ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

W. Roome
Nokia Bell Labs
R. Yang
Yale University
July 3, 2017

Extensible Property Maps for the ALTO Protocol
draft-roome-alto-unified-props-new-01

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285] by generalizing the concept of "endpoint properties" to other entity domains, 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

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

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

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

This Internet-Draft will expire on January 4, 2018.

Table of Contents

1. Introduction

The ALTO protocol [RFC7285] introduced the concept of "properties" attached to "endpoint addresses," and defined the Endpoint Property Service (EPS) to allow 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. Furthermore, recent proposal ([I-D.ietf-alto-path-vector]) have suggested new classes of entities (ANE) with properties. The EPS cannot be extended to new entity domains. Instead, new services, with new request and response messages, would have to be defined for each new entity domain.

Second, the EPS is only defined as a POST-mode service. Clients must request the properties for an explicit set of addresses. By contrast, [RFC7285] defines a GET-mode Cost Map resource which returns all available costs, so a client can get the full set of costs once, and then lookup costs without querying the ALTO server. RFC7285 does not define an equivalent service for endpoint properties. And it is unlikely a property will be defined for every possible address. It is very likely that properties will only be

defined for a subset of addresses, and that subset would be small enough to enumerate. This is particularly true if blocks of 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 resource types, namely Property Maps (see Section 4) and Filtered Property Maps (see Section 5). The former are GET-mode resources which return the property values for all entities in a domain, and are analogous to the ALTO's Network Maps and Cost Maps. The latter are POST-mode resources which return the values for a set of properties and entities requested by the client, and are analogous to ALTO's Filtered Network Maps and Filtered Cost Maps.

Entity domains and property names are extensible, so that new 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 by the ALTO protocol.

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

An entity is an object with a (possibly empty) set of properties. Every entity is in a domain, such as the IPv4 and IPv6 domains, and has a unique address.

2.2. Domain

A domain is a family of entities. Two examples are the Internet address and PID domain (see Section 3.1 and Section 3.2) that this document will define. An additional example is the proposed domain of Abstract Network Elements associated with topology and routing, as suggested by [I-D.ietf-alto-path-vector].

2.3. Entity Address

Each entity has a unique address of the format:

```
domain-name : domain-specific-entity-address
```

Examples from the IP domain 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 denotes a JSON string with an entity address in this format.

The format of the second part of an entity address depends on the domain, and MUST be specified when registering a new 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 domain is registered.

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

2.4. Domain Name

Each 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 objects in the Internet address domain (Section 3.1).

The type DomainName denotes 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 in that domain, as well as any hierarchical or inheritance rules for those entities, MUST be specified at the same time.

2.5. Property Name

The space of property names associated with entities defined by this document is the same as, and is shared with, the endpoint property names defined by [RFC7285]. Thus entity property names are as defined in Section 10.8.2 of that document, and MUST be registered with the "ALTO Endpoint Property Type Registry" defined in Section 14.3 of that document.

The type PropertyName denotes a JSON string with a property name in this format.

A main design decision is that property names are defined in a single namespace, not specific to a domain, although some properties MAY only be applicable for particular domains. This design decision is to enforce a design that similar properties are named similarly.

The interpretation of the value of a property, however, MAY depend on the domain. For example, suppose the "geo-location" property is defined as the coordinates of a point, encoded as "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. When applied to an entity that represents a set of computers, such as a CIDR, the property would be the location of the center of that set. If it is necessary to represent the bounding box of a set of hosts, another property, such as "geo-region", SHOULD be defined.

2.6. Property Value

The property value MAY BE defined or undefined. If it is defined, it SHOULD BE a JSONString or a JSON "null" value. Otherwise, a protocol implementation SHOULD fail to parse the property value, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

2.7. Hierarchy and Inheritance

Entities in a given domain MAY form hiearchy based on entity address. Each domain MAY define its own hierarchy and inheritance semantics. Given a property, the semantics MUST NOT allow an entity with a defined property value to inherit the property value of another entity. Given a property, the semantics MAY allow an entity with an undefined property value to inherit the property value of another entity. An entity may not be able to inherit a property value if no other entities satisfy the inheritance conditions defined by the semantics. The hierarchy and inheritance semantics SHOULD be defined carefully to avoid multiple inheritance.

2.8. Relationship to Network Maps

[RFC7285] recognizes that some properties MAY be specific to an ALTO resource, such as a network map. Accordingly [RFC7285] defines the concept of "resource-specific endpoint properties" (Section 10.8.1), 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 property map as a whole. Thus all properties in a given property map depend on the same resource. Furthermore, entity addresses MAY depend on a network map (for example, the Abstract Network Elements suggested by [I-D.ietf-alto-path-vector]). Associating the dependency with the property map handles any entity address dependencies as well.

The "uses" field in an IRD entry defines the dependencies of a property map resource, and the "dependent-vtags" field in a property map response defines the dependencies of that map. These fields are defined in Sections 9.1.5 and 11.1 of [RFC7285], respectively.

This is similar to how RFC7285 handles dependencies between cost maps and network maps. Recall that cost maps present the costs between PIDs, and PID names depend on a network map. If an ALTO server provides the "routingcost" metric for the network maps "net1" and "net2", then the server defines two separate cost maps, one for "net1" and the other for "net2".

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 extensions defined in this document, would, instead, offer two different Property Maps for the "pid" property, one depending on "net1", the other on "net2".

3. Entity Domains

This document defines the following entity domains. For the definition of each domain, it includes the following template: domain name, domain-specific addresses, and hierarchy and inheritance semantics.

3.1. Internet Address Domains

The document defines two domains (IPv4 and IPv6) for Internet addresses. Both domains include individual addresses and blocks of addresses.

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 domains allow property values to be inherited. Specifically, if a property P is not defined for a specific Internet address IP, but P is defined for some block C which prefix-matches IP, then the address IP inherits the value of P defined for block C. If more than one such block defines a value for P, IP 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 property P is not defined for a block C, but is defined for some block C' prefix-matches 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.

```
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 define a property as not having a value for a particular entity. That is, a server MAY say that a property is "defined to have no value", as opposed to the property being "undefined". If that entity would inherit a value for that property, then the ALTO server MUST return a "null" value for that property, and an ALTO client MUST recognize a "null" value means "do not apply the inheritance rules for this property." If the entity would not inherit a value, the ALTO server MAY return "null" or MAY just omit the property. TODO: Discuss more.

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

3.1.4. Relationship to Network Maps

TODO: Need discussion. An Internet address domain MAY be associated with an ALTO network map resource. Logically, there is a map of Internet address entities to property values for each network map defined by the ALTO server, plus an additional property map for Internet address entities which are not associated with a network map. These maps are separate from each other. The prefixes in the property map do not have to correspond to the prefixes defining the network map's PIDs. For example, the property map for a network map MAY assign properties to "ipv4:192.0.2.0/24" even if that prefix is not associated with any PID in the network map.

3.2. PID Domain

The PID domain associates property values with the PIDs in a network map. Accordingly, this 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 domain is best for a property. In general, the Internet address domain is 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 the a provider's network, would best be associated with the PID domain.

3.4. ANE Domain

3.4.1. Domain Name

ane

3.4.2. Domain-Specific Entity Addresses

The entity address of ane domain is encoded as a JSON string. The string MUST be no more than 64 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), the at sign ('@', code point U+0040), the low line ('_', U+005F), or the '.' separator (U+002E). The '.' separator is reserved for future use and MUST NOT be used unless specifically indicated in this document, or an extension document.

3.4.3. Hierarchy and Inheritance

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

3.4.4. Relationship to Cost Map

TBA

4. Property Map Resource

A Property Map returns the properties defined for all entities in one or more domains. Note that Property Map Resource is not applicable to ANE domain.

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

4.1. Media Type

The media type of an ALTO Property Map resource is "application/alto-propmap+json".

4.2. HTTP Method

An ALTO Property Map resource 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 domain-types<1..*>;
  PropertyName prop-types<1..*>;
} PropertyMapCapabilities;
```

where "domain-types" is an array with the domains of the entities in this property map, and "prop-types" is an array with the names of the properties returned for entities in those domains. TODO: discuss semantics and requirements of multiple domains.

4.5. Uses

An array with the resource ID(s) of resource(s) with which the domains in this map are associated. In most cases, this array will have at most one ID, for example, for a network map resource. TODO: discuss semantics and requirements of multiple resources.

4.6. Response

If the 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. 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.

An ALTO Server MAY explicitly define a property as not having a value for a particular entity. That is, a server MAY say that a property is "defined to have no value", as opposed to the property being "undefined". If that entity would inherit a value for that property, then the ALTO server MUST return a "null" value for that property, and an ALTO client MUST recognize a "null" value means "do not apply the inheritance rules for this property." If the entity would not inherit a value, the ALTO server MAY return "null" or MAY just omit the property.

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 domain's inheritance rules to deduce those values.

5. Filtered Property Map Resource

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 and Section 7.7 give examples of filtered property map requests and responses.

5.1. Media Type

The media type of an ALTO Property Map resource is "application/alto-propmap+json".

5.2. HTTP Method

An ALTO Filtered Property Map resource 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 domains in this resource's "capabilities" field (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 (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

An array with the resource ID(s) of resource(s) with which the domains in this map are associated. In most cases, this array will have at most one ID, and it will be for a network map resource.

5.6. Response

The response is the same as for the property map (Section 4.6), except that it only includes the entities and properties requested by the client.

Also, the Filtered Property Map response MUST include all inherited property values for the specified entities (unlike the Full Property Map, the Filtered Property Map response does not include enough information for the client to calculate the inherited values).

Discussion Needed: sometimes the client can compute some inherited property values. In this case, can the Filter Property Map response only contain the uncomputable inherited property values instead of all of them?

6. Impact on Legacy ALTO Servers and ALTO Clients

6.1. Impact on Endpoint Property Service

The Property Maps defined in this document provide the same functionality as the Endpoint Property Service (EPS) defined in Section 11.4 of [RFC7285]. Accordingly, it is RECOMMENDED that the EPS be deprecated in favor of Property Maps. 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 depended upon, while global property names have no such prefix. The property map resources 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 "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 Endpoint Property Service MUST use that definition, and that syntax, in the EPS resource.

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 resource 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 resource for each one.

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 follows. Let CS be the set of all address blocks in the network map. If C is in CS, then the value of "pid" is the name of the PID associated with C. Otherwise, find the longest block C' in CS such that C' prefix-matches C, but is shorter than C. If there is such a block C', the value of "pid" is the name of the PID associated with C'. If not, then "pid" has no value for block C.

Note that although an ALTO server MAY provide a GET-mode property map resource 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
```

Figure 3: Example Network Map

7.2. Property Definitions

The examples in this section use four additional properties, "ISP", "ASN", "country" and "state", with the following values:

```
                            ISP    ASN   country   state
ipv4:192.0.2.0/24:    BitsRus    -      us       -
ipv4:192.0.2.0/28:        -    12345    -        NJ
ipv4:192.0.2.16/28:       -    12345    -        CT
ipv4:192.0.2.0:           -      -      -        PA
```

Figure 4: Example Property Values

7.3. Information Resource Directory (IRD)

The following IRD defines the relevant resources of the ALTO server. It provides two Property Map resources, one for the "ISP" and "ASN" properties, and another for the "country" and "state" properties. The server could have provided a Property Map resource 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": { ... },
   "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" : {
              "domain-types": [ "ipv4", "ipv6" ],
              "prop-types" : [  "country", "state" ]
            }
         },
         "isp-asn-property-map" : {
            "uri" : "http://alto.example.com/propmap/full/inet-ia",
            "media-type" : "application/alto-propmap+json",
            "capabilities" : {
              "domain-types": [ "ipv4", "ipv6" ],
              "prop-types" : [ "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" : {
              "domain-types": [ "ipv4", "ipv6" ],
              "prop-types" : [ "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" : {
              "domain-types" : [ "ipv4", "ipv6" ],
              "prop-types" : [ "pid" ]
            }
         },
         "availbw-property-map" : {
            "uri" : "http://alto.example.com/propmap/lookup/availbw",
            "media-type" : "application/alto-propmap+json",
            "accepts" : "application/alto-propmapparams+json",
            "capabilities" : {
              "domain-types" : [ "ane" ],
              "prop-types" : [ "availbw", "delay" ]
            }
         },
         "legacy-pid-property-map" : {
```

```
            "uri" : "http://alto.example.com/legacy/eps-pid",
            "media-type" : "application/alto-endpointprop+json",
            "accepts" : "application/alto-endpointpropparams+json",
            "capabilities" : {
              "prop-types" : [ "default-network-map.pid" ]
            }
         }
      }
```

Figure 5: 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 the response does not include the entity "ipv4:192.0.2.0", because it does not have a value for either of those properties. Also note that the entities "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28" are refinements of "ipv4:192.0.2.0/24", and hence inherit its value for "ISP" property. But because that value is inherited, it is not

```
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": {
    "ipv4:192.0.2.0/24":   {"ISP: "BitsRus"},
    "ipv4:192.0.2.0/28":   {"ASN": "12345"},
    "ipv4:192.0.2.16/28":  {"ASN": "12345"}
  }
}
```

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 none of the returned property values were explicitly defined; all values are derived by the inheritance rules for Internet address entities.

Also note the "ASN" property has the value "12345" for both the blocks "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28", so every address in the block "ipv4:192.0.2.0/27" has that property value. However the block "ipv4:192.0.2.0/27" itself does not have a value for "ASN": address blocks cannot inherit properties from blocks with longer prefixes, even if every such block has the same value.

```
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",
                 "ipv4:192.0.2.0/27",
                 "ipv4:192.0.2.0/28" ],
  "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"},
    "ipv4:192.0.2.0/27":  {"country": "us"},
    "ipv4:192.0.2.0/28":  {"ASN": "12345",
                           "country": "us",
                           "state": "NJ"}
 }
}
```

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 value of "pid" for the prefix "ipv4:192.0.2.0/26" is "pid1", even though all addresses in that block are in "pid2", because "ipv4:192.0.2.0/25" is the longest prefix in the network map which prefix-matches "ipv4:192.0.2.0/26", and that prefix is in "pid1".

```
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" : [
                "ipv4:192.0.2.0",
                "ipv4:192.0.2.16",
                "ipv4:192.0.2.64",
                "ipv4:192.0.2.128",
                "ipv4:192.0.2.0/26",
                "ipv4:192.0.2.0/30" ],
  "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": {
    "ipv4:192.0.2.0":     {"pid": "pid2"},
    "ipv4:192.0.2.16":    {"pid": "pid2"},
    "ipv4:192.0.2.64":    {"pid": "pid1"},
    "ipv4:192.0.2.128":   {"pid": "defaultpid"},
    "ipv4:192.0.2.0/26":  {"pid": "pid1"},
    "ipv4:192.0.2.0/30":  {"pid": "pid2"}
  }
}
```

7.8. Filtered Property Map Example #4

The following example uses the Filtered Property Map resource to request the "availbw" property for several abstract network elements.

```
POST /propmap/lookup/availbw 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:L001",
                "ane:Lae0",
                "ane:L3eb ],
  "properties" : [ "availbw" ]
}


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

{
  "property-map": {
    "ane:L001":    {"availbw": "55"},
    "ane:Lae0":    {"availbw": "70"},
    "ane:L3eb":    {"availbw": "40"}
  }
}
```

8. Security Considerations

As discussed in Section 15 of [RFC7285], properties MAY have sensitive customer-specific information. If this is the case, an ALTO Server MAY limit access to those properties by providing several different Property Maps. For non-sensitive properties, the ALTO Server would provide a URI which accepts requests from any client. Sensitive properties, on the other hand, would only be available via a secure URI which would require client authentication.

Also, while technically this document does not introduce any security risks not inherent in the Endpoint Property Service defined by [RFC7285], the GET-mode property map resource defined in this document does make it easier for a client to download large numbers of property values. Accordingly, an ALTO Server SHOULD limit GET-mode Property Maps to properties which do not contain sensitive data.

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
ipv4 ipv6 pid ane	See Section 3.1.1 See Section 3.1.2 See Section 3.2 See Section 3.4	See Section 3.1.3 See Section 3.1.3 None None

Table 2: ALTO Entity Domain Names.

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

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 EntityName, 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 2.4. Identifiers are to be recorded and displayed as strings.

Requests 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 2.3).
- 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.
- 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 Endpoint Property Type Registry

The ALTO Endpoint Property Type Registry was created by [RFC7285]. If possible, the name of that registry SHOULD be changed to "ALTO Entity Property Type Registry", to indicate that it is not restricted to Endpoint Properties. If it is not feasible to change the name, the description MUST be amended to indicate that it registers properties in all domains, rather than just the Internet address domain.

10. References

[I-D.ietf-alto-path-vector]
Bernstein, G., Chen, S., Gao, K., Lee, Y., Roome, W., Scharf, M., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector Cost Mode", draft-ietf-alto-path-vector-00 (work in progress), May 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Wendy Roome
Nokia Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: wendy@roome.com

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

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