the server during the initial allocation.

If a server received a request to assign an IPv6 address with a specified IPv6 Continuity service, but cannot fulfill the request, it must reply with the NoAddrsAvail status (refer to section 22.13 - Status Code Option in [RFC3315].

A server that does not support this option will discard it as well as the IA Address option that had this option encapsulated in one of its IAaddr-options field.

If a client does not receive the requested address, it must resend the request without the desired IPv6 Continuity Service option since it is not supported by the server. In that case, the host of the client cannot assume any IP continuity service behaviour for that address.

A server must not include the IPv6 Continuity Service option in the IAaddr-options field of an IA Address option, if not specifically requested previously by the client to which it is sending a message.

If a client receives an IA Address option from a server with the IPv6 Continuity Service option in the IAaddr-options field, without initially requesting a specific service using this option, it must discard the received IPv6 address.

If the mobile host has no preference regarding the type of continuity service it uses the 'AnyType' value as the specified type of continuity service. The Server will allocate an IPv6 address with some continuity service and must specify the type in IPv6 Continuity Service option encapsulated in the IAaddr-options field of the IA Address option. The method for selecting the type of continuity service is outside the scope of this specification.

3.2. IPv6 Prefix Type Specification

In this context, the IPv6 Continuity Service option is encapsulated in the IAprefix-options field of the IA_PD prefix option.

When in a message sent from a client to a server, the value of the IPv6 Continuity Service option indicates the type of continuity service required for the IPv6 prefix requested by the client.

When in a message sent from a server to a client, the value of the IPv6 Continuity Service option indicates the type of continuity service committed by the network for the associated IPv6 prefix. The value 'AnyType' can only appear in the message sent from the client to the server to indicate that the client has no specific preference. However, it cannot appear in a message sent from the server.

Once an IPv6 prefix type was requested and provided, any subsequent messages involving this prefix (lease renewal - for example) must include the IPv6 Continuity Service option with the same service type that was assigned by the server during the initial allocation.

If a server received a request to assign an IPv6 prefix with a specified IPv6 Continuity service, but cannot fulfill the request, it must reply with the NoAddrsAvail status.

A server that does not support this option will discard it as well as the IA_PD Prefix option that had this option encapsulated in one of its IAprefix-options field.

If a client does not receive the requested prefix, it must resend the request without the desired IPv6 Continuity Service option since it

is not supported by the server. In that case, the mobile router of the client cannot assume any IP continuity service behaviour for that prefix.

A server must not include the IPv6 Continuity Service option in the IAprefix-options field of an IA_PD Prefix option, if not specifically requested previously by the client to which it is sending a message.

If a client receives an IA_PD Prefix option from a server with the IPv6 Continuity Service option in the IAprefix-options field, without initially requesting a specific service using this option, it must discard the received IPv6 prefix.

If the mobile router has no preference regarding the type of continuity service it uses the 'AnyType' value as the specified type of continuity service. The Server will allocate an IPv6 prefix with some continuity service and must specify the type in IPv6 Continuity Service option encapsulated in the IAprefix-options field of the IA_PD Prefix option. The method for selecting the type of continuity service is outside the scope of this specification.

4. Anchor Preference Option

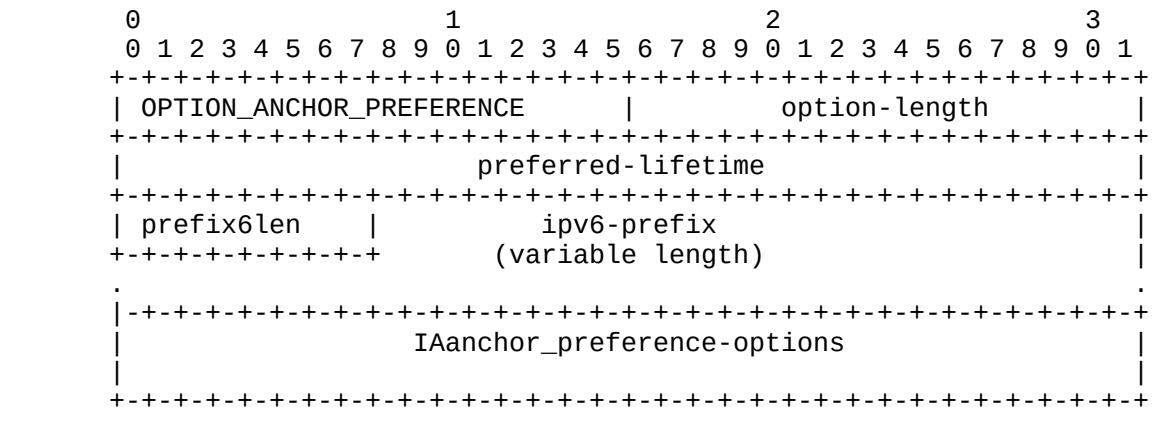
In a distributed mobility management environment that deploys multiple Mobility Anchors, each Mobility Anchor may have a set of IPv6 prefixes that is being used when assigning Session-lasting or Fixed source IPv6 addresses to hosts.

The selection of the Mobility Anchor that will serve a mobile host is performed by the network at various events like, the event of initial attachment of a mobile host to a network.

The Anchor Preference option enables a host to express its desire to receive a source IPv6 address with a specific IPv6 prefix. This is useful when the mobile host wishes to indicate to the network which Mobility Anchor should be used for anchoring its traffic and ensuring service continuity in the event of handoff between LANs with different IPv6 prefixes.

The network MAY respect this request but is not required to do so.

The format of the Anchor Preference option is:



option-code OPTION_ANCHOR_PREFERENCE (TBD)

option-len 5 + length of ipv6-prefix field + length of anchor_preference-options field

preferred-lifetime The preferred lifetime of the IPv6 address whose prefix is requested, expressed in units of seconds

prefix-length The length in bits of the ipv6-prefix. Typically allowed values are 0 to 128.

IPv6 prefix This is a variable length field that specifies the desired ipv6 prefix. The length is (prefix6len + 7) / 8. This field is padded with 0 bits up to the nearest octet boundary when prefix6len is not divisible by 8.

IAanchor_preference-option Options associated with this request

This option is used by the client in a request for a new IPv6 source address. The server replies with an IPv6 address that may or may not have the desired prefix.

An IPv6 prefix is requested only when the mobile host wishes to be anchored by a specific mobility anchor. The client must also indicate the type of mobility service it requires using the IPv6 Continuity Service option encapsulated in the IAanchor_preference-options field of the IA Address option.

When requesting an IPv6 prefix, only the ’Session-Lasting’ and ’Fixed’ types are legal.

The server must assign the IPv6 address of the requested type to the client, even if it does not fulfill the request for the specified prefix.

If a server received a request to use a specific IPv6 prefix and an IPv6 address type, but cannot assign an IPv6 address with that specified IPv6 Continuity it must reply with the NoAddrsAvail status.

A server that does not support this option will discard it.

If a client does not receive any address, it must assume that the the option is not supported by the server and use the IA Address option in subsequent requests.

5. Security Considerations

There are no specific security considerations for this option.

6. IANA Considerations

TBD

7. References

7.1. 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>.

7.2. Informative References

[I-D.ietf-dmm-ondemand-mobility] Yegin, A., Moses, D., Kweon, K., Lee, J., and J. Park, "On Demand Mobility Management", draft-ietf-dmm-ondemand-mobility-07 (work in progress), July 2016.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <http://www.rfc-editor.org/info/rfc3633>.

Authors' Addresses

Danny Moses
Intel
Petah Tikva
Israel

Email: danny.moses@intel.com

Alper Yegin
Istanbul
Turkey

Email: alper.yegin@yegin.org

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

S. Jeon
Sungkyunkwan University
S. Figueiredo
Altran Research
Y. Kim
Soongsil University
J. Kaippallimalil
Huawei

October 31, 2016

Use Cases and API Extension for Source IP Address Selection
draft-sijeon-dmm-use-cases-api-source-05.txt

Abstract

This draft specifies and analyzes the expected cases regarding the selection of a proper source IP address and address type by an application in a distributed mobility management (DMM) network. It also proposes a new Socket API to address further selection issues with three source IP address types defined in the on-demand mobility API draft.

Table of Contents

1. Introduction

Applications to select source IP address type in a mobile node (MN) need to consider IP session continuity and/or IP address reachability. [I-D.ietf-dmm-ondemand-mobility], defines three types of source IP addresses based on mobility management capabilities: fixed IP address, session-lasting IP address, and non-persistent IP address. Based on the address type requested by the application, the MN configures a proper source IP address. However, the source IP address type itself in a socket request may not be enough to convey all the requirements of an application. For example, more than one IP address of the same type requested may be available. It may be that as a result of mobility the MN can potentially obtain new IP

prefixes from different serving networks belonging to different subnets. This draft categorizes and analyzes use cases that an MN is likely to encounter. In addition, this draft proposes an extension that allows the application to express its preferences when more than one source address of a type is present.

2. Use Cases and Analysis

This section outlines use cases where an application on the MN tries to obtain a source IP address.

2.1. Application has no specific IP address type requirement or address preference

Applications such as text-based web browsing or information service, e.g. weather and stock information, as well as legacy applications belong to this category. Many applications use short-lived Internet connections with no requirements for session continuity or IP address reachability. Assigning a non-persistent IP address can be thus considered as default for MNs. However, it is subject to address assignment policy of a network operator. The suggested flag, IPV6_REQUIRE_NON-PERSISTENT_IP, defined in [I-D.ietf-dmm-ondemand-mobility] can used for expressing its preference to the IP stack.

2.2. Application has specific IP address type requirement and address preference

This category is for an application requiring IP session continuity with different granularity of IP address reachability. This case may be further divided in three sub-cases with regard to IP address type availability and/or address selection.

2.2.1. Case 1: there is no configured IP address based on a requested type in the IP stack, but there is a further selection preference by the application

Once an IP address is requested by an application regardless of any source IP address type defined in [I-D.ietf-dmm-ondemand-mobility], the network stack will configure an IP address after obtaining an IP prefix based on the requested source IP address type from the current serving gateway.

2.2.2. Case 2: there are one or more IP addresses configured with a requested type in the IP stack, and no selection preference by the application

This is the same as Case 1 described above, except the existence of more than one configured IP addresses belonging to the requested IP address type in the IP stack, e.g. due to different address assignment policy by an operator.

When a non-persistent IP address is requested, if an application requests a non-persistent IP address to the IP stack, the IP address is obtained from the serving IP gateway as the previous one is not maintained across gateway changes.

When a session-lasting IP address is requested, an expected sequence can be described as follows;

1. The MN has one or more session-lasting IP addresses configured in the IP stack.

2. If an application requests a session-lasting IP address to the IP stack, it will try to use an existing session-lasting IP address as it is already configured in the IP stack. If there are multiple available session-lasting IP addresses, the default address selection rules will be applied [RFC6724], e.g. with scope preference, longest prefix matching, and/or so on. The best-matched IP address among them will be selected and assigned to the application.

3. Subsequently, the MN moves to another serving network, and the previous (mobile) sessions are still in use. A new application requests a session-lasting IP address with flag, IPV6_REQUIRE_SESSION_LASTING_IP to the IP stack. The selection of the session-lasting IP address follows the same procedure as described in Step 2.

When a fixed IP address is requested, it will follow the same procedure with session-lasting IP address request as described.

2.2.3. Case 3: there are one or more IP addresses with a requested type configured in the IP stack, but there is a further selection preference by the application

Assume that there are one or multiple applications with session-lasting IP address running. A newly initiated application might get one of the session-lasting IP addresses being used, not initiating a protocol procedure, i.e. DHCP or SLAAC for a new session-lasting IP address to the network. On the contrary, the IP stack might try to get a new session-lasting IP address from the current serving gateway

by default. Acquiring a new session-lasting IP address may take some time (due to the exchange with the network) while using an existing one is instantaneous. On the other hand, using the existing one might yield less optimal routing. For example, the use of the IP address with an existing one configured might provide a suboptimal routing path as a result of a handover. This situation might not be preferred by newly initiated applications because the application incurs the costs of IP mobility even though the MN has not moved from the current serving network. Eventually, the new session is served by a remote IP mobility anchor with mobility management functions, though the MN has not moved yet.

If the application is allowed to further define its preference for an optimally routed, this situation can be avoided. See Section 3 for the proposed flag.

2.3. Gaps in the consistency with the default address selection

The need of an indication mechanism can be sought in the consistency with the former IETF standards. For example, in [RFC6724] where default behavior for IPv6 is specified, without a proper indication mechanism, following conflicts are expected to happen. In Rule 6 in [RFC6724], it is said that the matching label between source address of an IPv6 host and destination address is preferred among the combinations between other source addresses and destination address, where the label is a numeric value representing policies that prefer a particular source address prefix for use with a destination address prefix in [RFC6724]. In Rule 8 in [RFC6724], it is said that the longest matching prefix between source address of an IPv6 host and destination address is preferred among the combinations between other source addresses and destination address. Following Rules 6 and 8 may result in the selection of a source IP address with which packets that are sub-optimally routed.

3. Indications for expressing address preference requirement

When an application prefers a new IP address of the requested IP address type, additional indication flags should be delivered through the socket API interface.

To obtain an address that supports dynamic mobility using session-lasting IP address, a new address preference flag needs to be defined. The flag should be simple and useful while aligned with the three types of IP addresses. The objective of the hereby presented address preference flag is letting the IP stack check whether it has an available IP address assigned from the current serving network when the flag is received by an initiated application. If not, it

will trigger the IP stack to get a new IP address from the current serving network. We call it "ON_NET" property.

If the application requests an IP address with ON_NET flag set in the socket request, the IP address returned by the stack should conform to the address preference requirement. This should be the case even though other session-lasting IP addresses, not belonging to the current serving network are available. If there are multiple session-lasting IP addresses matched with ON_NET property, the default source address selection rules will be applied.

IPV6_XX_SRC_ON_NET

/* Require (or Prefer) an IP address based on a requested IP address type as source, assigned from the current serving network, whatever it has been assigned or should be assigned */

This flag aims to express the preference to check an IP address, being used by an application, previously assigned from the current serving network and to use it or to get an IP address from the current serving network, as well as enabling differentiated per-flow anchoring where an obtained session-lasting IP address might be used for all initiated session-lasting IP applications. The use of the flag can be combined together with the three types of IP address defined in [I-D.ietf-dmm-ondemand-mobility].

In [I-D.mccann-dmm-prefixcost], it proposes that the Router Advertisement signaling messages communicate the cost of maintaining a given prefix at the MN’s current point of attachment. The objective is to make a dynamic and optimal decision of address assignment and release, i.e. when to release old addresses and assign new ones. The proposed ON_NET property may present a way to deliver a prefix decision for an application, specifically from a routing distance point of view, to the IP stack.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

T.B.D.

6. Acknowledgements

We would like to thank Danny Moses, Marco Liebsch, Brian Haberman, Sri Gundavelli, Alexandru Petrescu for their valueable comments and suggestions on this work.

7. References

7.1. Normative References

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

7.2. Informative References

[I-D.ietf-dmm-ondemand-mobility]
Yegin, A., Moses, D., Kweon, K., Lee, J., and J. Park, "On Demand Mobility Management", draft-ietf-dmm-ondemand-mobility-07 (work in progress), July 2016.

[I-D.mccann-dmm-prefixcost]
McCann, P. and J. Kaippallimalil, "Communicating Prefix Cost to Mobile Nodes", draft-mccann-dmm-prefixcost-03 (work in progress), April 2016.

Authors' Addresses

Seil Jeon
Sungkyunkwan University
2066 Seobu-ro, Jangan-gu
Suwon, Gyeonggi-do
Korea

Email: seiljeon@skku.edu

Sergio Figueiredo
Altran Research
2, Rue Paul Dautier
Velizy-Villacoublay 78140
France

Email: sergio.figueiredo@altran.com

Younghan Kim
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul 156-743
Korea

Email: younghak@ssu.ac.kr

John Kaippallimalil
Huawei
5340 Legacy Dr., Suite 175
Plano, TX 75024
U.S

Email: john.kaippallimalil@huawei.com