ALTO WG W. Roome
Internet-Draft S. Randriamasy
Intended status: Standards Track Nokia Bell Labs
Expires: March 7, 2020 Y. Yang
Yale University
J. Zhang
Tongji University
K. Gao
Sichuan University
September 4, 2019

Unified Properties for the ALTO Protocol
draft-ietf-alto-unified-props-new-09

Abstract

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

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Copyright Notice

Table of Contents

1. Introduction 4
2. Overview: Basic Concepts 6
2.1. Entity 6
2.2. Entity Property 6
2.3. Property Map 7
2.4. Information Resource 7
2.5. Entity Domain 7
2.5.1. Resource-Specific Entity Domain 7
2.5.2. Relationship between Entity and Entity Domain 8
2.5.3. Aggregated Entity Domain 8
2.5.4. Resource-Specific Entity Property 9
2.6. Scope of Property Map 9
2.7. Entity Hierarchy and Property Inheritance 10
3. Protocol Specification: Basic Data Type 10
3.1. Entity Domain 10
3.1.1. Entity Domain Type 10
3.1.2. Entity Domain Name 11
3.1.3. Entity Identifier 11
3.1.4. Hierarchy and Inheritance 12
3.2. Entity Property 12
3.2.1. Entity Property Type 12
3.2.2. Entity Property Name 13
3.3. Information Resource Export 14
3.3.1. Resource-Specific Entity Domain Export 14
3.3.2. Entity Property Mapping Export 14
4. Entity Domain Types 14
4.1. Internet Address Domain Types 15
4.1.1. IPv4 Domain 15
4.1.2. IPv6 Domain 15
4.1.3. Hierarchy and Inheritance of Internet Address Domains 15
4.2. PID Domain 17

4.2.1. Entity Domain Type 17
4.2.2. Domain-Specific Entity Identifiers 17
4.2.3. Hierarchy and Inheritance 17
4.2.4. Relationship To Internet Addresses Domains 17
4.3. Internet Address Properties vs. PID Properties 17
5. Entity Domains and Property Mappings in Information Resources 18
5.1. Network Map Resource 18
5.1.1. Resource-Specific Entity Domain 18
5.1.2. Entity Property Mapping 18
5.2. Endpoint Property Resource 19
5.2.1. Resource-Specific Entity Domain 19
5.2.2. Entity Property Mapping 19
5.3. Property Map Resource 19
6. Property Map 19
6.1. Media Type 19
6.2. HTTP Method 20
6.3. Accept Input Parameters 20
6.4. Capabilities 20
6.5. Uses 20
6.6. Response 20
7. Filtered Property Map 22
7.1. Media Type 22
7.2. HTTP Method 22
7.3. Accept Input Parameters 22
7.4. Capabilities 23
7.5. Uses 23
7.6. Response 23
8. Impact on Legacy ALTO Servers and ALTO Clients 25
8.1. Impact on Endpoint Property Service 25
8.2. Impact on Resource-Specific Properties 25
8.3. Impact on Other Properties 25
9. Examples 25
9.1. Network Map 25
9.2. Property Definitions 26
9.3. Information Resource Directory (IRD) 27
9.4. Property Map Example 29
9.5. Filtered Property Map Example #1 30
9.6. Filtered Property Map Example #2 31
9.7. Filtered Property Map Example #3 32
9.8. Filtered Property Map Example #4 33
10. Security Considerations 34
11. IANA Considerations 35
11.1. application/alto-* Media Types 35
11.2. ALTO Entity Domain Type Registry 36
11.2.1. Consistency Procedure between ALTO Address Type
Registry and ALTO Entity Domain Type Registry . . . 37
11.2.2. ALTO Entity Domain Type Registration Process 38
11.3. ALTO Entity Property Type Registry 39

11.4. ALTO Resource-Specific Entity Domain Registries 40
11.4.1. Network Map 40
11.4.2. Endpoint Property 40
11.5. ALTO Resource Entity Property Mapping Registries 40
11.5.1. Network Map 41
12. Acknowledgments 41
13. Normative References 41
Authors' Addresses 42

1. Introduction

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

First, the EPS allows properties to be associated with only endpoints which are identified by individual communication addresses like IPv4 and IPv6 addresses. It is reasonable to think that collections of endpoints, as defined by CIDRs [RFC4632] or PIDs, may also have properties. Furthermore, recent ALTO use cases show that properties of network flows [RFC7011] and routing elements [RFC7921] are also very useful. Since the EPS cannot be extended to those generic entities, new services, with new request and response messages, would have to be defined for them.

Second, the EPS only allows endpoints identified by global communication addresses. However, many other generic entities like PIDs may not have global identifiers. Even for Internet addresses, there may be some local IP addresses and anycast IP addresses which are also not global unique.

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

This document specifies a new approach for defining and retrieving ALTO properties to address the three limitations:

- This document addresses the first limitation by introducing a generic concept called ALTO Entity which is a generalization of an endpoint to represent a PID, a network element, a cell in a cellular network, or other physical or logical objects used by ALTO. Each entity is included by a collection called ALTO Entity Domain. And each entity domain includes only one type of entities. Thus, each entity domain also has a type to indicate the type of entities in it.

- Additionally, this document addresses the second limitation by using resource-specific entity domains. A resource-specific entity domain is an entity domain exported by an existing ALTO information resource. And a resource-specific entity domain is named by its type and the resource id of the ALTO information resource which exports it. As each resource-specific entity domain name is unique, an entity can be uniquely identified by the name of a resource-specific entity domain and its domain-specific identifier.

- Finally, this document addresses the third limitation by defining two new types of ALTO information resources, namely Property Map (see Section 6) and Filtered Property Map (see Section 7). The former is a GET-mode resource which returns the property values for all entities in some entity domains, 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.

This approach is extensible, because new entity domain types can be defined without revising the protocol specification defined in this document, in the same way that new cost metrics and new endpoint properties can be defined without revising the protocol specification defined in [RFC7285].

This document subsumes the Endpoint Property Service defined in [RFC7285], although that service may be retained for legacy clients (see Section 8).

2. Overview: Basic Concepts

Before we define the specification of unified properties, there are several basic concepts which we need to introduce.

2.1. Entity

The entity concept generalizes the concept of the endpoint defined in Section 2.1 of [RFC7285]. An entity is an object that can be an endpoint and is identified by its network address, but can also be an object that has a defined mapping to a set of one or more network addresses or is even not related to any network address.

Examples of eligible entities are:

- a PID, defined in [RFC7285], that has a provider defined human readable abstract identifier defined by a ALTO network map, which maps a PID to a set of ipv4 and ipv6 addresses;

- an autonomous system (AS), that has an AS number (ASN) as its identifier and maps to a set of ipv4 and ipv6 addresses;

- a region representing a country, that is identified by its country code defined by ISO 3166 and maps to a set of cellular addresses;

- a TCP/IP network flow, that has a server defined identifier consisting of the defining TCP/IP 5-Tuple, , which is an example that all endpoints are entities while not all entities are endpoints;

- a routing element, that is specified in [RFC7921] and includes routing capability information;

- an abstract network element, that has a server defined identifier and represents a network node, link or their aggregation.

2.2. Entity Property

An entity property defines a property of an entity. It is similar to the endpoint property defined by Section 7.1 of [RFC7285], but can be general besides network-aware.

For example,

- an "ipv4" entity may have a property whose value is an Autonomous System (AS) number indicating the AS which this IPv4 address is owned by;

o a "pid" entity may have a property which indicates the central geographical location of endpoints included by it.

2.3. Property Map

An ALTO property map provides a set of properties for a set of entities. These entities may be in different types. For example, an ALTO property map may define the ASN property for both "ipv4" and "ipv6" entities.

2.4. Information Resource

This document uses the same definition of the information resource as defined by [RFC7285]. Each information resource usually has a JSON format representation following a specific schema defined by its media type.

For example, an ALTO network map resource is represented by a JSON objectof type InfoResourceNetworkMap defined by the media type "application/alto-networkmap+json".

2.5. Entity Domain

An entity domain defines a set of entities in the same type. This type is also called the type of this entity domain.

Using entity domains, an ALTO property map can indicate which entities the ALTO client can query to get their properties.

2.5.1. Resource-Specific Entity Domain

To define an entity domain, one naive solution is to enumerate all entities in this entity domain. But it is inefficient when the size of the entity domain is large.

To avoid enumerating all entities, this document introduces an approach called "Resource-Specific Entity Domain" to define entity domains:

Each information resource may define several types of entity domains. And for each type of entity domain, an information resource can define at most one entity domain. For example, an ALTO netowrk map resource can define an IPv4 domain, an IPv6 domain and a pid domain. In this document, these entity domains are called resource-specific entity domains. An ALTO property map only need to indicate which types of entity domain defined by which information resources can be queried, the ALTO client will know which entities are effective to be queried.

2.5.2. Relationship between Entity and Entity Domain

In this document, an entity is owned by exact one entity domain. It requires that when an ALTO client or server references an entity, it must indicate its entity domain explicitly. Even two entities in two different entity domains may reflect to the same physical or logical object, we treat them as different entities.

Because of this rule, although the resource-specific entity domain approach has no ambiguity, it may introduce redundancy.

2.5.3. Aggregated Entity Domain

Two entities in two different resource-specific entity domains may reflect to the same physical or logical object. For example, the IPv4 entity "192.0.2.34" in the IPv4 domain of the network map "netmap1" and the IPv4 entity "192.0.2.34" in the IPv4 domain of the network map "netmap2" should indicate the same Internet endpoint addressed by the IPv4 address "192.0.2.34".

Each entity in each resource-specific entity domain may only have part of properties of its associated physical or logical object. For example, the IPv4 entity in the IPv4 domain of the network map "netmap1" only has the PID property defined by "netmap1"; same to the IPv4 entity in the IPv4 domain of the network map "netmap2". If the ALTO client wants to get the complete properties, using the resource-specific entity domain, the ALTO client has to query the IPv4 entity "192.0.2.34" twice.

To simplify the query process of the ALTO client, this document introduces the concept "Aggregated Entity Domain". An aggregated entity domain defines a union set of entities coming from multiple resource-specific entity domains in the same type. An entity in the aggregated entity domain inherits all properties defined for its associated entity in each associated resource-specific entity domains. For example, the IPv4 entity "192.0.2.34" in the aggregated entity domain between the IPv4 domain of "netmap1" and the IPv4 domain of "netmap2" has PID properties defined by both "netmap1" and "netmap2".

Note that some resource-specific entity domains may not be able to be aggregated even if they are in the same type. For example, a property map "propmap1" may define the "asn" property on both PID domains "netmap1.pid" and "netmap2.pid". But the PID "pid1" in "netmap1.pid" and the PID with the same name in "netmap2.pid" have different "asn" property values. It does not make sense to define an aggregated PID domain between "netmap1.pid" and "netmap2.pid" to provide the "propmap1.asn" property because it is ambiguous.

2.5.4. Resource-Specific Entity Property

According to the example of the aggregated entity domain, an entity may have multiple properties in the same type but associated to different information resources. To distinguish them, this document uses the same approach proposed by Section 10.8.1 of [RFC7285], which is called "Resource-Specific Entity Property".

2.6. Scope of Property Map

Using entity domains to organize entities, an ALTO property map resource actually provides a set of properties for some entity domains. If we ignore the syntax sugar of the aggregated entity domain, we can consider an ALTO property map resource just provides a set of (ri, di) => (ro, po) mappings, where (ri, di) means a resource-specific entity domain of type di defined by the information resource ri, and (ro, po) means a resource-specific entity property po defined by the information resource ro.

For each (ri, di) => (ro, po) mapping, the scope of an ALTO property map resource must be one of cases in the following diagram:

```
                    domain.resource   domain.resource
                    (ri) = r          (ri) = this
                  +-----------------|-----------------+
    prop.resource | Export          | Non-exist       |
    (ro) = r      |                 |                 |
                  +-----------------|-----------------+
    prop.resource | Extend          | Define          |
    (ro) = this   |                 |                 |
                  +-----------------|-----------------+
```

where "this" points to the resulting property map resource, "r" presents an existing ALTO information resource other the resulting property map resource.

- o ri = ro = r ("export" mode): the property map resource just transforms the property mapping di => po defined by r into the unified representation format and exports it. For example: r = "netmap1", di = "ipv4", po = "pid". The property map resource exports the "ipv4 => pid" mapping defined by "netmap1".

- o ri = r, ro = this ("extend" mode): the property map extends properties of entities in the entity domain (r, di) and defines a new property po on them. For example: the property map resource ("this") defines a "geolocation" property on domain "netmap1.pid".

- o ri = ro = this ("define" mode): the property map defines a new intrinsic entity domain and defines property po for each entities in this domain. For example: the property map resource ("this") defines a new entity domain "asn" and defines a property "ipprefixes" on this domain.

- o ri = this, ro = r: in the scope of a property map resource, it does not make sense that another existing ALTO information resource defines a property for this property map resource.

2.7. Entity Hierarchy and Property Inheritance

Enumerating all individual effective entities are inefficient. Some types of entities have the hierarchy format, e.g., cidr, which stand for sets of individual entities. Many entities in the same hierarchical format entity sets may have the same proprety values. To reduce the size of the property map representation, this document introduces an approach called "Property Inheritance". Individual entities can inherit the property from its hierarchical format entity set.

3. Protocol Specification: Basic Data Type

3.1. Entity Domain

3.1.1. Entity Domain Type

An entity domain has a type, which is defined by a string that MUST be no more than 64 characters, and MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), hyphen ("-", U+002D), and low line ("_", U+005F). For example, the strings "ipv4", "ipv6", and "pid" are valid entity domain types.

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

An entity domain type defines the semantics of a type of entity domains. Each entity domain type MUST be registered with the IANA. The format of the entity identifiers (see Section 3.1.3) in that type of entity domains, as well as any hierarchical or inheritance rules (see Section 3.1.4) for those entities, MUST be specified at the same time.

3.1.2. Entity Domain Name

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

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

This document distinguish three types of entity domains: resource-specific entity domains, self-defined entity domain and aggregated entity domains. Their entity domain names are derived as follows.

Each ALTO information resource MAY define a resource-specific entity domain (which could be empty) in a given entity domain type. A resource-specific entity domain is identified by an entity domain name derived as follows. It MUST start with a resource ID using the ResourceID type defined in [RFC7285], followed by the "." separator (U+002E), followed by an EntityDomainType typed string. For example, if an ALTO server provides two network maps "netmap-1" and "netmap-2", they can define two different "pid" domains identified by "netmap-1.pid" and "netmap-2.pid" respectively. To be simplified, in the scope of a specific information resource, the resource-specific entity domain defined by itself can be identified by the "." EntityDomainTyep without the ResourceID.

When the associated information resource of a resource-specific entity domain is the current information resource itself, this resource-specific entity domain is a self-defined entity domain, and its ResourceID SHOULD be ignored from its entity domain name.

Given a set of ALTO information resources, there MAY be an aggregated entity domain in a given entity domain type amongst them. An aggregated entity domain is simply identified by its entity domain type. For example, given two network maps "net-map-1" and "net-map-2", "ipv4" and "ipv6" identify two aggregated Internet address entity domains (see Section 4.1) between them.

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

3.1.3. Entity Identifier

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

EntityID ::= EntityDomainName ':' DomainTypeSpecificEntityID

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

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

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

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

3.1.4. Hierarchy and Inheritance

To make the representation efficient, some types of entity domains MAY allow the ALTO client/server to use a hierarchical format entity identifier to represent a block of individual entities. e.g., In an IPv4 domain "net1.ipv4", a cidr "net1.ipv4:192.0.2.0/26" represents 64 individual IPv4 entities. In this case, the corresponding property inheritance rule MUST be defined for the entity domain type. The hierarchy and inheritance rule MUST have no ambiguity.

3.2. Entity Property

Each entity property has a type to indicate the encoding and the semantics of the value of this entity property, and has a name to be identified. One entity MAY have multiple properties in the same type.

3.2.1. Entity Property Type

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

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

To distinguish with the endpoint property type, the entity property type has the following features.

- Some entity property types may be applicable to entities in only particular types of entity domains, not all. For example, the "pid" property is not applicable to entities in a "pid" typed entity domain, but is applicable to entities in the "ipv4" or "ipv6" domains.

- The intended semantics of the value of a entity property may also depend on the the entity domain type of this entity. 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, identified by an address in the "ipv4" or "ipv6" entity domain, the property defines the host's location. However, when applied to an entity in a "pid" domain, the property would indicate the location of the center of all hosts in this "pid" entity.

3.2.2. Entity Property Name

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

```
EntityPropertyName ::= [ ResourceID ] '.' EntityPropertyType
```

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

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

When the associated information resource of the entity property is the current information resource itself, the ResourceID in the property name SHOULD be ignored. For example, the ".asn" property of an "ipv4" entity indicates the AS number of the AS which this IPv4 address is owned by.

3.3. Information Resource Export

Each information resource MAY export a set of entity domains and entity property mappings.

3.3.1. Resource-Specific Entity Domain Export

Each type of information resource MAY export several types of entity domains. For example, a network map resource defines a "pid" domain, a "ipv4" domain and a "ipv6" domain (which may be empty).

When a new ALTO information resource type is registered, if this type of information resource can export an existing type of entity domain, the corresponding document MUST define how to export such type of entity domain from such type of information resource.

When a new entity domain type is defined, if an existing type of information resource can export an entity domain in this entity domain type, the corresponding document MUST define how to export such type of entity domain from such type of information resource.

3.3.2. Entity Property Mapping Export

For each entity domain which could be exported by an information resource, this information resource MAY also export some mapping from this entity domain to some entity property. For example, a network map resource can map an "ipv4" entity to its "pid" property.

When a new ALTO information resource type is registered, if this type of information resource can export an entity domain in an existing entity domain type, and map entities in this entity domain to an existing type of entity property, the corresponding document MUST define how to export such type of an entity property.

When a new ALTO entity domain type or a new entity property type is defined, if an existing type of resource can export an entity domain in this entity domain type, and map entities in this entity domain to this type of entity property, the corresponding document MUST define how to export such type of an entity property.

4. Entity Domain Types

This document defines three entity domain types. The definition of each entity domain type below includes the following: (1) entity domain type name, (2) entity domain-specific entity identifiers, and (3) hierarchy and inheritance semantics. Since a global entity domain type defines a single global entity domain, we say entity domain instead of entity domain type.

4.1. Internet Address Domain Types

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

4.1.1. IPv4 Domain

4.1.1.1. Entity Domain Type

ipv4

4.1.1.2. Domain-Specific Entity Identifiers

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.

4.1.2. IPv6 Domain

4.1.2.1. Entity Domain Type

ipv6

4.1.2.2. Domain-Specific Entity Identifiers

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.

4.1.3. Hierarchy and Inheritance of Internet Address Domains

Both Internet address domains allow property values to be inherited. Specifically, if a property P is not defined for a specific Internet address I, but P is defined for 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.

4.2. 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.

4.2.1. Entity Domain Type

pid

4.2.2. Domain-Specific Entity Identifiers

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

4.2.3. Hierarchy and Inheritance

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

4.2.4. 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".

4.3. 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.

5. Entity Domains and Property Mappings in Information Resources

5.1. Network Map Resource

The ALTO network map resource defined by the media type "application/alto-networkmap+json" exports the following types of entity domains and entity property mappings.

5.1.1. Resource-Specific Entity Domain

An ALTO network map resource defines a "pid" domain, an "ipv4" domain and an "ipv6" domain by follows:

- o The defined "pid" domain includes all PIDs in keys of the "network-map" object.
- o The defined "ipv4" domain includes all IPv4 addresses appearing in the "ipv4" field of the endpoint address group of each PID.
- o The defined "ipv6" domain includes all IPv6 addresses appearing in the "ipv6" field of the endpoint address group of each PID.

5.1.2. Entity Property Mapping

For each of the preceding entity domains, an ALTO network map resource provides the properties mapping as follows:

ipv4 -> pid: An "networkmap" typed resource can map an "ipv4" entity to a "pid" property whose value is a PID defined by this "networkmap" resource and including the IPv4 address of this entity.

ipv6 -> pid: An "networkmap" typed resource can map an "ipv6" entity to a "pid" property whose value is a PID defined by this "networkmap" resource and including the IPv6 address of this entity.

5.2. Endpoint Property Resource

The ALTO endpoint property resource defined by the media type "application/alto-endpointprop+json" exports the following types of entity domains and entity property mappings.

5.2.1. Resource-Specific Entity Domain

An ALTO endpoint property resource defined an "ipv4" domain and an "ipv6" domain by follows:

- o The defined "ipv4" domain includes all IPv4 addresses appearing in keys of the "endpoint-properties" object.

- o The defined "ipv6" domain includes all IPv6 addresses appearing in keys of the "endpoint-properties" object.

5.2.2. Entity Property Mapping

For each of the preceding entity domains, an ALTO endpoint property resource exports the properties mapping from it to each supported global endpoint property. The property value is the corresponding global endpoint property value in the "endpiont-properties" object.

5.3. Property Map Resource

To avoid the nested reference and its potential complexity, this document does not specify the export rule of resource-specific entity domain and entity property mapping for the ALTO property map resource defined by the media type "application/alto-propmap+json" (see Section 6.1).

6. 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 9.4 gives an example of a property map request and its response.

6.1. Media Type

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

6.2. HTTP Method

The property map is requested using the HTTP GET method.

6.3. Accept Input Parameters

None.

6.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities:

```
object {
  EntityPropertyMapping mappings;
} PropertyMapCapabilities;

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

with fields:

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

6.5. Uses

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

6.6. Response

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

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

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

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

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

Specifically, a PropertyMapData object has one member for each entity in the property map. The entity's properties are encoded in the corresponding EntityProps object. EntityProps encodes one name/value pair for each property, where the property names are encoded as strings of type PropertyName. A protocol implementation SHOULD assume that the property value is either a JSONString or a JSON "null" value, and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

For each entity in the property map:

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

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

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

7. 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 9.5, Section 9.6, Section 9.7 and Section 9.8 give examples of filtered property map requests and responses.

7.1. Media Type

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

7.2. HTTP Method

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

7.3. 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 {
  EntityID             entities<1..*>;
  EntityPropertyName   properties<1..*>;
} ReqFilteredPropertyMap;
```

with fields:

entities: List of entity identifiers for which the specified properties are to be returned. The ALTO server MUST interpret entries appearing multiple times as if they appeared only once. The domain of each entity MUST be included in the list of entity domains in this resource's "capabilities" field (see Section 7.4).

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 7.4). 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.

7.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities, as defined in Section 6.4.

7.5. Uses

Same to the "uses" field of the Property Map resource (see Section 6.5).

7.6. 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:

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

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

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

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

  If an entity identifier 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 identifier.

- o A property name in "properties" in the request is invalid if this property name is not defined in the "properties" 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 6.6), except that:

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

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

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

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:

- o 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.

- o 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.

- o 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.

8. Impact on Legacy ALTO Servers and ALTO Clients

8.1. 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.

8.2. 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 defines the similar categories for entity properties. The difference is that there is no "global" entity properties but the "self-defined" entity properties as the special case of the "resource-specific" entity properties instead.

8.3. Impact on Other Properties

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

9. Examples

9.1. 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 Default Network Map

And another simple alternative network map:

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

Figure 4: Example Alternative Network Map

9.2. 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 5: Example Property Values for Internet Address Domains

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

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

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

Note that "-" means the value of the property for the entity is "undefined". So the entity would inherit a value for this property by the inheritance rule if possible. For example, the value of the "ISP" property for "ipv4:192.0.2.0" 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.

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

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

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

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

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

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


Figure 8: Example IRD

9.4. 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

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

9.5. Filtered Property Map Example #1

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

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

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

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

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

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


9.6. 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

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

9.7. 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" : [ "default-network-map.pid" ]
}

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

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

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

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

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

10. 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.

11. IANA Considerations

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

11.1. application/alto-* Media Types

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

Type	Subtype	Specification
application	alto-propmap+json	Section 6.1
application	alto-propmapparams+json	Section 7.3

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).

11.2. ALTO Entity Domain Type Registry

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

Identifier	Entity Identifier Encoding	Hierarchy & Inheritance
ipv4	See Section 4.1.1	See Section 4.1.3
ipv6	See Section 4.1.2	See Section 4.1.3
pid	See Section 4.2	None

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.

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

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

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

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

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

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

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

- o Whether the candidate entity identifier of the type 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 type is registered, the consistency with the ALTO Address Type Registry MUST be ensured by the following procedure:

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

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

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

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

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

  + Use the new ALTO entity domain type identifier to register a new ALTO entity domain type in the ALTO Entity Domain Type Registry following Section 11.2.2 of this document.

* If no (e.g., pid name, ane name or country code): Proceed with the ALTO Entity Domain Type registration as described in Section 11.2.2.

11.2.2. ALTO Entity Domain Type Registration Process

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

Registered ALTO entity domain type identifiers MUST conform to the syntactical requirements specified in Section 3.1.2. 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:

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

o Entity Identifier Encoding: The procedure for encoding the identifier of an entity of the registered type as an EntityID (see Section 3.1.3). If corresponding entity identifiers of an entity domain match a known "network" address type, the Entity Identifier Encoding of this domain identifier MUST include both Address Encoding and Prefix Encoding of the same identifier registered in the ALTO Address Type Registry [RFC7285]. For the purpose of defining properties, an individual entity identifier and the corresponding full-length prefix MUST be considered aliases for the same entity.

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

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

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

- o Security Considerations: In some usage scenarios, entity identifiers carried in ALTO Protocol messages may reveal information about an ALTO client or an ALTO service provider. Applications and ALTO service providers using addresses of the registered type should be 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.

11.3. 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 3.2.1. 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
pid	See Section 7.1.1 of [RFC7285]

Table 3: ALTO Entity Property Types.

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

- o Identifier: The unique id for the desired ALTO entity property type. The format MUST be as defined in Section 3.2.1 of this document. It includes the information of the applied ALTO entity domain and the property name.

- o 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.

This document requests registration of the identifier "pid", as shown in Table 3.

11.4. ALTO Resource-Specific Entity Domain Registries

11.4.1. Network Map

Media-type: application/alto-networkmap+json

Entity Domain Type	Intended Semantics
ipv4	See Section 5.1.1
ipv6	See Section 5.1.1
pid	See Section 5.1.1

Table 4: ALTO Network Map Resource-Specific Entity Domain.

11.4.2. Endpoint Property

Media-type: application/alto-endpointprop+json

Entity Domain Type	Intended Semantics
ipv4	See Section 5.2.1
ipv6	See Section 5.2.1

Table 5: ALTO Endpoint Property Resource-Specific Entity Domain.

11.5. ALTO Resource Entity Property Mapping Registries

11.5.1. Network Map

Media-type: application/alto-networkmap+json

Mapping Descriptor	Entity Domain Type	Property Type	Intended Semantics
ipv4 -> pid	ipv4	pid	See Section 5.1.2
ipv6 -> pid	ipv6	pid	See Section 5.1.2

Table 6: ALTO Network Map Entity Property Mapping.

12. Acknowledgments

The authors would like to thank discussions with Kai Gao, Qiao Xiang, Shawn Lin, Xin Wang, Danny Perez, and Vijay Gurbani. The authors thank Dawn Chen (Tongji University), and Shenshen Chen (Tongji/Yale University) for their contributions to earlier drafts.

13. 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>.

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

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

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

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

Authors' Addresses

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

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

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

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

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

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

Jingxuan Jensen Zhang
Tongji University
4800 Caoan Road
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

Kai Gao
Sichuan University
Chengdu 610000
China

Email: kaigao@scu.edu.cn