ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: July 6, 2019

W. Roome
Nokia Bell Labs
S. Chen
Tongji University
S. Randriamasy
Nokia Bell Labs
Y. Yang
Yale University
J. Zhang
Tongji University

January 2, 2019

Unified Properties for the ALTO Protocol
draft-ietf-alto-unified-props-new-06

Abstract

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

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 RFC 2119 [RFC2119].

Status of This Memo

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 two limitations.

First, it only allows properties to be associated with a particular domain of entities, namely individual IP addresses. It is reasonable to think that collections of endpoints, as defined by CIDRs [RFC4632] or PIDs, may also have properties. Since the EPS cannot be extended to new entity domains, new services, with new request and response messages, would have to be defined for new entity domains.

Second, 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 processes costs lookups without querying the ALTO

server. [RFC7285] does not define an equivalent 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. But in practice, it is highly unlikely that properties will be defined for every endpoint address. It is much more likely that properties will only be defined for a subset of endpoint addresses, and that subset would be small enough to be enumerated. This is particularly true if blocks of endpoint addresses with a common prefix (e.g., a CIDR) have the same value for a property. Furthermore, entities in other domains may very well be enumerable.

This document proposes a new approach to retrieve ALTO properties. Specifically, it defines two new types of resources, namely Property Map (see Section 4) and Filtered Property Map (see Section 5). The former is a GET-mode resource which returns the property values for all entities in a domain, and is analogous to a network map or a cost map in [RFC7285]. The latter is a POST-mode resource which returns the values for a set of properties and entities requested by the client, and is analogous to a filtered network map or a filtered cost map.

Additionally, this document introduces ALTO Entity Domains, where an entity is a generalization of an endpoint to also represent, a PID, a network element, or a cell in a cellular network, etc. As a consequence, ALTO Entity Domains defined in this document are a super-set of ALTO Address Types defined in [RFC7285]. Their exact relationship is specified in Section 9.2.1.

Entity domains and property names are extensible. New entity domains can be defined without revising the messages defined in this document, in the same way that new cost metrics and new endpoint properties can be defined without revising the messages defined in [RFC7285].

This proposal would subsume the Endpoint Property Service defined in [RFC7285], although that service may be retained for legacy clients (see Section 6).

2. Definitions and Concepts

2.1. Entity

The entity is a generalized concept of the endpoint defined in Section 2.1 of [RFC7285]. An entity is an object with a (possibly empty) set of properties. Each entity MUST be in one and only one domain, such as the IPv4 domain or the IPv6 domain, and has a unique address.

2.2. Entity Domain

An entity domain is a set of entities. Examples of domains are the Internet address domains (see Section 3.1 and the PID domain (see Section 3.2). This document will define the domains precisely below.

2.3. Domain Name

Each entity domain has a unique name. A domain name MUST be no more than 32 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). For example, the names "ipv4" and "ipv6" identify entities in the Internet address domains (see Section 3.1).

The type DomainName is used in this document to denote a JSON string with a domain name in this format.

Domain names MUST be registered with the IANA, and the format of the entity addresses (see Section 2.4) in that entity domain, as well as any hierarchical or inheritance rules (see Section 2.6) for those entities, MUST be specified at the same time.

2.4. Entity Address

Each entity has a unique address of the format:

```
    EntityAddr ::= DomainName : DomainSpecificEntityAddr
```

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

The type EntityAddr is used in this document to denote a JSON string with an entity address in this format.

The format of the second part of an entity address depends on the entity domain, and MUST be specified when registering a new entity domain. Addresses MAY be hierarchical, and properties MAY be inherited based on that hierarchy. Again, the rules defining any hierarchy or inheritance MUST be defined when the entity domain is registered.

Note that an entity address MAY have different textual representations, for a given entity domain. For example, the strings "ipv6:2001:db8::1" and "ipv6:2001:db8:0:0:0:0:0:1" refer to the same entity.

2.5. Property Type and Property Name

Every entity in some domain MAY have one or more properties. Every property MUST have a unique Property Type.

This document defines property types in the domain-specific context. This design is to enforce that each property type MUST be registered for a single specific entity domain. But multiple property types with the similar semantics MAY share the same Property Name in different entity domains. This design decision is adopted because of the following considerations:

- Some properties may only be applicable for particular entity domains, not all. For example, the "pid" property is not applicable for entities in the "pid" domain.

- The interpretation of the value of a property may depend on the entity domain. For different entity domains, not only the intended semantics but also the dependent resource types may be totally different. For example, suppose that the "geo-location" property is defined as the coordinates of a point, encoded as (say) "latitude longitude [altitude]." When applied to an entity that represents a specific host computer, such as an Internet address, the property defines the host's location and has no required dependency. However, when applied to an entity in the "pid" domain, the property would indicate the location of the center of all hosts in this "pid" entity and depend on a Network Map defining this "pid" entity.

To achieve this, each property type has a unique identifier encoded as the following format:

```
PropertyType ::= DomainName : PropertyName
```

The "DomainName" indicates which entity domain the property type applies to. The "PropertyName" SHOULD refer to the semantics of this property type. It does not have to be global unique. In other words, different property types could have the same property name applied to different entity domains, if they have the similar semantics. For example, the property types "ipv4:pid" and "ipv6:pid" have the same property name "pid" applied to both "ipv4" and "ipv6" domains.

Property types MUST be registered with the IANA, and the intended semantics, as well as the media types of dependent resources and the interpretation, MUST be specified at the same time.

2.6. Hierarchy and Inheritance

Entities in a given domain MAY form a hierarchy based on entity addresses, and introducing hierarchy allows the introduction of inheritance. Each entity domain MUST define its own hierarchy and inheritance rules when registered. The hierarchy and inheritance rule makes it possible for an entity to inherit a property value from another entity in the same domain.

2.7. Relationship with Other ALTO Resources

[RFC7285] recognizes that some properties for some entity domains MAY be specific to an ALTO resource, such as a network map. Accordingly Section 10.8.1 of [RFC7285] defines the concept of "resource-specific endpoint properties", and indicates that dependency by prefixing the property name with the ID of the resource on which it depends. That document defines one resource-specific property, namely the "pid" property, whose value is the name of the PID containing that endpoint in the associated network map.

This document takes a different approach. Instead of defining the dependency by qualifying the property name, this document attaches the dependency to the entity domains. Thus each resource-specific property of all entities in a specific domain depends on the same resources; the properties of entities in another domain may depend on another resource. For example, in a single property map, the "pid" property of all entities in an Internet address domain MUST depend on the same network map. Each property of all entities in the PID domain MUST also depend on a network map; but different properties may depend on different network maps.

Specifically, this document uses the "uses" and "dependent-vtags" fields defined in Sections 9.1.5 and 11.1 of [RFC7285], respectively, to specify the preceding dependency: the "uses" field of an IRD entry providing entity domain related resources (see Property Map and Filtered Property Map resources below) specifies the dependent resources, and the "dependent-vtags" field specifies dependency in message responses.

3. Entity Domains

This document defines three entity domains. The definition of each entity domain below includes the following: (1) domain name, (2) domain-specific addresses, and (3) hierarchy and inheritance semantics.

3.1. Internet Address Domains

The document defines two entity domains (IPv4 and IPv6) for Internet addresses. Both entity domains include individual addresses and blocks of addresses. Since the two domains use the same hierarchy and inheritance semantics, we define the semantics together, instead of repeating for each.

3.1.1. IPv4 Domain

3.1.1.1. Domain Name

ipv4

3.1.1.2. Domain-Specific Entity Addresses

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

3.1.2. IPv6 Domain

3.1.2.1. Domain Name

ipv6

3.1.2.2. Domain-Specific Entity Addresses

Individual addresses are strings as specified by Section 4 of [RFC5952]; blocks of addresses are prefix-match strings as specified in Section 7 of [RFC5952]. For the purpose of defining 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.

3.1.3. Hierarchy and Inheritance of ipv4/ipv6 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 some block C which prefix-matches I, then the address I inherits the value of P defined for block C. If more than one such block defines a value for P, I inherits the value of P in the block with the longest prefix. It is important to notice

that this longest prefix rule will ensure no multiple inheritance, and hence no ambiguity.

Address blocks can also inherit properties: if a property P is not defined for a block C, but is defined for some block C' which covers all IP addresses in C, and C' has a shorter mask than C, then block C inherits the property from C'. If there are several such blocks C', C inherits from the block with the longest prefix.

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:

o 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."

o 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.

3.2. 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.

3.2.1. Domain Name

pid

3.2.2. Domain-Specific Entity Addresses

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

3.2.3. Hierarchy and Inheritance

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

3.2.4. 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 some or all properties of a PID to the prefixes in that PID.

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".

3.3. Internet Address Properties vs. PID Properties

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

The PID domain 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 the Internet address domain. On the other hand, a property that explains why a PID was formed, or how it relates a provider's network, would best be associated with the PID domain.

4. 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 7.4 gives an example of a property map request and its response.

4.1. Media Type

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

4.2. HTTP Method

The property map is requested using the HTTP GET method.

4.3. Accept Input Parameters

None.

4.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities:

```
    object {
      DomainName entity-domains<1..*>;
      PropertyName properties<1..*>;
    } PropertyMapCapabilities;
```

where "entity-domains" is an array specifying the entity domains, and "properties" is an array specifying the property names returned for entities in those domains. The semantics is that each domain in "entity-domains" provides all properties defined in "properties". If a property in "properties" is NOT supported by a domain in "entity-

domains", the server can declare different property maps to conform to the semantics.

4.5. Uses

The "uses" field of a property map resource in an IRD entry specifies dependencies as discussed in Section 2.7. It is an array of the resource ID(s) of the resource(s) that properties of entities in domains specified in "entity-domains" depend on.

In a single property map, every property value of every entity depends on the same array of resources. Thus, if properties depend on different resources arrays would be provided, they MUST be split into different property maps.

Note that according to [RFC7285], a legacy ALTO server with two network maps, with resource IDs "net1" and "net2", could offer a single Endpoint Property Service for the two properties "net1.pid" and "net2.pid". An ALTO server which supports the property map resource defined in this document, would, instead, offer two different property maps for the "pid" property, one depending on "net1", and the other on "net2".

4.6. 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 {
  EntityAddr -> EntityProps;
} PropertyMapData;

object {
  PropertyName -> 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, the ALTO server returns the value defined for each of the properties specified in this resource's "capabilities" list. 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.

5. 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 7.5, Section 7.6, Section 7.7 and Section 7.8 give examples of filtered property map requests and responses.

5.1. Media Type

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

5.2. HTTP Method

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

5.3. Accept Input Parameters

The input parameters for a filtered property map request are supplied in the entity body of the POST request. 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 {
    EntityAddr     entities<1..*>;
    PropertyName   properties<1..*>;
  } ReqFilteredPropertyMap;
```

with fields:

entities: List of entity addresses 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 5.4).

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 5.4). 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.

5.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities, as defined in Section 4.4.

5.5. Uses

The "uses" field of a filtered property map is an array with the resource ID(s) of resource(s) that each domain in "entity-domains" depends on, in order to provide the properties specified in the "properties" capability. The same "uses" rule as defined by the property map resource applies (see Section 4.5).

5.6. 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 as follows:

- An entity address in "entities" in the request is invalid if:

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

  * The entity address is an invalid address in the entity domain.

  A valid entity address is never an error, even if this filtered property map resource does not define any properties for it.

If an entity address in "entities" in 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 this entity address.

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

  It is not an error that a filtered property map resource does not define a requested property's value for a particular entity. 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 4.6), except that:

- The "dependent-vtags" field in its "meta" field only includes the version tags of resources on which the requested properties of the entity domains depend, and the order MUST be consistent with the "uses" field of this filtered property map resource.

- It only includes the entities and properties requested by the client. If an entity in the request is an address block (e.g., an "ipv4" or "ipv6" entity), the response MUST cover properties for all addresses in this block.

It is important that the filtered property map response MUST include all inherited property values for the requested entities and all the entities which are able to inherit property values from them. 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 in the response is already covered by some other entities in the same response, it SHOULD be removed from the response for compactness. For example, 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.

6. Impact on Legacy ALTO Servers and ALTO Clients

6.1. 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.

6.2. 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 upon, while global property names have no such prefix. The property map and the filtered property map defined in this document do not distinguish between those two types of properties. Instead, if there is a dependency, it is indicated by the "uses" capability of a property map, and is shared by all properties and entity domains in that map. Accordingly, it is RECOMMENDED that resource-specific endpoint properties be deprecated, and no new resource-specific endpoint properties be defined.

6.3. Impact on the pid Property

Section 7.1.1 of [RFC7285] defines the resource-specific endpoint property name "pid", whose value is the name of the PID containing that endpoint. For compatibility with legacy clients, an ALTO server which provides the "pid" property via the EPS MUST use that definition, and that syntax.

However, when used with property maps, this document amends the definition of the "pid" property as follows.

First, the name of the property is simply "pid"; the name is not prefixed with the resource ID of a network map. The "uses" capability of the property map indicates the associated network map. This implies that a property map can only return the "pid" property for one network map; if an ALTO server provides several network maps, it MUST provide a Property Map for each of the network maps.

Second, a client MAY request the "pid" property for a block of addresses. An ALTO server determines the value of "pid" for an address block C as the rules defined in Section 5.6.

Note that although an ALTO server MAY provide a GET-mode property map which returns the entire map for the "pid" property, there is no need to do so, because that map is simply the inverse of the network map.

6.4. 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 blocks of addresses, rather than just individual addresses, which might change the semantics of a property.

7. Examples

7.1. Network Map

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

```
defaultpid:  ipv4:0.0.0.0/0  ipv6:::0/0
pid1:        ipv4:192.0.2.0/25
pid2:        ipv4:192.0.2.0/28  ipv4:192.0.2.16/28
pid3:        ipv4:192.0.3.0/28
pid4:        ipv4:192.0.3.16/28
```

Figure 3: Example Network Map

7.2. 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
ipv4:192.0.2.0:	-	-	-	PA
ipv4:192.0.3.0/28:	-	12346	-	TX
ipv4:192.0.3.16/28:	-	12346	-	MN

Figure 4: Example Property Values for Internet Address Domains

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

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

Figure 5: Example Property Values for 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 "ipv4:192.0.2.0" 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.

7.3. Information Resource Directory (IRD)

The following IRD defines the relevant resources of the ALTO server. It provides two property maps, one for the "ISP" and "ASN" properties, and another for the "country" and "state" properties. The server could have provided a single property map for all four properties, but did not, presumably because the organization that runs the ALTO server believes any given client is not interested in all four properties.

The server provides two filtered property maps. The first returns all four properties, and the second just returns 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 for the "pid" property does have a "uses" capability for

the default network map, because that defines the values of the "pid" property.

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

```
"meta" : {
    ...
    "default-alto-network-map" : "default-network-map"
},
"resources" : {
   "default-network-map" : {
      "uri" : "http://alto.example.com/networkmap",
      "media-type" : "application/alto-networkmap+json"
   },
   .... property map resources ....
   "country-state-property-map" : {
      "uri" : "http://alto.example.com/propmap/full/inet-cs",
      "media-type" : "application/alto-propmap+json",
      "capabilities" : {
        "entity-domains": [ "ipv4", "ipv6" ],
        "properties" : [  "country", "state" ]
      }
   },
   "isp-asn-property-map" : {
      "uri" : "http://alto.example.com/propmap/full/inet-ia",
      "media-type" : "application/alto-propmap+json",
      "capabilities" : {
        "entity-domains": [ "ipv4", "ipv6" ],
        "properties" : [ "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",
      "capabilities" : {
        "entity-domains": [ "ipv4", "ipv6" ],
        "properties" : [ "ISP", "ASN", "country", "state" ]
      }
   },
   "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" ]
      "capabilities" : {
        "entity-domains" : [ "ipv4", "ipv6" ],
```

```
              "properties" : [ "pid" ]
            }
         },
         "region-property-map": {
           "uri": "http://alto.exmaple.com/propmap/region",
           "media-type": "application/alto-propmap+json",
           "accepts": "application/alto-propmapparams+json",
           "uses" : [ "default-network-map" ],
           "capabilities": {
             "domain-types": [ "pid" ],
             "properties": [ "region" ]
           }
         },
         "legacy-pid-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" ]
            }
         }
      }
```

Figure 6: Example IRD

7.4. Property Map Example

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

Note that, to be compact, the response does not includes 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

   {
     "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"}
     }
   }
```

7.5. 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

{
  "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"}
  }
}
```

7.6. 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 listed in the request explicitly. 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

{
  "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"}
 }
}
```

7.7. Filtered Property Map Example #3

The following example uses the filtered property map resource to request the "pid" property for several IPv4 addresses and prefixes.

Note that the entity "ipv4:192.0.3.0/27" is redundant in the response. Although it can inherit a value of "defaultpid" for the "pid" property from the entity "ipv4:0.0.0.0/0", none of addresses in it is in "defaultpid". Because blocks "ipv4:192.0.3.0/28" and "ipv4:192.0.3.16/28" have already cover all addresses in that block. So an ALTO server who wants a compact response can omit this entity.

```
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",
                "ipv4:192.0.3.0/27" ],
  "properties" : [ "pid" ]
}
```

```
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": {
    "ipv4:192.0.2.128":   {"pid": "defaultpid"},
    "ipv4:192.0.2.0/27":  {"pid": "defaultpid"},
    "ipv4:192.0.3.0/28":  {"pid": "pid3"},
    "ipv4:192.0.3.16/28": {"pid": "pid4"}
  }
}
```

7.8. Filtered Property Map Example #4

The following example uses the filtered property map resource to request the "region" property for several PIDs defined in "default-network-map". The value of the "region" property for each PID is not defined by "default-network-map", but the reason why the PID is defined by the network operator.

```
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" : ["pid:pid1",
                "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": {
    "pid:pid1": {
      "region": "west"
    },
    "pid:pid2": {
      "region": "east"
    }
  }
}
```

8. 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 then should 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.

9. IANA Considerations

This document defines additional application/alto-* media types, and extends the ALTO endpoint property registry.

9.1. application/alto-* Media Types

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

Type	Subtype	Specification
application	alto-propmap+json	Section 4.1
application	alto-propmapparams+json	Section 5.3

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).

9.2. ALTO Entity Domain Registry

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

Identifier	Entity Address Encoding	Hierarchy & Inheritance	Mapping to ALTO Address Type
ipv4	See Section 3.1.1	See Section 3.1.3	Yes
ipv6	See Section 3.1.2	See Section 3.1.3	Yes
pid	See Section 3.2	None	No

Table 2: ALTO Entity Domains.

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

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

One potential issue of introducing the "ALTO Entity Domain Registry" is its relationship with the "ALTO Address Types Registry" already defined in Section 14.4 of [RFC7285]. In particular, the entity

address of an entity domain registered in the "ALTO Entity Domain 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 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 is registered in the ALTO Entity Domain Registry.

- If an ALTO entity domain 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 in a future document:

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

- Whether the candidate entity address of the entity domain is able to be an endpoint address, as defined in Sections 2.1 and 2.2 of [RFC7285].

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

- Test: Do corresponding entity addresses 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 identifier to be the found ALTO address type identifier.

      - If no: Define a new ALTO entity domain 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 identifier to register a new ALTO entity domain in the ALTO Entity Domain Registry following Section 9.2.2 of this document.

- If no (e.g., pid name, ane name or country code): Proceed with the ALTO Entity Domain registration as described in Section 9.2.2.

9.2.2. ALTO Entity Domain Registration Process

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

Registered ALTO entity domain identifiers MUST conform to the syntactical requirements specified in Section 2.3. Identifiers are to be recorded and displayed as strings.

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

- Identifier: The name of the desired ALTO entity domain.

- Entity Address Encoding: The procedure for encoding the address of an entity of the registered type as an EntityAddr (see Section 2.4). If corresponding entity addresses of an entity domain match a known "network" address type, the Entity Address 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]. For the purpose of defining properties, an individual entity address and the corresponding full-length prefix MUST be considered aliases for the same entity.

- 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.

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

- Security Considerations: In some usage scenarios, entity addresses 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 made aware 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.

9.3. ALTO Entity Property Type Registry

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

To distinguish with the "ALTO Endpoint Property Type Registry", each entry in this registry is an ALTO entity property type defined in Section 2.5. Thus, registered ALTO entity property type identifier MUST conform to the syntactical requirements specified in that section.

The initial registered ALTO entity property types are listed in Table 3.

Identifier	Intended Semantics	Dependencies and Interpretation
ipv4:pid	PID for the IPv4 entity	application/alto-networkmap+json, where the PID names are defined
ipv6:pid	PID for the IPv6 entity	application/alto-networkmap+json, where the PID names are defined

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 2.5 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.

- Dependencies and Interpretation: Dependent ALTO resources MAY be required by ALTO clients to interpret ALTO entity properties. Hence, a document defining a new type SHOULD provide a sequence of media types in which the dependent ALTO resources are and the guidance how ALTO clients use them to interpret the property.

This specification requests registration of the identifiers "ipv4:pid" and "ipv6:pid", as shown in Table 3.

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <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>.

[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>.

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

Shiwei Dawn Chen
Tongji University
4800 Caoan Road
Shanghai 201804
China

Email: dawn_chen_f@hotmail.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