ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: May 21, 2021

W. Roome
S. Randriamasy
Nokia Bell Labs
Y. Yang
Yale University
J. Zhang
Tongji University
K. Gao
Sichuan University
November 17, 2020

Unified properties for the ALTO protocol draft-ietf-alto-unified-props-new-14

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285] by generalizing the concept of "endpoint properties" to generic types of entities, and by presenting those properties as maps, similar to the network and cost maps in [RFC7285].

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] when, and only when, they appear in all capitals, as shown here.

When the words appear in lower case, they are to be interpreted with their natural language meanings.

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 May 21, 2021.

Copyright Notice

Table of Contents

1. Introduction

The ALTO protocol [RFC7285] introduces the concept of "properties" attached to "endpoint addresses", and defines the Endpoint Property Service (EPS) to allow ALTO clients to retrieve those properties. While useful, the EPS, as defined in [RFC7285], has at least three limitations.

First, the EPS allows properties to be associated with only endpoints that are identified by individual communication addresses like IPv4 and IPv6 addresses. It is reasonable to think that collections of endpoints, as defined by CIDRs [RFC4632] or PIDs, may also have properties. Furthermore, recent ALTO use cases show that properties of entities such as network flows [RFC7011] and routing elements [RFC7921] are also useful. Such cases are documented in [draft-gao-alto-fcs]. The current EPS however is restricted to individual endpoints and cannot be applied to those entities.

Second, the EPS only allows endpoints identified by global communication addresses. However, an endpoint address may be a local IP address or an anycast IP address which is also not globally unique. Additionally, an entity such as a PID may have an identifier that is not globally unique. That is, a same PID identifier may be used in multiple network maps, while in each network map, this PID identifier points to a different set of addresses. For example, PID "mypid10" may be defined in "netmap1" and "netmap2" while in each network map, "mypid10" covers a different set of addresses.

Third, the EPS is only defined as a POST-mode service. Clients must request the properties for an explicit set of endpoint addresses. By contrast, [RFC7285] defines a GET-mode cost map resource which returns all available costs, so a client can get a full set of costs once, and then process cost lookups without querying the ALTO server. [RFC7285] does not define a similar service for endpoint properties. At first, a map of endpoint properties might seem impractical, because it could require enumerating the property value for every possible endpoint. However, in practice, it is highly unlikely that properties will be defined for every endpoint address. It is much more likely that properties may be defined for only a subset of endpoint addresses, and the specification of properties uses an aggregation representation to allow enumeration. This is particularly true if blocks of endpoint addresses with a common prefix (e.g., a CIDR) have the same value for a property. Entities in other domains may very well allow aggregated representation and hence be enumerable as well.

To address the three limitations, this document specifies a protocol extension for defining and retrieving ALTO properties:

- The first limitation is addressed by introducing a generic concept called ALTO Entity, which generalizes an endpoint and may represent a PID, a network element, a cell in a cellular network, an abstracted network element as defined in [I-D.ietf-alto-path-vector], or other physical or logical objects involved in a network topology. Each entity is included in a collection called an ALTO Entity Domain. Since each ALTO Entity Domain includes only one type of entities, each Entity Domain can be classified by the type of entities in it.

- The second limitation is addressed by using resource-specific entity domains. A resource-specific entity domain contains entities that are defined and identified with respect to a given ALTO information resource, which provides scoping. For example, an entity domain containing PIDs is identified with respect to the network map in which these PIDs are defined. Likewise, an entity domain containing local IP addresses may be defined with respect to a local network map.

- The third limitation is addressed by defining two new types of ALTO information resources: Property Map, detailed in Section 7 and Filtered Property Map, detailed in Section 8. The former is a GET-mode resource that returns the property values for all entities in one or more entity domains, and is analogous to a network map or a cost map in [RFC7285]. The latter is a POST-mode resource that returns the values for sets of properties and

entities requested by the client, and is analogous to a filtered network map or a filtered cost map.

The protocol extension defined in this document is augmentable. New entity domain types can be defined without revising the specification defined in this document. Similarly, new cost metrics and new endpoint properties can be defined in other documents without revising the protocol specification defined in [RFC7285].

1.1. Terminology

- Transaction: A request/response exchange between an ALTO Client and an ALTO Server.
- Client: When used with a capital "C", this term refers to an ALTO Client.
- Server: When used with a capital "S", this term refers to an ALTO Server.
- FPM: An abbreviation for filtered property map.
- EPS: An abbreviation for Endpoint Property Service.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here. When the words appear in lower case, they are to be interpreted with their natural language meanings.

3. Basic Features of the Unified Property Extension

This section gives a high-level overview of the basic features involved in ALTO Entity Property Maps. It assumes the reader is familiar with the ALTO protocol [RFC7285]. The purpose of this extension is to allow conveying properties on objects that extend ALTO Endpoints and are called ALTO Entities, or entities for short.

The features introduced in this section can be used as standalone. However, in some cases, these features may depend on particular information resources and need to be defined with respect to them. To this end, Section 4 introduces additional features that extend the ones presented in the present section.

3.1. Entity

The concept of an ALTO Entity generalizes the concept of an ALTO Endpoint defined in Section 2.1 of [RFC7285]. An entity is an object that can be an endpoint that is defined by its network address, but can also be an object that has a defined mapping to a set of one or more network addresses or an object that is not even related to any network address. Thus, whereas all endpoints are entities, not all entities are endpoints.

Examples of entities are:

- an ALTO endpoint, defined in [RFC7285], that represents an application or a host identified by a communication address (e.g., an IPv4 or IPv6 address) in a network,
- a PID, defined in [RFC7285], that has a provider defined human-readable identifier specified by an ALTO network map, which maps a PID to a set of ipv4 and ipv6 addresses,
- an autonomous system (AS), that has an AS number (ASN) as its identifier and maps to a set of ipv4 and ipv6 addresses,
- a country with a code as specified in [ISO3166-1], to which applications such as CDN providers associate properties and capabilities,
- a TCP/IP network flow, that is identified by a TCP/IP 5-Tuple specifying its source and destination addresses and port numbers and the utilized protocol,
- a routing element, that is specified in [RFC7921] and is associated with routing capabilities information,
- an abstract network element, that represents an abstraction of a network part such as a routable network node, one or more links, a network domain or their aggregation.

3.2. Entity Domain

An entity domain defines a set of entities of the same semantic type. An entity domain is characterized by its type and identified by its name.

In this document, an entity must be owned by exactly one entity domain name. An entity identifier must point to exactly one entity. If two entities in two different entity domains refer to the same physical or logical object, they are treated as different entities.

For example, if an object has both an IPv4 and an IPv6 address, these two addresses will be treated as two entities, defined respectively in the "ipv4" and "ipv6" entity domains.

3.2.1. Entity Domain Type

The type of an entity domain type defines the semantics of a type of entity. Entity domain types can be defined in different documents. For example: the present document defines entity domain types "ipv4", "ipv6" and "pid" in sections Section 6.1 and Section 6.2. The entity domain type "ane", that defines Abstract Network Elements (ANEs), is introduced in [I-D.ietf-alto-path-vector]. The entity domain type that defines country codes is introduced in [draft-ietf-alto-cdni-request-routing-alto]. An entity domain type is expected to be registered at the IANA, as specified in section Section 12.2.2 and similarly to an ALTO address type.

3.2.2. Entity Domain Name

The name of an entity domain is defined in the scope of an ALTO server. An entity domain name can sometimes be identical to the name of its relevant entity domain type. This is the case when the entities of a domain have an identifier that points to the same object throughout all the information resources of the Server that provide entity properties for this domain. For example, a domain of type "ipv4" containing entities identified by a public IPv4 address can be named "ipv4" because its entities are uniquely identified by all the Server resources.

In some cases, a domain type and domain name must be different. Indeed, for some domain types, entities are defined relatively to a given information resource. As a consequence, entities in such domains may be defined in a resource handling this domain type but not in other resources handling this same domain type. Moreover, across different ALTO information resources handling a domain type, an entity identifier may point to different objects. This is the case for entities of domain type "pid". A PID is defined relatively to a network map. For example: an entity "mypid10" of domain type "pid" may be defined in a given network map resource and be undefined in other network maps, or may even map to a different set of endpoint addresses. In this case, naming an entity domain only by its type "pid" does not guarantee that its entities are owned by exactly one entity domain name. Section 4.2 and Section 5.1.2 of this document describe how a domain is uniquely identified by a name that associates the domain type and the related information resource.

3.3. Entity Property Type

An entity property defines a property of an entity. This is similar to the endpoint property defined in Section 7.1 of [RFC7285]. An entity property can convey either network-aware or network-agnostic information. Similarly to an entity domain, an entity property is characterized by its type and identified by its name. An entity property type is expected to be registered at the IANA, as specified in section Section 12.3.

Below are some examples with real and fictitious entity domain and property names:

- an entity in the "ipv4" domain may have a property whose value is an Autonomous System (AS) number indicating the AS that owns this IPv4 address and another property named "countrycode" indicating a country code mapping to this address,

- an entity identified by its country code in the "countrycode" domain, defined in [draft-ietf-alto-cdni-request-routing-alto] may have a property indicating what delivery protocol is used by a CDN,

- an entity in the "netmap1.pid" domain may have a property that indicates the central geographical location of the endpoints it includes.

It should be noted that some identifiers may be used for both an entity domain type and a property type. For example:

- the identifier "countrycode" may point to both the entity domain type "countrycode" and the property type "countrycode".

- the identifier "pid" may point to both the entity domain type "pid" and the property type "pid".

Likewise, a same identifier may point to both a domain name and a property name.

3.4. New information resource and media type: ALTO Property Map

This document introduces a new ALTO information resource named Property Map. An ALTO property map provides a set of properties on one or more sets of entities. A property may apply to different entity domain types and names. For example, an ALTO property map may define the "ASN" property for both "ipv4" and "ipv6" entity domains.

The present extension also introduces a new media type.

This document uses the same definition of an information resource as Section 9.1 of [RFC7285]. ALTO uses media types to uniquely indicate the data format used to encode the content to be transmitted between an ALTO server and an ALTO client in the HTTP entity body. In the present case, an ALTO property map resource is defined by the media type "application/alto-propmap+json".

A Property Map can be queried as a GET-mode resource, thus conveying values of all property values on all entities indicated in its capabilities. It can also be queried as a POST-mode resource, thus conveying a selection of properties on a selection of entities.

4. Advanced Features of the Unified Property Extension

4.1. Entity Identifier and Entity Domain Name

In [RFC7285], an endpoint has an identifier that is explicitly associated with the "ipv4" or "ipv6" address domain. Examples are "ipv4:192.0.2.14" and "ipv6:2001:db8::12".

In this document, an entity must be owned by exactly one entity domain name and an entity identifier must point to exactly one entity. To ensure this, an entity identifier is explicitly attached to the name of its entity domain and an entity domain type characterizes the semantics and identifier format of its entities.

The encoding format of an entity identifier is further specified in Section 5.1.3 of this document.

For instance:

- if an entity is an endpoint with example routable IPv4 address "192.0.2.14", its identifier is associated with domain name "ipv4" and is "ipv4:192.0.2.14",

- if an entity is a PID named "mypid10" in network map resource "netmap2", its identifier is associated with domain name "netmap2.pid" and is "netmap2.pid:mypid10".

4.2. Resource-Specific Entity Domain Name

Some entities are defined and identified uniquely and globally. This is the case for instance when entities are endpoints that are identified by a routable IPv4 or IPv6 address. The entity domain for such entities can be globally defined and named "ipv4" or "ipv6". Those entity domains are called resource-agnostic entity domains in this document, as they are not associated with any specific ALTO information resources.

Some other entities and entity types are only defined relatively to a given information resource. This is the case for entities of domain type "pid", that can only be understood with respect to the network map where they are defined. For example, a PID named "mypid10" may be defined to represent a set S1 of IP addresses in a network map resource named "netmap1". Another network map "netmap2" may use the same name "mypid10" and define it to represent another set S2 of IP addresses. The identifier "pid:mypid10" may thus point to different objects because the information on the originating information resource is lost.

To solve this ambiguity, the present extension introduces the concept of resources-specific entity domain. This concept applies to domain types where entities are defined relatively to a given information resource. It can also apply to entity domains that are defined locally, such as local networks of objects identified with a local IPv4 address.

In such cases, an entity domain type is explicitly associated with an identifier of the information resource where these entities are defined. Such an information resource is referred to as the "specific information resource". Using a resource-aware entity domain name, an ALTO property map can unambiguously identify distinct entity domains of the same type, on which entity properties may be queried. Examples of resource-specific entity domain names may look like: "netmap1.pid" or "netmap2.pid". Thus, a name association such as "netmap1.pid:mypid10" and "netmap2.pid:mypid10" allows to distinguish the two abovementioned PIDs that are both named "mypid10" but in two different resources, "netmap1" and "netmap2".

An information resource is defined in the scope of an ALTO Server and so is an entity domain name. The format of a resource-specific entity domain name is further specified in Section 5.1.2.

4.3. Resource-Specific Entity Property Value

Like entity domains, some types of properties are defined relatively to an information resource. That is, an entity may have a property of a given type, whose values are associated to different information resources.

For example, suppose entity "192.0.2.34" defined in the "ipv4" domain has a property of type "pid", whose value is the PID to which address "192.0.2.34" is attached in a network map. The mapping of network addresses to PIDs is specific to a network map and probably different from one network map resource to another one. So that if a property "pid" is defined for entity "192.0.2.34" in two different network

maps "netmap1" and "netmap2", the value for this property will likely be a different value in "netmap1" and "netmap2".

To support information resource dependent property values, this document uses the same approach as in Section 10.8.1 of [RFC7285] entitled "Resource-Specific Endpoint Properties". When a property value depends on a given information resource, the name of this property must be explicitly associated with the information resource that defines it.

For example, the property "pid" queried on entity "ipv4:192.0.2.34" and defined in both "netmap1" and "netmap2", can be named "netmap1.pid" and "netmap2.pid". This allows a Client to get a property of the same type but defined in different information resources with a single query. Specifications on the property name format are provided in Section 5.2.

4.4. Entity Hierarchy and Property Inheritance

For some domain types, entities can be grouped in a set and be defined by the identifier of this set. This is the case for domain types "ipv4" and "ipv6", where individual Internet addresses can be grouped in blocks. When a same property value applies to a whole set, a Server can define a property for the identifier of this set instead of enumerating all the entities and their properties. This allows a substantial reduction of transmission payload both for the Server and the Client. For example, all the entities included in the set defined by the address block "ipv6:2001:db8::1/64" share the same properties and values defined for this block.

Additionally, entity sets sometimes are related by inclusion, hierarchy or other relations. This allows defining inheritance rules for entity properties that propagate properties among related entity sets. The Server and the Client can use these inheritance rules for further payload savings. Entity hierarchy and property inheritance rules are specified in the documents that define the applicable domain types. The present document defines these rules for the "ipv4" and "ipv6" domain types.

This document introduces, for applicable domain types, "Entity Property Inheritance rules", with the following concepts: Entity Hierarchy, Property Inheritance and Property Value Unicity. A detailed specification of entity hierarchy and property inheritance rules is provided in Section 5.1.4.

4.4.1. Entity Hierarchy

An entity domain may allow using a single identifier to identify a set of individual entities. For example, a CIDR block can be used to identify a set of IPv4 or IPv6 entities. A CIDR block is called a hierarchical entity identifier, as it can reflect inclusion relations among entity sets. For example, the CIDR "ipv4:192.0.1.0/24" includes all the individual ipv4 entities identified by the CIDR "ipv4:192.0.1.0/26".

4.4.2. Property Inheritance

A property may be defined for a hierarchical entity identifier, while it may be undefined for individual entities covered by this identifier. In this case, these individual entities inherit the property value defined for the identifier that covers them. For example, suppose a property map defines the ASN property only for the hierarchical entity identifier "ipv4:192.0.1.0/24" but not for individual entities in this block. Suppose also that inheritance rules are specified for CIDR blocks in the "ipv4" domain type. When receiving this property map, a Client can infer that entity "ipv4:192.0.1.1" inherits the "ASN" property value of block "ipv4:192.0.1.0/24" because the address "ipv4:192.0.1.1" is included by the CIDR block "ipv4:192.0.1.0/24".

Property value inheritance rules also apply among entity sets. A property map may define values for an entity set belonging to a hierarchy but not for "sub" sets that are covered by this set identifier. In this case, inheritance rules must specify how entities in "sub" sets inherit property values from their "super" set. For instance, if the "ASN" property is defined only for the entity set identified by address block "ipv4:192.0.1.0/24", the entity set identified by "ipv4:192.0.1.0/30" and thus included in the former set, may inherit the "ASN" property values from set "ipv4:192.0.1.0/24".

4.4.3. Property Value Unicity

The inheritance rules must ensure that an entity belonging to a hierarchical set of entities inherits no more than one property value, for the sake of consistency. Indeed, a property map may define a property on a hierarchy of entity sets that inherit property values from one or more including sets in the upper levels. On the other hand, a property value, defined at a lower level of the hierarchy may be different from the value defined at an upper level. In such a case, a set in the lower level of the hierarchy may potentially end up with different values. This may be the case for address blocs with increasing prefix length, on which a property

value gets increasingly accurate and thus may differ. For example, a fictitious property such as "geo-location" or "average transfer volume" may be defined at a progressively finer grain for entity sets defined with progressively longer CIDR prefixes. It seems more interesting to have property values of progressively higher accuracy. A unicity rule, applied to the entity domain type must specify an arbitration rule among the different property values for an entity. An example illustrating the need for such rules is provided in Section 6.1.3.

4.5. Supported Properties on Entity Domains in Property Map Capabilities

A property type is not necessarily applicable to any domain type, or an ALTO Server may just not provide a property on all applicable domains. For instance, a property type reflecting link bandwidth is likely not defined on entities of a domain of type "country-code". Therefore an ALTO server providing Property Maps needs to specify the properties that can be queried on the different entity domains it supports.

This document explains how the Information Resources Directory (IRD) capabilities of a Property Map resource unambiguously expose what properties a Client can query on a given entity domain.

- a field named "mappings" lists the names of the entity domains supported by the Property Map,

- for each listed entity domain, a list of the names of the applicable properties is provided.

An example is provided in Section 10.3. The "mappings" field associates entity domains and properties that can be resource-agnostic or resource-specific. This allows a Client to formulate compact and unambiguous entity property queries, possibly relating to one or more information resources. In particular:

- it avoids a Client to query a property on entity domains on which it is not defined,

- it allows a Client to query, for an entity E, values for a property P that are defined in different information resources,

- it allows a Client to query a property P on entities that are defined in different information resources.

Further specifications are provided in Section 7.4.

4.6. Defining Information Resource

A Client willing to query properties on entities belonging to a domain needs to know how to retrieve these entities. To this end, the Client can look up the "mappings" field exposed in IRD capabilities of a property map, see Section 4.5. This field, in its keys, exposes all the entity domains supported by the property map. The syntax of the entity domain identifier specified in Section 5.1.2 allows the client to infer whether the entity domain is resource-specific or not. The Client can extract, if applicable, the identifier of the specific resource, query the resource and retrieve the entities. For example:

- an entity domain named "netmap1.ipv4" includes the IPv4 addresses that appear in the "ipv4" field of the endpoint address group of each PID in the network map "netmap1", and that cannot be recognized outside "netmap1" because, for instance, these are local non-routable addresses,

- an entity domain named "netmap1.pid" includes the PIDs listed in network map "netmap1".

- an entity domain named "ipv4" is resource-agnostic and covers all the routable IPv4 addresses.

Besides, it is also necessary to inform a Client about which associations of specific resources and entity domain types are allowed, because it is not possible to prevent a Server from exposing inappropriate associations. An informed Client will just ignore inappropriate associations exposed by a Server and avoid error-prone transactions with the Server.

For example, the association "costmap3.pid" is not allowed for the following reason: although a cost map exposes PID identifiers, it does not define them, that is, the set of addresses included in this PID. Neither does a cost map list all the PIDs on which properties can be queried, because a cost map only exposes PID pairs on which a queried cost type is defined. Therefore, the resource "costmap3" does not enable a Client to extract information on the existing PID entities or on the addresses they contain.

Instead, the cost map uses a network map, where all the PIDs used in a cost map are defined together with the addresses contained by the PIDs. This network map is qualified in this document as the Defining Information Resource for the entity domain of type "pid" and this concept is explained in Section 4.6.1.

4.6.1. Defining Information Resource and Media Type

For the reasons explained in the previous section, this document introduces the concept of defining information resource and media type.

A defining information resource for an entity domain D is the information resource where entities of D are defined. That is, all the information on the entities of D can be retrieved in this resource. This concept applies to resource-specific domains. This is useful for entity domain types that are by essence domain-specific, such as "pid" and "ane" domain types. It is also useful for resource-specific entity domains constructed from resource-agnostic domain types, such as network map specific domains of local IPv4 addresses.

The defining information resource of an entity domain D has the following specificities:

- it has an entry in the IRD,
- it defines the entities of D,
- it does not use another information resource that defines these entities,
- it defines and exposes entity identifiers that are all persistent.
- its media type is unique and equal to the one that is specified for the defining information resource of an entity domain type.

A fundamental attribute of a defining information resource is its media type. There is a unique association between an entity domain type and the media type of its defining information resource. When an entity domain type allows associations with defining information resources, the document that defines this entity domain type specifies the media type of the potential defining information resource. Likewise, the IANA registration of an entity domain type also specifies the media type of potential defining information resources.

When the Client wants to use a resource-specific entity domain, it needs to be cognizant of the media-type of its defining information resource. If the Server exposes resources a resource specific entity domain with a non-compliant media type for the domain type, the Client can avoid transaction errors by ignoring them.

4.6.2. Examples of specific resources media-types

Here are some examples of specific information resource types associated to entity domain types and their media type.

- For entity domain type "pid": the media type of the specific resource is "application/alto-networkmap+json", because PIDs are defined in network map resources.

- For entity domain types "ipv4" and "ipv6": the media type of the specific resource is "application/alto-networkmap+json", because IPv4 and IPv6 addresses covered by the Server are defined in network map resources.

- For entities of domain type "ane": [I-D.ietf-alto-path-vector] defines entities named "ANE", where ANE stands for Abstracted Network Element, and the entity domain type "ane". When an ANE has a persistent identifier, say, "entity-4", the latter is provided by the Server as a value of the "persistent-entity-id" property of the ANE. Further properties may be queried on an ANE with a persistent entity ID. These properties are available from a persistent property map, that defines properties on a specific "ane" domain. Together with the persistent identifier, the Server also provides the property map resource identifier where the "ane" domain containing "entity-4" is defined. The definition of the "ane" entity domain containing "entity-4" is thus specific to the property map. Therefore, for entities of domain type "ane" that have a persitent identifier, the media type of the specific information resource is "application/alto-propmap+json".

4.7. Defining Information Resource for Resource-Specific Property Values

As explained in Section 4.3, a property type may take values that are resource specific. This is the case for property type "pid", whose values are by essence defined relatively to a specific network map. The PID value returned for an IPv4 address is specific to the network map defining the PID and may differ from one network map to another one. Property values may be specific to different types of information resources. For example: the value for property "pid" is specific to a network map. The value for property type "cdnifci-capab" is specific to the information resource "cdnifci-map", defined in [draft-ietf-alto-cdni-request-routing-alto], while network maps do not define property "fci-capability" for ipv4 addresses and a cdnifci-map does not define "pid" values for IPv4 addresses.

Thus, similarly to resource specific entity domains, the Client needs to be cognizant of appropriate associations of information resource and property types.

4.7.1. Examples of defining resources media-types for properties

Here are some examples of specific information resources types associated to entity property types and their media type.

- For property type "pid": the media type of the specific resource is "application/alto-networkmap+json", because PIDs are defined in network map resources.

- For property type "cdni-fci-capability": the media type of the specific resource is "application/alto-cdnifci+json"

5. Protocol Specification: Basic Data Type

5.1. Entity Domain

5.1.1. Entity Domain Type

An entity domain has a type, which is uniquely identified by a string that MUST be no more than 64 characters, and MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), hyphen ('-', U+002D), and low line ('_', U+005F). The '.' separator MUST NOT be used unless specifically indicated in a further extension document.

For example, the strings "ipv4", "ipv6", and "pid" are valid entity domain types. "ipv4.anycast" and "pid.local" are invalid.

The type EntityDomainType is used in this document to denote a JSON string meeting the preceding requirements.

An entity domain type defines the semantics of a type of entity, independently of any specifying resource. Each entity domain type MUST be registered with the IANA. The format of the entity identifiers (see Section 5.1.3) in that type of entity domain, as well as any hierarchical or inheritance rules (see Section 5.1.4) for those entities, MUST be specified at the same time.

5.1.2. Entity Domain Name

As said in Section 3.2 when introducing entity domains, an entity domain is characterized by its type and identified by its name.

This document distinguishes three categories of entity domains: resource-specific entity domains, resource-agnostic entity domains and self-defined entity domains. Their entity domain names are constructed as specified in the following sub-sections.

Each entity domain is identified by a unique entity domain name which is a string of the following format:

```
EntityDomainName ::= [ [ ResourceID ] '.' ] EntityDomainType
```

Where the presence and construction of component:

```
"[ [ ResourceID ] '.' ]"
```

depends on the category of entity domain.

Note that the '.' separator is not allowed in EntityDomainType and hence there is no ambiguity on whether an entity domain name refers to a resource-agnostic entity domain or a resource-specific entity domain.

Note also that the resource ID format defined in Section 10.1 of [RFC7285] specifies that: "the '.' separator is reserved for future use and MUST NOT be used unless specifically indicated in this document, or an extension document". The present extension keeps the format specification of [RFC7285], hence the '.' separator MUST NOT be used in an information resource ID.

5.1.2.1. Resource-specific Entity Domain

A resource-specific entity domain is identified by an entity domain name constructed as follows. It MUST start with a resource ID using the ResourceID type defined in Section 10.2 of [RFC7285], followed by the '.' separator (U+002E), followed by a string of the type EntityDomainType specified in Section 5.1.1.

For example, if an ALTO server provides two network maps "netmap-1" and "netmap-2", these network maps can define two resource-specific domains of type "pid", respectively identified by "netmap-1.pid" and "netmap-2.pid".

5.1.2.2. Resource-agnostic Entity Domain

A resource-agnostic entity domain contains entities that are identified independently of any information resource. Hence, the identifier of a resource-agnostic entity domain is simply the identifier of its entity domain type. For example, "ipv4" and "ipv6"

identify the two resource-agnostic Internet address entity domains defined in Section 6.1.

5.1.2.3. Self-defined Entity Domain

A property map can define properties on entities that are specific to a unique information resource, which is the property map itself. This may be the case when an ALTO Server provides properties on a set of entities that are defined only in this property map, are not relevant to another one and do not depend on another specific resource.

For example: a specialised property map may define a domain of type "ane", defined in [I-D.ietf-alto-path-vector], that contains a set of ANEs representing data centers, that each have a persistent identifier and are relevant only to this property map.

In this case, the entity domain is qualified as "self-defined". The identifier of a self-defined entity domain can be of the format:

```
EntityDomainName ::= .EntityDomainType
```

where '.' indicates that the entity domain only exists within the property map resource using it.

A self-defined entity domain can be viewed as a particular case of resource-specific entity domain, where the specific resource is the current resource that uses this entity domain. In that case, for the sake of simplification, the component "ResourceID" SHOULD be omitted in its entity domain name.

5.1.3. Entity Identifier

Entities in an entity domain are identified by entity identifiers (EntityID) of the following format:

```
EntityID ::= EntityDomainName ':' DomainTypeSpecificEntityID
```

Examples from the Internet address entity domains include individual IP addresses such as "net1.ipv4:192.0.2.14" and "net1.ipv6:2001:db8::12", as well as address blocks such as "net1.ipv4:192.0.2.0/26" and "net1.ipv6:2001:db8::1/48".

The format of the second part of an entity identifier depends on the entity domain type, and MUST be specified when defining a new entity domain type and registering it with the IANA. Identifiers MAY be hierarchical, and properties MAY be inherited based on that

hierarchy. The rules defining any hierarchy or inheritance MUST be defined when the entity domain type is registered.

The type EntityID is used in this document to denote a JSON string representing an entity identifier in this format.

Note that two entity identifiers with different valid textual representations may refer to the same entity, for a given entity domain. For example, the strings "net1.ipv6:2001:db8::1" and "net1.ipv6:2001:db8:0:0:0:0:0:1" refer to the same entity in the "ipv6" entity domain.

5.1.4. Hierarchy and Inheritance

To simplify the representation, some types of entity domains allow the ALTO Client and Server to use a hierarchical entity identifier format to represent a block of individual entities. For instance, in an IPv4 domain "net1.ipv4", a CIDR "net1.ipv4:192.0.2.0/26" covers 64 individual IPv4 entities. In this case, the corresponding property inheritance rule MUST be defined for the entity domain type. The hierarchy and inheritance rule MUST have no ambiguity.

5.2. Entity Property

Each entity property has a type to indicate the encoding and the semantics of the value of this entity property, and has a name to identify it.

5.2.1. Entity Property Type

The type EntityPropertyType is used in this document to indicate a string denoting an entity property type. The string MUST be no more than 32 characters, and it MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), the hyphen ('-', U+002D), the colon (':', U+003A), or the low line ('_', U+005F). Note that the '.' separator is not allowed because it is reserved to separate an entity property type and an information resource identifier when an entity property is resource-specific.

Each entity property type MUST be registered with the IANA. The intended semantics of the entity property type MUST be specified at the same time.

Identifiers prefixed with "priv:" are reserved for Private Use [RFC5226] without a need to register with IANA. All other identifiers for entity property types appearing in an HTTP request or response with an "application/alto-*" media type MUST be registered

in the "ALTO Entity Property Type Registry", defined in Section 12.3. For an entity property identifier with the "priv:" prefix, an additional string (e.g., company identifier or random string) MUST follow the prefix to reduce potential collisions, that is, the string "priv:" alone is not a valid endpoint property identifier.

To distinguish from the endpoint property type, the entity property type has the following characteristics:

- Some entity property types are applicable to entities in particular entity domain types only. For example, the property type "pid" is applicable to entities in the entity domain types "ipv4" or "ipv6" while is not applicable to entities in an entity domain of type "pid".

- The intended semantics of the value of an entity property may also depend on the entity domain type. For example, suppose that a property named "geo-location" is defined as the coordinates of a point, encoded as: "latitude longitude [altitude]." When applied to an entity that represents a specific host computer, identified by an address in an entity domain of type "ipv4" or "ipv6", the "geo-location" property would define the host's location. However, when applied to an entity in a "pid" domain type, the property would indicate the location of the center of all hosts in this "pid" entity.

5.2.2. Entity Property Name

Each entity property is identified by an entity property name, which is a string of the following format:

EntityPropertyName ::= [ResourceID] '.' EntityPropertyType

Similar to the endpoint property type defined in Section 10.8 of [RFC7285], each entity property may be defined by either the property map itself (self-defined) or some other specific information resource (resource-specific).

The entity property name of a resource-specific entity property starts with a string of the type ResourceID defined in [RFC7285], followed by the '.' separator (U+002E) and a EntityDomainType typed string. For example, the "pid" properties of an "ipv4" entity defined by two different maps "net-map-1" and "net-map-2" are identified by "net-map-1.pid" and "net-map-2.pid" respectively.

The specific information resource of an entity property may be the current information resource itself, that is, the property map defining the property. In that case, the ResourceID in the property

name SHOULD be ignored. For example, the property name ".asn" applied to an entity identitifed by its IPv4 address, indicates the AS number of the AS that "owns" the entity, where the returned AS number is defined by the property map itself.

5.2.3. Format for Entity Property Value

[RFC7285] in Section 11.4.1.6, specifies that an implementation of the Endpoint Property Service specified in [RFC7285] SHOULD assume that the property value is a JSONString and fail to parse if it is not. This document extends the format of a property value by allowing it to be a JSONValue instead of just a JSONString.

6. Entity Domain Types Defined in this Document

This document requires the definition of each entity domain type MUST include (1) the entity domain type name and (2) domain-specific entity identifiers, and MAY include (3) hierarchy and inheritance semantics optionally. This document defines three initial entity domain types as follows.

6.1. Internet Address Domain Types

The document defines two entity domain types (IPv4 and IPv6) for Internet addresses. Both types are resource-agnostic entity domain types and hence define corresponding resource-agnostic entity domains as well. Since the two domains use the same hierarchy and inheritance semantics, we define the semantics together, instead of repeating for each.

6.1.1. IPv4 Domain

6.1.1.1. Entity Domain Type

ipv4

6.1.1.2. Domain-Specific Entity Identifiers

Individual addresses are strings as specified by the IPv4Addresses rule of Section 3.2.2 of [RFC3986]; Hierarchical addresses are prefix-match strings as specified in Section 3.1 of [RFC4632]. To define properties, an individual Internet address and the corresponding full-length prefix are considered aliases for the same entity. An individual Internet address and the corresponding full-length prefix are considered aliases for the same entity on which to define properties. Thus, "ipv4:192.0.2.0" and "ipv4:192.0.2.0/32" are equivalent.

6.1.2. IPv6 Domain

6.1.2.1. Entity Domain Type

ipv6

6.1.2.2. Domain-Specific Entity Identifiers

Individual addresses are strings as specified by Section 4 of [RFC5952]; Hierarchical addresses are prefix-match strings as specified in Section 7 of [RFC5952]. To define properties, an individual Internet address and the corresponding 128-bit prefix are considered aliases for the same entity. That is, "ipv6:2001:db8::1" and "ipv6:2001:db8::1/128" are equivalent, and have the same set of properties.

6.1.3. Hierarchy and Inheritance of Internet Address Domains

Both Internet address domains allow property values to be inherited. Specifically, if a property P is not defined for a specific Internet address I, but P is defined for a hierarchical Internet address C which prefix-matches I, then the address I inherits the value of P defined for the hierarchical address C. If more than one such hierarchical addresses define a value for P, I inherits the value of P in the hierarchical address with the longest prefix. Note that this longest prefix rule ensures no multiple value inheritances, and hence no ambiguity.

Hierarchical addresses can also inherit properties: if a property P is not defined for the hierarchical address C, but is defined for another hierarchical address C' which covers all IP addresses in C, and C' has a shorter prefix length than C, then C MAY inherit the property from C'. If there are multiple such hierarchical addresses like C', C MUST inherit from the hierarchical address having the longest prefix length.

As an example, suppose that a server defines a property P for the following entities:

```
ipv4:192.0.2.0/26: P=v1
ipv4:192.0.2.0/28: P=v2
ipv4:192.0.2.0/30: P=v3
ipv4:192.0.2.0:    P=v4
```

Figure 1: Defined Property Values.

Then the following entities have the indicated values:

```
ipv4:192.0.2.0:    P=v4
ipv4:192.0.2.1:    P=v3
ipv4:192.0.2.16:   P=v1
ipv4:192.0.2.32:   P=v1
ipv4:192.0.2.64:   (not defined)
ipv4:192.0.2.0/32: P=v4
ipv4:192.0.2.0/31: P=v3
ipv4:192.0.2.0/29: P=v2
ipv4:192.0.2.0/27: P=v1
ipv4:192.0.2.0/25: (not defined)
```

Figure 2: Inherited Property Values.

An ALTO server MAY explicitly indicate a property as not having a value for a particular entity. That is, a server MAY say that property P of entity X is "defined to have no value", instead of "undefined". To indicate "no value", a server MAY perform different behaviours:

- If that entity would inherit a value for that property, then the ALTO server MUST return a "null" value for that property. In this case, the ALTO client MUST recognize a "null" value as "no value" and "do not apply the inheritance rules for this property."

- If the entity would not inherit a value, then the ALTO server MAY return "null" or just omit the property. In this case, the ALTO client cannot infer the value for this property of this entity from the Inheritance rules. So the client MUST interpret that this property has no value.

If the ALTO server does not define any properties for an entity, then the server MAY omit that entity from the response.

6.1.4. Defining Information Resource Media Type for domain types IPv4 and IPv6

Entity domain types "ipv4" and "ipv6" both allow to define resource specific entity domains. When resource specific domains are defined with entities of domain type "ipv4" or "ipv6", the defining information resource for an entity domain of type "ipv4" or "ipv6" MUST be a Network Map. The media type of a defining information resource is therefore:

application/alto-networkmap+json

6.2. PID Domain

The PID domain associates property values with the PIDs in a network map. Accordingly, this entity domain always depends on a network map.

6.2.1. Entity Domain Type

pid

6.2.2. Domain-Specific Entity Identifiers

The entity identifiers are the PID names of the associated network map.

6.2.3. Hierarchy and Inheritance

There is no hierarchy or inheritance for properties associated with PIDs.

6.2.4. Defining Information Resource Media Type for Domain Type PID

The entity domain type "pid" allows to define resource specific entity domains. When resource specific domains are defined with entities of domain type "pid", the defining information resource for entity domain type "pid" MUST be a Network Map. The media type of a defining information resource is therefore:

application/alto-networkmap+json

6.2.5. Relationship To Internet Addresses Domains

The PID domain and the Internet address domains are completely independent; the properties associated with a PID have no relation to the properties associated with the prefixes or endpoint addresses in that PID. An ALTO server MAY choose to assign all the properties of a PID to the prefixes in that PID or only some of these properties.

For example, suppose "PID1" consists of the prefix "ipv4:192.0.2.0/24", and has the property "P" with value "v1". The Internet address entities "ipv4:192.0.2.0" and "ipv4:192.0.2.0/24" in the IPv4 domain MAY have a value for the property "P", and if they do, it is not necessarily "v1".

6.3. Internet Address Properties vs. PID Properties

Because the Internet address and PID domains relate to completely distinct domain types, the question may arise as to which entity domain type is the best for a property. In general, the Internet address domain types are RECOMMENDED for properties that are closely related to the Internet address, or are associated with, and inherited through, hierarchical addresses.

The PID domain type is RECOMMENDED for properties that arise from the definition of the PID, rather than from the Internet address prefixes in that PID.

For example, because Internet addresses are allocated to service providers by blocks of prefixes, an "ISP" property would be best associated with Internet address domain types. On the other hand, a property that explains why a PID was formed, or how it relates to a provider's network, would best be associated with the PID domain type.

7. Property Map

A property map returns the properties defined for all entities in one or more domains, e.g., the "location" property of entities in "pid" domain, and the "ASN" property of entities in "ipv4" and "ipv6" domains.

Section 10.4 gives an example of a property map request and its response.

7.1. Media Type

The media type of a property map is "application/alto-propmap+json".

7.2. HTTP Method

The property map is requested using the HTTP GET method.

7.3. Accept Input Parameters

None.

7.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities:

```
    object {
      EntityPropertyMapping mappings;
    } PropertyMapCapabilities;

    object-map {
      EntityDomainName -> EntityPropertyName<1..*>;
    } EntityPropertyMapping
```

with fields:

mappings: A JSON object whose keys are names of entity domains and values are the supported entity properties of the corresponding entity domains.

7.5. Uses

The "uses" field of a property map resource in an IRD entry specifies dependent resources of this property map. It is an array of the resource ID(s) of the resource(s).

7.6. Response

If the entity domains in this property map depend on other resources, the "dependent-vtags" field in the "meta" field of the response MUST be an array that includes the version tags of those resources, and the order MUST be consistent with the "uses" field of this property map resource. The data component of a property map response is named "property-map", which is a JSON object of type PropertyMapData, where:

```
    object {
      PropertyMapData property-map;
    } InfoResourceProperties : ResponseEntityBase;

    object-map {
      EntityID -> EntityProps;
    } PropertyMapData;

    object {
      EntityPropertyName -> JSONValue;
    } EntityProps;
```

The ResponseEntityBase type is defined in Section 8.4 of [RFC7285].

Specifically, a PropertyMapData object has one member for each entity in the property map. The entity's properties are encoded in the corresponding EntityProps object. EntityProps encodes one name/value pair for each property, where the property names are encoded as

strings of type PropertyName. A protocol implementation SHOULD assume that the property value is either a JSONString or a JSON "null" value, and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

For each entity in the property map:

- If the entity is in a resource-specific entity domain, the ALTO server SHOULD only return self-defined properties and resource-specific properties which depend on the same resource as the entity does. The ALTO client SHOULD ignore the resource-specific property in this entity if their mapping is not registered in the ALTO Resource Entity Property Transfer Registry of the type of the corresponding resource.

- If the entity is in a shared entity domain, the ALTO server SHOULD return self-defined properties and all resource-specific properties defined for all resource-specific entities which have the same domain-specific entity identifier as this entity does.

For efficiency, the ALTO server SHOULD omit property values that are inherited rather than explicitly defined; if a client needs inherited values, the client SHOULD use the entity domain's inheritance rules to deduce those values.

8. Filtered Property Map

A filtered property map returns the values of a set of properties for a set of entities selected by the client.

Section 10.5, Section 10.6, Section 10.7 and Section 10.8 give examples of filtered property map requests and responses.

8.1. Media Type

The media type of a property map resource is "application/alto-propmap+json".

8.2. HTTP Method

The filtered property map is requested using the HTTP POST method.

8.3. Accept Input Parameters

The input parameters for a filtered property map request are supplied in the entity body of the POST request. The input parameters for a filtered property map request are supplied in the POST request's

entity body. This document specifies the input parameters with a data format indicated by the media type "application/alto-propmapparams+json", which is a JSON object of type ReqFilteredPropertyMap:

```
object {
  EntityID             entities<1..*>;
  EntityPropertyName   properties<1..*>;
} ReqFilteredPropertyMap;
```

with fields:

entities: List of entity identifiers for which the specified properties are to be returned. The ALTO server MUST interpret entries appearing multiple times as if they appeared only once. The domain of each entity MUST be included in the list of entity domains in this resource's "capabilities" field (see Section 8.4).

properties: List of properties to be returned for each entity. Each specified property MUST be included in the list of properties in this resource's "capabilities" field (see Section 8.4). The ALTO server MUST interpret entries appearing multiple times as if they appeared only once.

Note that the "entities" and "properties" fields MUST have at least one entry each.

8.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities, as defined in Section 7.4.

8.5. Uses

Same to the "uses" field of the Property Map resource (see Section 7.5).

8.6. Filtered Property Map Response

The response MUST indicate an error, using ALTO protocol error handling, as defined in Section 8.5 of [RFC7285], if the request is invalid.

Specifically, a filtered property map request can be invalid in the following cases:

- An entity identifier in the "entities" field of the request is invalid if:

* The domain of this entity is not defined in the "entity-domains" capability of this resource in the IRD,

* The entity identifier is not valid for the entity domain.

A valid entity identifier does never generate an error, even if the filtered property map resource does not define any properties for it.

If an entity identifier in the "entities" field of the request is invalid, the ALTO server MUST return an "E_INVALID_FIELD_VALUE" error defined in Section 8.5.2 of [RFC7285], and the "value" field of the error message SHOULD indicate the provided invalid entity identifier.

o A property name in the "properties" field of the request is invalid if this property name is not defined in the "properties" capability of this resource in the IRD.

When a filtered property map resource does not define a value for a property requested on a particular entity, it is not an error. In this case, the ALTO server MUST omit that property from the response for that endpoint.

If a property name in "properties" in the request is invalid, the ALTO server MUST return an "E_INVALID_FIELD_VALUE" error defined in Section 8.5.2 of [RFC7285]. The "value" field of the error message SHOULD indicate the property name.

The response to a valid request is the same as for the Property Map (see Section 7.6), except that:

o If the requested entities include entities in the shared entity domain, the "dependent-vtags" field in its "meta" field MUST include version tags of all dependent resources appearing in the "uses" field.

o If the requested entities only include entities in resource-specific entity domains, the "dependent-vtags" field in its "meta" field MUST include the version tags of the resources on which the requested resource-specific entity domains and the requested resource-specific properties are dependent on.

o The response only includes the entities and properties requested by the client. If an entity in the request is identified by a hierarchical identifier (e.g., a "ipv4" or "ipv6" prefix), the response MUST cover properties for all identifiers in this hierarchical identifier.

The filtered property map response MUST include all the inherited property values for the requested entities and all the entities which are able to inherit property values from the requested entities. To achieve this goal, the ALTO server MAY follow three rules:

- If a property for a requested entity is inherited from another entity not included in the request, the response SHOULD include this property for the requested entity. For example, A full property map may skip a property P for an entity A (e.g., ipv4:192.0.2.0/31) if P can be derived using inheritance from another entity B (e.g., ipv4:192.0.2.0/30). A filtered property map request may include only A but not B. In such a case, the property P SHOULD be included in the response for A.

- If there are entities covered by a requested entity but having different values for the requested properties, the response SHOULD include all those entities and the different property values for them. For example, considering a request for property P of entity A (e.g., ipv4:192.0.2.0/31), if P has value v1 for A1=ipv4:192.0.2.0/32 and v2 for A2=ipv4:192.0.2.1/32, then, the response SHOULD include A1 and A2.

- If an entity address in the response is already covered by other entities addresses in the same response, it SHOULD be removed from the response, for the sake of compactness. In the previous example, the entity A = ipv4:192.0.2.0/31 SHOULD be removed because A1 and A2 cover all the addresses in A.

An ALTO client should be aware that the entities in the response MAY be different from the entities in its request.

8.7. Entity property type defined in this document

This document defines the entity property type "pid"

The intended semantics are the same as in [RFC7285]

The defining information resource for property type MUST be a network map.

The media type of a defining information resource is therefore:

application/alto-networkmap+json

This document requests a IANA registration for this property

9. Impact on Legacy ALTO Servers and ALTO Clients

9.1. Impact on Endpoint Property Service

Since the property map and the filtered property map defined in this document provide the functionality of the Endpoint Property Service (EPS) defined in Section 11.4 of [RFC7285], it is RECOMMENDED that the EPS be deprecated in favor of Property Map and Filtered Property Map. However, ALTO servers MAY provide an EPS for the benefit of legacy clients.

9.2. Impact on Resource-Specific Properties

Section 10.8 of [RFC7285] defines two categories of endpoint properties: "resource-specific" and "global". Resource-specific property names are prefixed with the ID of the resource they depend on, while global property names have no such prefix. The property map and the filtered property map defined in this document define similar categories of entity properties. The difference is that entity property maps do not define "global" entity properties. Instead, they define "self-defined" entity properties as a special case of "resource-specific" entity properties, where the specific resource is the property map itself. This means that "self-defined" properties are defined within the scope of the property map.

9.3. Impact on Other Properties

In general, there should be little or no impact on other previously defined properties. The only consideration is that properties can now be defined on hierarchical entity identifiers, rather than just individual entity identifiers, which might change the semantics of a property.

10. Examples

10.1. Network Map

The examples in this section use a very simple default network map:

```
defaultpid:  ipv4:0.0.0.0/0  ipv6::/0
pid1:        ipv4:192.0.2.0/25
pid2:        ipv4:192.0.2.0/27
pid3:        ipv4:192.0.3.0/28
pid4:        ipv4:192.0.3.16/28
```

Figure 3: Example Default Network Map

And another simple alternative network map:

```
defaultpid:  ipv4:0.0.0.0/0  ipv6:::0/0
pid1:        ipv4:192.0.2.0/27
pid2:        ipv4:192.0.3.0/27
```

Figure 4: Example Alternative Network Map

10.2. Property Definitions

Beyond "pid", the examples in this section use four additional properties for Internet address domains, "ISP", "ASN", "country" and "state", with the following values:

	ISP	ASN	country	state
ipv4:192.0.2.0/23:	BitsRus	-	us	-
ipv4:192.0.2.0/28:	-	12345	-	NJ
ipv4:192.0.2.16/28:	-	12345	-	CT
ipv4:192.0.2.1:	-	-	-	PA
ipv4:192.0.3.0/28:	-	12346	-	TX
ipv4:192.0.3.16/28:	-	12346	-	MN

Figure 5: Example Property Values for Internet Address Domains

And the examples in this section use the property "region" for the PID domain of the default network map with the following values:

	region
pid:defaultpid:	-
pid:pid1:	us-west
pid:pid2:	us-east
pid:pid3:	us-south
pid:pid4:	us-north

Figure 6: Example Property Values for Default Network Map's PID Domain

Note that "-" means the value of the property for the entity is "undefined". So the entity would inherit a value for this property by the inheritance rule if possible. For example, the value of the "ISP" property for "ipv4:192.0.2.1" is "BitsRus" because of "ipv4:192.0.2.0/24". But the "region" property for "pid:defaultpid" has no value because no entity from which it can inherit.

Similar to the PID domain of the default network map, the examples in this section use the property "ASN" for the PID domain of the alternative network map with the following values:

	ASN
pid:defaultpid:	-
pid:pid1:	12345
pid:pid2:	12346

Figure 7: Example Property Values for Alternative Network Map's PID Domain

10.3. Information Resource Directory (IRD)

The following IRD defines ALTO Server information resources that are relevant to the Entity Property Service. It provides two property maps: one for the "ISP" and "ASN" properties, and another one for the "country" and "state" properties. The server could have provided a single property map for all four properties, but does not, presumably because the organization that runs the ALTO server believes that a client is not necessarily interested in getting all four properties.

The server provides several filtered property maps. The first returns all four properties, and the second returns only the "pid" property for the default network map.

The filtered property maps for the "ISP", "ASN", "country" and "state" properties do not depend on the default network map (it does not have a "uses" capability), because the definitions of those properties do not depend on the default network map. The Filtered Property Map providing the "pid" property does have a "uses" capability for the default network map, because the default network map defines the values of the "pid" property.

Note that for legacy clients, the ALTO server provides an Endpoint Property Service for the "pid" property defined on the endpoints of the default network map.

The server provides another filtered Property map resource, named "ane-dc-property-map", that returns a fictitious properties named "storage-capacity", "ram" and "cpu" for ANEs that have a persistent identifier. The entity domein to which the ANEs belong is "self-defined" and valid only within the property map.

```
    "meta" : {
      ...
      "default-alto-network-map" : "default-network-map"
    },
    "resources" : {
      "default-network-map" : {
        "uri" : "http://alto.example.com/networkmap/default",
        "media-type" : "application/alto-networkmap+json"
```

```
        },
        "alt-network-map" : {
          "uri" : "http://alto.example.com/networkmap/alt",
          "media-type" : "application/alto-networkmap+json"
        },
        .... property map resources ....
        "ia-property-map" : {
          "uri" : "http://alto.example.com/propmap/full/inet-ia",
          "media-type" : "application/alto-propmap+json",
          "uses": [ "default-network-map", "alt-network-map" ],
          "capabilities" : {
            "mappings": {
              "ipv4": [ ".ISP", ".ASN" ],
              "ipv6": [ ".ISP", ".ASN" ]
            }
          }
        },
        "iacs-property-map" : {
          "uri" : "http://alto.example.com/propmap/lookup/inet-iacs",
          "media-type" : "application/alto-propmap+json",
          "accepts": "application/alto-propmapparams+json",
          "uses": [ "default-network-map", "alt-network-map" ],
          "capabilities" : {
            "mappings": {
              "ipv4": [ ".ISP", ".ASN", ".country", ".state" ],
              "ipv6": [ ".ISP", ".ASN", ".country", ".state" ]
            }
          }
        },
        "region-property-map": {
          "uri": "http://alto.example.com/propmap/lookup/region",
          "media-type": "application/alto-propmap+json",
          "accepts": "application/alto-propmapparams+json",
          "uses" : [ "default-network-map", "alt-network-map" ],
          "capabilities": {
            "mappings": {
              "default-network-map.pid": [ ".region" ],
              "alt-network-map.pid": [ ".ASN" ],
            }
          }
        },
        "ip-pid-property-map" : {
          "uri" : "http://alto.example.com/propmap/lookup/pid",
          "media-type" : "application/alto-propmap+json",
          "accepts" : "application/alto-propmapparams+json",
          "uses" : [ "default-network-map", "alt-network-map" ],
          "capabilities" : {
            "mappings": {
```

```
            "ipv4": [ "default-network-map.pid",
                      "alt-network-map.pid" ],
            "ipv6": [ "default-network-map.pid",
                      "alt-network-map.pid" ]
          }
        }
      },
      "legacy-endpoint-property" : {
        "uri" : "http://alto.example.com/legacy/eps-pid",
        "media-type" : "application/alto-endpointprop+json",
        "accepts" : "application/alto-endpointpropparams+json",
        "capabilities" : {
          "properties" : [ "default-network-map.pid",
                           "alt-network-map.pid" ]
        }
      },
      "ane-dc-property-map": {
        "uri" : "http://alto.example.com/propmap/lookup/ane-dc",
        "media-type" : "application/alto-propmap+json",
        "accepts": "application/alto-propmapparams+json",
        "capabilities": {
          "mappings": {
            ".ane" : [ "storage-capacity", "ram", "cpu" ]
          }
        },
      }
    }
```

Figure 8: Example IRD

10.4. Full Property Map Example

The following example uses the properties and IRD defined above in Section 10.3 to retrieve a Property Map for entities with the "ISP" and "ASN" properties.

Note that, to be compact, the response does not include the entity "ipv4:192.0.2.0", because values of all those properties for this entity are inherited from other entities.

Also note that the entities "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28" are merged into "ipv4:192.0.2.0/27", because they have the same value of the "ASN" property. The same rule applies to the entities "ipv4:192.0.3.0/28" and "ipv4:192.0.3.0/28". Both of "ipv4:192.0.2.0/27" and "ipv4:192.0.3.0/27" omit the value for the "ISP" property, because it is inherited from "ipv4:192.0.2.0/23".

```
GET /propmap/full/inet-ia HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {"resource-id": "default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"},
      {"resource-id": "alt-network-map",
       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"}
    ]
  },
  "property-map": {
    "ipv4:192.0.2.0/23":   {".ISP": "BitsRus"},
    "ipv4:192.0.2.0/27":   {".ASN": "12345"},
    "ipv4:192.0.3.0/27":   {".ASN": "12346"}
  }
}
```

10.5. Filtered Property Map Example #1

The following example uses the filtered property map resource to request the "ISP", "ASN" and "state" properties for several IPv4 addresses.

Note that the value of "state" for "ipv4:192.0.2.0" is the only explicitly defined property; the other values are all derived by the inheritance rules for Internet address entities.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ipv4:192.0.2.0",
                 "ipv4:192.0.2.1",
                 "ipv4:192.0.2.17" ],
  "properties" : [ ".ISP", ".ASN", ".state" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {"resource-id": "default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"},
      {"resource-id": "alt-network-map",
       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"}
    ]
  },
  "property-map": {
    "ipv4:192.0.2.0":
          {".ISP": "BitsRus", ".ASN": "12345", ".state": "PA"},
    "ipv4:192.0.2.1":
          {".ISP": "BitsRus", ".ASN": "12345", ".state": "NJ"},
    "ipv4:192.0.2.17":
          {".ISP": "BitsRus", ".ASN": "12345", ".state": "CT"}
  }
}
```

10.6. Filtered Property Map Example #2

The following example uses the filtered property map resource to request the "ASN", "country" and "state" properties for several IPv4 prefixes.

Note that the property values for both entities "ipv4:192.0.2.0/26" and "ipv4:192.0.3.0/26" are not explicitly defined. They are inherited from the entity "ipv4:192.0.2.0/23".

Also note that some entities like "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28" in the response are not explicitly listed in the request. The response includes them because they are refinements of the requested entities and have different values for the requested properties.

The entity "ipv4:192.0.4.0/26" is not included in the response, because there are neither entities which it is inherited from, nor entities inherited from it.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ipv4:192.0.2.0/26",
                 "ipv4:192.0.3.0/26",
                 "ipv4:192.0.4.0/26" ],
  "properties" : [ ".ASN", ".country", ".state" ]
}

HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {"resource-id": "default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"},
      {"resource-id": "alt-network-map",
       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"}
    ]
  },
  "property-map": {
    "ipv4:192.0.2.0/26":  {".country": "us"},
    "ipv4:192.0.2.0/28":  {".ASN": "12345",
                           ".state": "NJ"},
    "ipv4:192.0.2.16/28": {".ASN": "12345",
                           ".state": "CT"},
    "ipv4:192.0.2.0":     {".state": "PA"},
    "ipv4:192.0.3.0/26":  {".country": "us"},
    "ipv4:192.0.3.0/28":  {".ASN": "12345",
                           ".state": "TX"},
    "ipv4:192.0.3.16/28": {".ASN": "12345",
                           ".state": "MN"}
  }
}
```

10.7. Filtered Property Map Example #3

The following example uses the filtered property map resource to request the "default-network-map.pid" property and the "alt-network-map.pid" property for a set of IPv4 addresses and prefixes.

Note that the entity "ipv4:192.0.3.0/27" is decomposed into two entities "ipv4:192.0.3.0/28" and "ipv4:192.0.3.16/28", as they have different "default-network-map.pid" property values.

```
POST /propmap/lookup/pid HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [
                "ipv4:192.0.2.128",
                "ipv4:192.0.2.0/27",
                "ipv4:192.0.3.0/27" ],
  "properties" : [ "default-network-map.pid",
                   "alt-network-map.pid ]
}

HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {"resource-id": "default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"},
      {"resource-id": "alt-network-map",
       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"}
    ]
  },
  "property-map": {
    "ipv4:192.0.2.128":   {"default-network-map.pid": "defaultpid",
                           "alt-network-map.pid": "defaultpid"},
    "ipv4:192.0.2.0/27":  {"default-network-map.pid": "pid2",
                           "alt-network-map.pid": "pid1"},
    "ipv4:192.0.3.0/28":  {"default-network-map.pid": "pid3",
                           "alt-network-map.pid": "pid2"},
    "ipv4:192.0.3.16/28": {"default-network-map.pid": "pid4",
                           "alt-network-map.pid": "pid2"}
  }
}
```

10.8. Filtered Property Map Example #4

Here is an example of using the filtered property map to query the regions for several PIDs in "default-network-map". The "region" property is specified as a "self-defined" property, i.e., the values of this property are defined by this property map resource.

```
POST /propmap/lookup/region HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : ["default-network-map.pid:pid1",
                "default-network-map.pid:pid2"],
  "properties" : [ ".region" ]
}

HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json

{
  "meta" : {
    "dependent-vtags" : [
       {"resource-id": "default-network-map",
        "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"}
    ]
  },
  "property-map": {
    "default-network-map.pid:pid1": {
      ".region": "us-west"
    },
    "default-network-map.pid:pid2": {
      ".region": "us-east"
    }
  }
}
```

10.9. Filtered Property Map for ANEs Example #5

The following example uses the filtered property map resource "ane-dc-property-map" to request properties "storage-capacity" and "cpu" on several ANEs defined in this property map.

```
POST /propmap/lookup/ane-dc HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [".ane:dc21", ".ane:dc45.srv9", ".ane:dc6.srv-cluster8"]
  "properties" : [ "storage-capacity", "cpu"]
}


  HTTP/1.1 200 OK
  Content-Length: ###
  Content-Type: application/alto-propmap+json

  {
    "meta" : {
    },
    "property-map": {
      ".ane:dc21":
        {"storage-capacity" : 40000 Gbytes, "cpu" : 500 Cores},
      ".ane:dc45.srv9":
        {"storage-capacity" : 100 Gbytes, "cpu" : 20 Cores},
      ".ane:dc6.srv-cluster8":
           {"storage-capacity" : 6000 Gbytes, "cpu" : 100 Cores},
    }
  }
```

11. Security Considerations

Both Property Map and Filtered Property Map defined in this document fit into the architecture of the ALTO base protocol, and hence the Security Considerations (Section 15 of [RFC7285]) of the base protocol fully apply: authenticity and integrity of ALTO information (i.e., authenticity and integrity of Property Maps), potential undesirable guidance from authenticated ALTO information (e.g., potentially imprecise or even wrong value of a property such as geo-location), confidentiality of ALTO information (e.g., exposure of a potentially sensitive entity property such as geo-location), privacy for ALTO users, and availability of ALTO services should all be considered.

A particular fundamental security consideration when an ALTO server provides a Property Map is to define precisely the policies on who can access what properties for which entities. Security mechanisms such as authentication and confidentiality mechanisms should then be applied to enforce the policy. For example, a policy can be that a

property P can be accessed only by its owner (e.g., the customer who is allocated a given IP address). Then, the ALTO server will need to deploy corresponding mechanisms to realize the policy. The policy may allow non-owners to access a coarse-grained value of the property P. In such a case, the ALTO server may provide a different URI to provide the information.

12. IANA Considerations

This document defines additional application/alto-* media types. It defines an ALTO Entity Domain Type Registry that extends the ALTO Address Type Registry defined in [RFC7285]. It also defines an ALTO Entity Property Type Registry that extends the ALTO endpoint property registry defined in [RFC7285].

12.1. application/alto-* Media Types

This document registers two additional ALTO media types, listed in Table 1.

Type	Subtype	Specification
application	alto-propmap+json	Section 7.1
application	alto-propmapparams+json	Section 8.3

Table 1: Additional ALTO Media Types.

Type name: application

Subtype name: This document registers multiple subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC7159].

Security considerations: Security considerations related to the generation and consumption of ALTO Protocol messages are discussed in Section 15 of [RFC7285].

Interoperability considerations: This document specifies formats of conforming messages and the interpretation thereof.

Published specification: This document is the specification for these media types; see Table 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Additional information:

Magic number(s): n/a

File extension(s): This document uses the mime type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): n/a

Person & email address to contact for further information: See Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force (mailto:iesg@ietf.org).

12.2. ALTO Entity Domain Type Registry

This document requests IANA to create and maintain the "ALTO Entity Domain Type Registry", listed in Table 2.

Identifier	Entity Identifier Encoding	Hierarchy & Inheritance
ipv4	See Section 6.1.1	See Section 6.1.3
ipv6	See Section 6.1.2	See Section 6.1.3
pid	See Section 6.2	None

Table 2: ALTO Entity Domains Types

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO entity domains types. Second, it states the requirements for allocated entity domains types.

12.2.1. Consistency Procedure between ALTO Address Type Registry and ALTO Entity Domain Type Registry

One potential issue of introducing the "ALTO Entity Domain Type Registry" is its relationship with the "ALTO Address Types Registry" already defined in Section 14.4 of [RFC7285]. In particular, the entity identifier of a type of an entity domain registered in the "ALTO Entity Domain Type Registry" MAY match an address type defined in "ALTO Address Type Registry". It is necessary to precisely define and guarantee the consistency between "ALTO Address Type Registry" and "ALTO Entity Domain Registry".

We define that the ALTO Entity Domain Type Registry is consistent with ALTO Address Type Registry if two conditions are satisfied:

- When an address type is already or able to be registered in the ALTO Address Type Registry [RFC7285], the same identifier MUST be used when a corresponding entity domain type is registered in the ALTO Entity Domain Type Registry.

- If an ALTO entity domain type has the same identifier as an ALTO address type, their addresses encoding MUST be compatible.

To achieve this consistency, the following items MUST be checked before registering a new ALTO entity domain type in a future document:

- Whether the ALTO Address Type Registry contains an address type that can be used as an identifier for the candidate entity domain type identifier. This has been done for the identifiers "ipv4" and "ipv6" of Table 2.

- Whether the candidate entity domain type identifier can potentially be an endpoint address type, as defined in Sections 2.1 and 2.2 of [RFC7285].

When a new ALTO entity domain type is registered, the consistency with the ALTO Address Type Registry MUST be ensured by the following procedure:

- Test: Do corresponding entity domain type identifiers match a known "network" address type?

  * If yes (e.g., cell, MAC or socket addresses):

    + Test: Is such an address type present in the ALTO Address Type Registry?

- If yes: Set the new ALTO entity domain type identifier to be the found ALTO address type identifier.

- If no: Define a new ALTO entity domain type identifier and use it to register a new address type in the ALTO Address Type Registry following Section 14.4 of [RFC7285].

+ Use the new ALTO entity domain type identifier to register a new ALTO entity domain type in the ALTO Entity Domain Type Registry following Section 12.2.2 of this document.

* If no (e.g., pid name, ane name or country code): Proceed with the ALTO Entity Domain Type registration as described in Section 12.2.2.

12.2.2. ALTO Entity Domain Type Registration Process

New ALTO entity domain types are assigned after IETF Review [RFC5226] to ensure that proper documentation regarding the new ALTO entity domain types and their security considerations has been provided. RFCs defining new entity domain types SHOULD indicate how an entity in a registered type of domain is encoded as an EntityID, and, if applicable, the rules defining the entity hierarchy and property inheritance. Updates and deletions of ALTO entity domains types follow the same procedure.

Registered ALTO entity domain type identifiers MUST conform to the syntactical requirements specified in Section 5.1.2. Identifiers are to be recorded and displayed as strings.

Requests to the IANA to add a new value to the Entity Domain Type registry MUST include the following information:

o Identifier: The name of the desired ALTO entity domain type.

o Entity Identifier Encoding: The procedure for encoding the identifier of an entity of the registered domain type as an EntityID (see Section 5.1.3). If corresponding entity identifiers of an entity domain type match a known "network" address type, the Entity Identifier Encoding of this domain identifier MUST include both Address Encoding and Prefix Encoding of the same identifier registered in the ALTO Address Type Registry [RFC7285]. To define properties, an individual entity identifier and the corresponding full-length prefix MUST be considered aliases for the same entity.

o Hierarchy: If the entities form a hierarchy, the procedure for determining that hierarchy.

- Inheritance: If entities can inherit property values from other entities, the procedure for determining that inheritance.

- Media type of defining information resource: some entity domain types allow their entity domain type name to be combined with an information resource name to define a resource-specific entity domain. Such an information resource is called "defining information resource". In this case, the authorized media type of specific information resources MUST be specified in the document defining the entity domain type. When this entity domain type allows combinations with defining resources, this must be indicated and the conditions fully specified in the document. The defining information resource for an entity domain type is the one that:

  * has an entry in the IRD,

  * defines these entities,

  * does not use another information resource that defines these entities,

  * defines and exposes entity identifiers that are all persistent.

  * has a unique media type equal to the one specified in the document defining the entity domain type.

  This information is useful when Servers indicate resource specific entity domains in the property map capabilities. Clients need to know if the combination of information resource type and entity domain type is allowed. See also, Section 4.6 and Section 5.1 for more information.

- Mapping to ALTO Address Type: A boolean value to indicate if the entity domain type can be mapped to the ALTO address type with the same identifier.

- Security Considerations: In some usage scenarios, entity identifiers carried in ALTO Protocol messages may reveal information about an ALTO client or an ALTO service provider. Applications and ALTO service providers using addresses of the registered type should be cognizant of how (or if) the addressing scheme relates to private information and network proximity.

This specification requests registration of the identifiers "ipv4", "ipv6" and "pid", as shown in Table 2.

12.3. ALTO Entity Property Type Registry

This document requests IANA to create and maintain the "ALTO Entity Property Type Registry", listed in Table 3.

This registry extends the "ALTO Endpoint Property Type Registry", defined in [RFC7285], in that a property type is defined on one or more entity domains, rather than just on IPv4 and IPv6 Internet address domains. An entry in this registry is an ALTO entity property type defined in Section 5.2.1. Thus, a registered ALTO entity property type identifier MUST conform to the syntactical requirements specified in that section.

Identifier	Intended Semantics
pid	See Section 7.1.1 of [RFC7285]

Table 3: ALTO Entity Property Types.

Requests to the IANA to add a new value to the registry MUST include the following information:

- Identifier: The unique id for the desired ALTO entity property type. The format MUST be as defined in Section 5.2.1 of this document. It includes the information of the applied ALTO entity domain and the property name.

- Intended Semantics: ALTO entity properties carry with them semantics to guide their usage by ALTO clients. Hence, a document defining a new type SHOULD provide guidance to both ALTO service providers and applications utilizing ALTO clients as to how values of the registered ALTO entity property should be interpreted.

- Security Considerations: ALTO entity properties expose information to ALTO clients. ALTO service providers should be cognizant of the security ramifications related to the exposure of an entity property.

In security considerations, the request should also discuss the sensitivity of the information, and why it is required for ALTO-based operations. Regarding this discussion, the request SHOULD follow the recommendations of Section 14.3. ALTO Endpoint Property Type Registry in [RFC7285].

This document requests registration of the identifier "pid", listed in Table 3. Semantics for this property are documented in

Section 7.1.1 of [RFC7285]. No security issues related to the exposure of a "pid" identifier are considered, as it is exposed with the Network Map Service defined and mandated in [RFC7285].

13. Acknowledgments

The authors would like to thank Dawn Chen (Tongji University), and Shenshen Chen (Tongji/Yale University) for their contributions to earlier drafts. Thank you also to Qiao Xiang (Yale University), Shawn Lin, Xin Wang and Vijay Gurbani (Vail systems) for fruitful discussions. Last, big thanks to Danny Perez (University of Campinas) and Luis Contreras (Telefonica) for their substantial review feedback and suggestions to improve this document.

14. References

14.1. Normative References

[ISO3166-1] The International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes, ISO 3166-1:2013", 2013.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <https://www.rfc-editor.org/info/rfc7921>.

[RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <https://www.rfc-editor.org/info/rfc8008>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

14.2. Informative References

[draft-gao-alto-fcs]
Zhang, J., Gao, K., Wang, J., Xiang, Q., and Y. Yang, "ALTO Extension: Flow-based Cost Query", September 2016.

[draft-ietf-alto-cdni-request-routing-alto]
Roome, J., Yang, Y., Ma, K., Peterson, J., and J. Zhang, "Content Delivery Network Interconnection (CDNI) Request Routing: CDNI Footprint and Capabilities Advertisement using ALTO (work in progress)", 2020.

[I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector", draft-ietf-alto-path-vector-09 (work in progress), November 2019, <http://www.ietf.org/internet-drafts/draft-ietf-alto-path-vector-09.txt>.

Authors' Addresses

Wendy Roome
Nokia Bell Labs (Retired)
124 Burlington Rd
Murray Hill, NJ 07974
USA

Phone: +1-908-464-6975
Email: wendy@wdroome.com

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

Email: Sabine.Randriamasy@nokia-bell-labs.com

Y. Richard Yang
Yale University
51 Prospect Street
New Haven, CT 06511
USA

Phone: +1-203-432-6400
Email: yry@cs.yale.edu

Jingxuan Jensen Zhang
Tongji University
4800 Caoan Road
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

Kai Gao
Sichuan University
Chengdu 610000
China

Email: kaigao@scu.edu.cn