

ALTO & CDNI WGs
Internet-Draft
Intended status: Standards Track
Expires: 21 August 2022

J. Seedorf
HFT Stuttgart - Univ. of Applied Sciences
Y. Yang
Yale University
K. Ma
Ericsson
J. Peterson
NeuStar
J. Zhang
Tongji University
17 February 2022

Content Delivery Network Interconnection (CDNI) Request Routing: CDNI
Footprint and Capabilities Advertisement using ALTO
draft-ietf-alto-cdni-request-routing-alto-22

Abstract

The Content Delivery Networks Interconnection (CDNI) framework in RFC 6707 defines a set of protocols to interconnect CDNs to achieve multiple goals, including extending the reach of a given CDN. A CDNI Request Routing Footprint & Capabilities Advertisement interface (FCI) is needed to achieve the goals of a CDNI. RFC 8008 defines the FCI semantics and provides guidelines on the FCI protocol, but the exact protocol is not specified. This document defines a new Application-Layer Traffic Optimization (ALTO) service, called "CDNI Advertisement Service", that provides an implementation of the FCI, following the guidelines defined in RFC 8008.

Status of This Memo

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

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

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

This Internet-Draft will expire on 21 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology and Background	4
2.1. Terminology	4
2.2. Semantics of FCI Advertisement	5
2.3. ALTO Background and Benefits	6
3. CDNI Advertisement Service	8
3.1. Media Type	9
3.2. HTTP Method	9
3.3. Accept Input Parameters	9
3.4. Capabilities	9
3.5. Uses	9
3.6. Response	9
3.7. Examples	12
3.7.1. IRD	12
3.7.2. A Basic Example	15
3.7.3. Incremental Updates	17
4. CDNI Advertisement Service using ALTO Network Map	18
4.1. Network Map Footprint Type: altopid	19
4.2. Examples	19
4.2.1. ALTO Network Map for CDNI Advertisements	19
4.2.2. ALTO PID Footprints in CDNI Advertisements	20
4.2.3. Incremental Updates	21
5. Filtered CDNI Advertisement using CDNI Capabilities	23
5.1. Media Type	23
5.2. HTTP Method	23
5.3. Accept Input Parameters	23
5.4. Capabilities	24
5.5. Uses	24
5.6. Response	24
5.7. Examples	25
5.7.1. A Basic Example	25
5.7.2. Incremental Updates	26

6.	Query Footprint Properties using ALTO Property Map Service	28
6.1.	Representing Footprint Objects as Property Map Entities	28
6.1.1.	ASN Domain	29
6.1.2.	COUNTRYCODE Domain	30
6.2.	Representing CDNI Capabilities as Property Map Entity Properties	30
6.2.1.	Defining Information Resource Media Type for Property Type cdni-capabilities	30
6.2.2.	Intended Semantics of Property Type cdni-capabilities	31
6.3.	Examples	31
6.3.1.	Property Map	31
6.3.2.	Filtered Property Map	32
6.3.3.	Incremental Updates	34
7.	IANA Considerations	35
7.1.	application/alto-cdni+json Media Type	35
7.2.	application/alto-cdnifilter+json Media Type	36
7.3.	CDNI Metadata Footprint Type Registry	38
7.4.	ALTO Entity Domain Type Registry	38
7.5.	ALTO Entity Property Type Registry	39
8.	Security Considerations	39
9.	References	40
9.1.	Normative References	41
9.2.	Informative References	42
	Acknowledgments	43
	Contributors	43
	Authors' Addresses	43

1. Introduction

The ability to interconnect multiple content delivery networks (CDNs) has many benefits, including increased coverage, capability, and reliability. The Content Delivery Networks Interconnection (CDNI) framework [RFC6707] defines four interfaces to interconnect CDNs: (1) the CDNI Request Routing Interface, (2) the CDNI Metadata Interface, (3) the CDNI Logging Interface, and (4) the CDNI Control Interface.

Among these four interfaces, the CDNI Request Routing Interface provides key functions, as specified in [RFC6707]: "The CDNI Request Routing interface enables a Request Routing function in an Upstream CDN to query a Request Routing function in a Downstream CDN to determine if the Downstream CDN is able (and willing) to accept the delegated Content Request. It also allows the Downstream CDN to control what should be returned to the User Agent in the redirection message by the upstream Request Routing function." At a high level, the scope of the CDNI Request Routing Interface, therefore, contains two main tasks: (1) determining if the dCDN (downstream CDN) is

willing to accept a delegated content request, and (2) redirecting the content request coming from a uCDN (upstream CDN) to the proper entry point or entity in the dCDN.

Correspondingly, the Request Routing Interface is broadly divided into two functionalities: (1) the CDNI Footprint & Capabilities Advertisement interface (FCI) defined in [RFC8008], and (2) the CDNI Request Routing Redirection interface (RI) defined in [RFC7975]. This document focuses on the first functionality (CDNI FCI).

Specifically, CDNI FCI allows both an advertisement from a dCDN to a uCDN (push) and a query from a uCDN to a dCDN (pull) so that the uCDN knows whether it can redirect a particular user request to that dCDN.

A key component in defining CDNI FCI is defining objects describing the footprints and capabilities of a dCDN. Such objects are already defined in Section 5 of [RFC8008]. However, no protocol is defined to transport and update such objects between a uCDN and a dCDN.

To define such a protocol, this document specifies an extension of the Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol by introducing a new ALTO service called "CDNI Advertisement Service".

Section 2.3 discusses the benefits in using ALTO as a transport protocol.

2. Terminology and Background

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

The design of CDNI FCI transport using ALTO assumes an understanding of both FCI semantics and ALTO. Hence, this document starts with a non-normative review for both.

2.1. Terminology

The document uses the CDNI terms defined in [RFC6707], [RFC8006] and [RFC8008]. Also, the document uses the ALTO terms defined in [RFC7285] and [I-D.ietf-alto-unified-props-new]. This document uses the following abbreviations:

* ALTO: Application-Layer Traffic Optimization

- * ASN: Autonomous System Number
- * CDN: Content Delivery Network
- * CDNI: CDN Interconnection
- * dCDN: Downstream CDN
- * FCI: CDNI FCI, CDNI Request Routing Footprint & Capabilities Advertisement interface
- * IRD: Information Resource Directory in ALTO
- * PID: Provider-defined Identifier in ALTO
- * uCDN: Upstream CDN

2.2. Semantics of FCI Advertisement

[RFC8008] defines the semantics of CDNI FCI, provides guidance on what Footprint and Capabilities mean in a CDNI context, and specifies the requirements on the CDNI FCI transport protocol. The definitions in [RFC8008] depend on [RFC8006]. Below is a non-normative review of key related points of [RFC8008] and [RFC8006]. For detailed information and normative specification, the reader should refer to these two RFCs.

- * Multiple types of mandatory-to-implement footprints (i.e., ipv4cidr, ipv6cidr, asn, and countrycode) are defined in [RFC8006]. A "Set of IP-prefixes" can contain both full IP addresses (i.e., a /32 for IPv4 or a /128 for IPv6) and IP prefixes with an arbitrary prefix length. There must also be support for multiple IP address versions, i.e., IPv4 and IPv6, in such a footprint.
- * Multiple initial types of capabilities are defined in [RFC8008] including (1) Delivery Protocol, (2) Acquisition Protocol, (3) Redirection Mode, (4) Capabilities related to CDNI Logging, and (5) Capabilities related to CDNI Metadata. They are required in all cases and, therefore, considered as mandatory-to-implement capabilities for all CDNI FCI implementations.
- * Footprint and capabilities are defined together and cannot be interpreted independently from each other. Specifically, [RFC8008] integrates footprint and capabilities with an approach of "capabilities with footprint restrictions", by expressing capabilities on a per footprint basis.

- * Specifically, for all mandatory-to-implement footprint types, footprints can be viewed as constraints for delegating requests to a dCDN: A dCDN footprint advertisement tells the uCDN the limitations for delegating a request to the dCDN. For IP prefixes or Autonomous System Numbers (ASNs), the footprint signals to the uCDN that it should consider the dCDN a candidate only if the IP address of the request routing source falls within the prefix set or ASN, respectively. The CDNI specifications do not define how a given uCDN determines what address ranges are in a particular ASN. Similarly, for country codes, a uCDN should only consider the dCDN a candidate if it covers the country of the request routing source. The CDNI specifications do not define how a given uCDN determines the country of the request routing source. Different types of footprint constraints can be combined together to narrow the dCDN candidacy, i.e., the uCDN should consider the dCDN a candidate only if the request routing source satisfies all the types of footprint constraints in the advertisement.
- * Given that a large part of Footprint and Capabilities Advertisement may happen in contractual agreements, the semantics of CDNI Footprint and Capabilities advertisement refers to answering the following question: what exactly still needs to be advertised by the CDNI FCI? For instance, updates about temporal failures of part of a footprint can be useful information to convey via the CDNI FCI. Such information would provide updates on information previously agreed in contracts between the participating CDNs. In other words, the CDNI FCI is a means for a dCDN to provide changes/updates regarding a footprint and/or capabilities that it has previously agreed to serve in a contract with a uCDN. Hence, server push and incremental encoding will be necessary techniques.

2.3. ALTO Background and Benefits

Application-Layer Traffic Optimization (ALTO) [RFC7285] defines an approach for conveying network layer (topology) information to "guide" the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. Usually, it is assumed that an ALTO server conveys information that these applications cannot measure or have difficulty measuring themselves [RFC5693].

Originally, ALTO was motivated by optimizing cross-ISP traffic generated by P2P applications [RFC5693]. However, ALTO can also be used for improving the request routing in CDNs. In particular, Section 5 of [RFC7971] explicitly mentions ALTO as a candidate protocol to improve the selection of a CDN surrogate or origin.

The following reasons make ALTO a suitable candidate protocol for dCDN selection as part of CDNI request routing and, in particular, for an FCI protocol:

- * **Application Layer-oriented:** ALTO is a protocol specifically designed to improve application layer traffic (and application layer connections among hosts on the Internet) by providing additional information to applications that these applications could not easily retrieve themselves. This matches the need of CDNI, where a uCDN wants to improve application layer CDN request routing by using information (provided by a dCDN) that the uCDN could not easily obtain otherwise. Hence, ALTO can help a uCDN to select a proper dCDN by first providing dCDNs' capabilities as well as footprints (see Section 3) and then providing costs of surrogates in a dCDN by ALTO cost maps.
- * **Security:** The identification between uCDNs and dCDNs is an important requirement (see Section 8). ALTO maps can be signed and hence provide inherent origin protection. Please see Section 15.1.2 of [RFC7285] for detailed protection strategies.
- * **RESTful design:** The ALTO protocol has undergone extensive revisions in order to provide a RESTful design regarding the client-server interaction specified by the protocol. It is flexible and extensible enough to handle existing and potential future data formats defined by CDNI. It can provide the consistent client-server interaction model for other existing CDNI interfaces or potential future extensions and therefore reduce the learning cost for both users and developers, although they are not in the scope of this document. A CDNI FCI interface based on ALTO would inherit this RESTful design. Please see Section 3.
- * **Error-handling:** The ALTO protocol provides extensive error-handling in the whole request and response process (see Section 8.5 of [RFC7285]). A CDNI FCI interface based on ALTO would inherit this extensive error-handling framework. Please see Section 5.
- * **Map Service:** The semantics of an ALTO network map is an exact match for the needed information to convey a footprint by a dCDN, in particular, if such a footprint is being expressed by IP-prefix ranges. Please see Section 4.
- * **Filtered Map Service:** The ALTO map filtering service would allow a uCDN to query only for parts of an ALTO map. For example, the ALTO filtered property map service can enable a uCDN to query properties of a part of footprints efficiently. Please see Section 6.

- * Server-initiated notifications and incremental updates: When the footprint or the capabilities of a dCDN change (i.e., unexpectedly from the perspective of a uCDN), server-initiated notifications would enable a dCDN to inform a uCDN about such changes directly. Consider the case where - due to failure - part of the footprint of the dCDN is not functioning, i.e., the CDN cannot serve content to such clients with reasonable QoS. Without server-initiated notifications, the uCDN might still use a recent network and cost map from the dCDN, and therefore redirect requests to the dCDN which it cannot serve. Similarly, the possibility for incremental updates would enable efficient conveyance of the aforementioned (or similar) status changes by the dCDN to the uCDN. The newest design of ALTO supports server pushed incremental updates [RFC8895].
- * Content availability on hosts: A dCDN might want to express CDN capabilities in terms of certain content types (e.g., codecs/formats, or content from certain content providers). ALTO Entity Property Map [I-D.ietf-alto-unified-props-new] would enable a dCDN to make such information available to a uCDN. This would enable a uCDN to access whether a dCDN has the capabilities for a given type of content requested.
- * Resource availability on hosts or links: The capabilities on links (e.g., maximum bandwidth) or caches (e.g., average load) might be useful information for a uCDN for optimized dCDN selection. For instance, if a uCDN receives a streaming request for content with a certain bitrate, it needs to know if it is likely that a dCDN can fulfill such stringent application-level requirements (i.e., can be expected to have enough consistent bandwidth) before it redirects the request. In general, if ALTO could convey such information via ALTO Entity Property Map [I-D.ietf-alto-unified-props-new], it would enable more sophisticated means for dCDN selection with ALTO. ALTO Path Vector Extension [I-D.ietf-alto-path-vector] is designed to allow ALTO clients to query information such as capacity regions for a given set of flows.

3. CDNI Advertisement Service

The ALTO protocol relies upon the ALTO Information Service framework which consists of multiple services. All ALTO services are "provided through a common transport protocol, messaging structure and encoding, and transaction model" [RFC7285]. The ALTO protocol specification defines multiple initial services, e.g., the ALTO network map service and cost map service.

This document defines a new ALTO service, called "CDNI Advertisement Service", which conveys JSON [RFC8259] objects of media type "application/alto-cdni+json". These JSON objects are used to transport BaseAdvertisementObject objects defined in [RFC8008]. This document specifies how to transport such BaseAdvertisementObject objects via the ALTO protocol with the ALTO "CDNI Advertisement Service". Similar to other ALTO services, this document defines the ALTO information resource for the "CDNI Advertisement Service" as follows.

Note that the encoding of BaseAdvertisementObject reuses the one defined in [RFC8008] and therefore also follows the recommendations of I-JSON (Internet JSON) [RFC7493], which is required by [RFC8008].

3.1. Media Type

The media type of the CDNI Advertisement resource is "application/alto-cdni+json" (see Section 7).

3.2. HTTP Method

A CDNI Advertisement resource is requested using the HTTP GET method.

3.3. Accept Input Parameters

There are no applicable Accept Input parameters.

3.4. Capabilities

There are no applicable capabilities.

3.5. Uses

The "uses" field MUST NOT appear unless the CDNI Advertisement resource depends on other ALTO information resources. If the CDNI Advertisement resource has dependent resources, the resource IDs of its dependent resources MUST be included into the "uses" field. This document only defines one potential dependent resource for the CDNI Advertisement resource. See Section 4 for details of when and how to use it. Future documents may extend the CDNI Advertisement resource and allow other dependent resources.

3.6. Response

The "meta" field of a CDNI Advertisement response MUST include the "vtag" field defined in Section 10.3 of [RFC7285]. This field provides the version of the retrieved CDNI FCI resource.

If a CDNI Advertisement response depends on other ALTO information resources, it MUST include the "dependent-vtags" field, whose value is an array to indicate the version tags of the resources used, where each resource is specified in "uses" of its Information Resource Directory (IRD) entry.

The data component of an ALTO CDNI Advertisement response is named "cdni-advertisement", which is a JSON object of type CDNIAdvertisementData:

```
object {  
  CDNIAdvertisementData cdni-advertisement;  
} InfoResourceCDNIAdvertisement : ResponseEntityBase;  
  
object {  
  BaseAdvertisementObject capabilities-with-footprints<0..*>;  
} CDNIAdvertisementData;
```

Specifically, a CDNIAdvertisementData object is a JSON object that includes only one property named "capabilities-with-footprints", whose value is an array of BaseAdvertisementObject objects. It provides capabilities with footprint restrictions for uCDN to decide the dCDN selection. If the value of this property is an empty array, it means the corresponding dCDN cannot provide any mandatory-to-implement CDNI capabilities for any footprints.

The syntax and semantics of BaseAdvertisementObject are well defined in Section 5.1 of [RFC8008]. A BaseAdvertisementObject object includes multiple properties, including capability-type, capability-value, and footprints, where footprints are defined in Section 4.2.2.2 of [RFC8006].

To be self-contained, below is an equivalent specification of BaseAdvertisementObject described in the ALTO-style notation (see Section 8.2 of [RFC7285]). As mentioned above, the normative specification of BaseAdvertisementObject is in [RFC8008].

```
object {  
  JSONString capability-type;  
  JSONValue capability-value;  
  Footprint footprints<0..*>;  
} BaseAdvertisementObject;  
  
object {  
  JSONString footprint-type;  
  JSONString footprint-value<1..*>;  
} Footprint;
```

For each BaseAdvertisementObject, the ALTO client MUST interpret footprints appearing multiple times as if they appeared only once. If footprints in a BaseAdvertisementObject is null or empty or not appearing, the ALTO client MUST understand that the capabilities in this BaseAdvertisementObject have the "global" coverage, i.e., the corresponding dCDN can provide them for any request routing source.

Note: Further optimization of BaseAdvertisement objects to effectively provide the advertisement of capabilities with footprint restrictions is certainly possible. For example, these two examples below both describe that the dCDN can provide capabilities ["http/1.1", "https/1.1"] for the same footprints. However, the latter one is smaller in its size.

EXAMPLE 1

```
{
  "meta": {...},
  "cdni-advertisement": {
    "capabilities-with-footprints": [
      {
        "capability-type": "FCI.DeliveryProtocol",
        "capability-value": {
          "delivery-protocols": [
            "http/1.1"
          ]
        },
        "footprints": [
          <Footprint objects>
        ]
      },
      {
        "capability-type": "FCI.DeliveryProtocol",
        "capability-value": {
          "delivery-protocols": [
            "https/1.1"
          ]
        },
        "footprints": [
          <Footprint objects>
        ]
      }
    ]
  }
}
```

EXAMPLE 2

```
{
```

```
"meta": {...},
"cdni-advertisement": {
  "capabilities-with-footprints": [
    {
      "capability-type": "FCI.DeliveryProtocol",
      "capability-value": {
        "delivery-protocols": [
          "https/1.1",
          "http/1.1"
        ]
      },
      "footprints": [
        <Footprint objects>
      ]
    }
  ]
}
```

Since such optimizations are not required for the basic interconnection of CDNs, the specifics of such mechanisms are outside the scope of this document.

This document only requires the ALTO server to provide the initial FCI-specific CDNI Payload Types defined in [RFC8008] as the mandatory-to-implement CDNI capabilities.

3.7. Examples

3.7.1. IRD

Below is the IRD of a simple, example ALTO server. The server provides both base ALTO information resources (e.g., network maps) and CDNI FCI related information resources (e.g., CDNI Advertisement resources), demonstrating a single, integrated environment.

Specifically, the IRD announces nine information resources as follows:

- * two network maps,
- * one CDNI Advertisement resource without dependency,
- * one CDNI Advertisement resource depending on a network map,
- * one filtered CDNI Advertisement resource to be defined in Section 5,

- * one property map including "cdni-capabilities" as its entity property,
- * one filtered property map including "cdni-capabilities" and "pid" as its entity properties, and
- * two update stream services:
 - one for updating CDNI Advertisement resources,
 - one for updating property maps

```
GET /directory HTTP/1.1
Host: alto.example.com
Accept: application/alto-directory+json,application/alto-error+json
```

```
HTTP/1.1 200 OK
Content-Length: 3531
Content-Type: application/alto-directory+json
```

```
{
  "meta": {
    "default-alto-network-map": "my-default-network-map"
  },
  "resources": {
    "my-default-network-map": {
      "uri": "https://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },
    "my-eu-netmap": {
      "uri": "https://alto.example.com/myeunetmap",
      "media-type": "application/alto-networkmap+json"
    },
    "my-default-cdnifci": {
      "uri": "https://alto.example.com/cdnifci",
      "media-type": "application/alto-cdni+json"
    },
    "my-cdnifci-with-pid-footprints": {
      "uri": "https://alto.example.com/networkcdnifci",
      "media-type": "application/alto-cdni+json",
      "uses": [ "my-eu-netmap" ]
    },
    "my-filtered-cdnifci": {
      "uri": "https://alto.example.com/cdnifci/filtered",
      "media-type": "application/alto-cdni+json",
      "accepts": "application/alto-cdnifilter+json"
    },
    "cdnifci-property-map": {
```

```
"uri": "https://alto.example.com/propmap/full/cdnifci",
"media-type": "application/alto-propmap+json",
"uses": [ "my-default-cdni" ],
"capabilities": {
  "mappings": {
    "ipv4": [ "my-default-cdni.cdni-capabilities" ],
    "ipv6": [ "my-default-cdni.cdni-capabilities" ],
    "countrycode": [
      "my-default-cdni.cdni-capabilities" ],
    "asn": [ "my-default-cdni.cdni-capabilities" ]
  }
},
"filtered-cdnifci-property-map": {
  "uri": "https://alto.example.com/propmap/lookup/cdnifci-pid",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "uses": [ "my-default-cdni", "my-default-network-map" ],
  "capabilities": {
    "mappings": {
      "ipv4": [ "my-default-cdni.cdni-capabilities",
        "my-default-network-map.pid" ],
      "ipv6": [ "my-default-cdni.cdni-capabilities",
        "my-default-network-map.pid" ],
      "countrycode": [
        "my-default-cdni.cdni-capabilities" ],
      "asn": [ "my-default-cdni.cdni-capabilities" ]
    }
  }
},
"update-my-cdni-fci": {
  "uri": "https://alto.example.com/updates/cdnifci",
  "media-type": "text/event-stream",
  "accepts": "application/alto-updatestreamparams+json",
  "uses": [
    "my-default-network-map",
    "my-eu-netmap",
    "my-default-cdnifci",
    "my-filtered-cdnifci",
    "my-cdnifci-with-pid-footprints"
  ],
  "capabilities": {
    "incremental-change-media-types": {
      "my-default-network-map": "application/json-patch+json",
      "my-eu-netmap": "application/json-patch+json",
      "my-default-cdnifci":
        "application/merge-patch+json,application/json-patch+json",
      "my-filtered-cdnifci":
```

```

        "application/merge-patch+json,application/json-patch+json",
        "my-cdnifci-with-pid-footprints":
        "application/merge-patch+json,application/json-patch+json"
    }
}
},
"update-my-props": {
    "uri": "https://alto.example.com/updates/properties",
    "media-type": "text/event-stream",
    "uses": [
        "cdnifci-property-map",
        "filtered-cdnifci-property-map"
    ],
    "capabilities": {
        "incremental-change-media-types": {
            "cdnifci-property-map":
            "application/merge-patch+json,application/json-patch+json",
            "filtered-cdnifci-property-map":
            "application/merge-patch+json,application/json-patch+json"
        }
    }
}
}
}
}

```

3.7.2. A Basic Example

This basic example demonstrates a simple CDNI Advertisement resource, which does not depend on other resources. There are three BaseAdvertisementObjects in this resource and these objects' capabilities are http/1.1 delivery protocol, [http/1.1, https/1.1] delivery protocol, and https/1.1 acquisition protocol, respectively.

```

GET /cdnifci HTTP/1.1
Host: alto.example.com
Accept: application/alto-cdni+json,application/alto-error+json

```

```

HTTP/1.1 200 OK
Content-Length: 1411
Content-Type: application/alto-cdni+json

```

```

{
  "meta": {
    "vtag": {
      "resource-id": "my-default-cdnifci",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },

```

```
"cdni-advertisement": {
  "capabilities-with-footprints": [
    {
      "capability-type": "FCI.DeliveryProtocol",
      "capability-value": {
        "delivery-protocols": [
          "http/1.1"
        ]
      },
      "footprints": [
        {
          "footprint-type": "ipv4cidr",
          "footprint-value": [ "192.0.2.0/24" ]
        },
        {
          "footprint-type": "ipv6cidr",
          "footprint-value": [ "2001:db8::/32" ]
        }
      ]
    },
    {
      "capability-type": "FCI.DeliveryProtocol",
      "capability-value": {
        "delivery-protocols": [
          "https/1.1",
          "http/1.1"
        ]
      },
      "footprints": [
        {
          "footprint-type": "ipv4cidr",
          "footprint-value": [ "198.51.100.0/24" ]
        }
      ]
    },
    {
      "capability-type": "FCI.AcquisitionProtocol",
      "capability-value": {
        "acquisition-protocols": [
          "https/1.1"
        ]
      },
      "footprints": [
        {
          "footprint-type": "ipv4cidr",
          "footprint-value": [ "203.0.113.0/24" ]
        }
      ]
    }
  ]
}
```



```

    }
  ]
}
}

```

3.7.3. Incremental Updates

A benefit of using ALTO to provide CDNI Advertisement resources is that such resources can be updated using ALTO incremental updates [RFC8895]. Below is an example that also shows the benefit of having both JSON merge patch and JSON patch to encode updates.

At first, an ALTO client requests updates for "my-default-cdnifci", and the ALTO server returns the "control-uri" followed by the full CDNI Advertisement response. Then when there is a change in the delivery-protocols in that http/1.1 is removed (from [http/1.1, https/1.1] to only https/1.1) due to maintenance of the http/1.1 clusters, the ALTO server regenerates the new CDNI Advertisement resource and pushes the full replacement to the ALTO client. Later on, the ALTO server notifies the ALTO client that "192.0.2.0/24" is added into the "ipv4" footprint object for delivery-protocol https/1.1 by sending the change encoded by JSON patch to the ALTO client.

```

POST /updates/cdnifci HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 94

```

```

{
  "add": {
    "my-cdnifci-stream": {
      "resource-id": "my-default-cdnifci"
    }
  }
}

```

```

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

```

```

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/3141592653589"}

```

```

event: application/alto-cdni+json,my-cdnifci-stream
data: { ... full CDNI Advertisement resource ... }

```

```

event: application/alto-cdni+json,my-cdnifci-stream
data: {
  data: "meta": {
    data: "vtag": {
      data: "tag": "dasdfa10ce8b059740bddsfasd8eb1d47853716"
    data: }
    data: },
    data: "cdni-advertisement": {
      data: "capabilities-with-footprints": [
        data: {
          data: "capability-type": "FCI.DeliveryProtocol",
          data: "capability-value": {
            data: "delivery-protocols": [
              data: "https/1.1"
            data: ]
          data: },
          data: "footprints": [
            data: { "footprint-type": "ipv4cidr",
              data: "footprint-value": [ "203.0.113.0/24" ]
            data: }
          data: ]
        data: },
        data: { ... other CDNI advertisement object ... }
      data: ]
    data: }
  data: }

```

```

event: application/json-patch+json,my-cdnifci-stream
data: [
  data: { "op": "replace",
    data: "path": "/meta/vtag/tag",
    data: "value": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
  data: },
  data: { "op": "add",
    data: "path": "/cdni-advertisement/capabilities-with-footprints
/0/footprints/0/footprint-value/-",
    data: "value": "192.0.2.0/24"
  data: }
  data: ]

```

4. CDNI Advertisement Service using ALTO Network Map

4.1. Network Map Footprint Type: altopid

The ALTO protocol defines a concept called Provider-defined Identifier (PID) to represent a group of IPv4 or IPv6 addresses which can be applied the same management policy. The PID is an alternative to the pre-defined CDNI footprint types (i.e., ipv4cidr, ipv6cidr, asn, and countrycode).

To leverage this concept, this document defines a new CDNI Footprint Type called "altopid". A CDNI Advertisement resource can depend on an ALTO network map resource and use "altopid" footprints to compress its CDNI Footprint Payload.

Specifically, the "altopid" footprint type indicates that the corresponding footprint value is a list of PIDNames as defined in [RFC7285]. These PIDNames are references of PIDs in a network map resource. Hence a CDNI Advertisement resource using "altopid" footprints depends on a network map. For such a CDNI Advertisement resource, the resource id of its dependent network map MUST be included in the "uses" field of its IRD entry, and the "dependent-vtags" field with a reference to this network map MUST be included in its response (see the example in Section 4.2.2).

4.2. Examples

The following examples use the same IRD given in Section 3.7.1.

4.2.1. ALTO Network Map for CDNI Advertisements

Below provides a sample network map whose resource id is "my-eu-netmap". This map is referenced by the CDNI Advertisement example in Section 4.2.2.

```
GET /myeunetmap HTTP/1.1
Host: alto.example.com
Accept: application/alto-networkmap+json,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: 344
Content-Type: application/alto-networkmap+json
```

```
{
  "meta": {
    "vtag": [
      { "resource-id": "my-eu-netmap",
        "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ]
  },
  "network-map": {
    "south-france" : {
      "ipv4": [ "192.0.2.0/24", "198.51.100.0/25" ],
      "ipv6": [ "2001:db8::/32" ]
    },
    "germany": {
      "ipv4": [ "203.0.113.0/24" ]
    }
  }
}
```

4.2.2. ALTO PID Footprints in CDNI Advertisements

This example shows a CDNI Advertisement resource that depends on a network map described in Section 4.2.1.

```
GET /networkcdnifci HTTP/1.1
Host: alto.example.com
Accept: application/alto-cdni+json,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: 736
Content-Type: application/alto-cdni+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-eu-netmap",
        "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ]
  },
  "cdn-advertisement": {
    "capabilities-with-footprints": [
      { "capability-type": "FCI.DeliveryProtocol",
        "capability-value": [ "https/1.1" ],
        "footprints": [
          { "footprint-type": "altopid",
            "footprint-value": [ "south-france" ]
          }
        ]
      },
      { "capability-type": "FCI.AcquisitionProtocol",
        "capability-value": [ "https/1.1" ],
        "footprints": [
          { "footprint-type": "altopid",
            "footprint-value": [ "germany", "south-france" ]
          }
        ]
      }
    ]
  }
}
```

4.2.3. Incremental Updates

In this example, the ALTO client is interested in changes of "my-cdnifci-with-pid-footprints" and its dependent network map "my-eu-netmap". Considering two changes, the first one is to change footprints of the https/1.1 delivery protocol capability, and the second one is to remove the "south-france" PID from the footprints of the https/1.1 acquisition protocol capability.

```
POST /updates/cdnifci HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 185

{
  "add": {
    "my-eu-netmap-stream": {
      "resource-id": "my-eu-netmap"
    },
    "my-netmap-cdnifci-stream": {
      "resource-id": "my-cdnifci-with-pid-footprints"
    }
  }
}

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/3141592653590"}

event: application/alto-networkmap+json,my-eu-netmap-stream
data: { ... full Network Map of my-eu-netmap ... }

event: application/alto-cdnifci+json,my-netmap-cdnifci-stream
data: { ... full CDNI Advertisement resource ... }

event: application/json-patch+json,my-netmap-cdnifci-stream
data: [
data:   { "op": "replace",
data:     "path": "/meta/vtag/tag",
data:     "value": "dasdfal0ce8b059740bddsfasd8ebld47853716"
data:   },
data:   { "op": "add",
data:     "path":
data:       "/cdni-advertisement/capabilities-with-footprints
data:       /0/footprints/0/footprint-value/-",
data:     "value": "germany"
data:   }
data: ]

event: application/json-patch+json,my-netmap-cdnifci-stream
data: [
data:   { "op": "replace",
```

```
data:      "path": "/meta/vtag/tag",
data:      "value": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
data:    },
data:    { "op": "remove",
data:      "path":
data:        "/cdni-advertisement/capabilities-with-footprints
/1/footprints/0/footprint-value/1"
data:    }
data:  ]
```

5. Filtered CDNI Advertisement using CDNI Capabilities

Sections 3 (Section 3) and 4 (Section 4) describe CDNI Advertisement Service which can be used to enable a uCDN to get capabilities with footprint restrictions from dCDNs. However, since always getting full CDNI Advertisement resources from dCDNs is inefficient, this document introduces a new service named "Filtered CDNI Advertisement Service", to allow a client to filter a CDNI Advertisement resource using a client-given set of CDNI capabilities. For each entry of the CDNI Advertisement response, an entry will only be returned to the client if it contains at least one of the client given CDNI capabilities. The relationship between a filtered CDNI Advertisement resource and a CDNI Advertisement resource is similar to the relationship between a filtered network/cost map and a network/cost map.

5.1. Media Type

A filtered CDNI Advertisement resource uses the same media type defined for the CDNI Advertisement resource in Section 3.1: "application/alto-cdni+json".

5.2. HTTP Method

A filtered CDNI Advertisement resource is requested using the HTTP POST method.

5.3. Accept Input Parameters

The input parameters for a filtered CDNI Advertisement resource 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-cdnifilter+json" which is a JSON object of type ReqFilteredCDNIAdvertisement, where:

```
object {  
  JSONString capability-type;  
  JSONValue capability-value;  
} CDNICapability;  
  
object {  
  CDNICapability cdni-capabilities<0..*>;  
} ReqFilteredCDNIAdvertisement;
```

with fields:

capability-type: The same as Base Advertisement Object's capability-type defined in Section 5.1 of [RFC8008].

capability-value: The same as Base Advertisement Object's capability-value defined in Section 5.1 of [RFC8008].

cdni-capabilities: A list of CDNI capabilities defined in Section 5.1 of [RFC8008] for which footprints are to be returned. If this list is empty, the ALTO server MUST interpret it as a request for the full CDNI Advertisement resource. The ALTO server MUST interpret entries appearing in this list multiple times as if they appeared only once. If the ALTO server does not define any footprints for a CDNI capability, it MUST omit this capability from the response.

5.4. Capabilities

There are no applicable capabilities.

5.5. Uses

Same to the "uses" field of the CDNI Advertisement resource (see Section 3.5).

5.6. Response

If the request is invalid, the response MUST indicate an error, using ALTO protocol error handling specified in Section 8.5 of [RFC7285].

Specifically, a filtered CDNI Advertisement request is invalid if:

- * the value of "capability-type" is null;
- * the value of "capability-value" is null;
- * the value of "capability-value" is inconsistent with "capability-type".

When a 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 CDNI capability.

The ALTO server returns a filtered CDNI Advertisement resource for a valid request. The format of a filtered CDNI Advertisement resource is the same as a full CDNI Advertisement resource (See Section 3.6.)

The returned filtered CDNI Advertisement resource MUST contain all the BaseAdvertisementObject objects satisfying the following condition: The CDNI capability object of each included BaseAdvertisementObject object MUST follow two constraints:

- * The "cdni-capabilities" field of the input includes a CDNI capability object X having the same capability type as it.
- * All the mandatory properties in its capability value is a superset of mandatory properties in capability value of X semantically.

See Section 5.7.1 for a concrete example.

The version tag included in the "vtag" field of the response MUST correspond to the full CDNI Advertisement resource from which the filtered CDNI Advertisement resource is provided. This ensures that a single, canonical version tag is used independently of any filtering that is requested by an ALTO client.

5.7. Examples

The following examples use the same IRD example as in Section 3.7.1.

5.7.1. A Basic Example

This example filters the full CDNI Advertisement resource in Section 3.7.2 by selecting only the http/1.1 delivery protocol capability. Only the second BaseAdvertisementObjects in the full resource will be returned because the second object's capability is http/1.1 and https/1.1 delivery protocols which is the superset of https/1.1 delivery protocol.

```
POST /cdnifci/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-cdni+json
Content-Type: application/cdnifilter+json
Content-Length: 176

{
```

```
"cdni-capabilities": [
  {
    "capability-type": "FCI.DeliveryProtocol",
    "capability-value": {
      "delivery-protocols": [ "https/1.1" ]
    }
  }
]
}

HTTP/1.1 200 OK
Content-Length: 570
Content-Type: application/alto-cdni+json

{
  "meta": {
    "vtag": {
      "resource-id": "my-filtered-cdnifci",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "cdni-advertisement": {
    "capabilities-with-footprints": [
      {
        "capability-type": "FCI.DeliveryProtocol",
        "capability-value": {
          "delivery-protocols": [
            "https/1.1",
            "http/1.1"
          ]
        },
        "footprints": [
          {
            "footprint-type": "ipv4cidr",
            "footprint-value": [ "198.51.100.0/24" ]
          }
        ]
      }
    ]
  }
}
```

5.7.2. Incremental Updates

In this example, the ALTO client only cares about the updates of one advertisement object for delivery protocol capability whose value includes "https/1.1". Thus, it adds its limitation of capabilities in "input" field of the POST request.

```
POST /updates/cdnifci HTTP/1.1
Host: fcialtoupdate.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 346
```

```
{
  "add": {
    "my-filtered-fci-stream": {
      "resource-id": "my-filtered-cdnifci",
      "input": {
        "cdni-capabilities": [
          {
            "capability-type": "FCI.DeliveryProtocol",
            "capability-value": {
              "delivery-protocols": [ "https/1.1" ]
            }
          }
        ]
      }
    }
  }
}
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream
```

```
event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/3141592653590"}
```

```
event: application/alto-cdni+json,my-filtered-fci-stream
data: { ... filtered CDNI Advertisement resource ... }
```

```
event: application/json-patch+json,my-filtered-fci-stream
data: [
data: {
data:   "op": "replace",
data:   "path": "/meta/vtag/tag",
data:   "value": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
data: },
data: { "op": "add",
data:   "path":
data:   "/cdni-advertisement/capabilities-with-footprints
data:   /0/footprints/0/footprint-value/-",
data:   "value": "192.0.2.0/24"
data: }
```

data:]

6. Query Footprint Properties using ALTO Property Map Service

Besides the requirement of retrieving footprints of given capabilities, another common requirement for uCDN is to query CDNI capabilities of given footprints.

Considering each footprint as an entity with properties including CDNI capabilities, a natural way to satisfy this requirement is to use the ALTO property map as defined in [I-D.ietf-alto-unified-props-new]. This section describes how ALTO clients look up properties for individual footprints. First, it describes how to represent footprint objects as entities in the ALTO property map. Then it describes how to represent footprint capabilities as entity properties in the ALTO property map. Finally, it provides examples of the full property map and the filtered property map supporting CDNI capabilities, and their incremental updates.

6.1. Representing Footprint Objects as Property Map Entities

A footprint object has two properties: footprint-type and footprint-value. A footprint-value is an array of footprint values conforming to the specification associated with the registered footprint type ("ipv4cidr", "ipv6cidr", "asn", "countrycode", and "altopid").

Considering each ALTO entity defined in [I-D.ietf-alto-unified-props-new] also has two properties: entity domain type and domain-specific identifier, a straightforward approach to represent a footprint as an ALTO entity is to represent its footprint-type as an entity domain type, and its footprint value as a domain-specific identifier.

Each existing footprint type can be represented as an entity domain type as follows:

- * According to [I-D.ietf-alto-unified-props-new], "ipv4" and "ipv6" are two predefined entity domain types, which can be used to represent "ipv4cidr" and "ipv6cidr" footprints respectively. Note that both "ipv4" and "ipv6" domains can include not only hierarchical addresses but also individual addresses. Therefore, a "ipv4cidr" or "ipv6cidr" footprint with the longest prefix can also be represented by an individual address entity. When the uCDN receives a property map with individual addresses in an "ipv4" or "ipv6" domain, it can translate them as corresponding "ipv4cidr" or "ipv6cidr" footprints with the longest prefix.

- * "pid" is also a predefined entity domain type, which can be used to represent "altopid" footprints. Note that "pid" is a resource-specific entity domain. To represent an "altopid" footprint, the specifying information resource of the corresponding "pid" entity domain MUST be the dependent network map used by the CDNI Advertisement resource providing this "altopid" footprint.
- * However, no existing entity domain type can represent "asn" and "countrycode" footprints. To represent footprint-type "asn" and "countrycode", this document registers two new entity domains in Section 7 in addition to the ones in [I-D.ietf-alto-unified-props-new].

Here is an example of representing a footprint object of "ipv4cidr" type as a set of "ipv4" entities in the ALTO property map. The representation of the footprint object of "ipv6cidr" type is similar.

```
{ "footprint-type": "ipv4cidr",  
  "footprint-value": ["192.0.2.0/24", "198.51.100.0/24"]  
} --> "ipv4:192.0.2.0/24", "ipv4:198.51.100.0/24"
```

And here is an example of corresponding footprint object of "ipv4cidr" type represented by an individual address in an "ipv4" domain in the ALTO property map. The translation of the entities in an "ipv6" domain is similar.

```
"ipv4:203.0.113.100" --> {  
  "footprint-type": "ipv4cidr",  
  "footprint-value": ["203.0.113.100/32"]  
}
```

6.1.1. ASN Domain

The ASN domain associates property values with Autonomous Systems in the Internet.

6.1.1.1. Entity Domain Type

The entity domain type of the ASN domain is "asn" (in lowercase).

6.1.1.2. Domain-Specific Entity Identifiers

The entity identifier of an entity in an ASN domain MUST be encoded as a string consisting of the characters "as" (in lowercase) followed by the ASN [RFC6793] as a decimal number without leading zeros.

6.1.1.3. Hierarchy and Inheritance

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

6.1.2. COUNTRYCODE Domain

The COUNTRYCODE domain associates property values with countries.

6.1.2.1. Entity Domain Type

The entity domain type of the COUNTRYCODE domain is "countrycode" (in lowercase).

6.1.2.2. Domain-Specific Entity Identifiers

The entity identifier of an entity in a COUNTRYCODE domain is encoded as an ISO 3166-1 alpha-2 code [ISO3166-1] in lowercase.

6.1.2.3. Hierarchy and Inheritance

There is no hierarchy or inheritance for properties associated with country codes.

6.2. Representing CDNI Capabilities as Property Map Entity Properties

This document defines a new entity property type called "cdni-capabilities". An ALTO server can provide a property map resource mapping the "cdni-capabilities" entity property type for a CDNI Advertisement resource that it provides to an "ipv4", "ipv6", "asn" or "countrycode" entity domain.

6.2.1. Defining Information Resource Media Type for Property Type cdni-capabilities

The entity property type "cdni-capabilities" allows defining resource-specific entity properties. When resource-specific entity properties are defined with entity property type "cdni-capabilities", the defining information resource for a "cdni-capabilities" property MUST be a CDNI Advertisement resource provided by the ALTO server. The media type of the defining information resource for a "cdni-capabilities" property is therefore:

application/alto-cdni+json

6.2.2. Intended Semantics of Property Type cdni-capabilities

A "cdni-capabilities" property for an entity is to indicate all the CDNI capabilities that a corresponding CDNI Advertisement resource provides for the footprint represented by this entity. Thus, the value of a "cdni-capabilities" property MUST be a JSON array. Each element in a "cdni-capabilities" property MUST be an JSON object as format of CDNICapability (see Section 5.3). The value of a "cdni-capabilities" property for an "ipv4", "ipv6", "asn", "countrycode" or "altopid" entity MUST include all the CDNICapability objects satisfying the following conditions: (1) they are provided by the defining CDNI Advertisement resource; and (2) the represented footprint object of this entity is in their footprint restrictions.

6.3. Examples

The following examples use the same IRD example given by Section 3.7.1.

6.3.1. Property Map

This example shows a full property map in which entities are footprints and entities' property is "cdni-capabilities".

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

HTTP/1.1 200 OK
Content-Length: 1522
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "meta": {
      "dependent-vtags": [
        { "resource-id": "my-default-cdnifci",
          "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62" }
      ]
    },
    "countrycode:us": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.DeliveryProtocol",
          "capability-value": {
            "delivery-protocols": ["http/1.1"]} }
      ],
      "ipv4:192.0.2.0/24": {
        "my-default-cdnifci.cdni-capabilities": [
```

```

    { "capability-type": "FCI.DeliveryProtocol",
      "capability-value": {
        "delivery-protocols": ["http/1.1"]}
    },
    "ipv4:198.51.100.0/24": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.DeliveryProtocol",
          "capability-value": {
            "delivery-protocols": ["https/1.1", "http/1.1"]}
        }
      ],
    },
    "ipv4:203.0.113.0/24": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.AcquisitionProtocol",
          "capability-value": {
            "acquisition-protocols": ["http/1.1"]}
        }
      ],
    },
    "ipv6:2001:db8::/32": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.DeliveryProtocol",
          "capability-value": {
            "delivery-protocols": ["http/1.1"]}
        }
      ],
    },
    "asn:as64496": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.DeliveryProtocol",
          "capability-value": {
            "delivery-protocols": ["https/1.1", "http/1.1"]}
        }
      ],
    }
  }
}

```

6.3.2. Filtered Property Map

This example uses the filtered property map service to get "pid" and "cdni-capabilities" properties for two footprints "ipv4:192.0.2.0/24" and "ipv6:2001:db8::/32".


```
POST /propmap/lookup/cdnifci-pid HTTP/1.1
Host: alto.example.com
Content-Type: application/alto-propmapparams+json
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: 181

{
  "entities": [
    "ipv4:192.0.2.0/24",
    "ipv6:2001:db8::/32"
  ],
  "properties": [ "my-default-cdnifci.cdni-capabilities",
                  "my-default-networkmap.pid" ]
}

HTTP/1.1 200 OK
Content-Length: 796
Content-Type: application/alto-propmap+json

{
  "property-map": {
    "meta": {
      "dependent-vtags": [
        { "resource-id": "my-default-cdnifci",
          "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62" },
        { "resource-id": "my-default-networkmap",
          "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf63" }
      ]
    },
    "ipv4:192.0.2.0/24": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.DeliveryProtocol",
          "capability-value": { "delivery-protocols": ["http/1.1"]} },
      ],
      "my-default-networkmap.pid": "pid1"
    },
    "ipv6:2001:db8::/32": {
      "my-default-cdnifci.cdni-capabilities": [
        { "capability-type": "FCI.DeliveryProtocol",
          "capability-value": { "delivery-protocols": ["http/1.1"]} },
      ],
      "my-default-networkmap.pid": "pid3"
    }
  }
}
```

6.3.3. Incremental Updates

In this example, the ALTO client is interested in updates for the properties "cdni-capabilities" and "pid" of two footprints "ipv4:192.0.2.0/24" and "countrycode:fr".

```
POST /updates/properties HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 339

{
  "add": {
    "fci-propmap-stream": {
      "resource-id": "filtered-cdnifci-property-map",
      "input": {
        "properties": [ "my-default-cdnifci.cdni-capabilities",
                        "my-default-networkmap.pid" ],
        "entities": [ "ipv4:192.0.2.0/24",
                      "ipv6:2001:db8::/32" ]
      }
    }
  }
}

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/1414213562373"}

event: application/alto-cdni+json,fci-propmap-stream
data: { ... filtered property map ... }

event: application/merge-patch+json,fci-propmap-stream
data: {
data:   "property-map": {
data:     "meta": {
data:       "dependent-vtags": [
data:         { "resource-id": "my-default-cdnifci",
data:           "tag": "2beeac8ee23c3ddle98a73fd30df80ece9fa5627"},
data:         { "resource-id": "my-default-networkmap",
data:           "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf63"}
data:       ]
data:     },
data:   },
```

```

data:      "ipv4:192.0.2.0/24": {
data:      "my-default-cdnifci.cdni-capabilities": [
data:      { "capability-type": "FCI.DeliveryProtocol",
data:      "capability-value": {
data:      "delivery-protocols": ["http/1.1", "https/1.1"]}}}
data:      }
data:      }
data:      }

event: application/json-patch+json,fci-propmap-stream
data: [
data:  { "op": "replace",
data:      "path": "/meta/dependent-vtags/0/tag",
data:      "value": "61b23185a50dc7b334577507e8ff8c3b409e4"
data:  },
data:  { "op": "replace",
data:      "path":
data:      "/property-map/countrycode:fr/my-default-networkmap.pid",
data:      "value": "pid5"
data:  }
data: ]

```

7. IANA Considerations

This document defines two new media types: "application/alto-cdni+json", as described in Section 7.1, and "application/cdnifilter+json", as described in Section 7.2. It also defines a new CDNI metadata footprint type (Section 7.3), two new ALTO entity domain types (Section 7.4), and a new ALTO entity property type (Section 7.5).

7.1. application/alto-cdni+json Media Type

Type name:
application

Subtype name:
alto-cdni+json

Required parameters:
N/A

Optional parameters:
N/A

Encoding considerations:
Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations:

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

Interoperability considerations:

N/A

Published specification:

Section 3 of RFCthis

Applications that use this media type:

ALTO servers and ALTO clients [RFC7285] either stand alone or are embedded within other applications that provides CDNI interfaces for uCDNs or dCDNs.

Fragment identifier considerations:

N/A

Additional information:

Magic number(s): N/A

File extension(s): N/A

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).

7.2. application/alto-cdnifilter+json Media Type

Type name:

application

Subtype name:

alto-cdnifilter+json

Required parameters:
N/A

Optional parameters:
N/A

Encoding considerations:
Encoding considerations are identical to those specified for the
"application/json" media type. See [RFC8259].

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

Interoperability considerations:
N/A

Published specification:
Section 5 of RFCthis

Applications that use this media type:
ALTO servers and ALTO clients [RFC7285] either stand alone or are
embedded within other applications that provides CDNI interfaces
for uCDNs or dCDNs and supports CDNI capability-based filtering.

Fragment identifier considerations:
N/A

Additional information:
Magic number(s): N/A

File extension(s): N/A

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).

7.3. CDNI Metadata Footprint Type Registry

This document updates the CDNI Metadata Footprint Types Registry created by Section 7.2 of [RFC8006]. A new footprint type is to be registered, listed in Table 1.

Footprint Type	Description	Specification
altopid	A list of PID names	Section 4 of RFCthis

Table 1: CDNI Metadata Footprint Type

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

7.4. ALTO Entity Domain Type Registry

This document updates the ALTO Entity Domain Type Registry created by Section 11.2 of [I-D.ietf-alto-unified-props-new]. Two new entity domain types are to be registered, listed in Table 2.

Identifier	Entity Address Encoding	Hierarchy & Inheritance	Media Type of Defining Resource	Mapping to ALTO Address Type
asn	See Section 6.1.1.2 of RFCthis	None	None	false
countrycode	See Section 6.1.2.2 of RFCthis	None	None	false

Table 2: Additional ALTO Entity Domain Types

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

7.5. ALTO Entity Property Type Registry

This document updates the ALTO Entity Property Type Registry created by Section 11.3 of [I-D.ietf-alto-unified-props-new]. A new entity property type is to be registered, listed in Table 3.

Identifier	Intended Semantics	Media Type of Defining Resource
cdni-capabilities	Section 6.2 of RFCthis	application/alto-cdni+json

Table 3: Additional ALTO Entity Property Type

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

8. Security Considerations

As an extension of the base ALTO protocol [RFC7285], this document fits into the architecture of the base protocol. And hence Security Considerations of the base protocol (Section 15 of [RFC7285]) fully apply when this extension is provided by an ALTO server.

In the context of CDNI Advertisement, the following security risk scenarios should be considered:

- * For authenticity and integrity of ALTO information, an attacker may disguise itself as an ALTO server for a dCDN (e.g., by starting a man-in-the-middle attack), and provide false capabilities and footprints to a uCDN using the CDNI Advertisement service. Such false information may lead a uCDN to (1) select an incorrect dCDN to serve user requests, or (2) skip uCDNs in good conditions. To address this risk, protection strategies in Section 15.1.2 of [RFC7285] can be applied.
- * For potential undesirable guidance from authenticated ALTO information, a dCDN can provide a uCDN with limited capabilities and smaller footprint coverage so that the dCDN can avoid transferring traffic for a uCDN which they should have to transfer. To reduce this risk, the protection strategies in Section 15.2.2 of [RFC7285] can be considered.
- * For confidentiality and privacy of ALTO information, footprint properties integrated with ALTO property maps may expose network location identifiers (e.g., IP addresses or fine-grained PIDs).

To address this risk, the protection strategy for risk types (1) and (3) as described in Section 15.3 of [RFC7285] can be considered.

- * For availability of ALTO services, an attacker may conduct service degradation attacks using services defined in this document to disable ALTO services of a network. It may request potentially large, full CDNI Advertisement resources from an ALTO server in a dCDN continuously, to consume the bandwidth resources of that ALTO server. It may also query filtered property map services with many smaller individual footprints, to consume the computation resources of the ALTO server. To mitigate these risks, the protection strategies in Section 15.5.2 of [RFC7285] can be applied.

Although protection strategies as described in Section 15 of [RFC7285] should be applied to address aforementioned security and privacy considerations, two special cases need to be included as follows:

- * As required by section 7 of [RFC8008],

"All protocols that implement these capabilities and footprint advertisement objects are REQUIRED to provide integrity and authentication services."

Therefore, the uCDN (ALTO Client) MUST be authenticated to the dCDN (ALTO Server). And the dCDN (ALTO Server) MUST support HTTP Digest Authentication and MAY also support TLS mutual authentication. The authentication method will need to be negotiated out of band and is out of scope for this document, as is the approach for provisioning and managing these credentials.

- * One specific information leakage risk introduced by this document could not be addressed by these strategies. In particular, if a dCDN signs agreements with multiple uCDNs without any isolation, this dCDN may disclose extra information of one uCDN to another one. In that case, one uCDN may redirect requests which should not have to be served by this dCDN to it.

To reduce the risk, a dCDN SHOULD isolate full/filtered CDNI Advertisement resources for different uCDNs. It could consider generating URIs of different full/filtered CDNI Advertisement resources by hashing its company ID, a uCDN's company ID as well as their agreements. A dCDN SHOULD avoid exposing all full/filtered CDNI Advertisement resources in one of its IRDs.

9. References

9.1. Normative References

- [I-D.ietf-alto-unified-props-new]
Roome, W., Randriamasy, S., Yang, Y. R., Zhang, J. J., and K. Gao, "An ALTO Extension: Entity Property Maps", Work in Progress, Internet-Draft, draft-ietf-alto-unified-props-new-22, 25 January 2022,
<<https://datatracker.ietf.org/doc/html/draft-ietf-alto-unified-props-new-22>>.
- [ISO3166-1]
ISO (International Organization for Standardization), ., "ISO 3166-1: Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", 2020.
- [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/rfc/rfc2119>>.
- [RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012,
<<https://www.rfc-editor.org/rfc/rfc6793>>.
- [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/rfc/rfc7285>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015,
<<https://www.rfc-editor.org/rfc/rfc7493>>.
- [RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016,
<<https://www.rfc-editor.org/rfc/rfc8006>>.
- [RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016,
<<https://www.rfc-editor.org/rfc/rfc8008>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/rfc/rfc8259>>.
- [RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <<https://www.rfc-editor.org/rfc/rfc8895>>.

9.2. Informative References

- [I-D.ietf-alto-path-vector] Gao, K., Lee, Y., Randriamasy, S., Yang, Y. R., and J. J. Zhang, "An ALTO Extension: Path Vector", Work in Progress, Internet-Draft, draft-ietf-alto-path-vector-21, 2 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-alto-path-vector-21>>.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <<https://www.rfc-editor.org/rfc/rfc5693>>.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <<https://www.rfc-editor.org/rfc/rfc6707>>.
- [RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <<https://www.rfc-editor.org/rfc/rfc7971>>.
- [RFC7975] Niven-Jenkins, B., Ed. and R. van Brandenburg, Ed., "Request Routing Redirection Interface for Content Delivery Network (CDN) Interconnection", RFC 7975, DOI 10.17487/RFC7975, October 2016, <<https://www.rfc-editor.org/rfc/rfc7975>>.

Acknowledgments

The authors thank Matt Caulfield, Danny Alex Lachos Perez, Daryl Malas and Sanjay Mishra for their timely reviews and invaluable comments. Big thanks also to ALTO WG Chairs (Qin Wu and Vijay Gurbani), and all the directorate reviewers and IESG reviewers (Martin Duke, Erik Kline, Martin Vigoureux, Murray Kucherawy, Roman Danyliw, Zaheduzzaman Sarker, Eric Vyncke, and Francesca Palombini), for their thorough reviews, discussions, guidance and shepherding, that further improve this document.

Jan Seedorf has been partially supported by the GreenICN project (GreenICN: Architecture and Applications of Green Information Centric Networking), a research project supported jointly by the European Commission under its 7th Framework Program (contract no. 608518) and the National Institute of Information and Communications Technology (NICT) in Japan (contract no. 167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the European Commission, or NICT.

This document has also been supported by the Coordination Support Action entitled 'Supporting European Experts Presence in International Standardisation Activities in ICT' ("StandlCT.eu") funded by the European Commission under the Horizon 2020 Programme with Grant Agreement no. 780439. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the European Commission.

Contributors

Xiao Shawn Lin
Huawei
2222 Newjinqiao Rd
Shanghai
200125
China
Phone: +86-15316812351
Email: x.shawn.lin@gmail.com

Authors' Addresses

Jan Seedorf
HFT Stuttgart - Univ. of Applied Sciences
Schellingstrasse 24
70174 Stuttgart
Germany
Phone: +49-0711-8926-2801
Email: jan.seedorf@hft-stuttgart.de

Y. Richard Yang
Yale University
51 Prospect Street
New Haven, CT 06511
United States of America
Phone: +1-203-432-6400
Email: yry@cs.yale.edu
URI: <http://www.cs.yale.edu/~yry/>

Kevin J. Ma
Ericsson
43 Nagog Park
Acton, MA 01720
United States of America
Phone: +1-978-844-5100
Email: kevin.j.ma.ietf@gmail.com

Jon Peterson
NeuStar
1800 Sutter St Suite 570
Concord, CA 94520
United States of America
Email: jon.peterson@neustar.biz

Jingxuan Jensen Zhang
Tongji University
4800 Cao'an Hwy
Shanghai
201804
China
Email: jingxuan.zhang@tongji.edu.cn

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 18, 2020

S. Randriamasy
Nokia Bell Labs
R. Yang
Yale University
Q. Wu
Huawei
L. Deng
China Mobile
N. Schwan
Thales Deutschland
March 17, 2020

Application-Layer Traffic Optimization (ALTO) Cost Calendar
draft-ietf-alto-cost-calendar-21

Abstract

This document is an extension to the base Application-Layer Traffic Optimization (ALTO) protocol. It extends the ALTO cost information service so that applications decide not only 'where' to connect, but also 'when'. This is useful for applications that need to perform bulk data transfer and would like to schedule these transfers during an off-peak hour, for example. This extension introduces ALTO Cost Calendar, with which an ALTO Server exposes ALTO cost values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes, are specified in the Information Resources Directory and ALTO Server responses.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 18, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Some recent known uses	4
1.2. Terminology	5
2. Requirements Language	5
3. Overview of ALTO Cost Calendars and terminology	5
3.1. ALTO Cost Calendar overview	6
3.2. ALTO Cost Calendar information features	6
3.3. ALTO Calendar design characteristics	7
3.3.1. ALTO Cost Calendar for all cost modes	9
3.3.2. Compatibility with legacy ALTO Clients	10
4. ALTO Calendar specification: IRD extensions	10
4.1. Calendar attributes in the IRD resource capabilities . .	11
4.2. Calendars in a delegate IRD	12
4.3. Example IRD with ALTO Cost Calendars	13
5. ALTO Calendar specification: Service Information Resources .	17
5.1. Calendar extensions for Filtered Cost Maps (FCM)	17
5.1.1. Calendar extensions in Filtered Cost Map requests . .	17
5.1.2. Calendar extensions in Filtered Cost Map responses .	18
5.1.3. Use case and example: FCM with a bandwidth Calendar .	21
5.2. Calendar extensions in the Endpoint Cost Service	23
5.2.1. Calendar specific input in Endpoint Cost requests . .	23
5.2.2. Calendar attributes in the Endpoint Cost response . .	24
5.2.3. Use case and example: ECS with a routingcost Calendar	25
5.2.4. Use case and example: ECS with a multi-cost Calendar	
for routingcost and owdelay	27
6. IANA Considerations	29
7. Security Considerations	29
8. Operational Considerations	31
9. Acknowledgements	32
10. References	32
10.1. Normative References	32

10.2. Informative References	33
Authors' Addresses	35

1. Introduction

The base Application-Layer Traffic Optimization (ALTO) protocol specified in [RFC7285] provides guidance to overlay applications that need to select one or several hosts from a set of candidates able to provide a desired resource. This guidance is based on parameters that affect performance and efficiency of the data transmission between the hosts such as the topological distance. The goal of ALTO is to improve the Quality of Experience (QoE) in the application while optimizing resource usage in the underlying network infrastructure.

The ALTO protocol in [RFC7285] specifies a network map which defines groupings of endpoints in provider-defined network regions identified by Provider-defined Identifiers (PIDs). The Cost Map Service, Endpoint Cost Service (ECS) and Endpoint Ranking Service then provide ISP-defined costs and rankings for connections among the specified endpoints and PIDs and thus incentives for application clients to connect to ISP preferred locations, for instance, to reduce their costs. For the reasons outlined in the ALTO problem statement [RFC5693] and requirement AR-14 of [RFC6708], ALTO does not disseminate network metrics that change frequently. In a network, the costs can fluctuate for many reasons having to do with instantaneous traffic load or due to diurnal patterns of traffic demand or planned events such as network maintenance, holidays or highly publicized events. Thus, an ALTO application wishing to use the Cost Map and Endpoint Cost Service at some future time will have to estimate the state of the network at that time, a process that is, at best, fragile and brittle since the application does not have any visibility into the state of the network. Providing network costs for only the current time thus may not be sufficient, in particular for applications that can schedule their traffic in a span of time, for example by deferring backups or other background traffic to off-peak hours.

In case the ALTO Cost value changes are predictable over a certain period of time and the application does not require immediate data transfer, it can save time to get the whole set of cost values over this period in one single ALTO response. Using this set to schedule data transfers allows optimizing the network resources usage and QoE. ALTO Clients and Servers can also minimize their workload by reducing and accordingly scheduling their data exchanges.

This document extends [RFC7285] to allow an ALTO Server to provide network costs for a given duration of time. A sequence of network

costs across a time span for a given pair of network locations is named an "ALTO Cost Calendar". The Filtered Cost Map Service and Endpoint Cost Service are extended to provide Cost Calendars. In addition to this functional ALTO enhancement, we expect to further save network and storage resources by gathering multiple Cost Values for one cost type into one single ALTO Server response.

In this document, an "ALTO Cost Calendar" is specified in terms of information resource capabilities that are applicable to time-sensitive ALTO metrics. An ALTO Cost Calendar exposes ALTO Cost Values in JSON arrays, see [RFC8259], where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the Information Resources Directory (IRD) and in the Server response to allow the ALTO Client to interpret the received ALTO values. Last, the extensions for ALTO Calendars are applicable to any Cost Mode and they ensure backwards compatibility with legacy ALTO Clients - those that only support [RFC7285].

In the rest of this document, Section 3 provides the design characteristics. Sections Section 4 and Section 5 define the formal specifications for the IRD and the information resources. IANA, security and operational considerations are addressed respectively in sections Section 6, Section 7 and Section 8.

1.1. Some recent known uses

A potential use case is implementing smart network services that allow applications to dynamically build end-to-end, virtual networks, to satisfy given demands, with no manual intervention. For example, data-transfer automation applications may need a network service to determine on the availability of bandwidth resources, to decide when to transfer their data sets. The SENSE project [SENSE-sdn-e2e-net] supports such applications by requiring that a network provides services such as the Time-Bandwidth-Product (TBP) service, which informs applications of bandwidth availability during a specific time period. ALTO Calendars can support this service if the Calendar start date and duration cover the period of interest of the requesting application.

The need of future scheduling of large scale traffic that can be addressed by the ALTO protocol is also motivated by Unicorn, a unified resource orchestration framework for multi-domain, geo-distributed data analytics, see [Unicorn-fgcs].

1.2. Terminology

- o ALTO transaction: A request/response exchange between an ALTO Client and an ALTO Server.
- o Client: When used with a capital "C", this term refers to an ALTO Client.
- o Calendar, Cost Calendar: When used with capitalized words, these terms refer to an ALTO Cost Calendar.
- o Calendared: this adjective qualifies information resources providing Cost Calendars and information on costs that are provided in the form of a Cost Calendar.
- o Endpoint (EP): An endpoint is defined as in Section 2.1 of [RFC7285]. It can be, for example, a peer, a CDN storage location, a physical server involved in a virtual server-supported application, a party in a resource-sharing swarm such as a computation grid, or an online multi-party game.
- o ECM: Is an abbreviation for Endpoint Cost Map.
- o FCM: Is an abbreviation for Filtered Cost Map.
- o Server: When used with a capital "S", this term refers to an ALTO Server.

2. Requirements Language

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

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

3. Overview of ALTO Cost Calendars and terminology

This section gives a high-level overview of the design. It assumes the reader is familiar with the ALTO protocol [RFC7285] and its Multi-Cost ALTO extension [RFC8189].

3.1. ALTO Cost Calendar overview

An ALTO Cost Calendar provided by the ALTO Server provides 2 information items:

- o an array of values for a given metric, where each value specifies the metric corresponding to a time interval, where the value array can sometimes be a cyclic pattern that repeats a certain number of times.
- o attributes describing the time scope of the Calendar, including the size and number of the intervals and the date of the starting point of the Calendar, allowing an ALTO Client to interpret the values properly.

An ALTO Cost Calendar can be used like a "time table" to figure out the best time to schedule data transfers and also to proactively manage application traffic given predictable events such as expected spike in traffic due to crowd gathering (concerts, sports, etc.), traffic-intensive holidays and network maintenance. A Calendar may be viewed as a synthetic abstraction of, for example, real measurements gathered over previous periods on which statistics have been computed. However, like for any schedule, unexpected network incidents may require the current ALTO Calendar to be updated and re-sent to the ALTO Clients needing it. The "ALTO Incremental Updates Using Server-Sent Events (SSE)" Service [I-D.ietf-alto-incr-update-sse] can be used to directly update the Calendar upon value changes, if supported by both the Server and the Client.

Most likely, the ALTO Cost Calendar would be used for the Endpoint Cost Service, assuming that a limited set of feasible Endpoints for a non-real time application is already identified, that they do not need to be accessed immediately and that their access can be scheduled within a given time period. The Filtered Cost Map Service is also applicable as long as the size of the Map allows it.

3.2. ALTO Cost Calendar information features

The Calendar attributes are provided in the Information Resources Directory (IRD) and in ALTO Server responses. The IRD announces attributes without date values in its information resources capabilities, whereas attributes with time dependent values are provided in the "meta" section of Server responses. The ALTO Cost Calendar attributes provide the following information:

- o attributes to describe the time scope of the Calendar value array:

- * "time-interval-size": the applicable time interval size for each Calendar value, defined in seconds, that can cover a wide range of values.
- * "number-of-intervals": the number of intervals provided in the Calendar.
- o "calendar-start-time": specifying when the Calendar starts, that is to which date the first value of the Cost Calendar is applicable.
- o "repeated": an optional attribute indicating how many iterations of the provided Calendar will have the same values. The Server may use it to allow the Client to schedule its next request and thus save its own workload by reducing processing of similar requests.

Attribute "repeated" may take a very high value if a Calendar represents a cyclic value pattern that the Server considers valid for a long period. In this case, the Server will only update the Calendar values once this period has elapsed or if an unexpected event occurs on the network. See Section 8 for more discussion.

3.3. ALTO Calendar design characteristics

The present document uses the notations defined in Section "8.2 Notation" of [RFC7285].

The extensions in this document encode requests and responses using JSON [RFC8259].

In the base protocol [RFC7285] section 11.2.3.6, an ALTO cost is specified as a generic JSONValue [RFC8259], to allow extensions. However, that section 11.2.3.6 states: "An implementation of the protocol in this document ([RFC7285]) SHOULD assume that the cost is a JSONNumber and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how costs of other data types are signaled".

The present document extends the definition of a legacy cost map given in [RFC7285] to allow a cost entry to be an array of values, with one value per time interval, instead of being just one number, when the Cost Calendar functionality is activated on this cost. Therefore the implementor of this extension MUST consider that a cost entry is an array of values if this cost has been queried as a Calendar.

Specifically, an implementation of this extension MUST parse the "number-of-intervals" attribute of the Calendar attributes in an IRD entry announcing a service providing a Cost Calendar for a given cost type. The implementation then will know that a cost entry of the service will be an array of values, and the expected size of the array is that specified by the "number-of-intervals" attribute. The following rules attempt to ensure consistency between the array size announced by the Server and the actual size of the array received by the Client:

- o The size of the array of values conveyed in a Cost Calendar and received by the Client MUST be equal to the value of attribute "number-of-intervals" indicated in the IRD for the requested cost type.
- o When the size of the array received by the Client is different from the expected size, the Client SHOULD ignore the received array.

To realize an ALTO Calendar, this document extends the IRD and the ALTO requests and responses for Cost Calendars.

This extension is designed to be lightweight and to ensure backwards compatibility with base protocol ALTO Clients and with other extensions. It relies on section 8.3.7 "Parsing of Unknown Fields" of [RFC7285] that writes: "Extensions may include additional fields within JSON objects defined in this document. ALTO implementations MUST ignore unknown fields when processing ALTO messages."

The Calendar-specific capabilities are integrated in the information resources of the IRD and in the "meta" member of ALTO responses to Cost Calendars requests. A Calendar and its capabilities are associated with a given information resource and within this information resource, with a given cost type. This design has several advantages:

- o it does not introduce a new mode,
- o it does not introduce new media types,
- o it allows an ALTO Server to offer, for a cost type, different Calendars with attributes that are specific to the information resources providing a Calendar for this cost type, instead of being globally specific to the cost type.

The applicable Calendared information resources are:

- o the Filtered Cost Map,

- o the Endpoint Cost Map.

The ALTO Server can choose in which frequency it provides cost Calendars to ALTO Clients. It may either provide Calendar updates starting at the request date, or carefully schedule its updates so as to take profit from a potential repetition/periodicity of Calendar values.

Since Calendar attributes are specific to an information resource, a Server may adapt the granularity of the calendared information so as to moderate the volume of exchanged data. For example: suppose a Server provides a Calendar for cost type name "routingcost". The Server may offer a Calendar in a Cost Map resource, which may be a voluminous resource, as an array of 6 intervals lasting each 4 hours. It may also offer a Calendar in an Endpoint Cost Map resource, which is potentially less voluminous, as a finer-grained array of 24 intervals lasting 1 hour each.

The ALTO Server does not support constraints on Calendars, provided Calendars are requested for numerical values, for two main reasons:

- o constraints on an array of values may be various: for instance, some Clients may refuse Calendars with one single value violating a constraint, where as other ones may tolerate Calendars with values violating constraints for example at given times. Therefore, expressing constraints in a way that covers all possible Client preferences is challenging,
- o if constraints were to be supported, the processing overhead would be substantial for the Server as it would have to parse all the values of the Calendar array before returning a response.

As providing the constraint functionality in conjunction with the Calendar functionality is not feasible for the reasons described above, the two features are mutually exclusive. The absence of constraints on Filtered Cost Map and Endpoint Cost Map Calendars reflects a divergence from the non-calendared information resources defined in [RFC7285] and extended in [RFC8189], that support optional constraints.

3.3.1. ALTO Cost Calendar for all cost modes

An ALTO Cost Calendar is well-suited for values encoded in the "numerical" mode. Actually, a Calendar can also represent metrics in other modes considered as compatible with time-varying values. For example, types of Cost values such as JSONBool can also be calendared, as their value may be 'true' or 'false' depending on

given time periods or likewise, values represented by strings, such as "medium", "high", "low", "blue", "open".

Note also that a Calendar is suitable as well for time-varying metrics provided in the "ordinal" mode, if these values are time-varying and the ALTO Server provides updates of cost value based preferences.

3.3.2. Compatibility with legacy ALTO Clients

The ALTO protocol extensions for Cost Calendars have been defined so as to ensure that Calendar-capable ALTO Servers can provide legacy ALTO Clients with legacy information resources as well. That is, a legacy ALTO Client can request resources and receive responses as specified in [RFC7285].

A Calendar-aware ALTO Server MUST implement the base protocol specified in [RFC7285].

A Calendar-aware ALTO Client MUST implement the base protocol specified in [RFC7285].

As a consequence, when a metric is available as a Calendar array, it also MUST be available as a single value as required by [RFC7285]. The Server, in this case, provides the current value of the metric to either Calendar-aware Clients not interested in future or time-based values, or Clients implementing [RFC7285] only.

For compatibility with legacy ALTO Clients specified in [RFC7285], calendared information resources are not applicable for full cost maps for the following reason: a legacy ALTO Client would receive a calendared cost map via an HTTP 'GET' command. As specified in section 8.3.7 of [RFC7285], it will ignore the Calendar Attributes indicated in the "meta" of the responses. Therefore, lacking information on Calendar attributes, it will not be able to correctly interpret and process the values of the received array of Calendar cost values.

Therefore, calendared information resources MUST be requested via the Filtered Cost Map Service or the Endpoint Cost Service, using a POST method.

4. ALTO Calendar specification: IRD extensions

The Calendar attributes in the IRD information resources capabilities carry dateless values. A Calendar is associated with an information resource rather than a cost type. For example, a Server can provide a "routingcost" Calendar for the Filtered Cost Map Service at a

granularity of one day and a "routingcost" Calendar for the Endpoint Cost Service at a finer granularity but for a limited number of endpoints. An example IRD with Calendar specific features is provided in Section 4.3.

4.1. Calendar attributes in the IRD resource capabilities

A Cost Calendar for a given cost type MUST be indicated in the IRD by an object of type CalendarAttributes. A CalendarAttributes object is represented by the "calendar-attributes" member of a resource entry. Member "calendar-attributes" is an array of CalendarAttributes objects. Each CalendarAttributes object lists a set of one or more cost types it applies to. A cost type name MUST NOT appear more than once in the "calendar-attributes" member of a resource entry; multiple appearances of a cost type name in the CalendarAttributes object of the "calendar-attributes" member MUST cause the ALTO Client to ignore any occurrences of this name beyond the first encountered occurrence. The Client SHOULD consider the CalendarAttributes object in the array containing the first encountered occurrence of a cost type as the valid one for this cost type. As an alternative, the Client may want to avoid the risks of erroneous guidance associated to the use of potentially invalid Calendar values. In this case, the Client MAY ignore the totality of occurrences of CalendarAttributes objects containing the cost type name and query the cost type using [RFC7285].

The encoding format for object CalendarAttributes, using JSON [RFC8259], is as follows:

```
CalendarAttributes calendar-attributes <1..*>;
```

```
object{
  JSONString cost-type-names <1..*>;
  JSONNumber time-interval-size;
  JSONNumber number-of-intervals;
} CalendarAttributes;
```

o "cost-type-names":

- * An array of one or more elements indicating the cost type names in the IRD entry to which the values of "time-interval-size" and "number-of-intervals" apply.

o "time-interval-size":

- * is the duration of an ALTO Calendar time interval in a unit of seconds. A "time-interval-size" value contains a non-negative JSONNumber. Example values are: 300 and 7200, meaning that

each Calendar value applies on a time interval that lasts 5 minutes and 2 hours, respectively. Since an interval size (e.g., 100 ms) can be smaller than the unit, the value specified may be a floating point (e.g., 0.1). Both ALTO Clients and Servers should be aware of potential precision issues caused by using floating point numbers; for example, the floating number 0.1 cannot be represented precisely using a finite number of binary bits. To improve interoperability and be consistent with [RFC7285] on the use of floating point numbers, the Server and the Client SHOULD use IEEE 754 double-precision floating point [IEEE.754.2008] to store this value.

- o "number-of-intervals":

- * is a strictly positive integer (greater or equal to 1), that indicates the number of values of the Cost Calendar array.

- An ALTO Server SHOULD specify the "time-interval-size" in the IRD as the smallest it is able to provide. A Client that needs a longer interval can aggregate multiple cost values to obtain it.

- Attribute "cost-type-names" is associated with "time-interval-size" and "number-of-intervals", because multiple cost types may share the same values for attributes "time-interval-size" and "number-of-intervals". To avoid redundancies, cost type names sharing the same values for "time-interval-size" and "number-of-intervals" are grouped in the "cost-type-names" attribute. In the example IRD provided in Section 4.3, the information resource "filtered-cost-map-calendar" provides a Calendar for cost type names "num-routingcost", "num-throughputrating" and "string-servicestatus". Cost type names "num-routingcost" and "num-throughputrating" are grouped in the "cost-type-names" attribute because they share the same values for "time-interval-size" and "number-of-intervals", which are respectively 7200 and 12.

- Multiplying "time-interval-size" by "number-of-intervals" provides the duration of the provided Calendar. For example, an ALTO Server may provide a Calendar for ALTO values changing every "time-interval-size" equal to 5 minutes. If "number-of-intervals" has the value 12, then the duration of the provided Calendar is 1 hour.

4.2. Calendars in a delegate IRD

It may be useful to distinguish IRD resources supported by the base ALTO protocol from resources supported by its extensions. To achieve this, one option, is that a "root" ALTO Server implementing [RFC7285] resources and running at a given domain, delegates "specialized" information resources such as the ones providing Cost Calendars, to

another ALTO Server running in a subdomain. The "root" ALTO Server can provide a Calendar-specific resource entry, that has a media-type of "application/alto-directory+json" and that specifies the URI allowing to retrieve the location of a Calendar-aware Server and discover its resources. This option is described in Section 9.2.4 "Delegation using IRDs" of [RFC7285].

This document provides an example, where a "root" ALTO Server runs in a domain called "alto.example.com". It delegates the announcement of Calendars capabilities to an ALTO Server running in a subdomain called "custom.alto.example.com". The location of the "delegate Calendar IRD" is assumed to be indicated in the "root" IRD by the resource entry: "custom-calendared-resources".

Another benefit of delegation is that some cost types for some resources may be more advantageous as Cost Calendars and it makes little sense to get them as a single value. For example, if a cost type has predictable and frequently changing values, calendared in short time intervals such as a minute, it saves time and network resources to track the cost values via a focused delegate Server rather than the more general "root" Server.

4.3. Example IRD with ALTO Cost Calendars

This section provides an example ALTO Server IRD that supports various cost metrics and cost modes. In particular, since [RFC7285] makes it mandatory, the Server uses metric "routingcost" in the "numerical" mode.

For illustrative purposes, this section introduces 3 other fictitious example metrics and modes that should be understood as examples and should not be used or considered as normative.

The cost type names used in the example IRD are as follows:

- o "num-routingcost": refers to metric "routingcost" in the numerical mode as defined in [RFC7285] and registered with IANA.
- o "num-owdelay": refers to fictitious performance metric "owdelay" in the "numerical" mode, to reflect the one-way packet transmission delay on a path. A related performance metric is currently under definition in [I-D.ietf-alto-performance-metrics].
- o "num-throughputrating": refers to fictitious metric "throughputrating" in the "numerical" mode, to reflect the provider preference in terms of end to end throughput.

- o "string-servicestatus": refers to fictitious metric "servicestatus" containing a string, to reflect the availability, defined by the provider, of for instance path connectivity.

The example IRD includes 2 particular URIs providing Calendars:

- o "https://custom.alto.example.com/calendar/costmap/filtered": a Filtered Cost Map in which Calendar capabilities are indicated for cost type names: "num-routingcost", "num-throughputrating" and "string-servicestatus",
- o "https://custom.alto.example.com/calendar/endpointcost/lookup": an Endpoint Cost Map in which Calendar capabilities are indicated for cost type names: "num-routingcost", "num-owdelay", "num-throughputrating", "string-servicestatus".

The design of the Calendar capabilities allows some Calendars with the same cost type name to be available in several information resources with different Calendar Attributes. This is the case for Calendars on "num-routingcost", "num-throughputrating" and "string-servicestatus", available in both the Filtered Cost map and Endpoint Cost Service, but with different time interval sizes for "num-throughputrating" and "string-servicestatus".

--- Client to Server request for IRD -----

```
GET /calendars-directory HTTP/1.1
Host: custom.alto.example.com
Accept: application/alto-directory+json,application/alto-error+json
```

--- Server response to Client -----

```
HTTP/1.1 200 OK
Content-Length: 2622
Content-Type: application/alto-directory+json
```

```
{
  "meta" : {
    "default-alto-network-map" : "my-default-network-map",
    "cost-types": {
      "num-routingcost": {
        "cost-mode" : "numerical",
        "cost-metric" : "routingcost"
      },
      "num-owdelay": {
        "cost-mode" : "numerical",
        "cost-metric": "owdelay"
      },
    },
  },
}
```

```
    "num-throughputrating": {
      "cost-mode" : "numerical",
      "cost-metric": "throughputrating"
    },
    "string-servicestatus": {
      "cost-mode" : "string",
      "cost-metric": "servicestatus"
    }
  }
},
"resources" : {
  "filtered-cost-map-calendar" : {
    "uri" :
      "https://custom.alto.example.com/calendar/costmap/filtered",
    "media-type" : "application/alto-costmap+json",
    "accepts" : "application/alto-costmapfilter+json",
    "capabilities" : {
      "cost-constraints" : true,
      "cost-type-names" : [ "num-routingcost",
                           "num-throughputrating",
                           "string-servicestatus" ],
      "calendar-attributes" : [
        { "cost-type-names" : [ "num-routingcost",
                               "num-throughputrating" ],
          "time-interval-size" : 7200,
          "number-of-intervals" : 12
        },
        { "cost-type-names" : [ "string-servicestatus" ],
          "time-interval-size" : 1800,
          "number-of-intervals" : 48
        }
      ],
    },
    "uses": [ "my-default-network-map" ]
  },
  "endpoint-cost-map-calendar" : {
    "uri" :
      "https://custom.alto.example.com/calendar/endpointcost/lookup",
    "media-type" : "application/alto-endpointcost+json",
    "accepts" : "application/alto-endpointcostparams+json",
    "capabilities" : {
      "cost-constraints" : true,
      "cost-type-names" : [ "num-routingcost",
                           "num-owdelay",
                           "num-throughputrating",
                           "string-servicestatus" ],
      "calendar-attributes" : [
        { "cost-type-names" : [ "num-routingcost" ],
```

```

        "time-interval-size" : 3600,
        "number-of-intervals" : 24
    },
    { "cost-type-names" : [ "num-owdelay" ],
      "time-interval-size" : 300,
      "number-of-intervals" : 12
    },
    { "cost-type-names" : [ "num-throughputrating" ],
      "time-interval-size" : 60,
      "number-of-intervals" : 60
    },
    { "cost-type-names" : [ "string-servicestatus" ],
      "time-interval-size" : 120,
      "number-of-intervals" : 30
    }
  ]
}
}
}
}

```

In this example IRD, for the Filtered Cost Map Service:

- o the Calendar for "num-routingcost" and "num-throughputrating" is an array of 12 values each provided on a time interval lasting 7200 seconds (2 hours).
- o the Calendar for "string-servicestatus": "is an array of 48 values each provided on a time interval lasting 1800 seconds (30 minutes).

For the Endpoint Cost Service:

- o the Calendar for "num-routingcost": is an array of 24 values each provided on a time interval lasting 3600 seconds (1 hour).
- o the Calendar for "num-owdelay": is an array of 12 values each provided on a time interval lasting 300 seconds (5 minutes).
- o the Calendar for "num-throughputrating": is an array of 60 values each provided on a time interval lasting 60 seconds (1 minute).
- o the Calendar for "string-servicestatus": "is an array of 30 values each provided on a time interval lasting 120 seconds (2 minutes).

Note that in this example IRD, member "cost-constraints" is present with a value set to "true" in both information resources "filtered-cost-map-calendar" and "endpoint-cost-map-calendar". Although a

Calendar-aware ALTO Server does not support constraints for the reasons explained in section Section 3.3, it MUST support constraints on cost types that are not requested as Calendars but are requested as specified in [RFC7285] and [RFC8189].

5. ALTO Calendar specification: Service Information Resources

This section documents extensions to two basic ALTO information resources (Filtered Cost Maps and Endpoint Cost Service) to provide calendared information services for them.

Both extensions return calendar start time (calendar-start-time, a point in time), which MUST be specified as an HTTP "Date" header field using the IMF-fixdate format specified in Section 7.1.1.1 of [RFC7231]. Note that the IMF-fixdate format uses "GMT", not "UTC", to designate the time zone, as in this example:

Date: Tue, 15 Nov 2019 08:12:31 GMT

5.1. Calendar extensions for Filtered Cost Maps (FCM)

A legacy ALTO Client requests and gets Filtered Cost Map responses as specified in [RFC7285].

5.1.1. Calendar extensions in Filtered Cost Map requests

The input parameters of a "legacy" request for a Filtered Cost Map, defined by object ReqFilteredCostMap in section 11.3.2 of [RFC7285], are augmented with one additional member. The same augmentation applies to object ReqFilteredCostMap defined in section 4.1.2 of [RFC8189].

A Calendar-aware ALTO Client requesting a Calendar on a given Cost Type for a Filtered Cost Map resource having Calendar capabilities MUST add the following field to its input parameters:

JSONBoolean calendared<1..*>;

This field is an array of 1 to N boolean values, where N is the number of requested metrics. N is greater than 1 when the Client and the Server also implement [RFC8189].

Each entry corresponds to the requested metric at the same array position. Each boolean value indicates whether or not the ALTO Server should provide the values for this cost type as a Calendar. The array MUST contain exactly N boolean values, otherwise, the Server returns an error.

This field MUST NOT be included if no member "calendar-attributes" is specified in this information resource.

If a value of field "calendared" is 'true' for a cost type name for which no Calendar attributes have been specified: an ALTO Server, whether it implements the extensions of this document or only implements [RFC7285], MUST ignore it and return a response with a single cost value as specified in [RFC7285].

If this field is not present, it MUST be assumed to have only values equal to 'false'.

A Calendar-aware ALTO Client that supports requests for only one cost type at a time and wants to request a Calendar MUST provide an array of 1 element:

```
"calendared" : [true],
```

A Calendar-aware ALTO Client that supports requests for more than one cost types at a time, as specified in [RFC8189] MUST provide an array of N values set to 'true' or 'false', depending whether it wants the applicable cost type values as a single or calendared value.

5.1.2. Calendar extensions in Filtered Cost Map responses

In a calendared ALTO Filtered Cost Map, a cost value between a source and a destination is a JSON array of JSON values. An ALTO Calendar values array has a number of values equal to the value of member "number-of-intervals" of the Calendar attributes that are indicated in the IRD. These attributes will be conveyed as metadata in the Filtered Cost Map response. Each element of the array is valid for the time-interval that matches its array position.

The FCM response conveys metadata among which:

- o some are not specific to Calendars and ensure compatibility with [RFC7285] and [RFC8189]
- o some are specific to Calendars.

The non Calendar-specific "meta" fields of a calendared Filtered Cost Map response MUST include at least:

- o if the ALTO Client requests cost values for one cost type at a time only: the "meta" fields specified in [RFC7285] for these information service responses:
 - * "dependent-vtags ",

- * "cost-type" field.
- o if the ALTO Client implements the Multi-Cost ALTO extension specified in [RFC8189] and requests cost values for several cost types at a time: the "meta" fields specified in [RFC8189] for these information service responses:
 - * "dependent-vtags ",
 - * "cost-type" field with value set to '{}', for backwards compatibility with [RFC7285].
 - * "multi-cost-types" field.

If the Client request does not provide member "calendared" or if it provides it with a value equal to 'false', for all the requested cost types, then the ALTO Server response is exactly as specified in [RFC7285] and [RFC8189].

If the value of member "calendared" is equal to 'false' for a given requested cost type, the ALTO Server MUST return, for this cost type, a single cost value as specified in [RFC7285].

If the value of member "calendared" is equal to 'true' for a given requested cost type, the ALTO Server returns, for this cost type, a cost value Calendar as specified above in this section. In addition to the above cited non Calendar specific "meta" members, the Server MUST provide a Calendar specific metadata field.

The Calendar-specific "meta" field that a calendared Filtered Cost Map response MUST include is a member called "calendar-response-attributes", that describes properties of the Calendar and where:

- o member "calendar-response-attributes" is an array of one or more objects of type "CalendarResponseAttributes".
- o each "CalendarResponseAttributes" object in the array is specified for one or more cost types for which the value of member "calendared", in object ReqFilteredCostMap provided in the Client request, is equal to 'true' and for which a Calendar is provided for the requested information resource.
- o the "CalendarResponseAttributes" object that applies to a cost type name has a corresponding "CalendarAttributes" object defined for this cost type name in the IRD capabilities of the requested information resource. This object is the entry, in the "calendar-attributes" array member of the IRD resource entry, that includes the name of the requested cost type. This corresponding

"CalendarAttributes" object has the same values as object "CalendarResponseAttributes" for members "time-interval-size" and "number-of-intervals". The members of the "CalendarResponseAttributes" object include all the members of the corresponding "CalendarAttributes" object.

The format of member "CalendarResponseAttributes" is defined as follows:

```
CalendarResponseAttributes calendar-response-attributes <1..*>;
```

```
object{
  [JSONString cost-type-names <1..*>;]
  JSONString calendar-start-time;
  JSONNumber time-interval-size;
  JSONNumber number-of-intervals;
  [JSONNumber repeated;]
} CalendarResponseAttributes;
```

Object CalendarResponseAttributes has the following attributes:

- o "cost-type-names": is an array of one or more cost-type-names to which the value of the other members of CalendarResponseAttributes apply and for which a Calendar has been requested. The value of this member is a subset of the "cost-type-names" member of the abovementioned corresponding "CalendarAttributes" object in the "calendar-attributes" array member in the IRD. This member MUST be present when Cost Calendars are provided for more than one cost type.
- o "calendar-start-time": indicates the date at which the first value of the Calendar applies. The value is a string that, as specified in Section 5, contains an HTTP "Date" header field using the IMF-fixdate format specified in Section 7.1.1.1 of [RFC7231]. The value provided for attribute "calendar-start-time" SHOULD NOT be later than the request date.
- o "time-interval-size": as specified in Section 4.1 and with the same value as in the abovementioned corresponding "CalendarAttributes" object.
- o "number-of-intervals": as specified in Section 4.1 and with the same value as in the abovementioned corresponding "CalendarAttributes" object.
- o "repeated": is an optional field provided for Calendars. It is an integer N greater or equal to '1' that indicates how many iterations of the Calendar value array starting at the date

indicated by "calendar-start-time" have the same values. The number N includes the iteration provided in the returned response.

For example: suppose the "calendar-start-time" member has value "Mon, 30 Jun 2019 00:00:00 GMT", the "time-interval-size" member has value '3600', the "number-of-intervals" member has value '24' and the value of member "repeated" is equal to '4'. This means that the Calendar values are the same on Monday, Tuesday, Wednesday and Thursday on a period of 24 hours starting at 00:00:00 GMT. The ALTO Client thus may use the same Calendar for the next 4 days starting at "calendar-start-time" and will only need to request a new one for Friday July 4th at 00:00:00 GMT.

Attribute "repeated" may take a very high value if a Calendar represents a cyclic value pattern that the Server considers valid for a long period and hence will only update once this period has elapsed or if an unexpected event occurs on the network. In the latter case, the Client will be notified if it uses the "ALTO Incremental Updates Using Server-Sent Events (SSE)" Service, specified in [I-D.ietf-alto-incr-update-sse]. To this end, it is RECOMMENDED that ALTO Servers providing ALTO Calendars also provide the "ALTO Incremental Updates Using Server-Sent Events (SSE)" Service that is specified in [I-D.ietf-alto-incr-update-sse]. Likewise, ALTO Clients capable of using ALTO Calendars SHOULD also use the SSE Service. See also discussion in Section 8 "Operational Considerations".

5.1.3. Use case and example: FCM with a bandwidth Calendar

An example of non-real-time information that can be provisioned in a Calendar is the expected path throughput. While the transmission rate can be measured in real time by end systems, the operator of a data center is in the position of formulating preferences for given paths, at given time periods to avoid traffic peaks due to diurnal usage patterns. In this example, we assume that an ALTO Client requests a Calendar of network-provider-defined throughput ratings, as specified in the IRD, to schedule its bulk data transfers as described in the use cases.

In the example IRD, Calendars for cost type name "num-throughputrating" are available for the information resources: "filtered-cost-calendar-map" and "endpoint-cost-map-calendar". The ALTO Client requests a Calendar for "num-throughputrating" via a POST request for a Filtered Cost Map.

We suppose in the present example that the ALTO Client sends its request on Tuesday July 1st 2019 at 13:15. The Server returns Calendars with arrays of 12 numbers for each source and destination

pair. The values for metric "throughputrating", in this example, are assumed to be encoded in 2 digits.

```
POST /calendar/costmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Length: 217
Content-Type: application/alto-costmapfilter+json
Accept: application/alto-costmap+json,application/alto-error+json
```

```
{
  "cost-type" : {"cost-mode" : "numerical",
                 "cost-metric" : "throughputrating"},
  "calendared" : [true],
  "pids" : {
    "srcs" : [ "PID1", "PID2" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 1043
Content-Type: application/alto-costmap+json
```

```
{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"}
    ],
    "cost-type" : {"cost-mode" : "numerical",
                   "cost-metric" : "throughputrating"},
    "calendar-response-attributes" : [
      {"calendar-start-time" : "Tue, 1 Jul 2019 13:00:00 GMT",
       "time-interval-size" : 7200,
       "number-of-intervals" : 12}
    ]
  },
  "cost-map" : {
    "PID1": { "PID1": [ 1, 12, 14, 18, 14, 14,
                       14, 18, 19, 20, 11, 12],
              "PID2": [13,  4, 15, 16, 17, 18,
                       19, 20, 11, 12, 13, 14],
              "PID3": [20, 20, 18, 14, 12, 12,
                       14, 14, 12, 12, 14, 16] },
    "PID2": { "PID1": [17, 18, 19, 10, 11, 12,
                       13, 14, 15, 16, 17, 18],
              "PID2": [20, 20, 18, 16, 14, 14,
```

```

        14, 16, 16, 16, 14, 16],
    "PID3": [20, 20, 18, 14, 12, 12,
             14, 14, 12, 12, 14, 16] }
  }
}
```

5.2. Calendar extensions in the Endpoint Cost Service

This document extends the Endpoint Cost Service, as defined in {11.5.1} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values. The media type {11.5.1.1} and HTTP method {11.5.1.2} are unchanged.

5.2.1. Calendar specific input in Endpoint Cost requests

The extensions to the requests for calendared Endpoint Cost Maps are the same as for the Filtered Cost Map Service, specified in Section 5.1.1 of this document. Likewise, the rules defined around the extensions to ECM requests are the same as those defined in Section 5.1.1 for FCM requests.

The ReqEndpointCostMap object for a calendared ECM request will have the following format:

```

object {
  [CostType cost-type;]
  [CostType multi-cost-types<1..*>;]
  [JSONBoolean calendared<1..*>;]
  EndpointFilter endpoints;
} ReqEndpointCostMap;

object {
  [TypedEndpointAddr srcs<0..*>;]
  [TypedEndpointAddr dsts<0..*>;]
} EndpointFilter;
```

Member "cost-type" is optional because, in the ReqEndpointCostMap object definition of this document, it is jointly present with member "multi-cost-types", to ensure compatibility with RFC 8189. In RFC8189, members "cost-type" and "multi-cost-types" are both optional and have to obey the rule specified in section 4.1.2 of 8189 saying that: "the Client MUST specify either "cost-type" or "multi-cost-types" but MUST NOT specify both".

The interpretation of member "calendared" is the same as for the ReqFilteredCostMap object defined in Section 5.1.1 of this document. The interpretation of the other members is the same as for object

ReqEndpointCostMap is defined in [RFC7285] and [RFC8189]. The type TypedEndpointAddr is defined in section 10.4.1 of [RFC7285].

For the reasons explained in section Section 3.3, a Calendar-aware ALTO Server does not support constraints. Therefore, member "[constraints]" is not present in the ReqEndpointCostMap object and member "constraints" MUST NOT be present in the input parameters of a request for an Endpoint Cost Calendar. If this member is present, the Server MUST ignore it.

5.2.2. Calendar attributes in the Endpoint Cost response

The "meta" field of a calendared Endpoint Cost response MUST include at least:

- o if the ALTO Client supports cost values for one cost type at a time only: the "meta" fields specified in {11.5.1.6} of [RFC7285] for the Endpoint Cost response:
 - * "cost-type" field.
- o if the ALTO Client supports cost values for several cost types at a time, as specified in [RFC8189] : the "meta" fields specified in [RFC8189] for the the Endpoint Cost response:
 - * "cost-type" field with value set to '{}', for backwards compatibility with [RFC7285].
 - * "multi-cost-types" field.

If the Client request does not provide member "calendared" or if it provides it with a value equal to 'false', for all the requested Cost Types, then the ALTO Server response is exactly as specified in [RFC7285] and [RFC8189].

If the ALTO Client provides member "calendared" in the input parameters with a value equal to 'true' for given requested Cost Types, the "meta" member of a calendared Endpoint Cost response MUST include, for these cost types, an additional member "calendar-response-attributes", the contents of which obey the same rules as for the Filtered Cost Map Service, specified in Section 5.1.2. The Server response is thus changed as follows, with respect to [RFC7285] and [RFC8189]:

- o the "meta" member has one additional field "CalendarResponseAttributes", as specified for the Filtered Cost Map Service,

- o the calendared costs are JSONArrays instead of the JSONNumbers format used by legacy ALTO implementations. All arrays have a number of values equal to 'number-of-intervals'. Each value corresponds to the cost in that interval.

If the value of member "calendared" is equal to 'false' for a given requested cost type, the ALTO Server MUST return, for this cost type, a single cost value as specified in [RFC7285].

5.2.3. Use case and example: ECS with a routingcost Calendar

Let us assume an Application Client is located in an end system with limited resources and having access to the network that is either intermittent or provides an acceptable quality in limited but predictable time periods. Therefore, it needs to schedule both its resource-greedy networking activities and its ALTO transactions.

The Application Client has the choice to trade content or resources with a set of Endpoints and needs to decide with which one it will connect and at what time. For instance, the Endpoints are spread in different time-zones, or have intermittent access. In this example, the 'routingcost' is assumed to be time-varying, with values provided as ALTO Calendars.

The ALTO Client associated with the Application Client queries an ALTO Calendar on 'routingcost' and will get the Calendar covering the 24 hours time period "containing" the date and time of the ALTO Client request.

For cost type "num-routingcost", the solicited ALTO Server has defined 3 different daily patterns each represented by a Calendar, to cover the week of Monday June 30th at 00:00 to Sunday July 6th 23:59:

- o C1 for Monday, Tuesday, Wednesday, Thursday, (weekdays)
- o C2 for Saturday, Sunday, (weekend)
- o C3 for Friday (maintenance outage on July 4, 2019 from 02:00:00 GMT to 04:00:00 GMT, or big holiday such as New Year evening).

In the following example, the ALTO Client sends its request on Tuesday July 1st 2019 at 13:15.

The "routingcost" values are assumed to be encoded in 3 digits.

```
POST /calendar/endpointcost/lookup HTTP/1.1
Host: custom.alto.example.com
Content-Length: 304
```

Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

```
{
  "cost-type" : {"cost-mode" : "numerical",
                 "cost-metric" : "routingcost"},
  "calendared" : [true],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2001:db8::10"
    ]
  }
}
```

HTTP/1.1 200 OK

Content-Length: 1351
Content-Type: application/alto-endpointcost+json

```
{
  "meta" : {
    "cost-type" : {"cost-mode" : "numerical",
                   "cost-metric" : "routingcost"},
    "calendar-response-attributes" : [
      {"calendar-start-time" : "Mon, 30 Jun 2019 00:00:00 GMT",
       "time-interval-size" : 3600,
       "number-of-intervals" : 24,
       "repeated": 4
      }
    ]
  },
  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89" : [100, 100, 100, 100, 100, 150,
                          200, 300, 300, 300, 300, 250,
                          250, 300, 300, 300, 300, 300,
                          400, 250, 250, 200, 150, 150],
      "ipv4:198.51.100.34" : [ 80, 80, 80, 80, 150, 150,
                              250, 400, 400, 450, 400, 200,
                              200, 350, 400, 400, 400, 350,
                              500, 200, 200, 200, 100, 100],
      "ipv4:203.0.113.45" : [300, 400, 250, 250, 200, 150,
                              150, 100, 100, 100, 100, 100,
                              100, 100, 100, 100, 100, 150,

```

```

        200, 300, 300, 300, 300, 250],
    "ipv6:2001:db8::10" : [200, 250, 300, 300, 300, 300,
        250, 300, 300, 300, 300, 350,
        300, 400, 250, 150, 100, 100,
        100, 150, 200, 250, 250, 300]
    }
  }
}

```

When the Client gets the Calendar for "routingcost", it sees that the "calendar-start-time" is Monday at 00h00 GMT and member "repeated" is equal to '4'. It understands that the provided values are valid until Thursday included and will only need to get a Calendar update on Friday.

5.2.4. Use case and example: ECS with a multi-cost Calendar for routingcost and owdelay

In this example, it is assumed that the ALTO Server implements multi-cost capabilities, as specified in [RFC8189]. That is, an ALTO Client can request and receive values for several cost types in one single transaction. An illustrating use case is a path selection done on the basis of 2 metrics: routing cost and owdelay.

As in the previous example, the IRD indicates that the ALTO Server provides "routingcost" Calendars in terms of 24 time intervals of 1 hour (3600 seconds) each.

For metric "owdelay", the IRD indicates that the ALTO Server provides Calendars in terms of 12 time intervals values lasting each 5 minutes (300 seconds).

In the following example transaction, the ALTO Client sends its request on Tuesday July 1st 2019 at 13:15.

This example assumes that the values of metric "owdelay" and "routingcost" are encoded in 3 digits.

```

POST calendar/endpointcost/lookup HTTP/1.1
Host: custom.alto.example.com
Content-Length: 390
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type" : {},
  "multi-cost-types" : [
    {"cost-mode" : "numerical", "cost-metric" : "routingcost"},

```

```

    {"cost-mode" : "numerical", "cost-metric" : "owdelay"}
  ],
  "calendared" : [true, true],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2001:db8::10"
    ]
  }
}

```

HTTP/1.1 200 OK

Content-Length: 2165

Content-Type: application/alto-endpointcost+json

```

{
  "meta" : {
    "multi-cost-types" : [
      {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
      {"cost-mode" : "numerical", "cost-metric" : "owdelay"}
    ],
    "calendar-response-attributes" : [
      {"cost-type-names" : [ "num-routingcost" ],
        "calendar-start-time" : "Mon, 30 Jun 2019 00:00:00 GMT",
        "time-interval-size" : 3600,
        "number-of-intervals" : 24,
        "repeated": 4 },
      {"cost-type-names" : [ "num-owdelay" ],
        "calendar-start-time" : "Tue, 1 Jul 2019 13:00:00 GMT",
        "time-interval-size" : 300,
        "number-of-intervals" : 12}
    ]
  },
  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89" : [[100, 100, 100, 100, 100, 150,
        200, 300, 300, 300, 300, 250,
        250, 300, 300, 300, 300, 300,
        400, 250, 250, 200, 150, 150],
        [ 20, 400, 20, 80, 80, 90,
        100, 90, 60, 40, 30, 20]],
      "ipv4:198.51.100.34" : [[ 80, 80, 80, 80, 150, 150,
        250, 400, 400, 450, 400, 200,
        200, 350, 400, 400, 400, 350,
        500, 200, 200, 200, 100, 100],

```



```

        [ 20, 20, 50, 30, 30, 30,
          30, 40, 40, 30, 20, 20]],
    "ipv4:203.0.113.45" : [[300, 400, 250, 250, 200, 150,
        150, 100, 100, 100, 100, 100,
        100, 100, 100, 100, 100, 150,
        200, 300, 300, 300, 300, 250],
        [100, 90, 80, 60, 50, 50,
          40, 40, 60, 90, 100, 80]],
    "ipv6:2001:db8::10" : [[200, 250, 300, 300, 300, 300,
        250, 300, 300, 300, 300, 350,
        300, 400, 250, 150, 100, 100,
        100, 150, 200, 250, 250, 300],
        [ 40, 40, 40, 40, 50, 50,
          50, 20, 10, 15, 30, 40]]
    }
}
}

```

When receiving the response, the Client sees that the Calendar values for metric "routingcost" are repeated for 4 iterations. Therefore, in its next requests until the "routingcost" Calendar is expected to change, the Client will only need to request a Calendar for "owdelay".

Without the ALTO Calendar extensions, the ALTO Client would have no clue on the dynamicity of the metric value change and would spend needless time requesting values at an inappropriate pace. In addition, without the Multi-Cost ALTO capabilities, the ALTO Client would duplicate this waste of time as it would need to send one request per cost metric.

6. IANA Considerations

This document does not define any new media types or introduce any new IANA considerations.

7. Security Considerations

As an extension of the base ALTO protocol [RFC7285], this document fits into the architecture of the base protocol, and hence the Security Considerations (Section 15) of [RFC7285] fully apply when this extension is provided by an ALTO Server. For example, the same authenticity and integrity considerations (Section 15.1 of [RFC7285]) still fully apply; the same considerations for the privacy of ALTO users (Section 15.4 of [RFC7285]) also still fully apply.

The calendaring information provided by this extension requires additional considerations on three security considerations discussed

in [RFC7285]: potential undesirable guidance to Clients (Section 15.2 of [RFC7285]), confidentiality of ALTO information (Section 15.2 of [RFC7285]), and availability of ALTO (Section 15.5 of [RFC7285]). For example, by providing network information in the future in a Calendar, this extension may improve availability of ALTO, when the ALTO Server is unavailable but related information is already provided in the Calendar.

For confidentiality of ALTO information, an operator should be cognizant that this extension may introduce a new risk: a malicious ALTO Client may get information for future events that are scheduled through Calendaring. Possessing such information, the malicious Client may use it to generate massive connections to the network at times where its load is expected to be high.

To mitigate this risk, the operator should address the risk of ALTO information being leaked to malicious Clients or third parties. As specified in Section 15.3.2 ("Protection Strategies") of [RFC7285], the ALTO Server should authenticate ALTO Clients and use the Transport Layer Security (TLS) protocol so that Man In The Middle (MITM) attacks to intercept an ALTO Calendar are not possible. [RFC7285] ensures the availability of such a solution in its Section 8.3.5. "Authentication and Encryption", which specifies that: "ALTO Server implementations as well as ALTO Client implementations MUST support the "https" URI scheme of [RFC2818] and Transport Layer Security (TLS) of [RFC5246]".

[RFC8446] specifies TLS 1.3 and writes in its section 1: "While TLS 1.3 is not directly compatible with previous versions, all versions of TLS incorporate a versioning mechanism which allows Clients and Servers to interoperably negotiate a common version if one is supported by both peers". ALTO Clients and Servers SHOULD support both TLS 1.3 [RFC8446] and TLS 1.2 [RFC5246], and MAY support and use newer versions of TLS as long as the negotiation process succeeds.

The operator should be cognizant that the preceding mechanisms do not address all security risks. In particular, they will not help in the case of "malicious Clients" possessing valid authentication credentials. The threat here is that legitimate Clients have become subverted by an attacker and are now 'bots' being asked to participate in a DDoS attack. The Calendar information now becomes valuable in knowing exactly when to perpetrate a DDoS attack. A mechanism such as a monitoring system that detects abnormal behaviors may still be needed.

To avoid malicious or erroneous guidance from ALTO information, an ALTO Client should be cognizant that using calendaring information can have risks: (1) Calendar values, especially in "repeated"

Calendars may be only statistical, and (2) future events may change. Hence, a more robust ALTO Client should adapt and extend protection strategies specified in Section 15.2 of [RFC7285]. For example, to be notified immediately when a particular ALTO value that the Client depends on changes, it is RECOMMENDED that both the ALTO Client and ALTO Server using this extension support "ALTO Incremental Updates Using Server-Sent Events(SSE)" Service [I-D.ietf-alto-incr-update-sse].

Another risk of erroneous guidance appears when the Server exposes an occurrence of a same cost type name in different elements of the Calendar objects array associated to an information resource. In this case, there is no way for the Client to figure out which Calendar object in the array is valid. The specification in this document recommends, in this case, that the Client uses the first encountered Calendar object occurrence containing the cost type name. However, the Client may want to avoid the risks of erroneous guidance associated to the use of potentially invalid Calendar values. To this end, as an alternative to the recommendation in this document, the Client MAY ignore the totality of occurrences of CalendarAttributes objects containing the cost type name and query this cost type using [RFC7285].

8. Operational Considerations

It is important that both the operator of the network and the operator of the applications consider both the feedback aspect and the prediction-based (uncertainty) aspect of using the Cost Calendar.

First consider the feedback aspect and consider the Cost Calendar as a traffic-aware map service (e.g., Google Maps). Using the service without considering its own effect, a large fleet can turn a not-congested road into a congested one; a large number of individual cars each choosing a road with light traffic ("cheap link") can also result in congestion or result in a less optimal global outcome (e.g., the Braess' Paradox [Braess-paradox]).

Next consider the prediction aspect. Conveying ALTO Cost Calendars tends to reduce the on-the-wire data exchange volume compared to multiple single cost ALTO transactions. An application using Calendars has a set of time-dependent values upon which it can plan its connections in advance with no need for the ALTO Client to query information at each time. Additionally, the Calendar response attribute "repeated", when provided, saves additional data exchanges in that it indicates that the ALTO Client does not need to query Calendars during a period indicated by this attribute. The preceding is true only when "accidents" do not happen.

Although individual network operators and application operators can choose their own approaches to address the aforementioned issues, this document recommends the following considerations. First, a typical approach to reducing instability and handling uncertainty is to ensure timely update of information. The SSE Service as discussed in Section 7 can handle updates, if supported by both the Server and the Client. Second, when a network operator updates the Cost Calendar and when an application reacts to the update, they should consider the feedback effects. This is the best approach even though there is theoretical analysis [Selfish-routing-Roughgarden-thesis] and Internet based evaluation [Selfish-routing-Internet-eval] showing that uncoordinated behaviors do not always cause substantial sub-optimal results.

High-resolution intervals may be needed when values change, sometimes during very small time intervals but in a significant manner. A way to avoid conveying too many entries is to leverage on the "repeated" feature. A Server can smartly set the Calendar start time and number of intervals so as to declare them "repeated" for a large number of periods, until the Calendar values change and are conveyed to requesting Clients.

The newer JSON Data Interchange Format specification [RFC8259] used in ALTO Calendars replaces the older one [RFC7159] used in the base ALTO protocol [RFC7285]. The newer JSON mandates UTF-8 text encoding to improve interoperability. Therefore, ALTO Clients and Servers implementations using UTF-{16,32} need to be cognizant of the subsequent interoperability risks and MUST switch to UTF-8 encoding, if they want to interoperate with Calendar-aware Servers and Clients.

9. Acknowledgements

The authors would like to thank Fred Baker, Li Geng, Diego Lopez, He Peng and Haibin Song for fruitful discussions and feedback on earlier draft versions. Dawn Chan, Kai Gao, Vijay Gurbani, Yichen Qian, Juergen Schoenwaelder, and Brian Weis and Jensen Zhang provided substantial review feedback and suggestions to the protocol design.

10. References

10.1. Normative References

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

- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/info/rfc8189>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [I-D.ietf-alto-incr-update-sse]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-ietf-alto-incr-update-sse-20 (work in progress), February 2020.
- [IEEE.754.2008]
"Standard for Binary Floating-Point Arithmetic, IEEE Standard 754", August 2008.

10.2. Informative References

- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.

- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <<https://www.rfc-editor.org/info/rfc5693>>.
- [RFC6708] Kiesel, S., Ed., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, DOI 10.17487/RFC6708, September 2012, <<https://www.rfc-editor.org/info/rfc6708>>.
- [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>>.
- [I-D.ietf-alto-performance-metrics]
WU, Q., Yang, Y., Dhody, D., Randriamasy, S., and L. Contreras, "ALTO Performance Cost Metrics", draft-ietf-alto-performance-metrics-09 (work in progress), March 2020.
- [I-D.xiang-alto-multidomain-analytics]
Xiang, Q., Zhang, J., Le, F., Yang, Y., and H. Newman, "Resource Orchestration for Multi-Domain, Exascale, Geo-Distributed Data Analytics", draft-xiang-alto-multidomain-analytics-03 (work in progress), March 2020.
- [SENSE-sdn-e2e-net]
"SDN for End-to-End Networked Science at the Exascale (SENSE)", <http://sense.es.net/overview>.
- [Braess-paradox]
Steinberg, R. and W. Zangwill, "The Prevalence of Braess' Paradox", Transportation Science Vol. 17 No. 3, August 1983.
- [Unicorn-fgcs]
Xiang, Q., Wang, T., Zhang, J., Newman, H., and Y. Liu, "Unicorn: Unified resource orchestration for multi-domain, geo-distributed data analytics", Future Generation of Computer Systems (FGCS) Volume 93, Pages 188-197, April 2019.
- [Selfish-routing-Roughgarden-thesis]
Roughgarden, T., "Selfish Routing", Dissertation Thesis, Cornell 2002, May 2002.

[Selfish-routing-Internet-eval]

Qiu, L., Yang, Y., Zhang, Y., and S. Shenker, "Selfish Routing in Internet-Like Environments", Proceedings of ACM SIGCOMM 2001, August 2001.

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

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

Richard Yang
Yale University
51 Prospect st
New Haven, CT 06520
USA

Email: yry@cs.yale.edu

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2020

W. Roome
Nokia Bell Labs
Y. Yang
Yale University
March 20, 2020

ALTO Incremental Updates Using Server-Sent Events (SSE)
draft-ietf-alto-incr-update-sse-22

Abstract

The Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol provides network related information, called network information resources, to client applications so that clients can make informed decisions in utilizing network resources. This document presents a mechanism to allow an ALTO server to push updates to ALTO clients, to achieve two benefits: (1) updates can be incremental, in that if only a small section of an information resource changes, the ALTO server can send just the changes; and (2) updates can be immediate, in that the ALTO server can send updates as soon as they are available.

Requirements Language

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

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 September 21, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terms	4
3. Background	6
3.1. Incremental Encoding: JSON Merge Patch	6
3.1.1. JSON Merge Patch Encoding	6
3.1.2. JSON Merge Patch ALTO Messages	7
3.2. Incremental Encoding: JSON Patch	10
3.2.1. JSON Patch Encoding	11
3.2.2. JSON Patch ALTO Messages	11
3.3. Multiplexing and Server Push: HTTP/2	13
3.4. Server Push: Server-Sent Event	14
4. Overview of Approach and High-level Protocol Message Flow . .	15
4.1. Update Stream Service Message Flow	16
4.2. Stream Control Service Message Flow	17
4.3. Service Announcement and Management Message Flow	18
5. Update Messages: Data Update and Control Update Messages . .	19
5.1. Generic ALTO Update Message Structure	19
5.2. ALTO Data Update Message	19
5.3. ALTO Control Update Message	21
6. Update Stream Service	22
6.1. Media Type	22
6.2. HTTP Method	22
6.3. Capabilities	22
6.4. Uses	23
6.5. Request: Accept Input Parameters	23
6.6. Response	25
6.7. Additional Requirements on Update Stream Service	27
6.7.1. Event Sequence Requirements	27
6.7.2. Cross-Stream Consistency Requirements	27
6.7.3. Multipart Update Requirements	28
6.8. Keep-Alive Messages	28

7.	Stream Control Service	29
7.1.	URI	29
7.2.	Media Type	30
7.3.	HTTP Method	30
7.4.	IRD Capabilities & Uses	30
7.5.	Request: Accept Input Parameters	30
7.6.	Response	31
8.	Examples	32
8.1.	Example: IRD Announcing Update Stream Services	32
8.2.	Example: Simple Network and Cost Map Updates	35
8.3.	Example: Advanced Network and Cost Map Updates	38
8.4.	Example: Endpoint Property Updates	41
8.5.	Example: Multipart Message Updates	45
9.	Operation and Processing Considerations	47
9.1.	Considerations for Choosing Data Update Messages	47
9.2.	Considerations for Client Processing Data Update Messages	48
9.3.	Considerations for Updates to Filtered Cost Maps	49
9.4.	Considerations for Updates to Ordinal Mode Costs	50
9.5.	Considerations for SSE Text Formatting and Processing	50
10.	Security Considerations	51
10.1.	Update Stream Server: Denial-of-Service Attacks	51
10.2.	ALTO Client: Update Overloading or Instability	52
10.3.	Stream Control: Spoofed Control Requests and Information Breakdown	52
11.	Requirements on Future ALTO Services to Use this Design	52
12.	IANA Considerations	53
12.1.	application/alto-updatestreamparams+json Media Type	53
12.2.	application/alto-updatestreamcontrol+json Media Type	54
13.	Contributors	55
14.	Acknowledgments	55
15.	Appendix: Design Decision: Not Allowing Stream Restart	56
16.	References	57
16.1.	Normative References	57
16.2.	Informative References	57
	Authors' Addresses	58

1. Introduction

The Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol provides network related information called network information resources to client applications so that clients may make informed decisions in utilizing network resources. For example, an ALTO server provides network and cost maps, where a network map partitions the set of endpoints into a manageable number of sets each defined by a Provider-Defined Identifier (PID), and a cost map provides directed costs between PIDs. Given network and cost maps, an ALTO client can obtain costs between endpoints by first using the network map to get the PID for each endpoint, and then using the cost map to get the

costs between those PIDs. Such costs can be used by the client to choose communicating endpoints with low network costs.

The ALTO protocol defines only an ALTO client pull model, without defining a mechanism to allow an ALTO client to obtain updates to network information resources, other than by periodically re-fetching them. In settings where an information resource may be large but only parts of it may change frequently (e.g., some entries of a cost map), complete re-fetching can be inefficient.

This document presents a mechanism to allow an ALTO server to push incremental updates to ALTO clients. Integrating server-push and incremental updates provides two benefits: (1) updates can be small, in that if only a small section of an information resource changes, the ALTO server can send just the changes; and (2) updates can be immediate, in that the ALTO server can send updates as soon as they are available.

While primarily intended to provide updates to GET-mode network and cost maps, the mechanism defined in this document can also provide updates to POST-mode ALTO services, such as the ALTO endpoint property and endpoint cost services. The mechanism can also support new ALTO services to be defined by future extensions, but a future service needs to satisfy requirements specified in Section 11.

The rest of this document is organized as follows. Section 3 gives background on the basic techniques used in this design: (1) JSON merge patch and JSON patch to allow incremental update; and (2) Server-Sent Events (SSE) [SSE] to allow server push. With the background, Section 4 gives a non-normative overview of the design. Section 5 defines individual messages in an update stream. Section 6 defines the update stream service; Section 7 defines the stream control service; Section 8 gives several examples to illustrate the two types of services. Section 9 describes operation and processing considerations by both ALTO servers and clients; Section 15 discusses a design feature that is not supported; Section 10 discusses security issues; Section 11 and Section 12 review the requirements for future ALTO services to use SSE and IANA considerations, respectively.

2. Terms

Besides the terminologies as defined in [RFC7285], this document also uses additional terminologies defined as follows:

Update Stream: A reliable, in-order HTTP/1.x compatible connection between an ALTO client and an ALTO server so that the server can push a sequence of update messages using [SSE] to the client.

Update Stream Server: This document refers to an ALTO server providing an update stream as an ALTO update stream server, or update stream server for short. Note that the ALTO server mentioned in this document refers to a general server that provides various kinds of services; it can be an update stream server or stream control server (see below); it can also be a server providing ALTO Information Resource Directory (IRD).

Update Message: A message that is either a data update message or a control update message.

Data Update Message: An update message that is for a single ALTO information resource and sent from the update stream server to the ALTO client when the resource changes. A data update message can be either a full-replacement message or an incremental-change message. Full replacement is a shorthand for a full-replacement message, and incremental change is a shorthand for an incremental-change message.

Full Replacement: A data update message for a resource that encodes the content of the resource in its original ALTO encoding.

Incremental Change: An data update message that specifies only the difference between the new content and the previous version. An incremental change can be encoded using either JSON merge patch or JSON patch in this document.

Stream Control Service: A service that provides an HTTP URI so that the ALTO client of an update stream can use it to send stream control requests to the ALTO server on the addition or removal of resources receiving update messages from the update stream. The ALTO server creates a new stream control resource for each update stream instance, assigns a unique URI to it, and sends the URI to the client as the first event in the stream. (Note that the Stream Control Service in ALTO has no association with the similarly named Stream Control Transmission Protocol [RFC4960].)

Stream Control: A shorthand for stream control service.

Stream Control Server: An ALTO server providing the stream control service.

Substream-ID: An ALTO client can assign a unique substream-id when requesting the addition of a resource receiving update messages from an update stream. The server puts the substream-id in each update event for that resource. Substream-id allows a client to use one update stream to receive updates to multiple requests for the same resource (i.e., with the same resource-id in an ALTO IRD), for example, for a POST-mode resource with different input parameters.

Data-ID: A subfield of the 'event' field of [SSE] to identify the ALTO data (object) to be updated. For an ALTO resource returning a multipart response, the data-id to identify the data (object) is the substream-id, in addition to the content-id of the object in the multipart response. The data-id of a single part response is just the substream-id.

Control Update Message: An update message for the update stream server to notify the ALTO client of related control information of the update stream. A control update message may be triggered by an internal event at the server, such as server overloading and hence the update stream server will no longer send updates for an information resource, or as a result of a client sending a request through the stream control service. The first message of an update stream is a control update message and provides the URI using which the ALTO client can send stream control requests to the stream control server.

3. Background

The design requires two basic techniques: encoding of incremental changes and server push. For incremental changes, existing techniques include JSON merge patch and JSON patch; this design uses both. For server push, existing techniques include HTTP/2 and [SSE]; this design adopts some design features of HTTP/2 but uses [SSE] as the basic server-push design. The rest of this section gives a non-normative summary of JSON merge patch, JSON patch, HTTP/2 and [SSE].

3.1. Incremental Encoding: JSON Merge Patch

To avoid always sending complete data, a server needs mechanisms to encode incremental changes, and JSON merge patch is one mechanism. [RFC7396] defines the encoding of incremental changes (called JSON merge patch objects) to be used by the HTTP PATCH method [RFC5789]. This document adopts from [RFC7396] only the JSON merge patch object encoding and does not use the HTTP PATCH method, as the updates are sent as events, instead of HTTP methods; also the updates are server-to-client in the updates, and PATCH semantics is more for client-to-server. Below is a non-normative summary of JSON merge patch objects; see [RFC7396] for the normative definition.

3.1.1. JSON Merge Patch Encoding

Informally, a JSON merge patch message consists of a JSON merge patch object (referred to as a patch in [RFC7396]), which defines how to transform one JSON value into another using a recursive merge patch algorithm. Specifically, the patch is computed by treating two JSON values (first one being the original, and the second being the

updated) as trees of nested JSON objects (dictionaries of name-value pairs), where the leaves are values (e.g., JSON arrays, strings, numbers) other than JSON objects and the path for each leaf is the sequence of keys leading to that leaf. When the second tree has a different value for a leaf at a path, or adds a new leaf, the patch has a leaf, at that path, with the new value. When a leaf in the first tree does not exist in the second tree, the JSON merge patch tree has a leaf with a JSON "null" value. Hence, in the patch, null as the value of a name/value pair will delete the element with "name" in the original JSON value. The patch does not have an entry for any leaf that has the same value in both versions. See the MergePatch pseudocode at the beginning of Section 2 of [RFC7396] for the formal specification of how to apply a given patch. As a result, if all leaf values are simple scalars, JSON merge patch is a quite efficient representation of incremental changes. It is less efficient when leaf values are arrays, because JSON merge patch replaces arrays in their entirety, even if only one entry changes.

3.1.2. JSON Merge Patch ALTO Messages

To provide both examples of JSON merge patch and a demonstration of the feasibility of applying JSON merge patch to ALTO, the sections below show the application of JSON merge patch to two key ALTO messages.

3.1.2.1. JSON Merge Patch Network Map Messages

Section 11.2.1.6 of [RFC7285] defines the format of an ALTO network map message. Assume a simple example ALTO message sending an initial network map:

```

{
  "meta" : {
    "vtag": {
      "resource-id" : "my-network-map",
      "tag" : "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "network-map" : {
    "PID1" : {
      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25" ]
    },
    "PID2" : {
      "ipv4" : [ "198.51.100.128/25" ]
    },
    "PID3" : {
      "ipv4" : [ "0.0.0.0/0" ],
      "ipv6" : [ "::/0" ]
    }
  }
}

```

Consider the following JSON merge patch update message, which (1) adds an ipv4 prefix "203.0.113.0/25" and an ipv6 prefix "2001:db8:8000::/33" to "PID1", (2) deletes "PID2", and (3) assigns a new "tag" to the network map:

```

{
  "meta" : {
    "vtag" : {
      "tag" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
    }
  },
  "network-map": {
    "PID1" : {
      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25",
                  "203.0.113.0/25" ],
      "ipv6" : [ "2001:db8:8000::/33" ]
    },
    "PID2" : null
  }
}

```

Applying the JSON merge patch update to the initial network map is equivalent to the following ALTO network map:

```

{
  "meta" : {
    "vtag": {
      "resource-id" : "my-network-map",
      "tag" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
    }
  },
  "network-map" : {
    "PID1" : {
      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25",
                  "203.0.113.0/25" ],
      "ipv6" : [ "2001:db8:8000::/33" ]
    },
    "PID3" : {
      "ipv4" : [ "0.0.0.0/0" ],
      "ipv6" : [ "::/0" ]
    }
  }
}

```

3.1.2.2. JSON Merge Patch Cost Map Messages

Section 11.2.3.6 of [RFC7285] defines the format of an ALTO cost map message. Assume a simple example ALTO message for an initial cost map:

```

{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "my-network-map",
        "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
      }
    ],
    "cost-type" : {
      "cost-mode" : "numerical",
      "cost-metric": "routingcost"
    },
    "vtag": {
      "resource-id" : "my-cost-map",
      "tag" : "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
    }
  },
  "cost-map" : {
    "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
    "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
    "PID3": { "PID1": 20, "PID2": 15 }
  }
}

```


The following JSON merge patch message updates the example cost map so that (1) the "tag" field of the cost map is updated, (2) the cost of PID1->PID2 is 9 instead of 5, (3) the cost of PID3->PID1 is no longer available, and (4) the cost of PID3->PID3 is defined as 1.

```
{
  "meta" : {
    "vtag": {
      "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
    }
  }
  "cost-map" : {
    "PID1" : { "PID2" : 9 },
    "PID3" : { "PID1" : null, "PID3" : 1 }
  }
}
```

Hence applying the JSON merge patch to the initial cost map is equivalent to the following ALTO cost map:

```
{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "my-network-map",
        "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
      }
    ],
    "cost-type" : {
      "cost-mode" : "numerical",
      "cost-metric": "routingcost"
    },
    "vtag": {
      "resource-id": "my-cost-map",
      "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
    }
  },
  "cost-map" : {
    "PID1": { "PID1": 1, "PID2": 9, "PID3": 10 },
    "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
    "PID3": { "PID2": 15, "PID3": 1 }
  }
}
```

3.2. Incremental Encoding: JSON Patch

3.2.1. JSON Patch Encoding

One issue of JSON merge patch is that it does not handle array changes well. In particular, JSON merge patch considers an array as a single object and hence can only replace an array in its entirety. When the change is to make a small change to an array such as the deletion of an element from a large array, whole-array replacement is inefficient. Consider the example in Section 3.1.2.1. To add a new entry to the `ipv4` array for `PID1`, the server needs to send a whole new array. Another issue is that JSON merge patch cannot change a value to be null, as the JSON merge patch processing algorithm (`MergePatch` in Section 3.1.1) interprets a null as a removal instruction. On the other hand, some ALTO resources can have null values, and it is possible that the update will want to change the new value to be null.

JSON patch [RFC6902] can address the preceding issues. It defines a set of operators to modify a JSON object. See [RFC6902] for the normative definition.

3.2.2. JSON Patch ALTO Messages

To provide both examples of JSON patch and a demonstration of the difference between JSON patch and JSON merge patch, the sections below show the application of JSON patch to the same updates shown in Section 3.1.2.

3.2.2.1. JSON Patch Network Map Messages

First consider the same update as in Section 3.1.2.1 for the network map. Below is the encoding using JSON patch:

```
[
  {
    "op": "replace",
    "path": "/meta/vtag/tag",
    "value": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
  },
  {
    "op": "add",
    "path": "/network-map/PID1/ipv4/2",
    "value": "203.0.113.0/25"
  },
  {
    "op": "add",
    "path": "/network-map/PID1/ipv6",
    "value": ["2001:db8:8000::/33"]
  },
  {
    "op": "remove",
    "path": "/network-map/PID2"
  }
]
```

3.2.2.2. JSON Patch Cost Map Messages

Compared with JSON merge patch, JSON patch does not encode cost map updates efficiently. Consider the cost map update shown in Section 3.1.2.2, the encoding using JSON patch is:

```
[
  {
    "op": "replace",
    "path": "/meta/vtag/tag",
    "value": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
  },
  {
    "op": "replace",
    "path": "/cost-map/PID1/PID2",
    "value": 9
  },
  {
    "op": "remove",
    "path": "/cost-map/PID3/PID1"
  },
  {
    "op": "replace",
    "path": "/cost-map/PID3/PID3",
    "value": 1
  }
]
```

3.3. Multiplexing and Server Push: HTTP/2

HTTP/2 ([RFC7540]) provides two related features: multiplexing and server push. In particular, HTTP/2 allows a client and a server to multiplex multiple HTTP requests and responses over a single TCP connection. The requests and responses can be interleaved on a block (frame) by block (frame) basis, by indicating the requests and responses in HTTP/2 messages, avoiding the head-of-line blocking problem encountered with HTTP/1.1. To achieve the same goal, this design introduces substream-id to allow a client to receive updates to multiple resources. HTTP/2 also provides a Server Push facility, to allow a server to send asynchronous updates.

Despite the two features of HTTP/2, this design chooses an HTTP/1.x-compatible design for the simplicity of HTTP/1.x. An HTTP/2 based design may more likely need to be implemented using a more complex HTTP/2 client library. In such a case, one approach for using Server Push for updates is for the update stream server to send each data update message as a separate Server Push item and let the client apply those updates as they arrive. An HTTP/2 client library may not necessarily inform a client application when the server pushes a resource. Instead, the library might cache the pushed resource, and only deliver it to the client when the client explicitly requests that URI. Further, it is more likely that an HTTP/2 based design may encounter issues with a proxy between the client and the server, in that Server Push is optional and can be

disabled by any proxy between the client and the server. This is not a problem for the intended use of Server Push: eventually the client will request those resources, so disabling Server Push just adds a delay. But this means that Server Push is not suitable for resources which the client does not know to request.

Thus this design leaves an HTTP/2 based design as a future work and focuses on ALTO updates on HTTP/1.x and [SSE].

3.4. Server Push: Server-Sent Event

Server-Sent Events (SSE) is a technique which can work with HTTP/1.1. The following is a non-normative summary of SSE; see [SSE] for its normative definition.

SSE enable a server to send new data to a client by "server-push". The client establishes an HTTP ([RFC7230], [RFC7231]) connection to the server and keeps the connection open. The server continually sends messages. Each message has one or more lines, where a line is terminated by a carriage-return immediately followed by a new-line, a carriage-return not immediately followed by a new-line, or a new-line not immediately preceded by a carriage-return. A message is terminated by a blank line (two line terminators in a row).

Each line in a message is of the form "field-name: string value". Lines with a blank field-name (that is, lines which start with a colon) are ignored, as are lines which do not have a colon. The protocol defines three field names: event, id, and data. If a message has more than one "data" line, the value of the data field is the concatenation of the values on those lines. There can be only one "event" and "id" line per message. The "data" field is required; the others are optional.

Figure 1 is a sample SSE stream, starting with the client request. The server sends three events and then closes the stream.

```
(Client request)
GET /stream HTTP/1.1
Host: example.com
Accept: text/event-stream

(Server response)
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: start
id: 1
data: hello there

event: middle
id: 2
data: let's chat some more ...
data: and more and more and ...

event: end
id: 3
data: goodbye
```

Figure 1: A Sample SSE stream.

4. Overview of Approach and High-level Protocol Message Flow

With the preceding background, this section now gives a non-normative overview of the update mechanisms and message flow to be defined in later sections of this document. Figure 2 gives the main components and overall message flow.

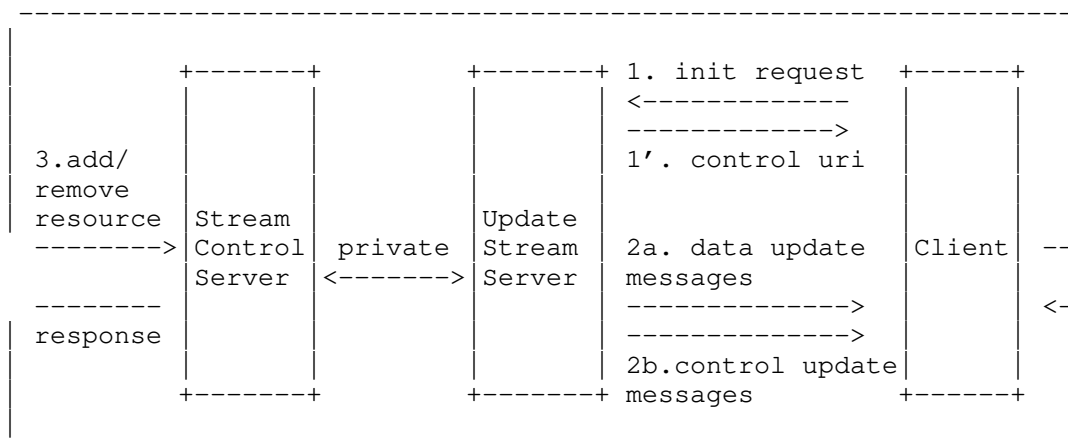


Figure 2: ALTO SSE Architecture and Message Flow.

4.1. Update Stream Service Message Flow

The building block of the update mechanism defined in this document is the update stream service (defined in Section 6), where each update stream service is a POST-mode service that provides update streams.

Note that the lines of the format **"** ... **"** are used to describe message flows in this section and the following sections.

**** Initial request: client -> update server **:**

When an ALTO client requests an update stream service, the ALTO client establishes a persistent connection to the update stream server and submits an initial update-stream request (defined in Section 6.5), creating an update stream. This initial request creating the update stream is labeled "1. init request" in Figure 2.

An update stream can provide updates to both GET-mode resources, such as ALTO network and cost maps, and POST-mode resources, such as ALTO endpoint property service. Also, to avoid creating too many update streams, this design allows an ALTO client to use one update stream to receive updates to multiple requests. In particular, the client may request to receive updates for the same resource but with different parameters for a POST-mode resource, in addition to being able to consolidate updates for multiple resources into a single stream. The updates for each request is called a substream, and hence, the update server needs an identifier to indicate the

substream when sending an update. To achieve this goal, the client assigns a unique substream-id when requesting updates to a resource in an update stream, and the server puts the substream-id in each update.

**** Data updates: update server -> client **:**

The objective of an update stream is to continuously push to an ALTO client data value changes to a set of resources, where the set of resources is specified by the ALTO client's requests. This document refers to messages sending such data-value changes as data update messages (defined in Section 5.2). Although an update stream may update one or more requests, each data update message updates only one request and is sent as a Server-Sent Event (SSE), as defined by [SSE]. A data update message is encoded either as a full replacement or as an incremental change. A full replacement uses the JSON message format defined by the ALTO protocol. There can be multiple encodings for incremental changes. The current design supports incremental changes using JSON merge patch ([RFC7396]) or JSON patch ([RFC6902]) to describe the changes of the resource. Future documents may define additional mechanisms for incremental changes. The update stream server decides when to send data update messages, and whether to send full replacements or incremental changes. These decisions can vary from resource to resource and from update to update. Since the transport is a HTTP/1.x compatible design, data update messages are delivered reliably and in order, and the lossless, sequential delivery of its messages allows the server to know the exact state of the client to compute the correct incremental updates. Figure 2 shows examples of data update messages (labeled "2a. data update messages") in the overall message flow.

**** Control updates: update server -> client **:**

An update stream can run for a long time, and hence there can be status changes at the update stream server side during the lifetime of an update stream; for example, the update stream server may encounter an error or need to shut down for maintenance. To support robust, flexible protocol design, this document allows the update stream server to send control update messages (defined in Section 5.3) in addition to data update messages to the ALTO client. Figure 2 shows that both data updates and control updates can be sent by the server to the client (labeled "2b. control update messages").

4.2. Stream Control Service Message Flow

**** Stream control: client -> stream control server **:**

In addition to control changes triggered from the update stream server side, in a flexible design, an ALTO client may initiate control changes as well, in particular, by adding or removing ALTO resources receiving updates. An ALTO client initiates such changes using the stream control service (defined in Section 7). Although one may use a design that the client uses the same HTTP connection to send the control requests, it requires stronger server support such as HTTP pipeline. For more flexibility, this document introduces stream control service. In particular, the update stream server of an update stream uses the first message to provide the URI of the stream control service (labeled "1': control uri" in Figure 2).

The ALTO client can then use the URI to ask the stream control server specified in the URI to request the update stream server to (1) send data update messages for additional resources, (2) stop sending data update messages for previously requested resources, or (3) gracefully stop and close the update stream altogether.

4.3. Service Announcement and Management Message Flow

**** Service announcements: IRD server -> client **:**

An update server may provide any number of update stream services, where each update stream may provide updates for a given subset of the ALTO server's resources. An ALTO server's Information Resource Directory (IRD) defines the update stream services and declares the set of resources for which each update stream service provides updates. The ALTO server selects the resource set for each update stream service. It is recommended that if a resource depends on one or more other resource(s) (indicated with the "uses" attribute defined in [RFC7285]), these other resource(s) should also be part of that update stream. Thus the update stream for a cost map should also provide updates for the network map on which that cost map depends.

**** Service management (server) **:**

An ALTO client may request any number of update streams simultaneously. Because each update stream consumes resources on the update stream server, an update stream server may require client authorization and/or authentication, limit the number of open update streams, close inactive streams, or redirect an ALTO client to another update stream server.

5. Update Messages: Data Update and Control Update Messages

This section defines the format of update messages sent from the server to the client. It first defines the generic structure of update messages (Section 5.1). It then defines the details of the data update messages (Section 5.2) and the control update messages (Section 5.3). These messages will be used in the next two sections to define the Update Stream Service (Section 6) and the Stream Control Service (Section 7).

5.1. Generic ALTO Update Message Structure

Both data update and control update messages from the server to the client have the same basic structure: each message includes a data field to provide data information, which is typically a JSON object; and an event field preceding the data field, to specify the media type indicating the encoding of the data field.

A data update message needs additional information to identify the ALTO data (object) to which the update message applies. To be generic, this document use a data-id to identify the ALTO data (object) to be updated; see below.

Hence, the event field of ALTO update message can include two sub-fields (media-type and data-id), where the two sub-fields are separated by a comma (',', U+002C):

```
media-type [ ',' data-id ]
```

According to Section 4.2 of [RFC6838], the comma character is not allowed in a media-type name. So there is no ambiguous when decoding of the two sub-fields.

Note that an update message does not use the SSE "id" field.

5.2. ALTO Data Update Message

A data update message is sent when a monitored resource changes. As discussed in the preceding section, the event field of a data update message includes two sub-fields: 'media-type' and 'data-id'.

The 'media-type' sub-field depends on whether the data update is a complete specification of the identified data, or an incremental patch (e.g., a JSON merge patch or JSON patch), if possible, describing the changes from the last version of the data. This document refers to these as full replacement and incremental change, respectively. The encoding of a full replacement is defined by its defining document (e.g., network and cost map messages by [RFC7285]),

and uses the media type defined in that document. The encoding of JSON merge patch is defined by [RFC7396], with the media type "application/merge-patch+json"; the encoding of JSON patch is defined by [RFC6902], with media type "application/json-patch+json".

The 'data-id' sub-field identifies the ALTO data to which the data update message applies.

First consider the case that the resource containing only a single JSON object. For example, since an ALTO client can request data updates for both a cost map resource (object) and its dependent network map resource (object) in the same update stream, to distinguish the updates, the client assigns a substream-id for each resource receiving data updates. Substream-ids MUST be unique within an update stream, but need not be globally unique. A substream-id is encoded as a JSON string with the same format as that of the type ResourceID (Section 10.2 of [RFC7285]). The type SubstreamID is used in this document to indicate a string of this format. The substream-id of a single JSON object is the 'data-id'.

As an example, assume that the ALTO client assigns substream-id "1" in its request to receive updates to the network map; and substream-id "2" to the cost map. Then the substream-ids are the data-ids indicating which objects will be updated. Figure 3 shows some examples of ALTO data update messages:

```
event: application/alto-networkmap+json,1
data: { ... full network map message ... }

event: application/alto-costmap+json,2
data: { ... full cost map message ... }

event: application/merge-patch+json,2
data: { ... JSON merge patch update for the cost map ... }
```

Figure 3: Examples of ALTO data update messages.

Next consider the case that a resource may include multiple JSON objects. This document considers the case that a resource may contain multiple components (parts) and they are encoded using the media type "multipart/related" [RFC2387]. Each part of this multipart response MUST be an HTTP message including a Content-ID header and a JSON object body. Each component requiring the update stream service (defined in Section 6) MUST be identified by a unique Content-ID to be defined in its defining document.

For a resource using the media type "multipart/related", the 'data-id' sub-field MUST be the concatenation of the substream-id, the '.' separator (U+002E) and the unique Content-ID in order.

5.3. ALTO Control Update Message

Control update messages have the media type "application/alto-updatestreamcontrol+json", and the data is of type UpdateStreamControlEvent:

```
object {  
  [String          control-uri;]  
  [SubstreamID     started<1..*>;]  
  [SubstreamID     stopped<1..*>;]  
  [String          description;]  
} UpdateStreamControlEvent;
```

control-uri: the URI providing stream control for this update stream (see Section 7). The server sends a control update message notifying the client of the control-uri. This control update message notifying the control-uri will be sent once and MUST be the first event in an update stream. If the URI value is NULL, the update stream server does not support stream control for this update stream; otherwise, the update stream server provides stream control through the given URI.

started: a list of substream-ids of resources. It notifies the ALTO client that the update stream server will start sending data update messages for each resource listed.

stopped: a list of substream-ids of resources. It notifies the ALTO client that the update stream server will no longer send data update messages for the listed resources. There can be multiple reasons for an update stream server to stop sending data update messages for a resource, including a request from the ALTO client using stream control (Section 6.7.1) or an internal server event.

description: a non-normative, human-readable text providing an explanation for the control event. When an update stream server stops sending data update messages for a resource, it is RECOMMENDED that the update stream server use the description field to provide details. There can be multiple reasons which trigger a "stopped" event; see above. The intention of this field is to provide a human-readable text for the developer and/or the administrator to diagnose potential problems.

6. Update Stream Service

An update stream service returns a stream of update messages, as defined in Section 5. An ALTO server's IRD (Information Resource Directory) MAY define one or more update stream services, which ALTO clients use to request new update stream instances. An IRD entry defining an update stream service MUST define the media type, HTTP method, and capabilities & uses as follows.

6.1. Media Type

The media type of an ALTO update stream service is "text/event-stream", as defined by [SSE].

6.2. HTTP Method

An ALTO update stream service is requested using the HTTP POST method.

6.3. Capabilities

The capabilities are defined as an object of type UpdateStreamCapabilities:

```
object {  
  IncrementalUpdateMediaTypes incremental-change-media-types;  
  Boolean support-stream-control;  
} UpdateStreamCapabilities;  
  
object-map {  
  ResourceID -> String;  
} IncrementalUpdateMediaTypes;
```

If this update stream can provide data update messages with incremental changes for a resource, the "incremental-change-media-types" field has an entry for that resource-id, and the value is the supported media types of the incremental change separated by commas. Normally this will be "application/merge-patch+json", "application/json-patch+json", or "application/merge-patch+json,application/json-patch+json", because, as described in Section 5, they are the only incremental change types defined by this document. However future extensions may define other types of incremental changes.

When choosing the media-types to encode incremental changes for a resource, the update stream server MUST consider the limitations of the encoding. For example, when a JSON merge patch specifies that the value of a field is null, its semantics is that the field is removed from the target, and hence the field is no longer defined

(i.e., undefined); see the MergePatch algorithm in Section 3.1.1 on how null value is processed. This, however, may not be the intended result for the resource, when null and undefined have different semantics for the resource. In such a case, the update stream server MUST choose JSON patch over JSON merge patch, if JSON patch is indicated as a capability of the update stream server; If the the server does not support JSON patch to handle such a case, the server then need to send a full replacement.

The "support-stream-control" field specifies whether the given update stream supports stream control. If "support-stream-control" field is "true", the update stream server will use the stream control specified in this document; else, the update stream server may use other mechanisms to provide the same functionality as stream control.

6.4. Uses

The "uses" attribute MUST be an array with the resource-ids of every resource for which this update stream can provide updates. Each resource specified in the "uses" MUST support full replacement: the update stream server can always send full replacement, and the ALTO client MUST accept full replacement.

This set may be any subset of the ALTO server's resources, and may include resources defined in linked IRDs. However, it is RECOMMENDED that the ALTO server selects a set that is closed under the resource dependency relationship. That is, if an update stream's "uses" set includes resource R1, and resource R1 depends on ("uses") resource R0, then the update stream's "uses" set SHOULD include R0 as well as R1. For example, an update stream for a cost map SHOULD also provide updates for the network map upon which that cost map depends.

6.5. Request: Accept Input Parameters

An ALTO client specifies the parameters for the new update stream by sending an HTTP POST body with the media type "application/alto-updatestreamparams+json". That body contains a JSON Object of type UpdateStreamReq, where:

```
object {  
  [AddUpdatesReq    add;]  
  [SubstreamID      remove<0...*>;]  
} UpdateStreamReq;  
  
object-map {  
  SubstreamID -> AddUpdateReq;  
} AddUpdatesReq;  
  
object {  
  ResourceID    resource-id;  
  [JSONString   tag;]  
  [Boolean      incremental-changes;]  
  [Object       input;]  
} AddUpdateReq;
```

add: specifies the resources (and the parameters for the resources) for which the ALTO client wants updates. In the scope of the same update stream, the ALTO client **MUST** assign a substream-id that is unique in the scope of the update stream (Section 5.2) for each entry, and use those substream-ids as the keys in the "add" field.

resource-id: the resource-id of an ALTO resource, and **MUST** be in the update stream's "uses" list (Section 6.4). If the resource-id is a GET-mode resource with a version tag (or "vtag"), as defined in Section 6.3 and Section 10.3 of [RFC7285], and the ALTO client has previously retrieved a version of that resource from the update stream server, the ALTO client **MAY** set the "tag" field to the tag part of the client's version of that resource. If that version is not current, the update stream server **MUST** send a full replacement before sending any incremental changes, as described in Section 6.7.1. If that version is still current, the update stream server **MAY** omit the initial full replacement.

incremental-changes: the ALTO client specifies whether it is willing to receive incremental changes from the update stream server for this substream. If the "incremental-changes" field is "true", the update stream server **MAY** send incremental changes for this substream. In this case, the client **MUST** support all incremental methods from the set announced in the server's capabilities for this resource; see Section 6.3 for server's announcement of potential incremental methods. If a client does not support all incremental methods from the set announced in the server's capabilities, the client can set "incremental-changes" to "false", and the update stream server then **MUST NOT** send incremental changes for that substream. The default value

for "incremental-changes" is "true", so to suppress incremental changes, the ALTO client MUST explicitly set "incremental-changes" to "false". An alternative design of incremental-changes control is a more fine-grained control, by allowing a client to select a subset of incremental methods from the set announced in the server's capabilities. But this alternative design is not adopted in this document, because it adds complexity to the server, which is more likely to be the bottleneck. Note that the ALTO client cannot suppress full replacement. When the ALTO client sets "incremental-changes" to "false", the update stream server MUST send a full replacement instead of an incremental change to the ALTO client. The update stream server MAY wait until more changes are available, and send a single full replacement with those changes. Thus an ALTO client which declines to accept incremental changes may not get updates as quickly as an ALTO client which does.

input: If the resource is a POST-mode service which requires input, the ALTO client MUST set the "input" field to a JSON Object with the parameters that the resource expects.

remove: it is used in update stream control requests (Section 7), and is not allowed in the update stream request. The update stream server SHOULD ignore this field if it is included in the request.

If a request has any errors, the update stream server MUST NOT create an update stream. Also, the update stream server will send an error response to the ALTO client as specified in Section 6.6.

6.6. Response

If the update stream request has any errors, the update stream server MUST return an HTTP "400 Bad Request" to the ALTO client. The body part of the HTTP response is the JSON object defined in Section 8.5.2 in [RFC7285]. Hence, an ALTO error response has the format:


```
HTTP/1.1 400 Bad Request
Content-Length: 131
Content-Type: application/alto-error+json
Connection: Closed
```

```
{
  "meta":{
    "code": "E_INVALID_FIELD_VALUE",
    "field": "add/my-network-map/resource-id",
    "value": "my-networkmap/#"
  }
}
```

Note that "field" and "value" are optional fields. If the "value" field exists, the "field" field MUST exist.

- o If an update stream request does not have an "add" field specifying one or more resources, the error code of the error message MUST be E_MISSING_FIELD and the "field" field SHOULD be "add". The update stream server MUST close the stream without sending any events.
- o If the "resource-id" field is invalid, or is not associated with the update stream, the error code of the error message MUST be E_INVALID_FIELD_VALUE; the "field" field SHOULD be the full path of the "resource-id" field and the "value" field SHOULD be the invalid resource-id. If there are more than one invalid resource-ids, the update stream server SHOULD pick one and return it. The update stream server MUST close the stream (i.e., TCP connection) without sending any events.
- o If the resource is a POST-mode service which requires input, the client MUST set the "input" field to a JSON Object with the parameters that that resource expects. If the "input" field is missing or invalid, the update stream server MUST return the same error response that that resource would return for missing or invalid input (see [RFC7285]). In this case, the update stream server MUST close the update stream without sending any events. If the input for several POST-mode resources are missing or invalid, the update stream server MUST pick one and return it.

The response to a valid request is a stream of update messages. Section 5 defines the update messages, and [SSE] defines how they are encoded into a stream.

An update stream server SHOULD send updates only when the underlying values change. However, it may be difficult for an update stream

server to guarantee that in all circumstances. Therefore a client MUST NOT assume that an update message represents an actual change.

6.7. Additional Requirements on Update Stream Service

6.7.1. Event Sequence Requirements

- o The first event MUST be a control update message with the URI of the update stream control service (see Section 7) for this update stream. Note that the value of the control-uri can be "null", indicating that there is no control stream service.
- o As soon as possible after the ALTO client initiates the connection, the update stream server checks the "tag" field for each added update request. If the "tag" field is not specified in an added update request, the update stream server MUST first send a full replacement for the request. If the "tag" field is specified, the client can accept incremental changes, and the server can compute an incremental update based on the "tag" (the server needs to ensure that for a POST resource with input, the "tag" should indicate the correct result for different inputs), the update stream server MAY omit the initial full replacement.
- o If this update stream provides updates for resource-ids R0 and R1, and if R1 depends on R0, then the update stream server MUST send the update for R0 before sending the related updates for R1. For example, suppose an update stream provides updates to a network map and its dependent cost maps. When the network map changes, the update stream server MUST send the network map update before sending the cost map updates.
- o When the ALTO client uses the stream control service to stop updates for one or more resources (Section 7), the ALTO client MUST send a stream control request. The update stream server MUST send a control update message whose "stopped" field has the substream-ids of all stopped resources.

6.7.2. Cross-Stream Consistency Requirements

If multiple ALTO clients create multiple update streams from the same update stream resource, and with the same update request parameters (i.e., same resource, same input), the update stream server MUST send the same updates to all of them. However, the update stream server MAY pack data items into different patch events, as long as the net result of applying those updates is the same.

For example, suppose two different ALTO clients create two different update streams for the same cost map, and suppose the update stream

server processes three separate cost point updates with a brief pause between each update. The server **MUST** send all three new cost points to both clients. But the update stream server **MAY** send a single patch event (with all three cost points) to one ALTO client, while sending three separate patch events (with one cost point per event) to the other ALTO client.

A update stream server **MAY** offer several different update stream resources that provide updates to the same underlying resource (that is, a resource-id may appear in the "uses" field of more than one update stream resource). In this case, those update stream resources **MUST** return the same update.

6.7.3. Multipart Update Requirements

This design allows any valid media type for full replacement. Hence, it supports ALTO resources using multipart to contain multiple JSON objects. This realizes the push benefit, but not the incremental encoding benefit of SSE.

JSON patch and merge patch provide the incremental encoding benefit but can be applied to only a single JSON object. If an update stream service supports a resource providing a multipart media type, which we refer to as a multipart resource, then the update stream service needs to handle the issue that the message of a full multipart resource can include multiple JSON objects. To address the issue, when an update stream service specifies that it supports JSON patch or merge patch incremental updates for a multipart resource, the service **MUST** ensure that (1) each part of a multipart message is a single JSON object, (2) each part is specified by a static content-id in the initial full message, (3) each data update event applies to only one part; and (4) each data update specifies substream-id.content-id as the 'event' field of the event, to identify the part to be updated.

6.8. Keep-Alive Messages

In an SSE stream, any line which starts with a colon (U+003A) character is a comment, and an ALTO client **MUST** ignore that line ([SSE]). As recommended in [SSE], an update stream server **SHOULD** send a comment line (or an event) every 15 seconds to prevent ALTO clients and proxy servers from dropping the HTTP connection. Note that although TCP also provides a Keep-alive function, the interval between TCP Keep-alive messages can depend on the OS configuration and varies. The preceding recommended SSE keep-alive allows the SSE client to detect the status of the update stream server with more certainty.

7. Stream Control Service

A stream control service allows an ALTO client to remove resources from the set of resources that are monitored by an update stream, or add additional resources to that set. The service also allows an ALTO client to gracefully shut down an update stream.

When an update stream server creates a new update stream, and if the update stream server supports stream control for the update stream, the update stream server creates a stream control service for that update stream. An ALTO client uses the stream control service to remove resources from the update stream instance, or to request updates for additional resources. An ALTO client cannot obtain the stream control service through the IRD. Instead, the first event that the update stream server sends to the ALTO client has the URI for the associated stream control service (see Section 5.3).

Each stream control request is an individual HTTP request. The ALTO client MAY send multiple stream control requests to the stream control server using the same HTTP connection.

7.1. URI

The URI for a stream control service, by itself, MUST uniquely specify the update stream instance which it controls. The stream control server MUST NOT use other properties of an HTTP request, such as cookies or the client's IP address, to determine the update stream. Furthermore, an update stream server MUST NOT reuse a control service URI once the associated update stream has been closed.

The ALTO client MUST evaluate a relative control URI reference [RFC3986] (for example, a URI reference without a host, or with a relative path) in the context of the URI used to create the update stream. The stream control service's host MAY be different from the update stream's host.

It is expected that there is an internal mechanism to map a stream control URI to the unique update stream instance to be controlled. For example, the update stream service may assign a unique, internal stream id to each update stream instance. However, the exact mechanism is left to the update stream service provider.

To prevent an attacker from forging a stream control URI and sending bogus requests to disrupt other update streams, the service should consider two security issues. First, if http, not https, is used, the stream control URI can be exposed to an on-path attacker. To address this issue, in a setting where the path from the server to

the client can traverse such an attacker, the server SHOULD use https. Second, even without direct exposure, an off-path attacker may guess valid stream control URIs. To address this issue, the server SHOULD choose stream control URIs with enough randomness, to make guessing difficult; the server SHOULD introduce mechanisms that detect repeated guesses indicating an attack (e.g., keeping track of the number of failed stream control attempts); please see <https://www.w3.org/TR/capability-urls/> .

7.2. Media Type

An ALTO stream control response does not have a specific media type.

7.3. HTTP Method

An ALTO update stream control resource is requested using the HTTP POST method.

7.4. IRD Capabilities & Uses

None (Stream control services do not appear in the IRD).

7.5. Request: Accept Input Parameters

A stream control service accepts the same input media type and input parameters as the update stream service (Section 6.5). The only difference is that a stream control service also accepts the "remove" field.

If specified, the "remove" field is an array of substream-ids the ALTO client previously added to this update stream. An empty "remove" array is equivalent to a list of all currently active resources; the update stream server responds by removing all resources and closing the stream.

An ALTO client MAY use the "add" field to add additional resources. The ALTO client MUST assign a unique substream-id to each additional resource. Substream-ids MUST be unique over the lifetime of this update stream: an ALTO client MUST NOT reuse a previously removed substream-id. The processing of an "add" resource is the same as discussed in Section 6.5 and Section 6.7.

If a request has any errors, the update stream server MUST NOT add or remove any resources from the associated update stream. Also, the stream control server will return an error response to the client as specified in Section 7.6.

7.6. Response

The stream control server MUST process the "add" field before the "remove" field. If the request removes all active resources without adding any additional resources, the update stream server MUST close the update stream. Thus an update stream cannot have zero resources.

If the request has any errors, the stream control server MUST return an HTTP "400 Bad Request" to the ALTO client. The body part of the HTTP response is the JSON object defined in Section 8.5.2 in [RFC7285]. An error response has the same format as specified in Section 6.6. Detailed error code and error information are specified as below.

- o If the "add" request does not satisfy the requirements in Section 6.5, the stream control server MUST return the ALTO error message defined in Section 6.6.
- o If any substream-id in the "remove" field was not added in a prior request, the error code of the error message MUST be `E_INVALID_FIELD_VALUE`; the "field" field SHOULD be "remove" and the "value" field SHOULD be an array of the invalid substream-ids. Thus it is illegal to "add" and "remove" the same substream-id in the same request. However, it is legal to remove a substream-id twice. To support the preceding checking, the update stream server MUST keep track of previously-used-but-now-closed substream-ids.
- o If any substream-id in the "add" field has been used before in this stream, the error code of the error message MUST be `E_INVALID_FIELD_VALUE`, the "field" field SHOULD be "add" and the "value" field SHOULD be an array of invalid substream-ids.
- o If the request has a non-empty "add" field and a "remove" field with an empty list of substream-ids (to replace all active resources with a new set, the client MUST explicitly enumerate the substream-ids to be removed), the error code of the error message MUST be `E_INVALID_FIELD_VALUE`; the "field" field SHOULD be "remove" and the "value" field SHOULD be an empty array.

If the request is valid but the associated update stream has been closed then the stream control server MUST return an HTTP "404 Not Found".

If the request is valid and the stream control server successfully processes the request without error, the stream control server should return either an HTTP "202 Accepted" response or an HTTP "204 No Content" response. The difference is that for the latter case, the

stream control server is sure that the update stream server has also processed the request. Regardless of 202 or 204 HTTP response, the final updates of related resources will be notified by the update stream server using its control update message(s), due to the modular design.

8. Examples

8.1. Example: IRD Announcing Update Stream Services

Below is an example IRD announcing three update stream services. The first, which is named "update-my-costs", provides updates for the network map, the "routingcost" and "hopcount" cost maps, and a filtered cost map resource. The second, which is named "update-my-prop", provides updates to the endpoint properties service. The third, which is named "update-my-pv", provides updates to a non-standard ALTO service returning a multipart response.

Note that in the "update-my-costs" update stream shown in the example IRD, the update stream server uses JSON patch for network map, and it uses JSON merge patch to update the other resources. Also, the update stream will only provide full replacements for "my-simple-filtered-cost-map".

Also, note that this IRD defines two filtered cost map resources. They use the same cost types, but "my-filtered-cost-map" accepts cost constraint tests, while "my-simple-filtered-cost-map" does not. To avoid the issues discussed in Section 9.3, the update stream provides updates for the second, but not the first.

This IRD also announces a non-standard ALTO service, which is named "my-pv". This service accepts an extended endpoint cost request as an input and returns a multipart response including an endpoint cost resource and a property map resource. This document does not rely on any other design details of this new service. In this document, the "my-pv" service is only used to illustrate how the update stream service provides updates to an ALTO resource returning a multipart response.

```
"my-network-map": {
  "uri": "https://alto.example.com/networkmap",
  "media-type": "application/alto-networkmap+json",
},
"my-routingcost-map": {
  "uri": "https://alto.example.com/costmap/routingcost",
  "media-type": "application/alto-costmap+json",
  "uses": ["my-networkmap"],
  "capabilities": {
```

```
    "cost-type-names": ["num-routingcost"]
  },
  "my-hopcount-map": {
    "uri": "https://alto.example.com/costmap/hopcount",
    "media-type": "application/alto-costmap+json",
    "uses": ["my-networkmap"],
    "capabilities": {
      "cost-type-names": ["num-hopcount"]
    }
  },
  "my-filtered-cost-map": {
    "uri": "https://alto.example.com/costmap/filtered/constraints",
    "media-type": "application/alto-costmap+json",
    "accepts": "application/alto-costmapfilter+json",
    "uses": ["my-networkmap"],
    "capabilities": {
      "cost-type-names": ["num-routingcost", "num-hopcount"],
      "cost-constraints": true
    }
  },
  "my-simple-filtered-cost-map": {
    "uri": "https://alto.example.com/costmap/filtered/simple",
    "media-type": "application/alto-costmap+json",
    "accepts": "application/alto-costmapfilter+json",
    "uses": ["my-networkmap"],
    "capabilities": {
      "cost-type-names": ["num-routingcost", "num-hopcount"],
      "cost-constraints": false
    }
  },
  "my-props": {
    "uri": "https://alto.example.com/properties",
    "media-type": "application/alto-endpointprops+json",
    "accepts": "application/alto-endpointpropparams+json",
    "capabilities": {
      "prop-types": ["priv:ietf-bandwidth"]
    }
  },
  "my-pv": {
    "uri": "https://alto.example.com/endpointcost/pv",
    "media-type": "multipart/related;
      type=application/alto-endpointcost+json",
    "accepts": "application/alto-endpointcostparams+json",
    "capabilities": {
      "cost-type-names": [ "path-vector" ],
      "ane-properties": [ "maxresbw", "persistent-entities" ]
    }
  }
}
```



```
    },
    "update-my-costs": {
      "uri": "https://alto.example.com/updates/costs",
      "media-type": "text/event-stream",
      "accepts": "application/alto-updatestreamparams+json",
      "uses": [
        "my-network-map",
        "my-routingcost-map",
        "my-hopcount-map",
        "my-simple-filtered-cost-map"
      ],
      "capabilities": {
        "incremental-change-media-types": {
          "my-network-map": "application/json-patch+json",
          "my-routingcost-map": "application/merge-patch+json",
          "my-hopcount-map": "application/merge-patch+json"
        },
        "support-stream-control": true
      }
    },
    "update-my-props": {
      "uri": "https://alto.example.com/updates/properties",
      "media-type": "text/event-stream",
      "uses": [ "my-props" ],
      "accepts": "application/alto-updatestreamparams+json",
      "capabilities": {
        "incremental-change-media-types": {
          "my-props": "application/merge-patch+json"
        },
        "support-stream-control": true
      }
    },
    "update-my-pv": {
      "uri": "https://alto.example.com/updates/pv",
      "media-type": "text/event-stream",
      "uses": [ "my-pv" ],
      "accepts": "application/alto-updatestreamparams+json",
      "capabilities": {
        "incremental-change-media-types": {
          "my-pv": "application/merge-patch+json"
        },
        "support-stream-control": true
      }
    }
  }
}
```

8.2. Example: Simple Network and Cost Map Updates

Given the update streams announced in the preceding example IRD, the section below shows an example of an ALTO client's request and the update stream server's immediate response, using the update stream resource "update-my-costs". In the example, the ALTO client requests updates for the network map and "routingcost" cost map, but not for the "hopcount" cost map. The ALTO client uses the ALTO server's resource-ids as the substream-ids. Because the client does not provide a "tag" for the network map, the update stream server must send a full replacement for the network map as well as for the cost map. The ALTO client does not set "incremental-changes" to "false", so it defaults to "true". Thus, the update stream server will send patch updates for the cost map and the network map.

```
POST /updates/costs HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 155

{ "add": {
  "my-network-map": {
    "resource-id": "my-network-map"
  },
  "my-routingcost-map": {
    "resource-id": "my-routingcost-map"
  }
}
}
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/3141592653589"}

event: application/alto-networkmap+json,my-network-map
data: {
data:   "meta" : {
data:     "vtag": {
data:       "resource-id" : "my-network-map",
data:       "tag" : "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
data:     }
data:   },
data:   "network-map" : {
```

```

data:      "PID1" : {
data:      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25" ]
data:      },
data:      "PID2" : {
data:      "ipv4" : [ "198.51.100.128/25" ]
data:      },
data:      "PID3" : {
data:      "ipv4" : [ "0.0.0.0/0" ],
data:      "ipv6" : [ "::/0" ]
data:      }
data:    }
data:  }
data: }

event: application/alto-costmap+json,my-routingcost-map
data: {
data:   "meta" : {
data:     "dependent-vtags" : [{
data:       "resource-id": "my-network-map",
data:       "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
data:     }],
data:     "cost-type" : {
data:       "cost-mode" : "numerical",
data:       "cost-metric": "routingcost"
data:     },
data:     "vtag": {
data:       "resource-id" : "my-routingcost-map",
data:       "tag" : "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
data:     }
data:   },
data:   "cost-map" : {
data:     "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
data:     "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
data:     "PID3": { "PID1": 20, "PID2": 15 }
data:   }
data: }

```

After sending those events immediately, the update stream server will send additional events as the maps change. For example, the following represents a small change to the cost map. PID1->PID2 is changed to 9 from 5, PID3->PID1 is no longer available and PID3->PID3 is now defined as 1:

```

event: application/merge-patch+json,my-routingcost-map
data: {
data:   "meta" : {
data:     "vtag": {
data:       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
data:     }
data:   },
data:   "cost-map": {
data:     "PID1" : { "PID2" : 9 },
data:     "PID3" : { "PID1" : null, "PID3" : 1 }
data:   }
data: }

```

As another example, the following represents a change to the network map: an ipv4 prefix "203.0.113.0/25" is added to PID1. It triggers changes to the cost map. The update stream server chooses to send an incremental change for the network map and send a full replacement instead of an incremental change for the cost map:

```

event: application/json-patch+json,my-network-map
data: {
data:   {
data:     "op": "replace",
data:     "path": "/meta/vtag/tag",
data:     "value" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
data:   },
data:   {
data:     "op": "add",
data:     "path": "/network-map/PID1/ipv4/2",
data:     "value": "203.0.113.0/25"
data:   }
data: }

event: application/alto-costmap+json,my-routingcost-map
data: {
data:   "meta" : {
data:     "vtag": {
data:       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
data:     }
data:   },
data:   "cost-map" : {
data:     "PID1": { "PID1": 1, "PID2": 3, "PID3": 7 },
data:     "PID2": { "PID1": 12, "PID2": 1, "PID3": 9 },
data:     "PID3": { "PID1": 14, "PID2": 8 }
data:   }
data: }

```

8.3. Example: Advanced Network and Cost Map Updates

This example is similar to the previous one, except that the ALTO client requests updates for the "hopcount" cost map as well as the "routingcost" cost map and provides the current version tag of the network map, so the update stream server is not required to send the full network map data update message at the beginning of the stream. In this example, the client uses the substream-ids "net", "routing" and "hops" for those resources. The update stream server sends the stream control URI and the full cost maps, followed by updates for the network map and cost maps as they become available:

```
POST /updates/costs HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 244

{ "add": {
  "net": {
    "resource-id": "my-network-map",
    "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
  },
  "routing": {
    "resource-id": "my-routingcost-map"
  },
  "hops": {
    "resource-id": "my-hopcount-map"
  }
}
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/2718281828459"}

event: application/alto-costmap+json, routing
data: { ... full routingcost cost map message ... }

event: application/alto-costmap+json, hops
data: { ... full hopcount cost map message ... }

(pause)

event: application/merge-patch+json, routing
data: {"cost-map": {"PID2" : {"PID3" : 31}}}

event: application/merge-patch+json, hops
data: {"cost-map": {"PID2" : {"PID3" : 4}}}
```

If the ALTO client wishes to stop receiving updates for the "hopcount" cost map, the ALTO client can send a "remove" request on the stream control URI:

```
POST /updates/streams/2718281828459 HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 24

{
  "remove": [ "hops" ]
}
```

```
HTTP/1.1 204 No Content
Content-Length: 0
```

(stream closed without sending data content)

The update stream server sends a "stopped" control update message on the original request stream to inform the ALTO client that updates are stopped for that resource:

```
event: application/alto-updatestreamcontrol+json
data: {
data:   "stopped": ["hops"]
data: }
```

Below is an example of an invalid stream control request. The "remove" field of the request includes an undefined substream-id and the stream control server will return an error response to the ALTO client.

```
POST /updates/streams/2718281828459 HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 31
{
  "remove": [ "properties" ]
}

HTTP/1.1 400 Bad Request
Content-Length: 89
Content-Type: application/alto-error+json

{
  "meta":{
    "code": "E_INVALID_FIELD_VALUE",
    "field": "remove",
    "value": "properties"
  }
}
```

If the ALTO client no longer needs any updates, and wishes to shut the update stream down gracefully, the client can send a "remove" request with an empty array:

```
POST /updates/streams/2718281828459 HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 17
```

```
{
  "remove": [ ]
}
```

```
HTTP/1.1 204 No Content
Content-Length: 0
```

(stream closed without sending data content)

The update stream server sends a final control update message on the original request stream to inform the ALTO client that all updates are stopped and then closes the stream:

```
event: application/alto-updatestreamcontrol+json
data: {
  data: "stopped": ["net", "routing"]
  data: }
```

(server closes stream)

8.4. Example: Endpoint Property Updates

As another example, here is how an ALTO client can request updates for the property "priv:ietf-bandwidth" for one set of endpoints and "priv:ietf-load" for another. The update stream server immediately sends full replacements with the property values for all endpoints. After that, the update stream server sends data update messages for the individual endpoints as their property values change.


```
POST /updates/properties HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamparams+json
Content-Length: 511
```

```
{ "add": {
  "props-1": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-bandwidth" ],
      "endpoints" : [
        "ipv4:198.51.100.1",
        "ipv4:198.51.100.2",
        "ipv4:198.51.100.3"
      ]
    }
  },
  "props-2": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-load" ],
      "endpoints" : [
        "ipv6:2001:db8:100::1",
        "ipv6:2001:db8:100::2",
        "ipv6:2001:db8:100::3"
      ]
    }
  }
}
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/1414213562373"}

event: application/alto-endpointprops+json,props-1
data: { "endpoint-properties": {
data:     "ipv4:198.51.100.1" : { "priv:ietf-bandwidth": "13" },
data:     "ipv4:198.51.100.2" : { "priv:ietf-bandwidth": "42" },
data:     "ipv4:198.51.100.3" : { "priv:ietf-bandwidth": "27" }
data:   } }

event: application/alto-endpointprops+json,props-2
data: { "endpoint-properties": {
data:     "ipv6:2001:db8:100::1" : { "priv:ietf-load": "8" },
data:     "ipv6:2001:db8:100::2" : { "priv:ietf-load": "2" },
data:     "ipv6:2001:db8:100::3" : { "priv:ietf-load": "9" }
data:   } }

  (pause)

event: application/merge-patch+json,props-1
data: { "endpoint-properties":
data:   {"ipv4:198.51.100.1" : {"priv:ietf-bandwidth": "3"}}
data: }

  (pause)

event: application/merge-patch+json,props-2
data: { "endpoint-properties":
data:   {"ipv6:2001:db8:100::3" : {"priv:ietf-load": "7"}}
data: }
```

If the ALTO client needs the "priv:ietf-bandwidth" property and the "priv:ietf-load" property for additional endpoints, the ALTO client can send an "add" request on the stream control URI:

```
POST /updates/streams/1414213562373" HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 448
```

```
{ "add": {
  "props-3": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-bandwidth" ],
      "endpoints" : [
        "ipv4:198.51.100.4",
        "ipv4:198.51.100.5"
      ]
    }
  },
  "props-4": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-load" ],
      "endpoints" : [
        "ipv6:2001:db8:100::4",
        "ipv6:2001:db8:100::5"
      ]
    }
  }
}
```

```
HTTP/1.1 204 No Content
Content-Length: 0
```

(stream closed without sending data content)

The update stream server sends full replacements for the two new resources, followed by incremental changes for all four requests as they arrive:

```
event: application/alto-endpointprops+json,props-3
data: { "endpoint-properties": {
data:   "ipv4:198.51.100.4" : { "priv:ietf-bandwidth": "25" },
data:   "ipv4:198.51.100.5" : { "priv:ietf-bandwidth": "31" },
data: } }
```

```
event: application/alto-endpointprops+json,props-4
data: { "endpoint-properties": {
data:   "ipv6:2001:db8:100::4" : { "priv:ietf-load": "6" },
data:   "ipv6:2001:db8:100::5" : { "priv:ietf-load": "4" },
data: } }
```

(pause)

```
event: application/merge-patch+json,props-3
data: { "endpoint-properties":
data:   {"ipv4:198.51.100.5" : {"priv:ietf-bandwidth": "15"}}
data: }
```

(pause)

```
event: application/merge-patch+json,props-2
data: { "endpoint-properties":
data:   {"ipv6:2001:db8:100::2" : {"priv:ietf-load": "9"}}
data: }
```

(pause)

```
event: application/merge-patch+json,props-4
data: { "endpoint-properties":
data:   {"ipv6:2001:db8:100::4" : {"priv:ietf-load": "3"}}
data: }
```

8.5. Example: Multipart Message Updates

This example shows how an ALTO client can request a non-standard ALTO service returning a multipart response. The update stream server immediately sends full replacements of the multipart response. After that, the update stream server sends data update messages for the individual parts of the response as the ALTO data (object) in each part changes.

```
POST /updates/pv HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamparams+json
Content-Length: 382
```

```
{
  "add": {
    "ecspvsub1": {
      "resource-id": "my-pv",
      "input": {
        "cost-type": {
          "cost-mode": "array",
          "cost-metric": "ane-path"
        },
        "endpoints": {
          "srcs": [ "ipv4:192.0.2.2" ],
          "dsts": [ "ipv4:192.0.2.89", "ipv4:203.0.113.45" ]
        },
        "ane-properties": [ "maxresbw", "persistent-entities" ]
      }
    }
  }
}
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data:      "https://alto.example.com/updates/streams/1414"}

event: multipart/related;boundary=example-pv;
      type=application/alto-endpointcost+json,ecspvsubl
data: --example-pv
data: Content-ID: ecsmap
data: Content-Type: application/alto-endpointcost+json
data:
data: { ... data (object) of an endpoint cost map ... }
data: --example-pv
data: Content-ID: propmap
data: Content-Type: application/alto-propmap+json
data:
data: { ... data (object) of a property map ... }
data: --example-pv--

      (pause)

event: application/merge-patch+json,ecspvsubl.ecsmap
data: { ... merge patch for updates of ecspvsubl.ecsmap ... }

event: application/merge-patch+json,ecspvsubl.propmap
data: { ... merge patch for updates of ecspvsubl.propmap ... }
```

9. Operation and Processing Considerations

9.1. Considerations for Choosing Data Update Messages

The update stream server should be cognizant of the effects of its update schedule, which includes both the choice of timing (i.e., when/what to trigger an update) and the choice of message format (i.e., given an update, send a full replacement or an incremental change). In particular, the update schedule can have effects on both the overhead and the freshness of information. To minimize overhead, the server may choose to batch a sequence of updates for resources that frequently change, by sending cumulative updates or a full replacement after a while. The update stream server should be cognizant that batching reduces the freshness of information. The server should also consider the effect of such delays on client behaviors (see below on client timeout on waiting for updates of dependent resources).

For incremental updates, this design allows both JSON patch and JSON merge patch for incremental changes. JSON merge patch is clearly superior to JSON patch for describing incremental changes to Cost Maps, Endpoint Costs, and Endpoint Properties. For these data structures, JSON merge patch is more space-efficient, as well as simpler to apply; There is no advantage allowing a server to use JSON patch for those resources.

The case is not as clear for incremental changes to network maps.

First, consider small changes such as moving a prefix from one PID to another. JSON patch could encode that as a simple insertion and deletion, while JSON merge patch would have to replace the entire array of prefixes for both PIDs. On the other hand, to process a JSON patch update, the ALTO client would have to retain the indexes of the prefixes for each PID. Logically, the prefixes in a PID are an unordered set, not an array; aside from handling updates, a client has no need to retain the array indexes of the prefixes. Hence to take advantage of JSON patch for network maps, ALTO clients would have to retain additional, otherwise unnecessary, data.

Second, consider more involved changes such as removing half of the prefixes from a PID. JSON merge patch would send a new array for that PID, while JSON patch would have to send a list of remove operations and delete the prefix one by one.

Therefore, each update stream server may decide on its own whether to use JSON merge patch or JSON patch according to the changes in network maps.

9.2. Considerations for Client Processing Data Update Messages

In general, when an ALTO client receives a full replacement for a resource, the ALTO client should replace the current version with the new version. When an ALTO client receives an incremental change for a resource, the ALTO client should apply those patches to the current version of the resource.

However, because resources can depend on other resources (e.g., cost maps depend on network maps), an ALTO client MUST NOT use a dependent resource if the resource on which it depends has changed. There are at least two ways an ALTO client can do that. The following paragraphs illustrate these techniques by referring to network and cost map messages, although these techniques apply to any dependent resources.

Note that when a network map changes, the update stream server **MUST** send the network map update message before sending the updates for the dependent cost maps (see Section 6.7.1).

One approach is for the ALTO client to save the network map update message in a buffer and continue to use the previous network map, and the associated cost maps, until the ALTO client receives the update messages for all dependent cost maps. The ALTO client then applies all network and cost map updates atomically.

Alternatively, the ALTO client **MAY** update the network map immediately. In this case, the cost maps using the network map become invalid because they are inconsistent with the current network map; hence, the ALTO client **MUST** mark each such dependent cost map as temporarily invalid and **MUST NOT** use that each such cost map until the ALTO client receives a cost map update message indicating that it is based on the new network map version tag.

The update stream server **SHOULD** send updates for dependent resources (i.e., the cost maps in the preceding example) in a timely fashion. However, if the ALTO client does not receive the expected updates, a simple recovery method is that the ALTO client closes the update stream connection, discards the dependent resources, and reestablishes the update stream. The ALTO client **MAY** retain the version tag of the last version of any tagged resources and give those version tags when requesting the new update stream. In this case, if a version is still current, the update stream server will not re-send that resource.

Although not as efficient as possible, this recovery method is simple and reliable.

9.3. Considerations for Updates to Filtered Cost Maps

If an update stream provides updates to a Filtered cost map which allows constraint tests, then an ALTO client **MAY** request updates to a Filtered cost map request with a constraint test. In this case, when a cost changes, the update stream server **MUST** send an update if the new value satisfies the test. If the new value does not, whether the update stream server sends an update depends on whether the previous value satisfied the test. If it did not, the update stream server **SHOULD NOT** send an update to the ALTO client. But if the previous value did, then the update stream server **MUST** send an update with a "null" value, to inform the ALTO client that this cost no longer satisfies the criteria.

An update stream server can avoid having to handle such a complicated behavior by offering update streams only for filtered cost maps which do not allow constraint tests.

9.4. Considerations for Updates to Ordinal Mode Costs

For an ordinal mode cost map, a change to a single cost point may require updating many other costs. As an extreme example, suppose the lowest cost changes to the highest cost. For a numerical mode cost map, only that one cost changes. But for an ordinal mode cost map, every cost might change. While this document allows an update stream server to offer incremental updates for ordinal mode cost maps, update stream server implementors should be aware that incremental updates for ordinal costs are more complicated than for numerical costs, and ALTO clients should be aware that small changes may result in large updates.

An update stream server can avoid this complication by only offering full replacements for ordinal cost maps.

9.5. Considerations for SSE Text Formatting and Processing

SSE was designed for events that consist of relatively small amounts of line-oriented text data, and SSE clients frequently read input one line-at-a-time. However, an update stream sends a full cost map as a single event, and a cost map may involve megabytes, if not tens of megabytes, of text. This has implications that the ALTO client and the update stream server may consider.

First, some SSE client libraries read all data for an event into memory, and then present it to the client as a character array. However, a client may not have enough memory to hold the entire JSON text for a large cost map. Hence an ALTO client SHOULD consider using an SSE library which presents the event data in manageable chunks, so the ALTO client can parse the cost map incrementally and store the underlying data in a more compact format.

Second, an SSE client library may use a low level, generic socket read library that stores each line of an event data, just in case the higher level parser may need the line delimiters as part of the protocol formatting. A server sending a complete cost map as a single line may then generate a multi-megabyte data "line", and such a long line may then require complex memory management at the client. It is RECOMMENDED that an update stream server limit the lengths of data lines.

Third, an SSE server may use a library which may put line breaks in places that would have semantic consequences for the ALTO updates;

see Section 11. The update stream server implementation **MUST** ensure that no line breaks are introduced to change the semantics.

10. Security Considerations

The Security Considerations (Section 15 of [RFC7285]) of the base protocol fully apply to this extension. For example, the same authenticity and integrity considerations (Section 15.1 of [RFC7285]) still fully apply; the same considerations for the privacy of ALTO users (Section 15.4 of [RFC7285]) also still fully apply.

The additional services (addition of update streams and stream control URIs) provided by this extension extend the attack surface described in Section 15.1.1 of [RFC7285]. Below we discuss the additional risks and their remedies.

10.1. Update Stream Server: Denial-of-Service Attacks

Allowing persistent update stream connections enables a new class of Denial-of-Service attacks.

For the update stream server, an ALTO client might create an unreasonable number of update stream connections, or add an unreasonable number of substream-ids to one update stream.

To avoid these attacks on the update stream server, the server **SHOULD** choose to limit the number of active streams and reject new requests when that threshold is reached. An update stream server **SHOULD** also choose to limit the number of active substream-ids on any given stream, or limit the total number of substream-ids used over the lifetime of a stream, and reject any stream control request which would exceed those limits. In these cases, the update stream server **SHOULD** return the HTTP status "503 Service Unavailable".

It is important to note that the preceding approach are not the only possibilities. For example, it may be possible for the update stream server to use somewhat more clever logic involving IP reputation, rate-limiting, and compartmentalizing the overall threshold into smaller thresholds that apply to subsets of potential clients.

While the preceding techniques prevent update stream DoS attacks from disrupting an update stream server's other services, it does make it easier for a DoS attack to disrupt the update stream service. Therefore an update stream server **MAY** prefer to restrict update stream services to authorized clients, as discussed in Section 15 of [RFC7285].

Alternatively, an update stream server MAY return the HTTP status "307 Temporary Redirect" to redirect the client to another ALTO server which can better handle a large number of update streams.

10.2. ALTO Client: Update Overloading or Instability

The availability of continuous updates can also cause overload for an ALTO client, in particular an ALTO client with limited processing capabilities. The current design does not include any flow control mechanisms for the client to reduce the update rates from the server. Under overloading, the client MAY choose to remove the information resources with high update rates.

Also, under overloading, the client may no longer be able to detect whether an information is still fresh or has become stale. In such a case, the client should be careful in how it uses the information to avoid stability or efficiency issues.

10.3. Stream Control: Spoofed Control Requests and Information Breakdown

An outside party which can read the update stream response, or which can observe stream control requests, can obtain the control URI and use that to send a fraudulent "remove" requests, thus disabling updates for the valid ALTO client. This can be avoided by encrypting the update stream and stream control requests (see Section 15 of [RFC7285]). Also, the update stream server echoes the "remove" requests on the update stream, so the valid ALTO client can detect unauthorized requests.

In general, as the architecture allows the possibility for the update stream server and the stream control server to be different entities, the additional risks should be evaluated and remedied. For example, the private communication path between the servers may be attacked, resulting in a risk of communications breakdown between them, as well as invalid or spoofed messages claiming to be on that private communications path. Proper security mechanisms, including confidentiality, authenticity, and integrity mechanisms should be considered.

11. Requirements on Future ALTO Services to Use this Design

Although this design is quite flexible, it has underlying requirements.

The key requirements are that (1) each data update message is for a single resource; (2) an incremental change can be applied only to a resource that is a single JSON object, as both JSON merge patch and

JSON patch can apply only to a single JSON object. Hence, if a future ALTO resource can contain multiple objects, then either each individual object also has a resource-id or an extension to this design is made.

At the low level encoding level, new line in SSE has its own semantics. Hence, this design requires that resource encoding does not include new lines that can confuse with SSE encoding. In particular, the data update message MUST NOT include "event: " or "data: " at a new line as part of data message.

If an update stream provides updates to a filtered cost map that allows constraint tests, the requirements for such services are stated in Section 9.3.

12. IANA Considerations

This document defines two new media-types, "application/alto-updatestreamparams+json", as described in Section 6.5, and "application/alto-updatestreamcontrol+json", as described in Section 5.3. All other media-types used in this document have already been registered, either for ALTO, JSON merge patch, or JSON patch.

12.1. application/alto-updatestreamparams+json Media Type

Type name: application

Subtype name: alto-updatestreamparams+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 10 of [RFCthis] and Section 15 of [RFC7285].

Interoperability considerations: [RFCthis] specifies format of conforming messages and the interpretation thereof.

Published specification: Section 6.5 of [RFCthis].

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

Fragment identifier considerations: n/a

Additional information:

Magic number(s): n/a

File extension(s): [RFCthis] 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).

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

12.2. application/alto-updatestreamcontrol+json Media Type

Type name: application

Subtype name: alto-updatestreamcontrol+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 10 of [RFCthis] and Section 15 of [RFC7285].

Interoperability considerations: [RFCthis] specifies format of conforming messages and the interpretation thereof.

Published specification: Section 5.3 of [RFCthis].

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

Fragment identifier considerations: n/a

Additional information:

Magic number(s): n/a

File extension(s): [RFCthis] 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).

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

13. Contributors

Section 2, Section 5.1, Section 5.2 and Section 8.5 of this document are based on contributions from Jingxuan Jensen Zhang, and he is considered an author.

14. Acknowledgments

Thank you to Dawn Chen (Tongji University), Shawn Lin (Tongji University) and Xiao Shi (Yale University) for their contributions to an earlier version of this document.

15. Appendix: Design Decision: Not Allowing Stream Restart

If an update stream is closed accidentally, when the ALTO client reconnects, the update stream server must resend the full maps. This is clearly inefficient. To avoid that inefficiency, the SSE specification allows an update stream server to assign an id to each event. When an ALTO client reconnects, the ALTO client can present the id of the last successfully received event, and the update stream server restarts with the next event.

However, that mechanism adds additional complexity. The update stream server must save SSE messages in a buffer, in case ALTO clients reconnect. But that mechanism will never be perfect: if the ALTO client waits too long to reconnect, or if the ALTO client sends an invalid id, then the update stream server will have to resend the complete maps anyway.

Furthermore, this is unlikely to be a problem in practice. ALTO clients who want continuous updates for large resources, such as full Network and cost maps, are likely to be things like P2P trackers. These ALTO clients will be well connected to the network; they will rarely drop connections.

Mobile devices certainly can and do drop connections and will have to reconnect. But mobile devices will not need continuous updates for multi-megabyte cost maps. If mobile devices need continuous updates at all, they will need them for small queries, such as the costs from a small set of media servers from which the device can stream the currently playing movie. If the mobile device drops the connection and reestablishes the update stream, the update stream server will have to retransmit only a small amount of redundant data.

In short, using event ids to avoid resending the full map adds a considerable amount of complexity to avoid a situation which is very rare. The complexity is not worth the benefit.

The Update Stream service does allow the ALTO client to specify the tag of the last received version of any tagged resource, and if that is still current, the update stream server need not retransmit the full resource. Hence ALTO clients can use this to avoid retransmitting full network maps. cost maps are not tagged, so this will not work for them. Of course, the ALTO protocol could be extended by adding version tags to cost maps, which would solve the retransmission-on-reconnect problem. However, adding tags to cost maps might add a new set of complications.

16. References

16.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.
- [RFC2387] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387, BCP 14, August 1998.
- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", RFC 6838, January 2013.
- [RFC6902] Bryan, P. and M. Nottingham, "JavaScript Object Notation (JSON) Patch", RFC 6902, April 2013.
- [RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.
- [RFC7396] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396, October 2014.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [SSE] Hickson, I., "Server-Sent Events (W3C)", W3C Recommendation 03 February 2015, February 2015.

16.2. Informative References

- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

- [RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.
- [RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.
- [RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015.

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

Y. Richard Yang
Yale University
51 Prospect St
New Haven CT
USA

Email: yry@cs.yale.edu

ALTO
Internet-Draft
Intended status: Experimental
Expires: 21 September 2022

K. Gao
Sichuan University
Y. Lee
Samsung
S. Randriamasy
Nokia Bell Labs
Y.R. Yang
Yale University
J. Zhang
Tongji University
20 March 2022

An ALTO Extension: Path Vector
draft-ietf-alto-path-vector-25

Abstract

This document is an extension to the base Application-Layer Traffic Optimization (ALTO) protocol. It extends the ALTO Cost Map and ALTO Property Map services so that an application can decide which endpoint(s) to connect based on not only numerical/ordinal cost values but also fine-grained abstract information of the paths. This is useful for applications whose performance is impacted by specified components of a network on the end-to-end paths, e.g., they may infer that several paths share common links and prevent traffic bottlenecks by avoiding such paths. This extension introduces a new abstraction called Abstract Network Element (ANE) to represent these components and encodes a network path as a vector of ANEs. Thus, it provides a more complete but still abstract graph representation of the underlying network(s) for informed traffic optimization among endpoints.

Status of This Memo

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

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

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

This Internet-Draft will expire on 21 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
2. Requirements Languages	6
3. Terminology	6
4. Requirements and Use Cases	7
4.1. Design Requirements	7
4.2. Sample Use Cases	10
4.2.1. Exposing Network Bottlenecks	11
4.2.2. Resource Exposure for CDN and Service Edge	15
5. Path Vector Extension: Overview	17
5.1. Abstract Network Element (ANE)	18
5.1.1. ANE Entity Domain	19
5.1.2. Ephemeral and Persistent ANEs	19
5.1.3. Property Filtering	20
5.2. Path Vector Cost Type	20
5.3. Multipart Path Vector Response	21
5.3.1. Identifying the Media Type of the Root Object	22
5.3.2. References to Part Messages	22
6. Specification: Basic Data Types	23
6.1. ANE Name	23
6.2. ANE Entity Domain	23
6.2.1. Entity Domain Type	23
6.2.2. Domain-Specific Entity Identifier	23
6.2.3. Hierarchy and Inheritance	23
6.2.4. Media Type of Defining Resource	23
6.3. ANE Property Name	24
6.4. Initial ANE Property Types	24
6.4.1. Maximum Reservable Bandwidth	24
6.4.2. Persistent Entity ID	25
6.4.3. Examples	25
6.5. Path Vector Cost Type	26

6.5.1.	Cost Metric: ane-path	26
6.5.2.	Cost Mode: array	27
6.6.	Part Resource ID and Part Content ID	27
7.	Specification: Service Extensions	27
7.1.	Notations	27
7.2.	Multipart Filtered Cost Map for Path Vector	28
7.2.1.	Media Type	28
7.2.2.	HTTP Method	28
7.2.3.	Accept Input Parameters	28
7.2.4.	Capabilities	29
7.2.5.	Uses	30
7.2.6.	Response	30
7.3.	Multipart Endpoint Cost Service for Path Vector	34
7.3.1.	Media Type	34
7.3.2.	HTTP Method	34
7.3.3.	Accept Input Parameters	34
7.3.4.	Capabilities	35
7.3.5.	Uses	35
7.3.6.	Response	35
8.	Examples	39
8.1.	Sample Setup	39
8.2.	Information Resource Directory	39
8.3.	Multipart Filtered Cost Map	42
8.4.	Multipart Endpoint Cost Service Resource	43
8.5.	Incremental Updates	48
8.6.	Multi-cost	50
9.	Compatibility with Other ALTO Extensions	52
9.1.	Compatibility with Legacy ALTO Clients/Servers	53
9.2.	Compatibility with Multi-Cost Extension	53
9.3.	Compatibility with Incremental Update	53
9.4.	Compatibility with Cost Calendar	53
10.	General Discussions	54
10.1.	Constraint Tests for General Cost Types	54
10.2.	General Multi-Resource Query	54
11.	Security Considerations	55
12.	IANA Considerations	57
12.1.	ALTO Cost Metric Registry	57
12.2.	ALTO Cost Mode Registry	58
12.3.	ALTO Entity Domain Type Registry	58
12.4.	ALTO Entity Property Type Registry	59
12.4.1.	New ANE Property Type: Maximum Reservable Bandwidth	59
12.4.2.	New ANE Property Type: Persistent Entity ID	60
13.	References	60
13.1.	Normative References	60
13.2.	Informative References	61
Appendix A.	Acknowledgments	64
Appendix B.	Revision Logs (To be removed before publication)	64

B.1.	Changes since -20	64
B.2.	Changes since -19	65
B.3.	Changes since -18	65
B.4.	Changes since -17	65
B.5.	Changes since -16	65
B.6.	Changes since -15	65
B.7.	Changes since -14	65
B.8.	Changes since -13	66
B.9.	Changes since -12	66
B.10.	Changes since -11	66
B.11.	Changes since -10	66
B.12.	Changes since -09	67
B.13.	Changes since -08	67
B.14.	Changes Since Version -06	67
Authors' Addresses	68

1. Introduction

Network performance metrics are crucial to assess the Quality of Experience (QoE) of applications. The ALTO protocol allows Internet Service Providers (ISPs) to provide guidance, such as topological distance between different end hosts, to overlay applications. Thus, the overlay applications can potentially improve the perceived QoE by better orchestrating their traffic to utilize the resources in the underlying network infrastructure.

Existing ALTO Cost Map (Section 11.2.3 of [RFC7285]) and Endpoint Cost Service (Section 11.5 of [RFC7285]) provide only cost information on an end-to-end path defined by its <source, destination> endpoints: The base protocol [RFC7285] allows the services to expose the topological distances of end-to-end paths, while various extensions have been proposed to extend the capability of these services, e.g., to express other performance metrics [I-D.ietf-alto-performance-metrics], to query multiple costs simultaneously [RFC8189], and to obtain the time-varying values [RFC8896].

While the existing extensions are sufficient for many overlay applications, the QoE of some overlay applications depends not only on the cost information of end-to-end paths, but also on particular components of a network on the paths and their properties. For example, job completion time, which is an important QoE metric for a large-scale data analytics application, is impacted by shared bottleneck links inside the carrier network as link capacity may impact the rate of data input/output to the job. We refer to such components of a network as Abstract Network Elements (ANE).

Predicting such information can be very complex without the help of ISPs, for example, [BOXOPT] has shown that finding the optimal bandwidth reservation for multiple flows can be NP-hard without further information than whether a reservation succeeds. With proper guidance from the ISP, an overlay application may be able to schedule its traffic for better QoE. In the meantime, it may be helpful as well for ISPs if applications could avoid using bottlenecks or challenging the network with poorly scheduled traffic.

Despite the claimed benefits, ISPs are not likely to expose raw details on their network paths: first for the sake of topology hiding requirement, second because it may increase volume and computation overhead, and last because applications do not necessarily need all the network path details and are likely not able to understand them.

Therefore, it is beneficial for both ISPs and applications if an ALTO server provides ALTO clients with an "abstract network state" that provides the necessary information to applications, while hiding the network complexity and confidential information. An "abstract network state" is a selected set of abstract representations of Abstract Network Elements traversed by the paths between <source, destination> pairs combined with properties of these Abstract Network Elements that are relevant to the overlay applications' QoE. Both an application via its ALTO client and the ISP via the ALTO server can achieve better confidentiality and resource utilization by appropriately abstracting relevant Abstract Network Elements. Server scalability can also be improved by combining Abstract Network Elements and their properties in a single response.

This document extends [RFC7285] to allow an ALTO server to convey "abstract network state", for paths defined by their <source, destination> pairs. To this end, it introduces a new cost type called "Path Vector" following the cost metric registration specified in [RFC7285] and the updated cost mode registration specified in [I-D.bw-alto-cost-mode]. A Path Vector is an array of identifiers that identifies an Abstract Network Element, which can be associated with various properties. The associations between ANEs and their properties are encoded in an ALTO information resource called Unified Property Map, which is specified in [I-D.ietf-alto-unified-props-new].

For better confidentiality, this document aims to minimize information exposure of an ALTO server when providing Path Vector service. In particular, this document enables and recommends that first ANEs are constructed on demand, and second an ANE is only associated with properties that are requested by an ALTO client. A Path Vector response involves two ALTO Maps: the Cost Map that contains the Path Vector results and the up-to-date Unified Property

Map that contains the properties requested for these ANEs. To enforce consistency and improve server scalability, this document uses the "multipart/related" content type defined in [RFC2387] to return the two maps in a single response.

As a single ISP may not have the knowledge of the full Internet paths between arbitrary endpoints, this document is mainly applicable 1) when there is a single ISP between the requested source and destination PIDs or endpoints, for example, ISP-hosted CDN/edge, tenant interconnection in a single public cloud platform, etc.; or 2) when the Path Vectors are generated from end-to-end measurement data.

2. Requirements Languages

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

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

3. Terminology

This document extends the ALTO base protocol [RFC7285] and the Unified Property Map extension [I-D.ietf-alto-unified-props-new]. In addition to the terms defined in these documents, this document also uses the following additional terms:

Abstract Network Element (ANE): An abstract representation for a component in a network that handles data packets and whose properties can potentially have an impact on the end-to-end performance of traffic. An ANE can be a physical device such as a router, a link or an interface, or an aggregation of devices such as a subnetwork or a data center.

The definition of Abstract Network Element is similar to Network Element defined in [RFC2216] in the sense that they both provide an abstract representation of specific components of a network. However, they have different criteria on how these particular components are selected. Specifically, a Network Element requires the components to be capable of exercising QoS control, while Abstract Network Element only requires the components to have an impact on the end-to-end performance.

ANE Name: A string that uniquely identifies an ANE in a specific

scope. An ANE can be constructed either statically in advance or on demand based on the requested information. Thus, different ANEs may only be valid within a particular scope, either ephemeral or persistent. Within each scope, an ANE is uniquely identified by an ANE Name, as defined in Section 6.1. Note that an ALTO client must not assume ANEs in different scopes but with the same ANE Name refer to the same component(s) of the network.

Path Vector: Path Vector, or ANE Path Vector, refers to a JSON array of ANE Names. It is a generalization of BGP path vector. While standard BGP path vector (Section 5.1.2 of [RFC4271]) specifies a sequence of autonomous systems for a destination IP prefix, the Path Vector defined in this extension specifies a sequence of ANEs either for a source Provider-Defined Identifier (PID) and a destination PID as in the CostMapData (11.2.3.6 in [RFC7285]), or for a source endpoint and a destination endpoint as in the EndpointCostMapData object (Section 11.5.1.6 of [RFC7285]).

Path Vector resource: An ALTO information resource (Section 8.1 of [RFC7285]) which supports the extension defined in this document.

Path Vector cost type: A special cost type, which is specified in Section 6.5. When this cost type is present in an IRD entry, it indicates that the information resource is a Path Vector resource. When this cost type is present in a Filtered Cost Map request or an Endpoint Cost Service request, it indicates each cost value must be interpreted as a Path Vector.

Path Vector request: The POST message sent to an ALTO Path Vector resource.

Path Vector response: A Path Vector response refers to the multipart/related message returned by a Path Vector resource.

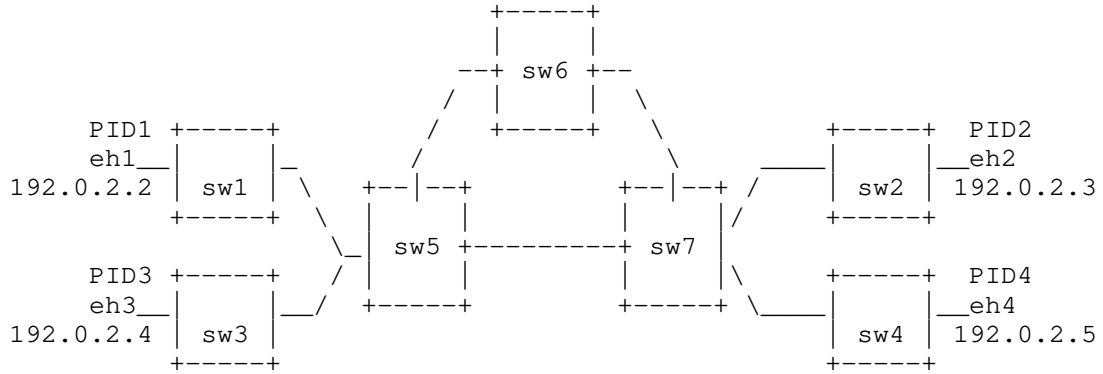
4. Requirements and Use Cases

4.1. Design Requirements

This section gives an illustrative example of how an overlay application can benefit from the extension defined in this document.

Assume that an application has control over a set of flows, which may go through shared links/nodes and share bottlenecks. The application seeks to schedule the traffic among multiple flows to get better performance. The constraints of feasible rate allocations of those flows will benefit the scheduling. However, Cost Maps as defined in [RFC7285] can not reveal such information.

Specifically, consider a network as shown in Figure 1. The network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches "sw1", "sw2", "sw3" and "sw4" are access switches, and sw5-sw7 form the backbone. End hosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of link eh1 -> sw1 and link sw1 -> sw5 is 150 Mbps, and the bandwidth of the other links is 100 Mbps.



$\text{bw}(\text{eh1} \rightarrow \text{sw1}) = \text{bw}(\text{sw1} \rightarrow \text{sw5}) = 150 \text{ Mbps}$
 $\text{bw}(\text{eh2} \rightarrow \text{sw2}) = \text{bw}(\text{eh3} \rightarrow \text{sw3}) = \text{bw}(\text{eh4} \rightarrow \text{sw4}) = 100 \text{ Mbps}$
 $\text{bw}(\text{sw1} \rightarrow \text{sw5}) = \text{bw}(\text{sw3} \rightarrow \text{sw5}) = \text{bw}(\text{sw2} \rightarrow \text{sw7}) = \text{bw}(\text{sw4} \rightarrow \text{sw7}) = 100 \text{ Mbps}$
 $\text{bw}(\text{sw5} \rightarrow \text{sw6}) = \text{bw}(\text{sw5} \rightarrow \text{sw7}) = \text{bw}(\text{sw6} \rightarrow \text{sw7}) = 100 \text{ Mbps}$

Figure 1: Raw Network Topology

The base ALTO topology abstraction of the network is shown in Figure 2. Assume the cost map returns an hypothetical cost type representing the available bandwidth between a source and a destination.



Figure 2: Base Topology Abstraction

Now assume the application wants to maximize the total rate of the traffic among a set of <source, destination> pairs, say "eh1 -> eh2" and "eh1 -> eh4". Let "x" denote the transmission rate of "eh1 -> eh2" and "y" denote the rate of "eh1 -> eh4". The objective function is

$$\max(x + y).$$

With the ALTO Cost Map, the cost between PID1 and PID2 and between PID1 and PID4 will both be 100 Mbps. The client can get a capacity region of

$$\begin{aligned} x &\leq 100 \text{ Mbps,} \\ y &\leq 100 \text{ Mbps.} \end{aligned}$$

With this information, the client may mistakenly think it can achieve a maximum total rate of 200 Mbps. However, this rate is infeasible, as there are only two potential cases:

- * Case 1: "eh1 -> eh2" and "eh1 -> eh4" take different path segments from "sw5" to "sw7". For example, if "eh1 -> eh2" uses path "eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2" and "eh1 -> eh4" uses path "eh1 -> sw1 -> sw5 -> sw7 -> sw4 -> eh4", then the shared bottleneck links are "eh1 -> sw1" and "sw1 -> sw5". In this case, the capacity region is:

$$\begin{aligned} x &\leq 100 \text{ Mbps} \\ y &\leq 100 \text{ Mbps} \\ x + y &\leq 150 \text{ Mbps} \end{aligned}$$

and the real optimal total rate is 150 Mbps.

- * Case 2: "eh1 -> eh2" and "eh1 -> eh4" take the same path segment from "sw5" to "sw7". For example, if "eh1 -> eh2" uses path "eh1 -> sw1 -> sw5 -> sw7 -> sw2 -> eh2" and "eh1 -> eh4" also uses path "eh1 -> sw1 -> sw5 -> sw7 -> sw4 -> eh4", then the shared bottleneck link is "sw5 -> sw7". In this case, the capacity region is:

$$\begin{aligned} x &\leq 100 \text{ Mbps} \\ y &\leq 100 \text{ Mbps} \\ x + y &\leq 100 \text{ Mbps} \end{aligned}$$

and the real optimal total rate is 100 Mbps.

Clearly, with more accurate and fine-grained information, the application can gain a better prediction of its traffic and may orchestrate its resources accordingly. However, to provide such information, the network needs to expose abstract information beyond the simple cost map abstraction. In particular:

- * The ALTO server must expose abstract information about the network paths that are traversed by the traffic between a source and a destination beyond a simple numerical value, which allows the overlay application to distinguish between Cases 1 and 2 and to compute the optimal total rate accordingly.
- * The ALTO server must allow the client to distinguish the common ANE shared by "eh1 -> eh2" and "eh1 -> eh4", e.g., "eh1 - sw1" and "sw1 - sw5" in Case 1.
- * The ALTO server must expose abstract information on the properties of the ANEs used by "eh1 -> eh2" and "eh1 -> eh4". For example, an ALTO server can either expose the available bandwidth between "eh1 - sw1", "sw1 - sw5", "sw5 - sw7", "sw5 - sw6", "sw6 - sw7", "sw7 - sw2", "sw7 - sw4", "sw2 - eh2", "sw4 - eh4" in Case 1, or expose 3 abstract elements "A", "B" and "C", which represent the linear constraints that define the same capacity region in Case 1.

In general, we can conclude that to support the multiple flow scheduling use case, the ALTO framework must be extended to satisfy the following additional requirements:

AR1: An ALTO server must provide the ANEs that are important to assess the QoE of the overlay application on the path of a <source, destination> pair.

AR2: An ALTO server must provide information to identify how ANEs are shared on the paths of different <source, destination> pairs.

AR3: An ALTO server must provide information on the properties that are important to assess the QoE of the application for ANEs.

The extension defined in this document specifies a solution to expose such abstract information.

4.2. Sample Use Cases

While the multiple flow scheduling problem is used to help identify the additional requirements, the extension defined in this document can be applied to a wide range of applications. This section highlights some use cases that are reported.

4.2.1. Exposing Network Bottlenecks

An important use case of the Path Vector extension is to expose network bottlenecks. Applications which need to perform large scale data transfers can benefit from being aware of the resource constraints exposed by this extension even if they have different objectives. One such example is the Worldwide LHC Computing Grid (WLCG), the largest example of a distributed computation collaboration in the research and education world.

Figure 3 illustrates an example of using ALTO Path Vector as an interface between the job optimizer for a data analytics system and the network manager. In particular, we assume the objective of the job optimizer is to minimize the job completion time.

In such a setting, the network-aware job optimizer (e.g., [CLARINET]) takes a query and generates multiple query execution plans (QEP). It can encode the QEPs as Path Vector requests that are sent to an ALTO server. The ALTO server obtains the routing information for the flows in a QEP and finds links, routers, or middleboxes (e.g., a stateful firewall) that can potentially become bottlenecks of the QEP (e.g., see [NOVA] and [G2] for mechanisms to identify bottleneck links under different settings). The resource constraint information is encoded in a Path Vector response and returned to the ALTO client.

With the network resource constraints, the job optimizer may choose the QEP with the optimal job completion time to be executed. It must be noted that the ALTO framework itself does not offer the capability to control the traffic. However, certain network managers may offer ways to enforce resource guarantees, such as on-demand tunnels (e.g., [SWAN]), demand vector (e.g., [HUG], [UNICORN]), etc. The traffic control interfaces and mechanisms are out of the scope of this document.

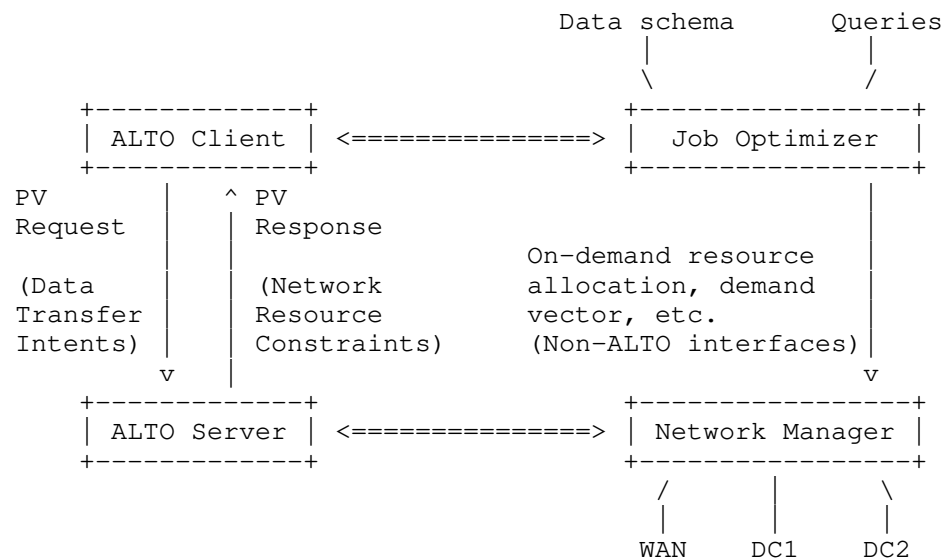


Figure 3: Example Use Case for Data Analytics

Another example is as illustrated in Figure 4. Consider a network consisting of multiple sites and a non-blocking core network, i.e., the links in the core network have sufficient bandwidth that they will not become the bottleneck of the data transfers.

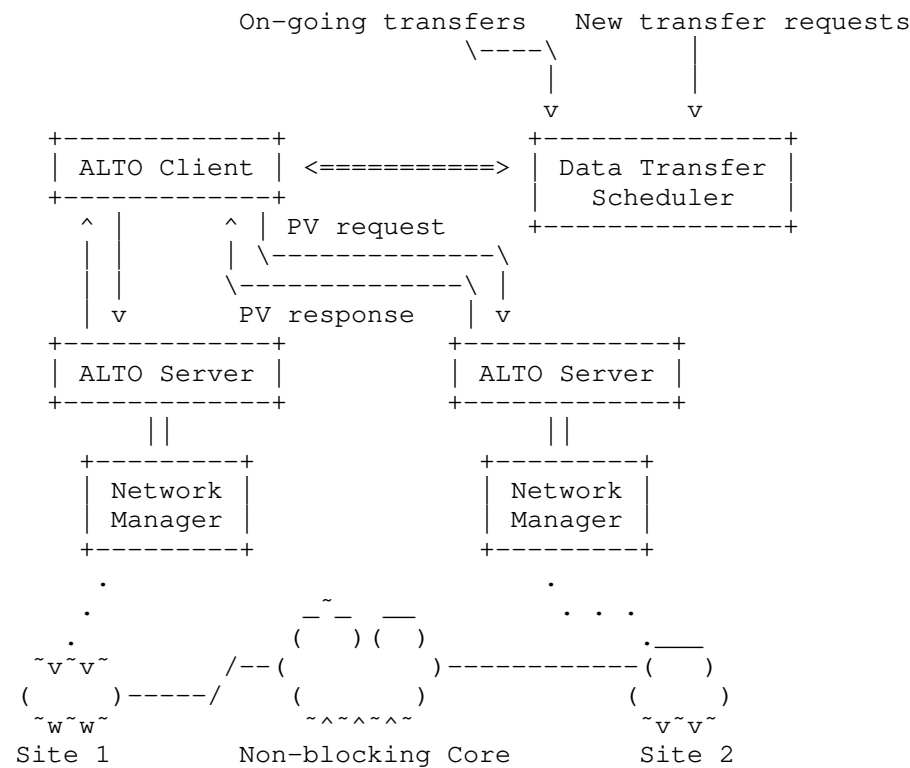
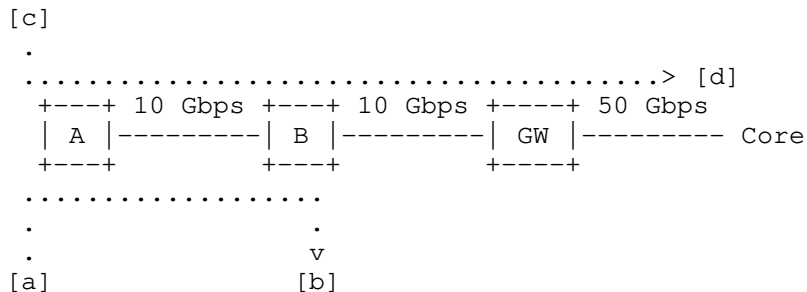


Figure 4: Example Use Case for Cross-site Bottleneck Discovery

Site 1:



Site 2:

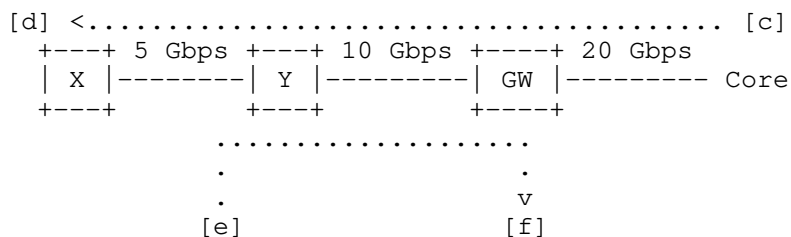


Figure 5: Example: Three Flows in Two Sites

With the Path Vector extension, a site can reveal the bottlenecks inside its own network with necessary information (such as link capacities) to the ALTO client, instead of providing the full topology and routing information, or no bottleneck information at all. The bottleneck information can be used to analyze the impact of adding/removing data transfer flows, e.g., using the [G2] framework. For example, assume hosts "a", "b", "c" are in site 1 and hosts "d", "e", "f" are in site 2, and there are 3 flows in two sites: "a -> b", "c -> d", "e -> f". For these flows, site 1 returns:

```

a: { b: [ane1] },
c: { d: [ane1, ane2, ane3] }

ane1: bw = 10 Gbps (link: A->B)
ane2: bw = 10 Gbps (link: B->GW)
ane3: bw = 50 Gbps (link: GW->Core)

```

and site 2 returns:

```
c: { d: [anei, aneii, aneiii] }  
e: { f: [aneiv] }
```

```
anei: bw = 5 Gbps (link Y->X)  
aneii: bw = 10 Gbps (link GW->Y)  
aneiii: bw = 20 Gbps (link Core->GW)  
aneiv: bw = 10 Gbps (link Y->GW)
```

With the information, the data transfer scheduler can use algorithms such as the theory on bottleneck structure [G2] to predict the potential throughput of the flows.

4.2.2. Resource Exposure for CDN and Service Edge

A growing trend in today's applications (2021) is to bring storage and computation closer to the end users for better QoE, such as Content Delivery Network (CDN), AR/VR, and cloud gaming, as reported in various documents (e.g., [SEREDGE] and [MOWIE]). Internet Service Providers may deploy multiple layers of CDN caches, or more generally service edges, with different latency and available resources including number of CPU cores, memory, and storage.

For example, Figure 6 illustrates a typical edge-cloud scenario where memory is measured in Gigabytes (G) and storage is measured in Terabytes (T). The "on-premise" edge nodes are closest to the end hosts and have the smallest latency, and the site-radio edge node and access central office (CO) have larger latency but more available resources.

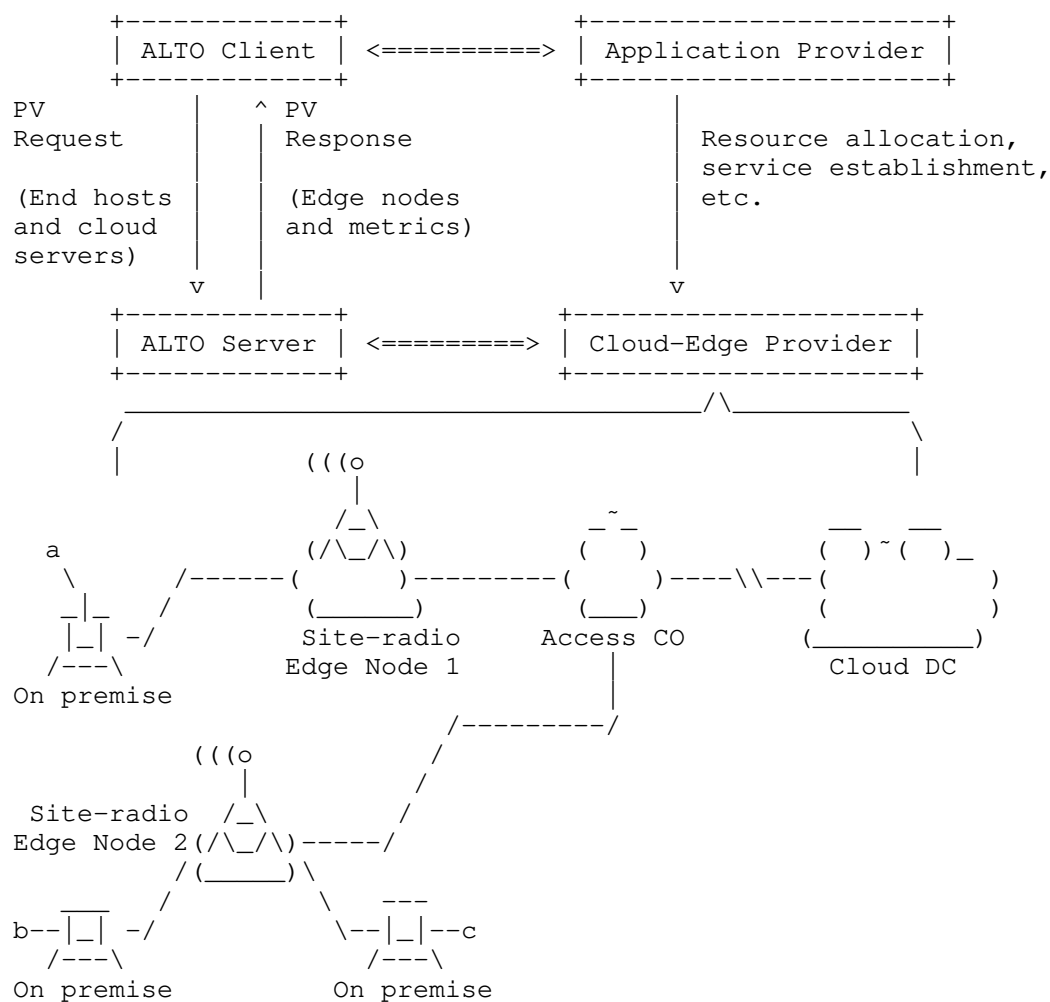


Figure 6: Example Use Case for Service Edge Exposure

```
a: { b: [ane1, ane2, ane3, ane4, ane5],  
      c: [ane1, ane2, ane3, ane4, ane6],  
      DC: [ane1, ane2, ane3] }  
b: { c: [ane5, ane4, ane6], DC: [ane5, ane4, ane3] }  
  
ane1: latency=5ms cpu=2 memory=8G storage=10T  
(on premise, a)  
  
ane2: latency=20ms cpu=4 memory=8G storage=10T  
(Site-radio Edge Node 1)  
  
ane3: latency=100ms cpu=8 memory=128G storage=100T  
(Access CO)  
  
ane4: latency=20ms cpu=4 memory=8G storage=10T  
(Site-radio Edge Node 2)  
  
ane5: latency=5ms cpu=2 memory=8G storage=10T  
(on premise, b)  
  
ane6: latency=5ms cpu=2 memory=8G storage=10T  
(on premise, c)
```

Figure 7: Example Service Edge Query Results

With the extension defined in this document, an ALTO server can selectively reveal the CDNs and service edges that reside along the paths between different end hosts and/or the cloud servers, together with their properties such as capabilities (e.g., storage, GPU) and available Service Level Agreement (SLA) plans. See Figure 7 for an example where the query is made for sources [a, b] and destinations [b, c, DC]. Here each ANE represents a service edge and the properties include access latency, available resources, etc. Note the properties here are only used for illustration purposes and are not part of this extension.

With the service edge information, an ALTO client may better conduct CDN request routing or offload functionalities from the user equipment to the service edge, with considerations on customized quality of experience.

5. Path Vector Extension: Overview

This section provides a non-normative overview of the Path Vector extension defined in this document. It is assumed that the readers are familiar with both the base protocol [RFC7285] and the Unified Property Map extension [I-D.ietf-alto-unified-props-new].

To satisfy the additional requirements listed in Section 4.1, this extension:

1. introduces the concept of Abstract Network Element (ANE) as the abstraction of components in a network whose properties may have an impact on the end-to-end performance of the traffic handled by those components,
2. extends the Cost Map and Endpoint Cost Service to convey the ANEs traversed by the path of a <source, destination> pair as Path Vectors, and
3. uses the Unified Property Map to convey the association between the ANEs and their properties.

Thus, an ALTO client can learn about the ANEs that are important to assess the QoE of different <source, destination> pairs by investigating the corresponding Path Vector value (AR1), identify common ANEs if an ANE appears in the Path Vectors of multiple <source, destination> pairs (AR2), and retrieve the properties of the ANEs by searching the Unified Property Map (AR3).

5.1. Abstract Network Element (ANE)

This extension introduces ANE as an indirect and network-agnostic way to specify a component or an aggregation of components of a network whose properties have an impact on the end-to-end performance for application traffic between endpoints.

ANEs allow ALTO servers to focus on common properties of different types of network components. For example, the throughput of a flow can be constrained by different components in a network: the capacity of a physical link, the maximum throughput of a firewall, the reserved bandwidth of an MPLS tunnel, etc. See the example below, assume the throughput of the firewall is 100 Mbps and the capacity for link (A, B) is also 100 Mbps, they result in the same constraint on the total throughput of f1 and f2. Thus, they are identical when treated as an ANE.



When an ANE is defined by an ALTO server, it is assigned an identifier by the ALTO server, i.e., a string of type ANEName as specified in Section 6.1, and a set of associated properties.

5.1.1. ANE Entity Domain

In this extension, the associations between ANE and the properties are conveyed in a Unified Property Map. Thus, ANEs must constitute an entity domain (Section 5.1 of [I-D.ietf-alto-unified-props-new]), and each ANE property must be an entity property (Section 5.2 of [I-D.ietf-alto-unified-props-new]).

Specifically, this document defines a new entity domain called "ane" as specified in Section 6.2 and defines two initial properties for the ANE entity domain.

5.1.2. Ephemeral and Persistent ANEs

By design, ANEs are ephemeral and not to be used in further requests to other ALTO resources. More precisely, the corresponding ANE names are no longer valid beyond the scope of a Path Vector response or the incremental update stream for a Path Vector request. Compared with globally unique ANE names, ephemeral ANE has several benefits including better privacy of the ISP's internal structure and more flexible ANE computation.

For example, an ALTO server may define an ANE for each aggregated bottleneck link between the sources and destinations specified in the request. For requests with different sources and destinations, the bottlenecks may be different but can safely reuse the same ANE names. The client can still adjust its traffic based on the information but is difficult to infer the underlying topology with multiple queries.

However, sometimes an ISP may intend to selectively reveal some "persistent" network components which, opposite to being ephemeral, have a longer life cycle. For example, an ALTO server may define an ANE for each service edge cluster. Once a client chooses to use a service edge, e.g., by deploying some user-defined functions, it may want to stick to the service edge to avoid the complexity of state transition or synchronization, and continuously query the properties of the edge cluster.

This document provides a mechanism to expose such network components as persistent ANEs. A persistent ANE has a persistent ID that is registered in a Property Map, together with their properties. See Section 6.2.4 and Section 6.4.2 for more detailed instructions on how to identify ephemeral ANEs and persistent ANEs.

5.1.3. Property Filtering

Resource-constrained ALTO clients (see Section 4.1.2 of [RFC7285]) may benefit from the filtering of Path Vector query results at the ALTO server, as an ALTO client may only require a subset of the available properties.

Specifically, the available properties for a given resource are announced in the Information Resource Directory as a new capability called "ane-property-names". The properties selected by a client as being of interest are specified in the subsequent Path Vector queries using the filter called 'ane-property-names'. The response includes and only includes the selected properties for the ANEs in the response.

The "ane-property-names" capability for Cost Map and for Endpoint Cost Service is specified in Section 7.2.4 and Section 7.3.4 respectively. The "ane-property-names" filter for Cost Map and Endpoint Cost Service is specified in Section 7.2.3 and Section 7.3.3 accordingly.

5.2. Path Vector Cost Type

For an ALTO client to correctly interpret the Path Vector, this extension specifies a new cost type called the Path Vector cost type.

The Path Vector cost type must convey both the interpretation and semantics in the "cost-mode" and "cost-metric" respectively. Unfortunately, a single "cost-mode" value cannot fully specify the interpretation of a Path Vector, which is a compound data type. For example, in programming languages such as C++ where there existed a JSON array type named JSONArray, a Path Vector will have the type of JSONArray<ANENAME>.

Instead of extending the "type system" of ALTO, this document takes a simple and backward compatible approach. Specifically, the "cost-mode" of the Path Vector cost type is "array", which indicates the value is a JSON array. Then, an ALTO client must check the value of the "cost-metric". If the value is "ane-path", it means that the JSON array should be further interpreted as a path of ANENAMES.

The Path Vector cost type is specified in Section 6.5.

5.3. Multipart Path Vector Response

For a basic ALTO information resource, a response contains only one type of ALTO resources, e.g., Network Map, Cost Map, or Property Map. Thus, only one round of communication is required: An ALTO client sends a request to an ALTO server, and the ALTO server returns a response, as shown in Figure 8.

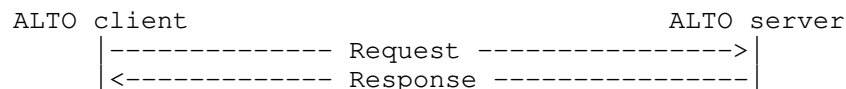


Figure 8: A Typical ALTO Request and Response

The extension defined in this document, on the other hand, involves two types of information resources: Path Vectors conveyed in an `InfoResourceCostMap` (defined in Section 11.2.3.6 of [RFC7285]) or an `InfoResourceEndpointCostMap` (defined in Section 11.5.1.6 of [RFC7285]), and ANE properties conveyed in an `InfoResourceProperties` (defined in Section 7.6 of [I-D.ietf-alto-unified-props-new]).

Instead of two consecutive message exchanges, the extension defined in this document enforces one round of communication. Specifically, the ALTO client must include the source and destination pairs and the requested ANE properties in a single request, and the ALTO server must return a single response containing both the Path Vectors and properties associated with the ANEs in the Path Vectors, as shown in Figure 9. Since the two parts are bundled together in one response message, their orders are interchangeable. See Section 7.2.6 and Section 7.3.6 for details.

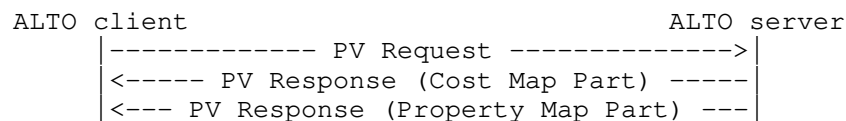


Figure 9: The Path Vector Extension Request and Response

This design is based on the following considerations:

1. ANEs may be constructed on demand, and potentially based on the requested properties (See Section 5.1 for more details). If sources and destinations are not in the same request as the properties, an ALTO server either cannot construct ANEs on-demand, or must wait until both requests are received.

2. As ANEs may be constructed on demand, mappings of each ANE to its underlying network devices and resources can be specific to the request. In order to respond to the Property Map request correctly, an ALTO server must store the mapping of each Path Vector request until the client fully retrieves the property information. The "stateful" behavior may substantially harm the server scalability and potentially lead to Denial-of-Service attacks.

One approach to realize the one-round communication is to define a new media type to contain both objects, but this violates modular design. This document follows the standard-conforming usage of "multipart/related" media type defined in [RFC2387] to elegantly combine the objects. Path Vectors are encoded in an InfoResourceCostMap or an InfoResourceEndpointCostMap, and the Property Map is encoded in an InfoResourceProperties. They are encapsulated as parts of a multipart message. The modular composition allows ALTO servers and clients to reuse the data models of the existing information resources. Specifically, this document addresses the following practical issues using "multipart/related".

5.3.1. Identifying the Media Type of the Root Object

ALTO uses media type to indicate the type of an entry in the Information Resource Directory (IRD) (e.g., "application/alto-costmap+json" for Cost Map and "application/alto-endpointcost+json" for Endpoint Cost Service). Simply putting "multipart/related" as the media type, however, makes it impossible for an ALTO client to identify the type of service provided by related entries.

To address this issue, this document uses the "type" parameter to indicate the root object of a multipart/related message. For a Cost Map resource, the "media-type" field in the IRD entry is "multipart/related" with the parameter "type=application/alto-costmap+json"; for an Endpoint Cost Service, the parameter is "type=application/alto-endpointcost+json".

5.3.2. References to Part Messages

As the response of a Path Vector resource is a multipart message with two different parts, it is important that each part can be uniquely identified. Following the designs of [RFC8895], this extension requires that an ALTO server assigns a unique identifier to each part of the multipart response message. This identifier, referred to as a Part Resource ID (See Section 6.6 for details), is present in the part message's "Content-ID" header. By concatenating the Part Resource ID to the identifier of the Path Vector request, an ALTO server/client can uniquely identify the Path Vector Part or the

Property Map part.

6. Specification: Basic Data Types

6.1. ANE Name

An ANE Name is encoded as a JSON string with the same format as that of the type PIDName (Section 10.1 of [RFC7285]).

The type ANENAME is used in this document to indicate a string of this format.

6.2. ANE Entity Domain

The ANE entity domain associates property values with the Abstract Network Elements in a Property Map. Accordingly, the ANE entity domain always depends on a Property Map.

It must be noted that the term "domain" here does not refer to a network domain. Rather, it is inherited from the "entity domain" defined in Sec 3.2 in [I-D.ietf-alto-unified-props-new] that represents the set of valid entities defined by an ALTO information resource (called the defining information resource).

6.2.1. Entity Domain Type

The Entity Domain Type is "ane".

6.2.2. Domain-Specific Entity Identifier

The entity identifiers are the ANE Names in the associated Property Map.

6.2.3. Hierarchy and Inheritance

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

6.2.4. Media Type of Defining Resource

The defining resource for entity domain type "ane" MUST be a Property Map, i.e., the media type of defining resources is:

application/alto-propmap+json

Specifically, for ephemeral ANEs that appear in a Path Vector response, their entity domain names MUST be exactly ".ane" and the defining resource of these ANEs is the Property Map part of the

multipart response. Meanwhile, for any persistent ANE whose defining resource is a Property Map resource, its entity domain name MUST have the format of "PROPMAP.ane" where PROPMAP is the resource ID of the defining resource. Persistent entities are "persistent" because standalone queries can be made by an ALTO client to their defining resource(s) when the connection to the Path Vector service is closed.

For example, the defining resource of an ephemeral ANE whose entity identifier is ".ane:NET1" is the Property Map part that contains this identifier. The defining resource of a persistent ANE whose entity identifier is "dc-props.ane:DC1" is the Property Map with the resource ID "dc-props".

6.3. ANE Property Name

An ANE Property Name is encoded as a JSON string with the same format as that of Entity Property Name (Section 5.2.2 of [I-D.ietf-alto-unified-props-new]).

6.4. Initial ANE Property Types

Two initial ANE property types are specified, "max-reservable-bandwidth" and "persistent-entity-id".

Note that these property types do not depend on any information resource. As such, the EntityPropertyName MUST only have the EntityPropertyType part.

6.4.1. Maximum Reservable Bandwidth

The maximum reservable bandwidth property ("max-reservable-bandwidth") stands for the maximum bandwidth that can be reserved for all the traffic that traverses an ANE. The value MUST be encoded as a non-negative numerical cost value as defined in Section 6.1.2.1 of [RFC7285] and the unit is bit per second (bps). If this property is requested by the ALTO client but not present for an ANE in the server response, it MUST be interpreted as that the property is not defined for the ANE.

This property can be offered in a setting where the ALTO server is part of a network system that provides on-demand resource allocation and the ALTO client is part of a user application. One existing example is [NOVA]: the ALTO server is part of an SDN controller and exposes a list of traversed network elements and associated link bandwidth to the client. The encoding in [NOVA] differs from the Path Vector response defined in this document that the Path Vector part and Property Map part are put in the same JSON object.

In such a framework, the ALTO server exposes resource (e.g., reservable bandwidth) availability information to the ALTO client. How the client makes resource requests based on the information and how the resource allocation is achieved respectively depend on interfaces between the management system and the users or a higher-layer protocol (e.g., SDN network intents or MPLS tunnels), which are out of the scope of this document.

6.4.2. Persistent Entity ID

The persistent entity ID property is the entity identifier of the persistent ANE which an ephemeral ANE presents (See Section 5.1.2 for details). The value of this property is encoded with the format EntityID defined in Section 5.1.3 of [I-D.ietf-alto-unified-props-new].

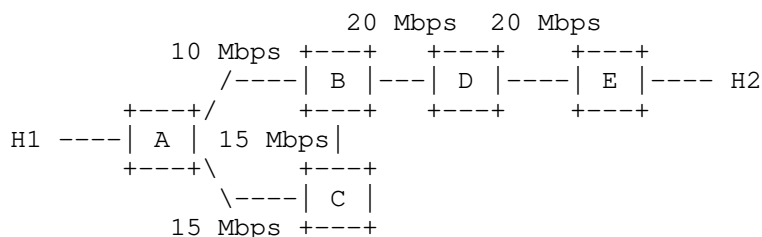
In this format, the entity ID combines:

- * a defining information resource for the ANE on which a "persistent-entity-id" is queried, which is the Property Map resource defining the ANE as a persistent entity, together with the properties;
- * the persistent name of the ANE in that Property Map.

With this format, the client has all the needed information for further standalone query properties on the persistent ANE.

6.4.3. Examples

To illustrate the use of "max-reservable-bandwidth", consider the following network with 5 nodes. Assume the client wants to query the maximum reservable bandwidth from H1 to H2. An ALTO server may split the network into two ANEs: "ane1" that represents the subnetwork with routers A, B, and C, and "ane2" that represents the subnetwork with routers B, D and E. The maximum reservable bandwidth for "ane1" is 15 Mbps (using path A->C->B) and the maximum reservable bandwidth for "ane2" is 20 Mbps (using path B->D->E).



To illustrate the use of "persistent-entity-id", consider the scenario in Figure 6. As the life cycle of service edges are typically long, they may contain information that is not specific to the query. Such information can be stored in an individual unified property map and later be accessed by an ALTO client.

For example, "ane1" in Figure 7 represents the on-premise service edge closest to host a. Assume the properties of the service edges are provided in a unified property map called "se-props" and the ID of the on-premise service edge is "9a0b55f7-7442-4d56-8a2c-b4cc6a8e3aa1", the "persistent-entity-id" of "ane1" will be "se-props.ane:9a0b55f7-7442-4d56-8a2c-b4cc6a8e3aa1". With this persistent entity ID, an ALTO client may send queries to the "se-props" resource with the entity ID ".ane:9a0b55f7-7442-4d56-8a2c-b4cc6a8e3aa1".

6.5. Path Vector Cost Type

This document defines a new cost type, which is referred to as the Path Vector cost type. An ALTO server MUST offer this cost type if it supports the extension defined in this document.

6.5.1. Cost Metric: ane-path

The cost metric "ane-path" indicates the value of such a cost type conveys an array of ANE names, where each ANE name uniquely represents an ANE traversed by traffic from a source to a destination.

An ALTO client MUST interpret the Path Vector as if the traffic between a source and a destination logically traverses the ANEs in the same order as they appear in the Path Vector.

When the Path Vector procedures defined in this document are in use, an ALTO server using the "ane-path" cost metric and the "array" cost mode (see Section 6.5.2) MUST return as the cost value a JSON array of ANEName and the client MUST also check that each element contained in the array is an ANEName (Section 6.1). Otherwise, the client MUST discard the response and SHOULD follow the instructions in Section 8.3.4.3 of [RFC7285] to handle the error.

6.5.2. Cost Mode: array

The cost mode "array" indicates that every cost value in the response body of a (Filtered) Cost Map or an Endpoint Cost Service MUST be interpreted as a JSON array object. While this cost mode can be applied to all cost metrics, additional specifications will be needed to clarify the semantics of the array cost mode when combined with cost metrics other than 'ane-path'.

6.6. Part Resource ID and Part Content ID

A Part Resource ID is encoded as a JSON string with the same format as that of the type ResourceID (Section 10.2 of [RFC7285]).

Even though the client-id assigned to a Path Vector request and the Part Resource ID MAY contain up to 64 characters by their own definition, their concatenation (see Section 5.3.2) MUST also conform to the same length constraint. The same requirement applies to the resource ID of the Path Vector resource, too. Thus, it is RECOMMENDED to limit the length of resource ID and client ID related to a Path Vector resource to 31 characters.

A Part Content ID conforms to the format of msg-id as specified in [RFC2387] and [RFC5322]. Specifically, it has the following format:

"<" PART-RESOURCE-ID "@" DOMAIN-NAME ">"

PART-RESOURCE-ID: PART-RESOURCE-ID has the same format as the Part Resource ID. It is used to identify whether a part message is a Path Vector or a Property Map.

DOMAIN-NAME: DOMAIN-NAME has the same format as dot-atom-text specified in Section 3.2.3 of [RFC5322]. It must be the domain name of the ALTO server.

7. Specification: Service Extensions

7.1. Notations

This document uses the same syntax and notations as introduced in Section 8.2 of RFC 7285 [RFC7285] to specify the extensions to existing ALTO resources and services.

7.2. Multipart Filtered Cost Map for Path Vector

This document introduces a new ALTO resource called multipart Filtered Cost Map resource, which allows an ALTO server to provide other ALTO resources associated with the Cost Map resource in the same response.

7.2.1. Media Type

The media type of the multipart Filtered Cost Map resource is "multipart/related" and the required "type" parameter MUST have a value of "application/alto-costmap+json".

7.2.2. HTTP Method

The multipart Filtered Cost Map is requested using the HTTP POST method.

7.2.3. Accept Input Parameters

The input parameters of the multipart Filtered Cost Map are supplied in the body of an HTTP POST request. This document extends the input parameters to a Filtered Cost Map, which is defined as a JSON object of type ReqFilteredCostMap in Section 4.1.2 of RFC 8189 [RFC8189], with a data format indicated by the media type "application/alto-costmapfilter+json", which is a JSON object of type PVReqFilteredCostMap:

```
object {  
  [EntityPropertyName ane-property-names<0..*>;]  
} PVReqFilteredCostMap : ReqFilteredCostMap;
```

with fields:

ane-property-names: A list of selected ANE properties to be included in the response. Each property in this list MUST match one of the supported ANE properties indicated in the resource's "ane-property-names" capability (Section 7.2.4). If the field is not present, it MUST be interpreted as an empty list.

Example: Consider the network in Figure 1. If an ALTO client wants to query the "max-reservable-bandwidth" between PID1 and PID2, it can submit the following request.

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
       application/alto-error+json
Content-Length: 201
Content-Type: application/alto-costmapfilter+json

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "pids": {
    "srcs": [ "PID1" ],
    "dsts": [ "PID2" ]
  },
  "ane-property-names": [ "max-reservable-bandwidth" ]
}
```

7.2.4. Capabilities

The multipart Filtered Cost Map resource extends the capabilities defined in Section 4.1.1 of [RFC8189]. The capabilities are defined by a JSON object of type PVFilteredCostMapCapabilities:

```
object {
  [EntityTypePropertyName ane-property-names<0..*>;]
} PVFilteredCostMapCapabilities : FilteredCostMapCapabilities;
```

with fields:

ane-property-names: Defines a list of ANE properties that can be returned. If the field is not present, it MUST be interpreted as an empty list, indicating the ALTO server cannot provide any ANE property.

This extension also introduces additional restrictions for the following fields:

cost-type-names: The "cost-type-names" field MUST include the Path Vector cost type, unless explicitly documented by a future extension. This also implies that the Path Vector cost type MUST be defined in the "cost-types" of the Information Resource Directory's "meta" field.

cost-constraints: If the "cost-type-names" field includes the Path Vector cost type, "cost-constraints" field MUST be "false" or not present unless specifically instructed by a future document.

testable-cost-type-names (Section 4.1.1 of [RFC8189]): If the "cost-type-names" field includes the Path Vector cost type and the "testable-cost-type-names" field is present, the Path Vector cost type MUST NOT be included in the "testable-cost-type-names" field unless specifically instructed by a future document.

7.2.5. Uses

This member MUST include the resource ID of the network map based on which the PIDs are defined. If this resource supports "persistent-entity-id", it MUST also include the defining resources of persistent ANEs that may appear in the response.

7.2.6. 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.

The "Content-Type" header of the response MUST be "multipart/related" as defined by [RFC2387] with the following parameters:

type: The type parameter is mandatory and MUST be "application/alto-costmap+json". Note that [RFC2387] permits both parameters with and without the double quotes.

start: The start parameter is as defined in [RFC2387] and is optional. If present, it MUST have the same value as the "Content-ID" header of the Path Vector part.

boundary: The boundary parameter is as defined in Section 5.1.1 of [RFC2046] and is mandatory.

The body of the response MUST consist of two parts:

- * The Path Vector part MUST include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-costmap+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Path Vector part MUST be a JSON object with the same format as defined in Section 11.2.3.6 of [RFC7285] when the "cost-type" field is present in the input parameters and MUST be a JSON object with the same format as defined in Section 4.1.3 of [RFC8189] if the "multi-cost-types" field is present. The JSON object MUST include the "vtag" field in the "meta" field, which provides the version tag of the returned CostMapData. The resource ID of the version tag MUST follow the format of

resource-id '.' part-resource-id

where "resource-id" is the resource Id of the Path Vector resource, and "part-resource-id" has the same value as the PART-RESOURCE-ID in the "Content-ID" of the Path Vector part. The "meta" field MUST also include the "dependent-vtags" field, whose value is a single-element array to indicate the version tag of the network map used, where the network map is specified in the "uses" attribute of the multipart Filtered Cost Map resource in IRD.

- * The Unified Property Map part MUST also include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-propmap+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Unified Property Map part is a JSON object with the same format as defined in Section 7.6 of [I-D.ietf-alto-unified-props-new]. The JSON object MUST include the "dependent-vtags" field in the "meta" field. The value of the "dependent-vtags" field MUST be an array of VersionTag objects as defined by Section 10.3 of [RFC7285]. The "vtag" of the Path Vector part MUST be included in the "dependent-vtags". If "persistent-entity-id" is requested, the version tags of the dependent resources that may expose the entities in the response MUST also be included.

The PropertyMapData has one member for each ANENAME that appears in the Path Vector part, which is an entity identifier belonging to the self-defined entity domain as defined in Section 5.1.2.3 of [I-D.ietf-alto-unified-props-new]. The EntityProps for each ANE has one member for each property that is both 1) associated with the ANE, and 2) specified in the "ane-property-names" in the request. If the Path Vector cost type is not included in the "cost-type" field or the "multi-cost-type" field, the "property-map" field MUST be present and the value MUST be an empty object ({}).

A complete and valid response MUST include both the Path Vector part and the Property Map part in the multipart message. If any part is NOT present, the client MUST discard the received information and send another request if necessary.

According to [RFC2387], the Path Vector part, whose media type is the same as the "type" parameter of the multipart response message, is the root object. Thus, it is the element the application processes first. Even though the "start" parameter allows it to be placed anywhere in the part sequence, it is RECOMMENDED that the parts

arrive in the same order as they are processed, i.e., the Path Vector part is always put as the first part, followed by the Property Map part. When doing so, an ALTO server MAY choose not to set the "start" parameter, which implies the first part is the root object.

Example: Consider the network in Figure 1. The response of the example request in Section 7.2.3 is as follows, where "ANE1" represents the aggregation of all the switches in the network.

```
HTTP/1.1 200 OK
Content-Length: 859
Content-Type: multipart/related; boundary=example-1;
              type=application/alto-costmap+json

--example-1
Content-ID: <costmap@alto.example.com>
Content-Type: application/alto-costmap+json

{
  "meta": {
    "vtag": {
      "resource-id": "filtered-cost-map-pv.costmap",
      "tag": "fb20b76204814e9db37a51151faaaef2"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "75ed013b3cb58f896e839582504f6228"
      }
    ],
    "cost-type": { "cost-mode": "array", "cost-metric": "ane-path" }
  },
  "cost-map": {
    "PID1": { "PID2": ["ANE1"] }
  }
}

--example-1
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "filtered-cost-map-pv.costmap",
        "tag": "fb20b76204814e9db37a51151faaaef2"
      }
    ]
  },
  "property-map": {
    ".ane:ANE1": { "max-reservable-bandwidth": 100000000 }
  }
}
```

7.3. Multipart Endpoint Cost Service for Path Vector

This document introduces a new ALTO resource called multipart Endpoint Cost Service, which allows an ALTO server to provide other ALTO resources associated with the Endpoint Cost Service resource in the same response.

7.3.1. Media Type

The media type of the multipart Endpoint Cost Service resource is "multipart/related" and the required "type" parameter MUST have a value of "application/alto-endpointcost+json".

7.3.2. HTTP Method

The multipart Endpoint Cost Service resource is requested using the HTTP POST method.

7.3.3. Accept Input Parameters

The input parameters of the multipart Endpoint Cost Service resource are supplied in the body of an HTTP POST request. This document extends the input parameters to an Endpoint Cost Service, which is defined as a JSON object of type ReqEndpointCost in Section 4.2.2 of [RFC8189], with a data format indicated by the media type "application/alto-endpointcostparams+json", which is a JSON object of type PVReqEndpointCost:

```
object {  
  [EntityPropertyName ane-property-names<0..*>;]  
} PVReqEndpointcost : ReqEndpointcostMap;
```

with fields:

ane-property-names: This document defines the "ane-property-names" in PVReqEndpointcost as the same as in PVReqFilteredCostMap. See Section 7.2.3.

Example: Consider the network in Figure 1. If an ALTO client wants to query the "max-reservable-bandwidth" between eh1 and eh2, it can submit the following request.

```
POST /ecs/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: 227
Content-Type: application/alto-endpointcostparams+json
```

```
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoints": {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [ "ipv4:192.0.2.18" ]
  },
  "ane-property-names": [ "max-reservable-bandwidth" ]
}
```

7.3.4. Capabilities

The capabilities of the multipart Endpoint Cost Service resource are defined by a JSON object of type `PVEndpointcostCapabilities`, which is defined as the same as `PVFilteredCostMapCapabilities`. See Section 7.2.4.

7.3.5. Uses

If this resource supports "persistent-entity-id", it MUST also include the defining resources of persistent ANEs that may appear in the response.

7.3.6. 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.

The "Content-Type" header of the response MUST be "multipart/related" as defined by [RFC7285] with the following parameters:

type: The type parameter MUST be "application/alto-endpointcost+json" and is mandatory.

start: The start parameter is as defined in Section 7.2.6.

boundary: The boundary parameter is as defined in Section 5.1.1 of [RFC2046] and is mandatory.

The body MUST consist of two parts:

- * The Path Vector part MUST include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-endpointcost+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Path Vector part MUST be a JSON object with the same format as defined in Section 11.5.1.6 of [RFC7285] when the "cost-type" field is present in the input parameters and MUST be a JSON object with the same format as defined in Section 4.2.3 of [RFC8189] if the "multi-cost-types" field is present. The JSON object MUST include the "vtag" field in the "meta" field, which provides the version tag of the returned EndpointCostMapData. The resource ID of the version tag MUST follow the format of

resource-id '.' part-resource-id

where "resource-id" is the resource Id of the Path Vector resource, and "part-resource-id" has the same value as the PART-RESOURCE-ID in the "Content-ID" of the Path Vector part.

- * The Unified Property Map part MUST also include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-propmap+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Unified Property Map part MUST be a JSON object with the same format as defined in Section 7.6 of [I-D.ietf-alto-unified-props-new]. The JSON object MUST include the "dependent-vtags" field in the "meta" field. The value of the "dependent-vtags" field MUST be an array of VersionTag objects as defined by Section 10.3 of [RFC7285]. The "vtag" of the Path Vector part MUST be included in the "dependent-vtags". If "persistent-entity-id" is requested, the version tags of the dependent resources that may expose the entities in the response MUST also be included.

The PropertyMapData has one member for each ANENAME that appears in the Path Vector part, which is an entity identifier belonging to the self-defined entity domain as defined in Section 5.1.2.3 of [I-D.ietf-alto-unified-props-new]. The EntityProps for each ANE has one member for each property that is both 1) associated with the ANE, and 2) specified in the "ane-property-names" in the request. If the Path Vector cost type is not included in the "cost-type" field or the "multi-cost-type" field, the "property-map" field MUST be present and the value MUST be an empty object ({}).

A complete and valid response MUST include both the Path Vector part and the Property Map part in the multipart message. If any part is NOT present, the client MUST discard the received information and send another request if necessary.

According to [RFC2387], the Path Vector part, whose media type is the same as the "type" parameter of the multipart response message, is the root object. Thus, it is the element the application processes first. Even though the "start" parameter allows it to be placed anywhere in the part sequence, it is RECOMMENDED that the parts arrive in the same order as they are processed, i.e., the Path Vector part is always put as the first part, followed by the Property Map part. When doing so, an ALTO server MAY choose not to set the "start" parameter, which implies the first part is the root object.

Example: Consider the network in Figure 1. The response of the example request in Section 7.3.3 is as follows.

```
HTTP/1.1 200 OK
Content-Length: 845
Content-Type: multipart/related; boundary=example-1;
              type=application/alto-endpointcost+json
```

```
--example-1
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "vtag": {
      "resource-id": "ecs-pv.ecs",
      "tag": "ec137bb78118468c853d5b622ac003f1"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "677fe5f4066848d282ece213a84f9429"
      }
    ],
    "cost-type": { "cost-mode": "array", "cost-metric": "ane-path" },
    "cost-map": {
      "ipv4:192.0.2.2": { "ipv4:192.0.2.18": ["ANE1"] }
    }
  }
}
```

```
--example-1
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json
```

```
{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "ecs-pv.ecs",
        "tag": "ec137bb78118468c853d5b622ac003f1"
      }
    ]
  },
  "property-map": {
    ".ane:ANE1": { "max-reservable-bandwidth": 100000000 }
  }
}
```

8. Examples

This section lists some examples of Path Vector queries and the corresponding responses. Some long lines are truncated for better readability.

8.1. Sample Setup

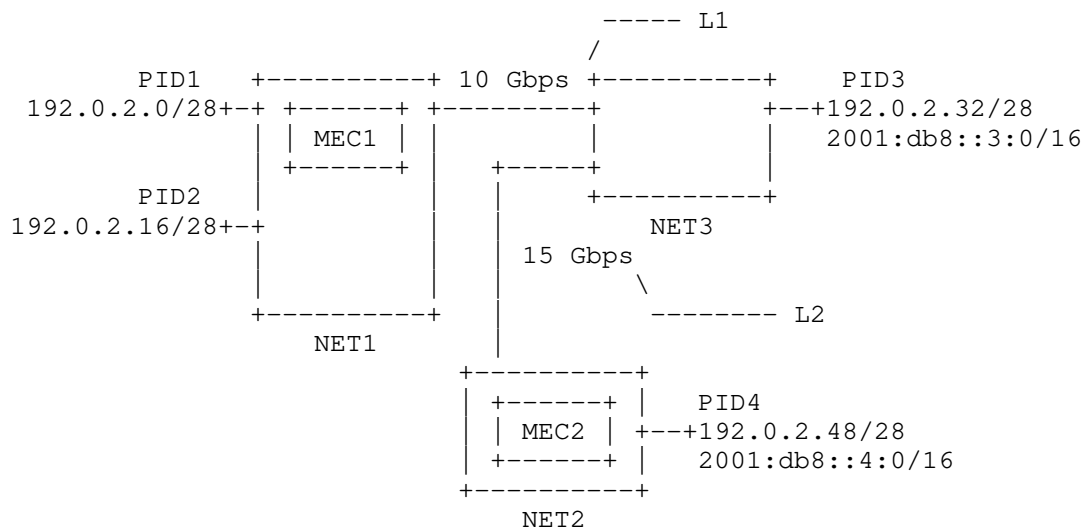


Figure 10: Examples of ANE Properties

In this document, Figure 10 is used to illustrate the message contents. There are 3 sub-networks (NET1, NET2 and NET3) and two interconnection links (L1 and L2). It is assumed that each sub-network has sufficiently large bandwidth to be reserved.

8.2. Information Resource Directory

To give a comprehensive example of the extension defined in this document, we consider the network in Figure 10. Assume that the ALTO server provides the following information resources:

- * "my-default-networkmap": A Network Map resource which contains the PIDs in the network.
- * "filtered-cost-map-pv": A Multipart Filtered Cost Map resource for Path Vector, which exposes the "max-reservable-bandwidth" property for the PIDs in "my-default-networkmap".

- * "ane-props": A filtered Unified Property resource that exposes the information for persistent ANEs in the network.
- * "endpoint-cost-pv": A Multipart Endpoint Cost Service for Path Vector, which exposes the "max-reservable-bandwidth" and the "persistent-entity-id" properties.
- * "update-pv": An Update Stream service, which provides the incremental update service for the "endpoint-cost-pv" service.
- * "multicost-pv": A Multipart Endpoint Cost Service with both Multi-Cost and Path Vector.

Below is the Information Resource Directory of the example ALTO server. To enable the extension defined in this document, the "path-vector" cost type (Section 6.5) is defined in the "cost-types" of the "meta" field, and is included in the "cost-type-names" of resources "filtered-cost-map-pv" and "endpoint-cost-pv".

```
{
  "meta": {
    "cost-types": {
      "path-vector": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      },
      "num-rc": {
        "cost-mode": "numerical",
        "cost-metric": "routingcost"
      }
    }
  },
  "resources": {
    "my-default-networkmap": {
      "uri": "https://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },
    "filtered-cost-map-pv": {
      "uri": "https://alto.example.com/costmap/pv",
      "media-type": "multipart/related;
        type=application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "capabilities": {
        "cost-type-names": [ "path-vector" ],
        "ane-property-names": [ "max-reservable-bandwidth" ]
      },
      "uses": [ "my-default-networkmap" ]
    }
  },
}
```

```
"ane-props": {
  "uri": "https://alto.example.com/ane-props",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "capabilities": {
    "mappings": {
      ".ane": [ "cpu" ]
    }
  }
},
"endpoint-cost-pv": {
  "uri": "https://alto.exmaple.com/endpointcost/pv",
  "media-type": "multipart/related;
    type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": [ "path-vector" ],
    "ane-property-names": [
      "max-reservable-bandwidth", "persistent-entity-id"
    ]
  },
  "uses": [ "ane-props" ]
},
"update-pv": {
  "uri": "https://alto.example.com/updates/pv",
  "media-type": "text/event-stream",
  "uses": [ "endpoint-cost-pv" ],
  "accepts": "application/alto-updatestreamparams+json",
  "capabilities": {
    "support-stream-control": true
  }
},
"multicost-pv": {
  "uri": "https://alto.exmaple.com/endpointcost/mcpv",
  "media-type": "multipart/related;
    type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": [ "path-vector", "num-rc" ],
    "max-cost-types": 2,
    "testable-cost-type-names": [ "num-rc" ],
    "ane-property-names": [
      "max-reservable-bandwidth", "persistent-entity-id"
    ]
  },
  "uses": [ "ane-props" ]
}
}
```

```
}
```

8.3. Multipart Filtered Cost Map

The following examples demonstrate the request to the "filtered-cost-map-pv" resource and the corresponding response.

The request uses the "path-vector" cost type in the "cost-type" field. The "ane-property-names" field is missing, indicating that the client only requests for the Path Vector but not the ANE properties.

The response consists of two parts. The first part returns the array of ANEName for each source and destination pair. There are two ANEs, where "L1" represents the interconnection link L1, and "L2" represents the interconnection link L2.

The second part returns an empty Property Map. Note that the ANE entries are omitted since they have no properties (See Section 3.1 of [I-D.ietf-alto-unified-props-new]).

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
       application/alto-error+json
Content-Length: 153
Content-Type: application/alto-costmapfilter+json
```

```
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "pids": {
    "srcs": [ "PID1" ],
    "dsts": [ "PID3", "PID4" ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 855
Content-Type: multipart/related; boundary=example-1;
             type=application/alto-costmap+json
```

```
--example-1
Content-ID: <costmap@alto.example.com>
Content-Type: application/alto-costmap+json
```

```

{
  "meta": {
    "vtag": {
      "resource-id": "filtered-cost-map-pv.costmap",
      "tag": "d827f484cb66ce6df6b5077cb8562b0a"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "c04bc5da49534274a6daeee8ea1dec62"
      }
    ],
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "cost-map": {
    "PID1": {
      "PID3": [ "L1" ],
      "PID4": [ "L1", "L2" ]
    }
  }
}
--example-1
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "filtered-cost-map-pv.costmap",
        "tag": "d827f484cb66ce6df6b5077cb8562b0a"
      }
    ]
  },
  "property-map": {
  }
}

```

8.4. Multipart Endpoint Cost Service Resource

The following examples demonstrate the request to the "endpoint-cost-pv" resource and the corresponding response.

The request uses the Path Vector cost type in the "cost-type" field, and queries the Maximum Reservable Bandwidth ANE property and the Persistent Entity property for two IPv4 source and destination pairs (192.0.2.34 -> 192.0.2.2 and 192.0.2.34 -> 192.0.2.50) and one IPv6 source and destination pair (2001:db8::3:1 -> 2001:db8::4:1).

The response consists of two parts. The first part returns the array of ANENAME for each valid source and destination pair. As one can see in Figure 10, flow 192.0.2.34 -> 192.0.2.2 traverses NET2, L1 and NET1, and flows 192.0.2.34 -> 192.0.2.50 and 2001:db8::3:1 -> 2001:db8::4:1 traverse NET2, L2 and NET3.

The second part returns the requested properties of ANEs. Assume NET1, NET2 and NET3 has sufficient bandwidth and their "max-reservable-bandwidth" values are set to a sufficiently large number (50 Gbps in this case). On the other hand, assume there are no prior reservation on L1 and L2, and their "max-reservable-bandwidth" values are the corresponding link capacity (10 Gbps for L1 and 15 Gbps for L2).

Both NET1 and NET2 have a mobile edge deployed, i.e., MEC1 in NET1 and MEC2 in NET2. Assume the ANENAME for MEC1 and MEC2 are "MEC1" and "MEC2" and their properties can be retrieved from the Property Map "ane-props". Thus, the "persistent-entity-id" property of NET1 and NET3 are "ane-props.ane:MEC1" and "ane-props.ane:MEC2" respectively.

```
POST /endpointcost/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;
       type=application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: 362
Content-Type: application/alto-endpointcostparams+json
```

```
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoints": {
    "srcs": [
      "ipv4:192.0.2.34",
      "ipv6:2001:db8::3:1"
    ],
    "dsts": [
      "ipv4:192.0.2.2",
      "ipv4:192.0.2.50",
      "ipv6:2001:db8::4:1"
    ]
  },
  "ane-property-names": [
    "max-reservable-bandwidth",
    "persistent-entity-id"
  ]
}
```

```
HTTP/1.1 200 OK
Content-Length: 1432
Content-Type: multipart/related; boundary=example-2;
             type=application/alto-endpointcost+json
```

```
--example-2
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "vtags": {
      "resource-id": "endpoint-cost-pv.ecs",
      "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
    },
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  }
}
```

```
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.34": {
      "ipv4:192.0.2.2": [ "NET3", "L1", "NET1" ],
      "ipv4:192.0.2.50": [ "NET3", "L2", "NET2" ]
    },
    "ipv6:2001:db8::3:1": {
      "ipv6:2001:db8::4:1": [ "NET3", "L2", "NET2" ]
    }
  }
}
--example-2
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
      },
      {
        "resource-id": "ane-props",
        "tag": "bf3c8c1819d2421c9a95a9d02af557a3"
      }
    ]
  },
  "property-map": {
    ".ane:NET1": {
      "max-reservable-bandwidth": 50000000000,
      "persistent-entity-id": "ane-props.ane:MEC1"
    },
    ".ane:NET2": {
      "max-reservable-bandwidth": 50000000000,
      "persistent-entity-id": "ane-props.ane:MEC2"
    },
    ".ane:NET3": {
      "max-reservable-bandwidth": 50000000000
    },
    ".ane:L1": {
      "max-reservable-bandwidth": 10000000000
    },
    ".ane:L2": {
      "max-reservable-bandwidth": 15000000000
    }
  }
}
```

```
}
```

Under certain scenarios where the traversal order is not crucial, an ALTO server implementation may choose to not follow strictly the physical traversal order and may even obfuscate the order intentionally to preserve its own privacy or conform to its own policies. For example, an ALTO server may choose to aggregate NET1 and L1 as a new ANE with ANE name "AGGR1", and aggregate NET2 and L2 as a new ANE with ANE name "AGGR2". The "max-reservable-bandwidth" of "AGGR1" takes the value of L1, which is smaller than that of NET1, and the "persistent-entity-id" of "AGGR1" takes the value of NET1. The properties of "AGGR2" are computed in a similar way and the obfuscated response is as shown below. Note that the obfuscation of Path Vector responses is implementation-specific and is out of the scope of this document, and developers may refer to Section 11 for further references.

```
HTTP/1.1 200 OK
Content-Length: 1263
Content-Type: multipart/related; boundary=example-2;
              type=application/alto-endpointcost+json
```

```
--example-2
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "vtags": {
      "resource-id": "endpoint-cost-pv.ecs",
      "tag": "bb975862fbe3422abf4dae386b132c1d"
    },
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.34": {
      "ipv4:192.0.2.2": [ "NET3", "AGGR1" ],
      "ipv4:192.0.2.50": [ "NET3", "AGGR2" ]
    },
    "ipv6:2001:db8::3:1": {
      "ipv6:2001:db8::4:1": [ "NET3", "AGGR2" ]
    }
  }
}
```

```
--example-2
```


Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

```
{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "bb975862fbe3422abf4dae386b132c1d"
      },
      {
        "resource-id": "ane-props",
        "tag": "bf3c8c1819d2421c9a95a9d02af557a3"
      }
    ]
  },
  "property-map": {
    ".ane:AGGR1": {
      "max-reservable-bandwidth": 10000000000,
      "persistent-entity-id": "ane-props.ane:MEC1"
    },
    ".ane:AGGR2": {
      "max-reservable-bandwidth": 15000000000,
      "persistent-entity-id": "ane-props.ane:MEC2"
    },
    ".ane:NET3": {
      "max-reservable-bandwidth": 50000000000
    }
  }
}
```

8.5. Incremental Updates

In this example, an ALTO client subscribes to the incremental update for the multipart Endpoint Cost Service resource "endpoint-cost-pv".

```
POST /updates/pv HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamparams+json
Content-Length: 112
```

```
{
  "add": {
    "ecspvsub1": {
      "resource-id": "endpoint-cost-pv",
      "input": <ecs-input>
    }
  }
}
```

Based on the server-side process defined in [RFC8895], the ALTO server will send the "control-uri" first using Server-Sent Event (SSE), followed by the full response of the multipart message.

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream
```

```
event: application/alto-updatestreamcontrol+json
data: {"control-uri": "https://alto.example.com/updates/streams/123"}
```

```
event: multipart/related;boundary=example-3;
      type=application/alto-endpointcost+json,ecspvsub1
data: --example-3
data: Content-ID: <ecsmapi@alto.example.com>
data: Content-Type: application/alto-endpointcost+json
data:
data: <endpoint-cost-map-entry>
data: --example-3
data: Content-ID: <propmap@alto.example.com>
data: Content-Type: application/alto-propmap+json
data:
data: <property-map-entry>
data: --example-3--
```

When the contents change, the ALTO server will publish the updates for each node in this tree separately, based on Section 6.7.3 of [RFC8895].

```
event: application/merge-patch+json, ecspvsubl.ecsmap@alto.example.com  
data: <Merge patch for endpoint-cost-map-update>
```

```
event: application/merge-patch+json, ecspvsubl.propmap@alto.example.com  
data: <Merge patch for property-map-update>
```

8.6. Multi-cost

The following examples demonstrate the request to the "multicost-pv" resource and the corresponding response.

The request asks for two cost types: the first is the Path Vector cost type, and the second is a numerical routing cost. It also queries the Maximum Reservable Bandwidth ANE property and the Persistent Entity property for two IPv4 source and destination pairs (192.0.2.34 -> 192.0.2.2 and 192.0.2.34 -> 192.0.2.50) and one IPv6 source and destination pair (2001:db8::3:1 -> 2001:db8::4:1).

The response consists of two parts. The first part returns a JSONArray that contains two JSONValue for each requested source and destination pair: the first JSONValue is a JSONArray of ANENames, which is the value of the Path Vector cost type, and the second JSONValue is a JSONNumber which is the value of the routing cost. The second part contains a Property Map that maps the ANEs to their requested properties.

```
POST /endpointcost/mcpv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;
       type=application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: 433
Content-Type: application/alto-endpointcostparams+json

{
  "multi-cost-types": [
    { "cost-mode": "array", "cost-metric": "ane-path" },
    { "cost-mode": "numerical", "cost-metric": "routingcost" }
  ],
  "endpoints": {
    "srcs": [
      "ipv4:192.0.2.34",
      "ipv6:2001:db8::3:1"
    ],
    "dsts": [
      "ipv4:192.0.2.2",
      "ipv4:192.0.2.50",
      "ipv6:2001:db8::4:1"
    ]
  },
  "ane-property-names": [
    "max-reservable-bandwidth",
    "persistent-entity-id"
  ]
}

HTTP/1.1 200 OK
Content-Length: 1350
Content-Type: multipart/related; boundary=example-4;
             type=application/alto-endpointcost+json

--example-4
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "vtags": {
      "resource-id": "endpoint-cost-pv.ecs",
      "tag": "84a4f9c14f9341f0983e3e5f43a371c8"
    },
    "multi-cost-types": [
      { "cost-mode": "array", "cost-metric": "ane-path" },
      { "cost-mode": "numerical", "cost-metric": "routingcost" }
    ]
  }
}
```

```

    ]
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.34": {
      "ipv4:192.0.2.2": [[ "NET3", "AGGR1" ], 3],
      "ipv4:192.0.2.50": [[ "NET3", "AGGR2" ], 2]
    },
    "ipv6:2001:db8::3:1": {
      "ipv6:2001:db8::4:1": [[ "NET3", "AGGR2" ], 2]
    }
  }
}
--example-4
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "84a4f9c14f9341f0983e3e5f43a371c8"
      },
      {
        "resource-id": "ane-props",
        "tag": "be157afa031443a187b60bb80a86b233"
      }
    ]
  },
  "property-map": {
    ".ane:AGGR1": {
      "max-reservable-bandwidth": 10000000000,
      "persistent-entity-id": "ane-props.ane:MEC1"
    },
    ".ane:AGGR2": {
      "max-reservable-bandwidth": 15000000000,
      "persistent-entity-id": "ane-props.ane:MEC2"
    },
    ".ane:NET3": {
      "max-reservable-bandwidth": 50000000000
    }
  }
}

```

9. Compatibility with Other ALTO Extensions

9.1. Compatibility with Legacy ALTO Clients/Servers

The multipart Filtered Cost Map resource and the multipart Endpoint Cost Service resource has no backward compatibility issue with legacy ALTO clients and servers. Although these two types of resources reuse the media types defined in the base ALTO protocol for the accept input parameters, they have different media types for responses. If the ALTO server provides these two types of resources, but the ALTO client does not support them, the ALTO client will ignore the resources without incurring any incompatibility problem.

9.2. Compatibility with Multi-Cost Extension

The extension defined in this document is compatible with the multi-cost extension [RFC8189]. Such a resource has a media type of either "multipart/related; type=application/alto-costmap+json" or "multipart/related; type=application/alto-endpointcost+json". Its "cost-constraints" field must either be "false" or not present and the Path Vector cost type must be present in the "cost-type-names" capability field but must not be present in the "testable-cost-type-names" field, as specified in Section 7.2.4 and Section 7.3.4.

9.3. Compatibility with Incremental Update

This extension is compatible with the incremental update extension [RFC8895]. ALTO clients and servers MUST follow the specifications given in Sections 5.2 and 6.7.3 of [RFC8895] to support incremental updates for a Path Vector resource.

9.4. Compatibility with Cost Calendar

The extension specified in this document is compatible with the Cost Calendar extension [RFC8896]. When used together with the Cost Calendar extension, the cost value between a source and a destination is an array of Path Vectors, where the k-th Path Vector refers to the abstract network paths traversed in the k-th time interval by traffic from the source to the destination.

When used with time-varying properties, e.g., maximum reservable bandwidth, a property of a single ANE may also have different values in different time intervals. In this case, if such an ANE has different property values in two time intervals, it MUST be treated as two different ANEs, i.e., with different entity identifiers. However, if it has the same property values in two time intervals, it MAY use the same identifier.

This rule allows the Path Vector extension to represent both changes of ANEs and changes of the ANEs' properties in a uniform way. The Path Vector part is calendared in a compatible way, and the Property Map part is not affected by the calendar extension.

The two extensions combined together can provide the historical network correlation information for a set of source and destination pairs. A network broker or client may use this information to derive other resource requirements such as Time-Block-Maximum Bandwidth, Bandwidth-Sliding-Window, and Time-Bandwidth-Product (TBP) (See [SENSE] for details).

10. General Discussions

10.1. Constraint Tests for General Cost Types

The constraint test is a simple approach to query the data. It allows users to filter the query result by specifying some boolean tests. This approach is already used in the ALTO protocol. [RFC7285] and [RFC8189] allow ALTO clients to specify the "constraints" and "or-constraints" tests to better filter the result.

However, the current syntax can only be used to test scalar cost types, and cannot easily express constraints on complex cost types, e.g., the Path Vector cost type defined in this document.

In practice, developing a bespoke language for general-purpose boolean tests can be a complex undertaking, and it is conceivable that there are some existing implementations already (the authors have not done an exhaustive search to determine whether there are such implementations). One avenue to develop such a language may be to explore extending current query languages like XQuery [XQuery] or JSONiq [JSONiq] and integrating these with ALTO.

Filtering the Path Vector results or developing a more sophisticated filtering mechanism is beyond the scope of this document.

10.2. General Multi-Resource Query

Querying multiple ALTO information resources continuously is a general requirement. Enabling such a capability, however, must address general issues like efficiency and consistency. The incremental update extension [RFC8895] supports submitting multiple queries in a single request, and allows flexible control over the queries. However, it does not cover the case introduced in this document where multiple resources are needed for a single request.

This extension gives an example of using a multipart message to encode the responses from two specific ALTO information resources: a Filtered Cost Map or an Endpoint Cost Service, and a Property Map. By packing multiple resources in a single response, the implication is that servers may proactively push related information resources to clients.

Thus, it is worth looking into the direction of extending the SSE mechanism as used in the incremental update extension [RFC8895], or upgrading to HTTP/2 [I-D.ietf-httpbis-http2bis] and HTTP/3 [I-D.ietf-quic-http], which provides the ability to multiplex queries and to allow servers proactively send related information resources.

Defining a general multi-resource query mechanism is out of the scope of this document.

11. Security Considerations

This document is an extension of the base ALTO protocol, so the Security Considerations [RFC7285] of the base ALTO protocol fully apply when this extension is provided by an ALTO server.

The Path Vector extension requires additional scrutiny on three security considerations discussed in the base protocol: confidentiality of ALTO information (Section 15.3 of [RFC7285]), potential undesirable guidance from authenticated ALTO information (Section 15.2 of [RFC7285]), and availability of ALTO service (Section 15.5 of [RFC7285]).

For confidentiality of ALTO information, a network operator should be aware that this extension may introduce a new risk: the Path Vector information, when used together with sensitive ANE properties such as capacities of bottleneck links, may make network attacks easier. For example, as the Path Vector information may reveal more fine-grained internal network structures than the base protocol, an attacker may identify the bottleneck link and start a distributed denial-of-service (DDoS) attack involving minimal flows to conduct the in-network congestion. Given the potential risk of leaking sensitive information, the Path Vector extension is mainly applicable in scenarios where 1) the ANE structures and ANE properties do not impose security risks to the ALTO service provider, e.g., not carrying sensitive information, or 2) the ALTO server and client have established a reliable trust relationship, for example, operated in the same administrative domain, or managed by business partners with legal contracts.

Three risk types are identified in Section 15.3.1 of [RFC7285]: (1) Excess disclosure of the ALTO service provider's data to an unauthorized ALTO client; (2) Disclosure of the ALTO service provider's data (e.g., network topology information or endpoint addresses) to an unauthorized third party; and (3) Excess retrieval of the ALTO service provider's data by collaborating ALTO clients. To mitigate these risks, an ALTO server MUST follow the guidelines in Section 15.3.2 of [RFC7285]. Furthermore, an ALTO server MUST follow the following additional protections strategies for risk types (1) and (3).

For risk type (1), an ALTO server MUST use the authentication methods specified in Section 15.3.2 of [RFC7285] to authenticate the identify of an ALTO client, and apply access control techniques to restrict unprivileged ALTO clients from retrieving sensitive Path Vector information. For settings where the ALTO server and client are not in the same trust domain, the ALTO server should reach agreements with the ALTO client on protecting the confidentiality before granting the access to Path Vector service with sensitive information. Such agreements may include legal contracts or Digital Right Management (DRM) techniques. Otherwise, the ALTO server MUST NOT offer the Path Vector service carrying sensitive information to the clients unless the potential risks are fully assessed and mitigated.

For risk type (3), an ALTO service provider must be aware that persistent ANEs may be used as "landmarks" in collaborative inferences. Thus, they should only be used when exposing public service access points (e.g., API gateways, CDNi) and/or when the granularity is coarse-grained (e.g., when an ANE represents an AS, a data center or a WAN). Otherwise, an ALTO server MUST use dynamic mappings from ephemeral ANE names to underlying physical entities. Specifically, for the same physical entity, an ALTO server SHOULD assign a different ephemeral ANE name when the entity appears in the responses to different clients or even for different request from the same client. A RECOMMENDED assignment strategy is to generate ANE names from random numbers.

Further, to protect the network topology from graph reconstruction (e.g., through isomorphic graph identification [BONDY]), the ALTO server SHOULD consider protection mechanisms to reduce information exposure or obfuscate the real information. When doing so, the ALTO server must be aware that information reduction/obfuscation may lead to potential Undesirable Guidance from Authenticated ALTO Information risk (Section 15.2 of [RFC7285]).

Thus, implementations of ALTO servers involving reduction or obfuscation of the Path Vector information SHOULD consider reduction/obfuscation mechanisms that can preserve the integrity of ALTO information, for example, by using minimal feasible region compression algorithms [NOVA] or obfuscation protocols [RESA][MERCATOR]. However, these obfuscation methods are experimental and their practical applicability of these methods to the generic capability provided by this extension is not fully assessed. The ALTO server MUST carefully verify that the deployment scenario satisfies the security assumptions of these methods before applying them to protect Path Vector services with sensitive network information.

For availability of ALTO service, an ALTO server should be cognizant that using Path Vector extension might have a new risk: frequent requesting for Path Vectors might consume intolerable amounts of the server-side computation and storage, which can break the ALTO server. For example, if an ALTO server implementation dynamically computes the Path Vectors for each request, the service providing Path Vectors may become an entry point for denial-of-service attacks on the availability of an ALTO server.

To mitigate this risk, an ALTO server may consider using optimizations such as precomputation-and-projection mechanisms [MERCATOR] to reduce the overhead for processing each query. Also, an ALTO server may also protect itself from malicious clients by monitoring the behaviors of clients and stopping serving clients with suspicious behaviors (e.g., sending requests at a high frequency).

The ALTO service providers must be aware that providing incremental updates of the "max-reservable-bandwidth" may provide information about other consumers of the network. For example, a change of the value may indicate one or more reservations has been made or changed. To mitigate this risk, an ALTO server can batch the updates and/or add a random delay before publishing the updates.

12. IANA Considerations

12.1. ALTO Cost Metric Registry

This document registers a new entry to the ALTO Cost Metric Registry, as instructed by Section 14.2 of [RFC7285]. The new entry is as shown below in Table 1.

Identifier	Intended Semantics	Security Considerations
ane-path	See Section 6.5.1	See Section 11

Table 1: ALTO Cost Metric Registry

12.2. ALTO Cost Mode Registry

This document registers a new entry to the ALTO Cost Mode Registry, as instructed by Section 4 of [I-D.bw-alto-cost-mode]. The new entry is as shown below in Table 2.

Identifier	Intended Semantics
array	See Section 6.5.2

Table 2: ALTO Cost Mode Registry

12.3. ALTO Entity Domain Type Registry

This document registers a new entry to the ALTO Domain Entity Type Registry, as instructed by Section 12.2 of [I-D.ietf-alto-unified-props-new]. The new entry is as shown below in Table 3.

Identifier	Entity Identifier Encoding	Hierarchy & Inheritance	Media Type of Defining Resource	Mapping to ALTO Address Type
ane	See Section 6.2.2	None	application/alto-propmap+json	false

Table 3: ALTO Entity Domain Type Registry

Identifier: See Section 6.2.1.

Entity Identifier Encoding: See Section 6.2.2.

Hierarchy: None

Inheritance: None

Media Type of Defining Resource: See Section 6.2.4.

Mapping to ALTO Address Type: This entity type does not map to ALTO address type.

Security Considerations: In some usage scenarios, ANE 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 ANEs will be made aware of how (or if) the addressing scheme relates to private information and network proximity, in further iterations of this document.

12.4. ALTO Entity Property Type Registry

Two initial entries "max-reservable-bandwidth" and "persistent-entity-id" are registered to the ALTO Domain "ane" in the "ALTO Entity Property Type Registry", as instructed by Section 12.3 of [I-D.ietf-alto-unified-props-new]. The two new entries are shown below in Table 4 and their details can be found in Section 12.4.1 and Section 12.4.2.

Identifier	Intended Semantics	Media Type of Defining Resource
max-reservable-bandwidth	See Section 6.4.1	application/alto-propmap+json
persistent-entity-id	See Section 6.4.2	application/alto-propmap+json

Table 4: Initial Entries for ane Domain in the ALTO Entity Property Types Registry

12.4.1. New ANE Property Type: Maximum Reservable Bandwidth

Identifier: "max-reservable-bandwidth"

Intended Semantics: See Section 6.4.1.

Media Type of Defining Resource: application/alto-propmap+json

Security Considerations: This property is essential for applications such as large-scale data transfers or overlay network interconnection to make better choice of bandwidth reservation. It may reveal the bandwidth usage of the underlying network and can potentially be leveraged to reduce the cost of conducting

denial-of-service attacks. Thus, the ALTO server MUST consider protection mechanisms including only providing the information to authorized clients, and information reduction and obfuscation as introduced in Section 11.

12.4.2. New ANE Property Type: Persistent Entity ID

Identifier: "persistent-entity-id"

Intended Semantics: See Section 6.4.2.

Media Type of Defining Resource: application/alto-propmap+json

Security Considerations: This property is useful when an ALTO server wants to selectively expose certain service points whose detailed properties can be further queried by applications. The entity IDs may consider sensitive information about the underlying network, and an ALTO server should follow the security considerations in Section 11 of [I-D.ietf-alto-unified-props-new].

13. References

13.1. Normative References

[I-D.bw-alto-cost-mode]

Boucadair, M. and Q. Wu, "A Cost Mode Registry for the Application-Layer Traffic Optimization (ALTO) Protocol", Work in Progress, Internet-Draft, draft-bw-alto-cost-mode-01, 4 March 2022, <<https://datatracker.ietf.org/doc/html/draft-bw-alto-cost-mode-01>>.

[I-D.ietf-alto-unified-props-new]

Roome, W., Randriamasy, S., Yang, Y. R., Zhang, J. J., and K. Gao, "An ALTO Extension: Entity Property Maps", Work in Progress, Internet-Draft, draft-ietf-alto-unified-props-new-24, 28 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-alto-unified-props-new-24>>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <<https://www.rfc-editor.org/rfc/rfc2046>>.

[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/rfc/rfc2119>>.

- [RFC2387] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387, DOI 10.17487/RFC2387, August 1998, <<https://www.rfc-editor.org/rfc/rfc2387>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/rfc/rfc5322>>.
- [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/rfc/rfc7285>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/rfc/rfc8189>>.
- [RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <<https://www.rfc-editor.org/rfc/rfc8895>>.
- [RFC8896] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", RFC 8896, DOI 10.17487/RFC8896, November 2020, <<https://www.rfc-editor.org/rfc/rfc8896>>.

13.2. Informative References

- [BONDY] Bondy, J.A. and R.L. Hemminger, "Graph reconstructiona survey", Journal of Graph Theory, Volume 1, Issue 3, pp 227-268 , 1977, <<https://doi.org/10.1002/jgt.3190010306>>.
- [BOXOPT] Xiang, Q., Yu, H., Aspnes, J., Le, F., Kong, L., and Y.R. Yang, "Optimizing in the dark: Learning an optimal solution through a simple request interface", Proceedings of the AAAI Conference on Artificial Intelligence 33, 1674-1681 , 2019, <<https://doi.org/10.1609/aaai.v33i01.33011674>>.

- [CLARINET] Viswanathan, R., Ananthanarayanan, G., and A. Akella, "CLARINET: WAN-Aware Optimization for Analytics Queries", In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), USENIX Association, Savannah, GA, 435-450 , 2016, <<https://dl.acm.org/doi/abs/10.5555/3026877.3026911>>.
- [G2] Ros-Giralt, J., Bohara, A., Yellamraju, S., Langston, M.H., Lethin, R., Jiang, Y., Tassiulas, L., Li, J., Tan, Y., and M. Veeraraghavan, "On the Bottleneck Structure of Congestion-Controlled Networks", Proceedings of the ACM on Measurement and Analysis of Computing Systems, Volume 3, Issue 3, pp 1-31 , 2019, <<https://dl.acm.org/doi/10.1145/3366707>>.
- [HUG] Chowdhury, M., Liu, Z., Ghodsi, A., and I. Stoica, "HUG: Multi-Resource Fairness for Correlated and Elastic Demands", 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16) (Santa Clara, CA, 2016), 407-424. , 2016, <<https://dl.acm.org/doi/10.5555/2930611.2930638>>.
- [I-D.ietf-alto-performance-metrics]
Wu, Q., Yang, Y. R., Lee, Y., Dhody, D., Randriamasy, S., and L. M. C. Murillo, "ALTO Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-26, 2 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-alto-performance-metrics-26>>.
- [I-D.ietf-httpbis-http2bis]
Thomson, M. and C. Benfield, "HTTP/2", Work in Progress, Internet-Draft, draft-ietf-httpbis-http2bis-07, 24 January 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-http2bis-07>>.
- [I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>>.
- [JSONiq] "The JSON Query language", 2020, <<https://www.jsoniq.org/>>.

- [MERCATOR] Xiang, Q., Zhang, J., Wang, X., Liu, Y., Guok, C., Le, F., MacAuley, J., Newman, H., and Y.R. Yang, "Toward Fine-Grained, Privacy-Preserving, Efficient Multi-Domain Network Resource Discovery", IEEE/ACM IEEE Journal on Selected Areas of Communication 37(8): 1924-1940, 2019, <<https://doi.org/10.1109/JSAC.2019.2927073>>.
- [MOWIE] Zhang, Y., Li, G., Xiong, C., Lei, Y., Huang, W., Han, Y., Walid, A., Yang, Y.R., and Z. Zhang, "MoWIE: Toward Systematic, Adaptive Network Information Exposure as an Enabling Technique for Cloud-Based Applications over 5G and Beyond", In Proceedings of the Workshop on Network Application Integration/CoDesign, ACM, Virtual Event USA, 20-27. , 2020, <<https://doi.org/10.1145/3405672.3409489>>.
- [NOVA] Gao, K., Xiang, Q., Wang, X., Yang, Y.R., and J. Bi, "An objective-driven on-demand network abstraction for adaptive applications", IEEE/ACM Transactions on Networking (TON) Vol 27, no. 2 (2019): 805-818., 2019, <<https://doi.org/10.1109/IWQoS.2017.7969117>>.
- [RESA] Xiang, Q., Zhang, J., Wang, X., Liu, Y., Guok, C., Le, F., MacAuley, J., Newman, H., and Y.R. Yang, "Fine-grained, multi-domain network resource abstraction as a fundamental primitive to enable high-performance, collaborative data sciences", Proceedings of the Super Computing 2018, 5:1-5:13 , 2019, <<https://doi.org/10.1109/SC.2018.000008>>.
- [RFC2216] Shenker, S. and J. Wroclawski, "Network Element Service Specification Template", RFC 2216, DOI 10.17487/RFC2216, September 1997, <<https://www.rfc-editor.org/rfc/rfc2216>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/rfc/rfc4271>>.
- [SENSE] "Software Defined Networking (SDN) for End-to-End Networked Science at the Exascale", 2019, <<https://www.es.net/network-r-and-d/sense/>>.
- [SEREDGE] Contreras, L., Baliosian, J., Martnez-Julia, P., and J. Serrat, "Computing at the Edge: But, what Edge?", In proceedings of the NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium. pp. 1-9. , 2020, <<https://doi.org/10.1109/NOMS47738.2020.9110342>>.

- [SWAN] Hong, C., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., and R. Wattenhofer, "Achieving High Utilization with Software-driven WAN", In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13), ACM, New York, NY, USA, 15-26. , 2013, <<http://doi.acm.org/10.1145/2486001.2486012>>.
- [UNICORN] Xiang, Q., Chen, S., Gao, K., Newman, H., Taylor, I., Zhang, J., and Y.R. Yang, "Unicorn: Unified Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics", 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI) (Aug. 2017), 1-6. , 2017, <<https://doi.org/10.1016/j.future.2018.09.048>>.
- [XQuery] "XQuery 3.1: An XML Query Language", 2017, <<https://www.w3.org/TR/xquery-31/>>.

Appendix A. Acknowledgments

The authors would like to thank discussions with Andreas Voellmy, Erran Li, Haibin Song, Haizhou Du, Jiayuan Hu, Qiao Xiang, Tianyuan Liu, Xiao Shi, Xin Wang, and Yan Luo. The authors thank Greg Bernstein, Dawn Chen, Wendy Roome, and Michael Scharf for their contributions to earlier drafts.

The authors would also like to thank Tim Chown, Luis Contreras, Roman Danyliw, Benjamin Kaduk, Erik Kline, Suresh Krishnan, Murray Kucherawy, Warren Kumari, Danny Lachos, Francesca Palombini, Eric Vyncke, Samuel Weiler, and Qiao Xiang whose feedback and suggestions are invaluable to improve the practicability and conciseness of this document, and Mohamed Boucadair, Martin Duke, Vijay Gurbani, Jan Seedorf, and Qin Wu who provide great support and guidance.

Appendix B. Revision Logs (To be removed before publication)

B.1. Changes since -20

Reivision -21

- * changes the normative requirement on protecting confidentiality of PV information with softer language

B.2. Changes since -19

Revision -20

- * changes the IANA registry information
- * adopts the comments from IESG reviews

B.3. Changes since -18

Revision -19

- * adds detailed examples for use cases
- * clarify terms with ambiguous meanings

B.4. Changes since -17

Revision -18

- * changes the specification for content-id to conform to [RFC2387] and [RFC5322]
- * adds IPv6 examples

B.5. Changes since -16

Revision -17

- * adds items for media type of defining resources in IANA considerations

B.6. Changes since -15

Revision -16

- * resolves the compatibility with the Multi-Cost extension (RFC 8189)
- * adds media types of defining resources for ANE property types (for IANA registration)

B.7. Changes since -14

Revision -15

- * fixes the IDNits warnings,

- * fixes grammar issues,
- * addresses the comments in the AD review.

B.8. Changes since -13

Revision -14

- * addresses the comments in the chair review,
- * fixes most issues raised by IDNits.

B.9. Changes since -12

Revision -13

- * changes the abstract based on the chairs' reviews
- * integrates Richard's responds to WGLC reviews

B.10. Changes since -11

Revision -12

- * clarifies the definition of ANEs in a similar way as how Network Elements is defined in [RFC2216]
- * restructures several paragraphs that are not clear (Sec 3, Path Vector bullet, Sec 4.2, Sec 5.1.3, Sec 6.2.4, Sec 6.4.2, Sec 9.3)
- * uses "ALTO Entity Domain Type Registry"

B.11. Changes since -10

Revision -11

- * replaces "part" with "components" in the abstract;
- * identifies additional requirements (AR) derived from the flow scheduling example, and introduces how the extension addresses the additional requirements
- * fixes the inconsistent use of "start" parameter in multipart responses;
- * specifies explicitly how to handle "cost-constraints";

- * uses the latest IANA registration mechanism defined in [I-D.ietf-alto-unified-props-new];
- * renames "persistent-entities" to "persistent-entity-id";
- * makes "application/alto-propmap+json" as the media type of defining resources for the "ane" domain;
- * updates the examples;
- * adds the discussion on ephemeral and persistent ANEs.

B.12. Changes since -09

Revision -10

- * revises the introduction which
 - extends the scope where the PV extension can be applied beyond the "path correlation" information
- * brings back the capacity region use case to better illustrate the problem
- * revises the overview to explain and defend the concepts and decision choices
- * fixes inconsistent terms, typos

B.13. Changes since -08

This revision

- * fixes a few spelling errors
- * emphasizes that abstract network elements can be generated on demand in both introduction and motivating use cases

B.14. Changes Since Version -06

- * We emphasize the importance of the path vector extension in two aspects:
 1. It expands the problem space that can be solved by ALTO, from preferences of network paths to correlations of network paths.
 2. It is motivated by new usage scenarios from both application's and network's perspectives.

- * More use cases are included, in addition to the original capacity region use case.
- * We add more discussions to fully explore the design space of the path vector extension and justify our design decisions, including the concept of abstract network element, cost type (reverted to -05), newer capabilities and the multipart message.
- * Fix the incremental update process to be compatible with SSE -16 draft, which uses client-id instead of resource-id to demultiplex updates.
- * Register an additional ANE property (i.e., persistent-entities) to cover all use cases mentioned in the draft.

Authors' Addresses

Kai Gao
Sichuan University
No.24 South Section 1, Yihuan Road
Chengdu
610000
China
Email: kaigao@scu.edu.cn

Young Lee
Samsung
South Korea
Email: younglee.tx@gmail.com

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
91460 Nozay
France
Email: sabine.randriamasy@nokia-bell-labs.com

Yang Richard Yang
Yale University
51 Prospect Street
New Haven, CT
United States of America
Email: yry@cs.yale.edu

Jingxuan Jensen Zhang
Tongji University
4800 Caoan Road
Shanghai
201804
China
Email: jingxuan.n.zhang@gmail.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: 1 September 2022

W. Roome
S. Randriamasy
Nokia Bell Labs
Y. Yang
Yale University
J. Zhang
Tongji University
K. Gao
Sichuan University
28 February 2022

An ALTO Extension: Entity Property Maps
draft-ietf-alto-unified-props-new-24

Abstract

This document specifies an extension to the base Application-Layer Traffic Optimization (ALTO) protocol that generalizes the concept of "endpoint properties", which were so far tied to IP addresses, to entities defined by a wide set of objects. Further, these properties are presented as maps, similar to the network and cost maps in the base ALTO protocol. While supporting the endpoints and related endpoint property service defined in RFC7285, the ALTO protocol is extended in two major directions. First, from endpoints restricted to IP addresses to entities covering a wider and extensible set of objects; second, from properties on specific endpoints to entire entity property maps. These extensions introduce additional features allowing entities and property values to be specific to a given information resource. This is made possible by a generic and flexible design of entity and property types.

Status of This Memo

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

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

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

This Internet-Draft will expire on 1 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
1.1. Terminology and notation	6
2. Requirements Language	6
3. Basic Features of the Entity Property Map Extension	7
3.1. Entity	7
3.2. Entity Domain	8
3.2.1. Entity Domain Type	8
3.2.2. Entity Domain Name	8
3.3. Entity Property Type	9
3.4. New Information Resource and Media Type: ALTO Property Map	10
4. Advanced Features of the Entity Property Map Extension	10
4.1. Entity Identifier and Entity Domain Name	10
4.2. Resource-Specific Entity Domain Name	11
4.3. Resource-Specific Entity Property Value	12
4.4. Entity Hierarchy and Property Inheritance	13
4.4.1. Entity Hierarchy	13
4.4.2. Property Inheritance	14
4.4.3. Property Value Unicity	14
4.5. Supported Properties on Entity Domains in Property Map Capabilities	15
4.6. Defining Information Resource for Resource-Specific Entity Domains	15
4.6.1. Defining Information Resource and its Media Type	16
4.6.2. Examples of Defining Information Resources and Their Media Type	17
4.7. Defining Information Resource for Resource-Specific Property Values	18
5. Protocol Specification: Basic Data Types	19
5.1. Entity Domain	19
5.1.1. Entity Domain Type	19
5.1.2. Entity Domain Name	20

5.1.3.	Entity Identifier	22
5.1.4.	Hierarchy and Inheritance	23
5.2.	Entity Property	23
5.2.1.	Entity Property Type	23
5.2.2.	Entity Property Name	24
5.2.3.	Format for Entity Property Value	25
6.	Entity Domain Types Defined in this Document	25
6.1.	Internet Address Domain Types	25
6.1.1.	Entity Domain Type: IPv4	25
6.1.2.	Entity Domain Type: IPv6	26
6.1.3.	Hierarchy and Inheritance of Internet Address Domains	26
6.1.4.	Defining Information Resource Media Type for domain types IPv4 and IPv6	28
6.2.	Entity Domain Type: PID	28
6.2.1.	Entity Domain Type Identifier	28
6.2.2.	Domain-Specific Entity Identifiers	28
6.2.3.	Hierarchy and Inheritance	28
6.2.4.	Defining Information Resource Media Type for Domain Type PID	28
6.2.5.	Relationship To Internet Addresses Domains	29
6.3.	Internet Address Properties vs. PID Properties	29
7.	Property Map	29
7.1.	Media Type	30
7.2.	HTTP Method	30
7.3.	Accept Input Parameters	30
7.4.	Capabilities	30
7.5.	Uses	30
7.6.	Response	30
8.	Filtered Property Map	32
8.1.	Media Type	32
8.2.	HTTP Method	32
8.3.	Accept Input Parameters	32
8.4.	Capabilities	34
8.5.	Uses	34
8.6.	Filtered Property Map Response	34
8.7.	Entity Property Type Defined in This Document	36
8.7.1.	Entity Property Type: pid	36
9.	Impact on Legacy ALTO Servers and ALTO Clients	37
9.1.	Impact on Endpoint Property Service	37
9.2.	Impact on Resource-Specific Properties	37
9.3.	Impact on Other Properties	37
10.	Examples	38
10.1.	Network Map	38
10.2.	Property Definitions	38
10.3.	Information Resource Directory (IRD)	39
10.4.	Full Property Map Example	42
10.5.	Filtered Property Map Example #1	43

10.6.	Filtered Property Map Example #2	44
10.7.	Filtered Property Map Example #3	45
10.8.	Filtered Property Map Example #4	47
10.9.	Filtered Property Map for ANEs Example #5	47
11.	Security Considerations	48
12.	IANA Considerations	50
12.1.	application/alto-propmap+json Media Type	51
12.2.	alto-propmapparams+json Media Type	52
12.3.	ALTO Entity Domain Type Registry	53
12.3.1.	Consistency Procedure between ALTO Address Type Registry and ALTO Entity Domain Type Registry	54
12.3.2.	ALTO Entity Domain Type Registration Process	56
12.4.	ALTO Entity Property Type Registry	57
13.	Acknowledgments	58
14.	References	59
14.1.	Normative References	59
14.2.	Informative References	60
Appendix A.	Features introduced with the Entity Property Maps extension	61
Authors' Addresses	62

1. Introduction

The ALTO protocol [RFC7285] introduces the concept of "properties" attached to "endpoint addresses". It also 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 that are further elaborated hereafter.

First, the EPS allows properties to be associated with only endpoints that are identified by individual communication addresses like IPv4 and IPv6 addresses. It is reasonable to think that collections of endpoints or Provider-Defined Identifiers (PIDs), may also have properties. Furthermore, recent ALTO use cases show that properties of entities such as abstracted network elements as defined in [I-D.ietf-alto-path-vector] are also useful. However, the current EPS is restricted to individual endpoints and cannot be applied to those entities.

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

Third, in section 11.4 of [RFC7285], the EPS is only defined as a POST-mode service. ALTO clients must request the properties for an explicit set of endpoint addresses. By contrast, [RFC7285], in section 11.2.3, defines a GET-mode cost map resource which returns all available costs, so an ALTO Client can retrieve a full set of costs once, and then process cost lookups without querying the ALTO server. [RFC7285] does not define a similar service for endpoint properties. At first, a map of endpoint properties might seem impractical, because it could require enumerating the property value for every possible endpoint. In particular, the number of endpoint addresses involved by an ALTO server can be quite large. To avoid enumerating a large number of endpoint addresses inefficiently, the ALTO server might define properties for a sufficiently large subset of endpoints and uses an aggregation representation to reference endpoints to allow efficient enumeration. This is particularly true if blocks of endpoint addresses with a common prefix have the same value for a property. Entities in other domains may very well allow aggregated representation and hence be enumerable as well.

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

- * The first limitation is addressed by introducing a generic concept called ALTO Entity, which generalizes an endpoint and may represent a PID, a network element, a cell in a cellular network, an abstracted network element [I-D.ietf-alto-path-vector], or other physical or logical objects involved in a network topology. Each entity is included in a collection called an ALTO entity domain. Since each ALTO entity domain includes only one type of entities, each entity domain can be classified by the type of enclosed entities.
- * The second limitation is addressed by using resource-specific entity domains. A resource-specific entity domain contains entities that are defined and identified with respect to a given ALTO information resource, which provides scoping. For example, an entity domain containing PIDs is identified with respect to the network map in which these PIDs are defined. Likewise, an entity domain containing local IP addresses may be defined with respect to a local network map.
- * The third limitation is addressed by defining two new types of ALTO information resources: Property Map (Section 7) and Filtered Property Map (Section 8). The former is a resource that is requested using the HTTP GET method, returns the property values for all entities in one or more entity domains, and is analogous to a network map or a cost map in Section 11.2 of [RFC7285]. The latter is a resource that that is requested using the HTTP POST

method, returns the values for sets of properties and entities requested by the client, and is analogous to a filtered network map or a filtered cost map.

The Entity Property Maps extension described in this document introduces a number of features that are summarized in Appendix A, where Table 4 lists the features and references the sections in this document that give their high-level and their normative description.

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

1.1. Terminology and notation

This document uses the following terms and abbreviations, that will be further defined in the document. While this document introduces the feature "entity property map", it will use both the term "property map" and "entity property map" to refer to this feature.

- * Transaction: A request/response exchange between an ALTO client and an ALTO server.
- * Client: When used with a capital "C", this term refers to an ALTO client. Note that expressions "ALTO client", "ALTO Client" and "Client" are equivalent.
- * Server: When used with a capital "S", this term refers to an ALTO server. Note that expressions "ALTO server", "ALTO Server" and "Server" are equivalent.
- * EPS: An abbreviation for Endpoint Property Service.

This document uses the semi-formal notation defined in Section 8.2 of [RFC7285].

2. Requirements Language

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

3. Basic Features of the Entity Property Map Extension

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

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

3.1. Entity

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

Examples of entities are:

- * an ALTO endpoint that represents an application or a host identified by a communication address (e.g., an IPv4 or IPv6 address) in a network,
- * a PID, defined in [RFC7285], that has a provider defined human-readable identifier specified by an ALTO network map, which maps a PID to a set of IPv4 and IPv6 addresses,
- * an Autonomous System (AS), that has an AS number (ASN) as its identifier and maps to a set of IPv4 and IPv6 addresses, that is defined in [I-D.ietf-alto-cdni-request-routing-alto],
- * a country with a code specified in [ISO3166-1], to which applications such as CDN providers associate properties and capabilities, that is defined in [I-D.ietf-alto-cdni-request-routing-alto],
- * a TCP/UDP network flow, that is identified by a TCP/UDP 5-tuple specifying its source and destination addresses and port numbers, and the IP protocol,

- * a routing element, that is specified in [RFC7921] and is associated with routing capabilities information,
- * an abstract network element, that is specified in [I-D.ietf-alto-path-vector] and that represents an abstraction of a network part such as a router, one or more links, a network domain or their aggregation.

Some of the example entities listed above have already been documented as ALTO entities. The other examples are provided for illustration as potential entities.

3.2. Entity Domain

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

In this document, an entity is owned by exactly one entity domain name. An entity identifier points to exactly one entity. If two entities in two different entity domains refer to the same physical or logical object, they are treated as different entities. For example, if an end host has both an IPv4 and an IPv6 address, these two addresses will be treated as two entities, defined respectively in the "ipv4" and "ipv6" entity domains.

3.2.1. Entity Domain Type

The type of an entity domain type defines the semantics of a type of entity. Entity domain types can be defined in different documents. For example: the present document defines entity domain types "ipv4", "ipv6" and "pid" in Section 6.1 and Section 6.2. The entity domain type "ane", that defines Abstract Network Elements (ANEs), is introduced in [I-D.ietf-alto-path-vector]. The entity domain type that defines country codes is introduced in [I-D.ietf-alto-cdni-request-routing-alto]. An entity domain type MUST be registered at the IANA, as specified in Section 12.3.2.

3.2.2. Entity Domain Name

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

"ipv4" because its entities are uniquely identified by all the Server resources.

In some cases, the name of an entity domain cannot be simply its entity domain type. Indeed, for some domain types, entities are defined relative to a given information resource. This is the case for entities of domain type "pid". A PID is defined relative to a network map. For example, an entity "mypid10" of domain type "pid" may be defined in a given network map and be undefined in other network maps. Or "mypid10" may even be defined in two different network maps and map, in each of these network maps, to a different set of endpoint addresses. In this case, naming an entity domain only by its type "pid" does not guarantee that its set of entities is owned by exactly one entity domain.

Section 4.2 and Section 5.1.2 describe how a domain is uniquely identified, across the ALTO server, by a name that associates the domain type and the related information resource.

3.3. Entity Property Type

An entity property defines a property of an entity. This is similar to the endpoint property defined in Section 7.1 of [RFC7285]. An entity property can convey either network-aware or network-agnostic information. Similar to an entity domain, an entity property is characterized by a type and identified by a name. An entity property type MUST be registered at the IANA, as specified in Section 12.4.

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

- * an entity in the "ipv4" domain type may have a property whose value is an Autonomous System (AS) number indicating the AS to which this IPv4 address belongs and another property named "countrycode" indicating a country code mapping to this address,
- * an entity identified by its country code in the entity domain type "countrycode", defined in [I-D.ietf-alto-cdni-request-routing-alto] may have a property indicating what delivery protocol is used by a CDN,
- * an entity in the "netmap1.pid" domain may have a property that indicates the central geographical location of the endpoints it includes.

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

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

Likewise, the same identifier may point to both a domain name and a property name. For example: the identifier "netmap10.pid" may point to either the domain defined by the PIDs of network map "netmap10" or to a property that returns, for an entity defined by its IPv4 address, the PID of netmap10 that contains this entity. Such cases will be further explained in Section 4.

3.4. New Information Resource and Media Type: ALTO Property Map

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

The present extension also introduces a new media type.

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

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

4. Advanced Features of the Entity Property Map Extension

This section gives a high-level overview of the advanced features involved in ALTO Entity Property Maps. Most of these features are defined to extend the ones defined in Section 3.

4.1. Entity Identifier and Entity Domain Name

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

In this document, example IPv4 and IPv6 addresses and prefixes are taken from the address ranges reserved for documentation by [RFC5737] and [RFC3849].

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

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

For instance:

- * if an entity is an endpoint with IPv4 address "192.0.2.14", its identifier is associated with entity domain name "ipv4" and is "ipv4:192.0.2.14",
- * if an entity is a PID named "mypid10" in network map resource "netmap2", its identifier is associated with entity domain name "netmap2.pid" and is "netmap2.pid:mypid10".

4.2. Resource-Specific Entity Domain Name

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

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

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

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

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

4.3. Resource-Specific Entity Property Value

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

For example, suppose entity "192.0.2.34" defined in the "ipv4" domain has a property of type "pid", whose value is the PID to which address "192.0.2.34" is attached in a network map. The mapping of network addresses to PIDs is specific to a network map and probably different from one network map resource to another one. Thus, if a property "pid" is defined for entity "192.0.2.34" in two different network maps "netmap1" and "netmap2", the value for this property can be a different value in "netmap1" and "netmap2".

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

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

4.4. Entity Hierarchy and Property Inheritance

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

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

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

4.4.1. Entity Hierarchy

An entity domain may allow using a single identifier to identify a set of related individual entities. For example, a CIDR block can be used to identify a set of IPv4 or IPv6 entities. A CIDR block is called a hierarchical entity identifier, as it can reflect inclusion relations among entity sets. That is, in an entity hierarchy, "supersets" are defined at upper levels and include "subsets" defined at lower levels." For example, the CIDR "ipv4:192.0.1.0/24" includes all the individual IPv4 entities identified by the CIDR "ipv4:192.0.1.0/26". This document will sometimes use the term "hierarchical address" to refer to a hierarchical entity identifier.

4.4.2. Property Inheritance

A property may be defined for a hierarchical entity identifier, while it may be undefined for individual entities covered by this identifier. In this case, these individual entities inherit the property value defined for the identifier that covers them. For example, suppose a property map defines a property P for which it assigns value V1 only for the hierarchical entity identifier "ipv4:192.0.1.0/24" but not for individual entities in this block. Suppose also that inheritance rules are specified for CIDR blocks in the "ipv4" domain type. When receiving this property map, a Client can infer that entity "ipv4:192.0.1.1" inherits the property value V1 of block "ipv4:192.0.1.0/24" because the address "ipv4:192.0.1.1" is included in the CIDR block "ipv4:192.0.1.0/24".

Property value inheritance rules also apply among entity sets. A property map may define values for an entity set belonging to a hierarchy but not for "subsets" that are covered by this set identifier. In this case, inheritance rules must specify how entities in "subsets" inherit property values from their "superset". For instance, suppose a property P is defined only for the entity set defined by address block "ipv4:192.0.1.0/24". We know that entity set "ipv4:192.0.1.0/30" is included in "ipv4:192.0.1.0/24". Therefore, the entities of "ipv4:192.0.1.0/30" may inherit the value of property P from set "ipv4:192.0.1.0/24", if an inheritance rule from "ipv4" CIDR blocks to included "ipv4" CIDR blocks, is specified.

4.4.3. Property Value Unicity

The inheritance rules must ensure that an entity belonging to a hierarchical set of entities inherits no more than one property value, for the sake of consistency. Indeed, a property map may define a property on a hierarchy of entity sets that inherit property values from one or more supersets (located at upper levels). On the other hand, a property value, defined on a subset (located at a lower level) may be different from the value defined on a superset. In such a case, subsets may potentially end up with different property values. This may be the case for address blocs with increasing prefix length, on which a property value gets increasingly accurate and thus may differ. For example, a fictitious property such as "geo-location" or "average transfer volume" may be defined at a progressively finer grain for lower level subsets of entities, defined with progressively longer CIDR prefixes. It seems more interesting to have property values of progressively higher accuracy. A unicity rule, applied to the entity domain type must specify an arbitration rule among the different property values for an entity. An example illustrating the need for such rules is provided in Section 6.1.3.

4.5. Supported Properties on Entity Domains in Property Map Capabilities

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

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

- * a field named "mappings" lists the names of the entity domains supported by the Property Map,
- * for each listed entity domain, a list of the names of the applicable properties is provided.

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

- * it prevents a Client from querying a property on entity domains on which it is not defined,
- * it allows a Client to query, for an entity E, values for a property P that are defined in several information resources,
- * it allows a Client to query a property P on entities that are defined in several information resources.

Further details are provided in Section 7.4.

4.6. Defining Information Resource for Resource-Specific Entity Domains

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

identifier of the specific resource, query the resource and retrieve the entities. For example:

- * an entity domain named "netmap1.ipv4" includes the IPv4 addresses that appear in the "ipv4" field of the endpoint address group of each PID in the network map "netmap1", and that have no meaning outside "netmap1" because, for instance, these are local addresses not reachable outside some private network,
- * an entity domain named "netmap1.pid" includes the PIDs listed in network map "netmap1".
- * an entity domain named "ipv4" is resource-agnostic and covers all the reachable IPv4 addresses.

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

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

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

4.6.1. Defining Information Resource and its Media Type

For the reasons explained in Section 4.6, this document introduces the concept of "Defining Information Resource and its Media Type".

A defining information resource for an entity domain D is the information resource where entities of D are defined. That is, all the information on the entities of D can be retrieved in this resource. A defining information resource is defined for resource-specific entity domains. It does not exist for entity domains that

are not resource-specific such as "ipv4" or "ipv6". Neither does it exist for entity domains that are covering entity identifiers already defined in other standardization documents, at it is the case for country code identifiers standardized in [ISO3166-1] or AS numbers allocated by the IANA. This is useful for entity domain types that are by essence domain-specific, such as the "pid" domain type. It is also useful for resource-specific entity domains constructed from resource-agnostic domain types, such as network map specific domains of local IPv4 addresses.

The defining information resource of a resource-specific entity domain D, when it exists, is unique and has the following specificities:

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

A fundamental characteristic of a defining information resource is its media type. There is a unique association between an entity domain type and the media type of its defining information resource. When an entity domain type allows associations with defining information resources, the media type of the potential defining information resource MUST be specified:

- * in the document that defines this entity domain type,
- * in the IANA ALTO Entity Domain Type Registry and related information.

When the Client wants to use a resource-specific entity domain, it needs to be cognizant of the media-type of its defining information resource. If the Server exposes a resource-specific entity domain with a non-compliant media type for the defining resource, the Client MUST ignore the entities from that entity domain to avoid errors.

4.6.2. Examples of Defining Information Resources and Their Media Type

Here are examples of defining information resource types and their media types associated to different entity domain types:

- * For entity domain type "pid": the media type of the specific resource is "application/alto-networkmap+json", because PIDs are defined in network map resources.
- * For entity domain types "ipv4" and "ipv6": the media type of the specific resource is "application/alto-networkmap+json", because IPv4 and IPv6 addresses covered by the Server are defined in network map resources.
- * For entities of domain type "ane": [I-D.ietf-alto-path-vector] defines entities named "ANE", where ANE stands for Abstracted Network Element, and the entity domain type "ane". An ANE may have a persistent identifier, say, "entity-4", that is provided by the Server as a value of the "persistent-entity-id" property of this ANE. Further properties may then be queried on an ANE by using its persistent entity ID. These properties are available from a persistent property map, that defines properties on a specific "ane" domain. Together with the persistent identifier, the Server also provides the property map resource identifier where the "ane" domain containing "entity-4" is defined. The definition of the "ane" entity domain containing "entity-4" is thus specific to the property map. Therefore, for entities of domain type "ane" that have a persistent identifier, the media type of the defining information resource is "application/alto-propmap+json".
- * Last, the entity domain types "asn" and "countrycode" defined in [I-D.ietf-alto-cdni-request-routing-alto] do not have a defining information resource. Indeed, the entity identifiers in these two entity domain types are already standardized in documents that the Client can use.

4.7. Defining Information Resource for Resource-Specific Property Values

As explained in Section 4.3, a property type may take values that are resource-specific. This is the case for property type "pid", whose values are by essence defined relative to a specific network map. That is, the PID value returned for an IPv4 address is specific to the network map defining this PID and may differ from one network map to another one.

Another example is provided in [I-D.ietf-alto-cdni-request-routing-alto] that defines property type "cdni-capabilities". The value of this property is specific to a CDNI advertisement resource, that provides a list of CDNI capabilities. The property is provided for entity domain types "ipv4", "ipv6", "asn" and "countrycode". A CDNI Advertisement resource does however not define PID values for IPv4 addresses while a network map does not define CDNI capabilities for IPv4 addresses.

Similar to resource-specific entity domains, the Client needs to be cognizant of appropriate associations of information resource and property types. Therefore, when specifying and registering a property type whose values are resource-specific, the media type of its defining information resource needs to be specified. For example:

- * The media type of the defining information resource for property type "pid" is "application/alto-networkmap+json".
- * The media type of the defining information resource for property type "cdni-capabilities" defined in [I-D.ietf-alto-cdni-request-routing-alto] is "application/alto-cdni+json".

5. Protocol Specification: Basic Data Types

5.1. Entity Domain

5.1.1. Entity Domain Type

An entity domain has a type, which is uniquely identified by a string that MUST be no more than 64 characters, and MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), the hyphen ('-', U+002D), the colon (':', U+003A), or the low line ('_', U+005F).

The usage of colon (':', U+003A) MUST obey the rules below:

- * The colon (':', U+003A) character MUST NOT appear more than once,
- * The colon character MUST NOT be used unless within the string "priv:",
- * The string "priv:" MUST NOT be used unless it starts the string that identifies an entity domain type,

- * For an entity domain type identifier with the "priv:" prefix , an additional string (e.g., company identifier or random string) MUST follow "priv:", to reduce potential collisions.

For example, the strings "ipv4", "ipv6", "pid" and "priv:example-test-edt", are valid entity domain types. "ipv4.anycast", "pid.local" and "priv:" are invalid.

Although "_", "-", "__--" are valid entity domain types, it is desirable to add characters such as alphanumeric ones, for better intelligibility.

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

An entity domain type defines the semantics of a type of entity, independently of any specifying resource. All entity domain types that are not prefixed with "priv:" MUST be registered with the IANA, in the "ALTO Entity Domain Type Registry", defined in Section 12.3, following the procedure specified in Section 12.3.2 of this document. The format of the entity identifiers (see Section 5.1.3) in that entity domain type, as well as any hierarchical or inheritance rules (see Section 5.1.4) for those entities, MUST be specified in the IANA registration.

Entity domain type identifiers prefixed with "priv:" are reserved for Private Use (see [RFC8126]) without a need to register with IANA. The definition of a private use entity domain type MUST apply the same way in all property maps of an IRD where it is present.

5.1.2. Entity Domain Name

As discussed in Section 3.2, an entity domain is characterized by a type and identified by a name.

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

Each entity domain is identified by a unique entity domain name. Borrowing the symbol "::=" from the Backus-Naur Form notation [RFC5511], the format of an entity domain name is defined as follows:

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

The presence and construction of the component

"[[ResourceID] '.']"

depends on the category of entity domain.

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

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

5.1.2.1. Resource-specific Entity Domain

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

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

5.1.2.2. Resource-agnostic Entity Domain

A resource-agnostic entity domain contains entities that are identified independently of any information resource. The identifier of a resource-agnostic entity domain is simply the identifier of its entity domain type. For example, "ipv4" and "ipv6" identify the two resource-agnostic Internet address entity domains defined in Section 6.1.

5.1.2.3. Self-defined Entity Domain

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

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

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

EntityDomainName ::= '.' EntityDomainType

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

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

5.1.3. Entity Identifier

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

EntityID ::= EntityDomainName ':' DomainTypeSpecificEntityID

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

The format of the second part of an entity identifier, DomainTypeSpecificEntityID, depends on the entity domain type, and MUST be specified when defining a new entity domain type and registering it with the IANA. Identifiers MAY be hierarchical, and properties MAY be inherited based on that hierarchy. The rules defining any hierarchy or inheritance MUST be defined when the entity domain type is registered.

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

Note that two entity identifiers with different valid textual representations may refer to the same entity, for a given entity domain. For example, the strings "net1.ipv6:2001:db8::1" and "net1.ipv6:2001:db8:0:0:0:0:1" refer to the same entity in the "ipv6" entity domain. Such equivalences should be established by the object represented by DomainTypeSpecificEntityID, for example, [RFC5952] establishes equivalence for IPv6 addresses, while [RFC4632] does so for IPv4 addresses.

5.1.4. Hierarchy and Inheritance

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

5.2. Entity Property

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

5.2.1. Entity Property Type

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

While, in Section 5.1.1, character ":" is allowed with restrictions on entity domain identifiers, it can be used without restrictions on entity property type identifiers. This relates to [RFC7285], where a Server can define properties on endpoints "ipv4" and "ipv6". In the present extension, there is a mapping of ALTO entity domain types "ipv4" and "ipv6", to ALTO address types "ipv4" and "ipv6". Properties defined on "ipv4" and "ipv6" endpoints should be re-usable on "ipv4" and "ipv6" entities. Forbidding the usage of ":" in a non-private entity property type identifier would not allow to use properties previously defined on "ipv4" and "ipv6" endpoints because their identifiers would be invalid.

Although ":" or "::-" are valid entity domain types, it is desirable to add characters such as alphanumeric ones, for better intelligibility.

Identifiers prefixed with "priv:" are reserved for Private Use [RFC8126] without a need to register with IANA. All other identifiers for entity property types MUST be registered in the "ALTO Entity Property Type Registry", defined in Section 12.4. The intended semantics of the entity property type MUST be specified in the IANA registration.

For an entity property identifier with the "priv:" prefix, an additional string (e.g., company identifier or random string) MUST follow the prefix to reduce potential collisions, that is, the string "priv:" alone is not a valid entity property identifier. The definition of a private use entity property type must apply the same way in all property maps of an IRD where it is present.

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

- * Some entity property types are applicable to entities in particular entity domain types only. For example, the property type "pid" is applicable to entities in the entity domain types "ipv4" or "ipv6" while is not applicable to entities in an entity domain of type "pid".
- * The intended semantics of the value of an entity property may also depend on the entity domain type. For example, suppose that a property named "geo-location" is defined as the coordinates of a point, encoded as: "latitude longitude [altitude]". When applied to an entity that represents a specific host computer, identified by an address in an entity domain of type "ipv4" or "ipv6", the "geo-location" property would define the host's location. However, when applied to an entity in a "pid" domain type, the property would indicate a location representative of all hosts in this "pid" entity.

5.2.2. Entity Property Name

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

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

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

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

The specific information resource of an entity property may be the current information resource itself, that is, the property map defining the property. In that case, the ResourceID in the property name SHOULD be omitted. For example, the property name ".asn" applied to an entity identified by its IPv4 address, indicates the AS number of the AS that "owns" the entity, where the returned AS number is defined by the property map itself.

5.2.3. Format for Entity Property Value

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

6. Entity Domain Types Defined in this Document

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

6.1. Internet Address Domain Types

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

6.1.1. Entity Domain Type: IPv4

6.1.1.1. Entity Domain Type Identifier

The identifier for this Entity Domain Type is "ipv4".

6.1.1.2. Domain-Specific Entity Identifiers

Individual addresses are strings as specified by the IPv4address rule in Section 3.2.2 of [RFC3986]; Hierarchical addresses are strings as specified by the prefix notation in Section 3.1 of [RFC4632]. To define properties, an individual Internet address and the corresponding full-length prefix are considered aliases for the same entity on which to define properties. Thus, "ipv4:192.0.2.0" and "ipv4:192.0.2.0/32" are equivalent.

6.1.2. Entity Domain Type: IPv6

6.1.2.1. Entity Domain Type Identifier

The identifier for this Entity Domain Type is "ipv6".

6.1.2.2. Domain-Specific Entity Identifiers

Individual addresses are strings as specified by Section 4 of [RFC5952]; Hierarchical addresses are strings as specified by IPv6 address prefixes notation in Section 2.3 of [RFC4291]. To define properties, an individual Internet address and the corresponding 128-bit prefix are considered aliases for the same entity. That is, "ipv6:2001:db8::1" and "ipv6:2001:db8::1/128" are equivalent, and have the same set of properties.

6.1.3. Hierarchy and Inheritance of Internet Address Domains

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

Hierarchical addresses can also inherit properties: if a property P is not defined for the hierarchical address C, but is defined for a set of hierarchical addresses, where each address C' in the set contains all IP addresses in C, and C' has a shorter prefix length than C, then C MUST inherit the property P from the C' having the longest prefix length.

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

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

Figure 1: Defined Property Values.

Then the following entities have the indicated values:

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

Figure 2: Inherited Property Values.

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

- * If entity X would inherit a value for property P, and if the ALTO server decides to say that "X has no value for P", then the ALTO server MUST return a "null" value for that property on X. In this case, the ALTO client MUST recognize the JSON "null" value as "no value" and interpret it as "do not apply the inheritance rules for this property on X".
- * If the entity would not inherit a value, then the ALTO server MAY return "null" or just omit the property. In this case, the ALTO client cannot infer the value for this property of this entity from the Inheritance rules. So, the client MUST interpret that this property has no value.

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

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

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

application/alto-networkmap+json

6.2. Entity Domain Type: PID

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

6.2.1. Entity Domain Type Identifier

The identifier for this Entity Domain Type is "pid"

6.2.2. Domain-Specific Entity Identifiers

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

6.2.3. Hierarchy and Inheritance

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

6.2.4. Defining Information Resource Media Type for Domain Type PID

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

application/alto-networkmap+json

6.2.5. Relationship To Internet Addresses Domains

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

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

6.3. Internet Address Properties vs. PID Properties

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

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

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

7. Property Map

A property map returns the properties defined for all entities in one or more domains, e.g., the "location" property of entities in "pid" domain, and the "ASN" property of entities in "ipv4" and "ipv6" domains. Section 10.4 gives an example of a property map request and its response.

Downloading the whole property map is a way for the Client to obtain the Entity IDs that can be used as input for a Filtered Property Map request. However, a whole property map may be too voluminous for a Client that only wants the list of applicable Entity IDs. How to obtain the list of entities of a filtered property map in a simplified response is specified in Section 8.

7.1. Media Type

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

7.2. HTTP Method

The property map is requested using the HTTP GET method.

7.3. Accept Input Parameters

A Property Map has no Accept Input parameters.

7.4. Capabilities

The capabilities are defined by an object of type `PropertyMapCapabilities`:

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

with fields:

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

7.5. Uses

The "uses" field of a property map resource in an IRD entry specifies the resources in this same IRD on which this property map directly depends. It is an array of resource ID(s). This array identifies the defining information resources associated with the resource-specific entity domains and properties that are indicated in this resource.

7.6. Response

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

where:

```
object {  
  PropertyMapData property-map;  
} InfoResourceProperties : ResponseEntityBase;  
  
object-map {  
  EntityID -> EntityProps;  
} PropertyMapData;  
  
object {  
  EntityPropertyName -> JSONValue;  
} EntityProps;
```

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

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

For each entity in the property map:

- * If the entity is in a resource-specific entity domain, the ALTO server MUST only return self-defined properties and resource-specific properties which depend on the same resource as the entity does. The ALTO client MUST ignore any resource-specific property for this entity if the mapping between this resource-specific property and this entity is not indicated, in the IRD, in the "mappings" capability of the property map resource.
- * If the entity identifier is resource-agnostic, the ALTO server SHOULD return the self-defined properties and all the resource-specific properties that are defined in the property defining information resources indicated, in the IRD, in the "mappings" capability of the property map resource, unless property values can be omitted upon some inheritance rules.

The ALTO server MAY omit property values that are inherited rather than explicitly defined, in order to achieve more compact encoding. As a consequence, the ALTO Client MUST NOT assume inherited property values will all be present. If the Client needs inherited values, it MUST use the entity domain's inheritance rules to deduce those values.

8. Filtered Property Map

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

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

While the IRD lists all the names of the supported properties, it only lists the names of the supported entity domains and not the entity IDs. A client, sometimes, may only want to know what entity IDs it can provide as input to a filtered property map request but wants to avoid the burden of downloading the full property map. Or it may want to check whether some given entity IDs are eligible for a query. To support such a case, the filtered property map supports a light weight response, with empty property values.

8.1. Media Type

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

8.2. HTTP Method

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

8.3. Accept Input Parameters

The input parameters for a filtered property map request are supplied in the entity body of the POST request. 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. ReqFilteredPropertyMap is designed to support the following cases of client requests:

- * The client wants the value of a selected set of properties on a selected set of entities,
- * The client wants all properties values on all the entities,

- * The client wants all entities on which a property is defined but is not interested in their property values,
- * The Client wants to cross-check whether some entity IDs are present in the Filtered Property Map but is not interested in their property values.

The third case is equivalent to querying the whole unfiltered property map, which can also be achieved with a GET request. Some Clients however, may prefer to systematically make filtered property map queries, where filtering parameters may sometimes be empty.

The JSON object `ReqFilteredPropertyMap` is specified as follows:

```
object {  
  EntityID          entities<0..*>;  
  [EntityPropertyName properties<0..*>;]  
} ReqFilteredPropertyMap;
```

with fields:

entities: List of entity identifiers for which the specified properties are to be returned. If the list is empty, the ALTO Server MUST interpret the list as if it contained a list of all entities currently defined in the filtered property map. The domain of each entity MUST be included in the list of entity domains in this resource's "capabilities" field (see Section 8.4). The ALTO server MUST interpret entries appearing multiple times as if they appeared only once.

properties: List of properties to be returned for each entity. If the list is empty, the ALTO Server MUST interpret the list as if it contained a list of all properties currently defined in the filtered property map. Each specified property MUST be included in the list of properties in this resource's "capabilities" field (see Section 8.4). The ALTO server MUST interpret entries appearing multiple times as if they appeared only once. This field is optional. If it is absent, the Server returns a property value equal to the literal string "{}" for all the entity IDs of the "entities" field on which at least one property is defined.

Note that the field "properties" is optional. When in addition, the "entities" field is an empty list, it corresponds to a query for all applicable entity IDs of the filtered property map, with no current interest on any particular property. When the "entities" field is not empty, it allows the Client to check whether the listed entity IDs can be used as input to a filtered property map query.

8.4. Capabilities

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

8.5. Uses

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

8.6. Filtered Property Map Response

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

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

- * The input field "entities" is absent from the Client request. In this case, the Server MUST return an "E_MISSING_FIELD" error as defined in Section 8.5.2 of [RFC7285].
- * An entity identifier in the "entities" field of the request is invalid. This occurs when:
 - The domain of this entity is not defined in the "entity-domains" capability of this resource in the IRD,
 - The entity identifier is not valid for the entity domain.

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

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

- * A property name in the "properties" field of the request is invalid. This occurs when this property name is not defined in the "properties" capability of this resource in the IRD.

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

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

Some identifiers can be interpreted as both an entity name and a property name, as it is the case for "pid" if it would be erroneously used alone. In such a case, the Server SHOULD follow Section 8.5.2 of [RFC7285], that says: "For an E_INVALID_FIELD_VALUE error, the server may include an optional field named "field" in the "meta" field of the response, to indicate the field that contains the wrong value."

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

- * If the requested entities include entities with a resource-agnostic identifier, the "dependent-vtags" field in its "meta" field MUST include version tags of all dependent resources appearing in the "uses" field.
- * If the requested entities only include entities in resource-specific entity domains, the "dependent-vtags" field in its "meta" field MUST include the version tags of the resources on which the requested resource-specific entity domains and the requested resource-specific properties are dependent on.
- * The response only includes the entities and properties requested by the client. If an entity in the request is identified by a hierarchical identifier (e.g., a "ipv4" or "ipv6" prefix), the response MUST return all properties that are present on any address covered by the prefix, even though some of those properties may not be present on all addresses covered by the prefix.
- * When the input member "properties" is absent from the client request, the Server returns a property map containing all the requested entity identifiers on which one or more properties are defined. For all the entities of the returned map, the returned property value is equal to '{}'.

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

- * If a property for a requested entity is inherited from another entity not included in the request, the response MUST 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 MUST be included in the response for A.
- * If there are entities covered by a requested entity but having different values for the requested properties, the response MUST 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.

For the sake of response compactness, the ALTO server SHOULD obey the following rule:

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

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

8.7. Entity Property Type Defined in This Document

This document defines the entity property type "pid". This property type extends the ALTO Endpoint Property Type "pid" defined in section 7.1.1 of [RFC7285] as follows: the property has the same semantics and applies to IPv4 and IPv6 addresses; the difference is that the IPv4 and IPv6 addresses have evolved from the status of endpoints to the status of entities.

The defining information resource for property type MUST be a network map. This document requests a IANA registration for this property

8.7.1. Entity Property Type: pid

1. Identifier: pid
2. Semantics: the intended semantics are the same as in [RFC7285] for the ALTO Endpoint Property Type "pid"
3. Media type of defining information resource: application/alto-networkmap+json
4. Security considerations: for entity property type "pid" are the same as documented in [RFC7285] for the ALTO Endpoint Property Type "pid".

9. Impact on Legacy ALTO Servers and ALTO Clients

9.1. Impact on Endpoint Property Service

Since the Property Map and the Filtered Property Map defined in this document provide a functionality that covers the EPS defined in Section 11.4 of [RFC7285], ALTO servers may prefer to provide Property Map and Filtered Property Map in place of EPS. However, for the legacy endpoint properties, it is recommended that ALTO servers also provide EPS so that legacy clients can still be supported.

9.2. Impact on Resource-Specific Properties

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

9.3. Impact on Other Properties

In the present extension, properties can be defined on sets of entity addresses, rather than just individual endpoint addresses as initially defined in [RFC7285]. This might change the semantics of a property. These sets can be for example hierarchical IP address blocks. For instance, a property such as fictitious "geo-location", defined on a set of IP addresses would have a value corresponding to a location representative of all the addresses in this set.

10. Examples

In this document, the HTTP message bodies of all the examples use Unix-style line-ending character (%x0A) as the line separator.

10.1. Network Map

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

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

Figure 3: Example Default Network Map

And another simple alternative network map:

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

Figure 4: Example Alternative Network Map

10.2. Property Definitions

Beyond "pid", the examples in this section use four additional fictitious property types for entities of domain type "ipv4": "countrycode", "ASN", "ISP", and "state". These properties are assumed to be resource-agnostic so their name is identical to their type. The entities have the following values:

	ISP	ASN	countrycode	state
ipv4:192.0.2.0/23:	BitsRus	-	us	-
ipv4:192.0.2.0/28:	-	65543	-	NJ
ipv4:192.0.2.16/28:	-	65543	-	CT
ipv4:192.0.2.1:	-	-	-	PA
ipv4:192.0.3.0/28:	-	65544	-	TX
ipv4:192.0.3.16/28:	-	65544	-	MN

Figure 5: Example Property Values for Internet Address Domains

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

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

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

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

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

	ASN
pid:defaultpid:	-
pid:pid1:	65543
pid:pid2:	65544

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

10.3. Information Resource Directory (IRD)

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

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

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

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

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

The other property maps in the returned IRD are here for purposes of illustration.

```
GET /directory HTTP/1.1
Host: alto.example.com
Accept: application/alto-directory+json,application/alto-error+json
```

```
HTTP/1.1 200 OK
Content-Length: 2713
Content-Type: application/alto-directory+json
```

```
{
  "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"
    },
    "ia-property-map" : {
      "uri" : "http://alto.example.com/propmap/full/inet-ia",
      "media-type" : "application/alto-propmap+json",
      "capabilities" : {
        "mappings": {
          "ipv4": [ ".ISP", ".ASN" ],

```

```
        "ipv6": [ ".ISP", ".ASN" ]
      }
    },
    "iacs-property-map" : {
      "uri" : "http://alto.example.com/propmap/lookup/inet-iacs",
      "media-type" : "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "capabilities" : {
        "mappings": {
          "ipv4": [ ".ISP", ".ASN", ".countrycode", ".state" ],
          "ipv6": [ ".ISP", ".ASN", ".countrycode", ".state" ]
        }
      }
    },
    "region-property-map": {
      "uri": "http://alto.example.com/propmap/lookup/region",
      "media-type": "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "uses" : [ "default-network-map", "alt-network-map" ],
      "capabilities": {
        "mappings": {
          "default-network-map.pid": [ ".region" ],
          "alt-network-map.pid": [ ".ASN" ]
        }
      }
    },
    "ip-pid-property-map" : {
      "uri" : "http://alto.example.com/propmap/lookup/pid",
      "media-type" : "application/alto-propmap+json",
      "accepts" : "application/alto-propmapparams+json",
      "uses" : [ "default-network-map", "alt-network-map" ],
      "capabilities" : {
        "mappings": {
          "ipv4": [ "default-network-map.pid",
                    "alt-network-map.pid" ],
          "ipv6": [ "default-network-map.pid",
                    "alt-network-map.pid" ]
        }
      }
    },
    "legacy-endpoint-property" : {
      "uri" : "http://alto.example.com/legacy/eps-pid",
      "media-type" : "application/alto-endpointprop+json",
      "accepts" : "application/alto-endpointpropparams+json",
      "capabilities" : {
        "properties" : [ "default-network-map.pid",
                        "alt-network-map.pid" ]
      }
    }
  },
  "legacy-endpoint-property" : {
    "uri" : "http://alto.example.com/legacy/eps-pid",
    "media-type" : "application/alto-endpointprop+json",
    "accepts" : "application/alto-endpointpropparams+json",
    "capabilities" : {
      "properties" : [ "default-network-map.pid",
                      "alt-network-map.pid" ]
    }
  }
}
```

```

    }
  },
  "ane-dc-property-map": {
    "uri" : "http://alto.example.com/propmap/lookup/ane-dc",
    "media-type" : "application/alto-propmap+json",
    "accepts": "application/alto-propmapparams+json",
    "capabilities": {
      "mappings": {
        ".ane" : [ "storage-capacity", "ram", "cpu" ]
      }
    }
  }
}
}
}

```

Figure 8: Example IRD

10.4. Full Property Map Example

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

Note that, to be compact, the response does not include the entity "ipv4:192.0.2.1" 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 "ipv4:192.0.3.16/28". 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: 418
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": "65543"},
    "ipv4:192.0.3.0/27": {".ASN": "65544"}
  }
}
```

10.5. Filtered Property Map Example #1

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

Note that the value of "state" for "ipv4:192.0.2.1" 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: 158
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: 540
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": "65543", ".state": "NJ"},
    "ipv4:192.0.2.1":
      {".ISP": "BitsRus", ".ASN": "65543", ".state": "PA"},
    "ipv4:192.0.2.17":
      {".ISP": "BitsRus", ".ASN": "65543", ".state": "CT"}
  }
}
```

10.6. Filtered Property Map Example #2

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

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

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

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

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: 174
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", ".countrycode", ".state" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: 774
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": {".countrycode": "us"},
    "ipv4:192.0.2.0/28": {".ASN": "65543",
                        ".state": "NJ"},
    "ipv4:192.0.2.16/28": {".ASN": "65543",
                          ".state": "CT"},
    "ipv4:192.0.2.1": {".state": "PA"},
    "ipv4:192.0.3.0/26": {".countrycode": "us"},
    "ipv4:192.0.3.0/28": {".ASN": "65544",
                        ".state": "TX"},
    "ipv4:192.0.3.16/28": {".ASN": "65544",
                          ".state": "MN"}
  }
}
```

10.7. Filtered Property Map Example #3

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

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

```
POST /propmap/lookup/pid HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: 222
Content-Type: application/alto-propmapparams+json
```

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

```
HTTP/1.1 200 OK
Content-Length: 774
Content-Type: application/alto-propmap+json
```

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

10.8. Filtered Property Map Example #4

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

```
POST /propmap/lookup/region HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: 132
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: 326
Content-Type: application/alto-propmap+json
```

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

10.9. Filtered Property Map for ANEs Example #5

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

```
POST /propmap/lookup/ane-dc HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: 155
Content-Type: application/alto-propmapparams+json
```

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

```
HTTP/1.1 200 OK
Content-Length: 295
Content-Type: application/alto-propmap+json
```

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

11. Security Considerations

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

ALTO clients using this extension should in addition be aware that the entity properties they require may convey more details than the endpoint properties conveyed by using [RFC7285]. Client requests may reveal details on their activity or plans thereof, that a malicious

Server, that is in a position to do so, may monetize or use for attacks or undesired surveillance. Likewise, ALTO Servers expose entities and properties related to specific parts of the infrastructure that reveal details on capabilities, locations, or resource availability. These details may be maliciously used for competition purposes, or to cause resource shortage or undesired publication.

To address these concerns, the Property Maps provided by this extension require additional attention on two security considerations discussed in [RFC7285]: "potential undesirable guidance from authenticated ALTO information" (Section 15.2 of [RFC7285]) and "confidentiality of ALTO information" (Section 15.3 of [RFC7285]). Threats to the availability of the ALTO Service caused by highly demanding queries should be addressed as specified in Section 15.5 of [RFC7285].

- * Potential undesirable guidance from authenticated ALTO information: it can be caused by Property values that change over time and thus lead to performance degradation or system rejection of application requests.

To avoid these consequences, a more robust ALTO client should adopt and extend protection strategies specified in Section 15.2 of [RFC7285]. For example, to be notified immediately when a particular ALTO value that the Client depends on changes, it is RECOMMENDED that both the ALTO Client and ALTO Server using this extension implement "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)" [RFC8895].

- * Confidentiality of ALTO information: 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. Another way is to expose highly abstracted coarse-grained property values to all Clients while restricting access to URIs exposing more fine-grained values to authorized Clients. Restricted access URIs may be gathered in delegate IRDs as specified in Section 9.2.4 of [RFC7285].

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 that do not contain sensitive data.

Section 12 of this document specifies that the ALTO service provider **MUST** be aware of the potential sensitivity of exposed entity domains and properties. Section 12.2.2. (ALTO Entity Domain Type Registration Process) of this document specifies that when the registration of an entity domain type is requested at the IANA, the request **MUST** include security considerations that show awareness of how the exposed entity addresses may be related to private information about an ALTO client or an infrastructure service provider. Likewise, Section 12.3. (ALTO Entity Property Type Registry) of this document specifies that when the registration of a property type is requested at the IANA, the request **MUST** include security considerations that explain why this property type is required for ALTO-based operations.

The risk of ALTO information being leaked to malicious Clients or third parties is addressed similarly to Section 7 of [RFC8896]. ALTO clients and servers **SHOULD** support TLS 1.3 [RFC8446].

12. IANA Considerations

This document defines additional application/alto-* media types, that are listed in Table 1. It defines an ALTO Entity Domain Type Registry that extends the ALTO Address Type Registry defined in [RFC7285]. It also defines an ALTO Entity Property Type Registry that extends the ALTO endpoint property registry defined in [RFC7285].

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

Table 1: Additional ALTO Media Types.

12.1. application/alto-propmap+json Media Type

Type name:

application

Subtype name:

alto-propmap+json

Required parameters:

n/a

Optional parameters:

n/a

Encoding considerations:

Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations:

Security considerations related to the generation and consumption of ALTO Protocol messages are discussed in Section 15 of [RFC7285] and Section 11 of this document.

Interoperability considerations:

n/a

Published specification:

This document is the specification for this media type. See Section 7.1.

Applications that use this media type:

ALTO servers and ALTO clients [RFC7285], either stand alone or embedded within other applications, when the queried resource is a property map, whether filtered or not.

Fragment identifier considerations:

n/a

Additional information:

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

Person & email address to contact for further information:

See Authors' Addresses section.

Intended usage:
COMMON

Restrictions on usage:
n/a

Author:
See Authors' Addresses section.

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

12.2. alto-propmapparams+json Media Type

Type name:
application

Subtype name:
alto-propmapparams+json

Required parameters:
n/a

Optional parameters:
n/a

Encoding considerations:
Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations:
Security considerations related to the generation and consumption of ALTO Protocol messages are discussed in Section 15 of [RFC7285] and Section 11 of this document.

Interoperability considerations:
n/a

Published specification:
This document is the specification for this media type. See Section 8.3.

Applications that use this media type:
ALTO servers and ALTO clients [RFC7285], either stand alone or embedded within other applications, when the queried resource is a filtered property map. This media-type indicates the data format used by the ALTO client to supply the property map filtering parameters.

Fragment identifier considerations:
n/a

Additional information:
Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

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

Intended usage:
COMMON

Restrictions on usage:
n/a

Author:
See Authors' Addresses section.

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

12.3. ALTO Entity Domain Type Registry

This document requests IANA to create and maintain the "ALTO Entity Domain Type Registry", listed in Table 2. The first line lists information items that must be provided with each registered entity domain type. Section 12.3.2 specifies how to document these items and provides guidance on the security considerations item that must be documented in addition.

Identifier	Entity Identifier Encoding	Hierarchy & Inheritance	Media Type of Defining Resource	Mapping to ALTO Address Type
ipv4	See Section 6.1.1	See Section 6.1.3	application/alto-networkmap+json	true
ipv6	See Section 6.1.2	See Section 6.1.3	application/alto-networkmap+json	true
pid	See Section 6.2	None	application/alto-networkmap+json	false

Table 2: ALTO Entity Domain Types

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

As specified in Section 5.1.1, identifiers prefixed with "priv:" are reserved for Private Use without a need to register with IANA

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

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

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

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

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

- * Whether the ALTO Address Type Registry contains an address type that can be used as an identifier for the candidate entity domain type identifier. This has been done for the identifiers "ipv4" and "ipv6" of Table 2.
- * Whether the candidate entity domain type identifier can potentially be an endpoint address type, as defined in Sections 2.1 and 2.2 of [RFC7285].

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

- * Test: Do corresponding entity domain type identifiers match a known "network" address type?
 - If yes (e.g., cell, MAC or socket addresses):
 - o 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].
 - o Use the new ALTO entity domain type identifier to register a new ALTO entity domain type in the ALTO Entity Domain Type Registry following Section 12.3.2 of this document.
 - If no (e.g., pid name, ane name or country code): Proceed with the ALTO Entity Domain Type registration as described in Section 12.3.2.

12.3.2. ALTO Entity Domain Type Registration Process

New ALTO entity domain types are assigned after IETF Review [RFC8126] 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 MUST indicate how an entity in a registered type of domain is encoded as an EntityID, and, if applicable, the rules defining the entity hierarchy and property inheritance. Updates and deletions of ALTO entity domains types follow the same procedure.

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

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

- * Identifier: The name of the desired ALTO entity domain type.
- * Entity Identifier Encoding: The procedure for encoding the identifier of an entity of the registered domain type as an EntityID (see Section 5.1.3). If corresponding entity identifiers of an entity domain type match a known "network" address type, the Entity Identifier Encoding of this domain identifier MUST include both Address Encoding and Prefix Encoding of the same identifier registered in the ALTO Address Type Registry [RFC7285]. To define properties, an individual entity identifier and the corresponding full-length prefix MUST be considered aliases for the same entity.
- * Hierarchy: If the entities form a hierarchy, the procedure for determining that hierarchy.
- * Inheritance: If entities can inherit property values from other entities, the procedure for determining that inheritance.
- * Media type of defining information resource: Some entity domain types allow an entity domain name to be combined with an information resource name to define a resource-specific entity domain. Such an information resource is called "defining information resource", defined in Section 4.6. For each entity domain type, the potential defining information resources have one common media type. This unique common media type is specific to the entity domain type and MUST be specified.
- * Mapping to ALTO Address Type: A boolean value to indicate if the entity domain type can be mapped to the ALTO address type with the same identifier.

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

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

12.4. ALTO Entity Property Type Registry

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

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

As specified in Section 5.2.1, identifiers prefixed with "priv:" are reserved for Private Use without a need to register with IANA.

The first line of Table 3 lists information items that must be provided with each registered entity property type.

Identifier	Intended Semantics	Media Type of Defining Resource
pid	See Section 7.1.1 of [RFC7285]	application/alto-networkmap+json

Table 3: ALTO Entity Property Types.

New ALTO entity property types are assigned after IETF Review [RFC8126] to ensure that proper documentation regarding the new ALTO entity property types and their security considerations has been provided. RFCs defining new entity property types SHOULD indicate how a property of a registered type is encoded as a property name. Updates and deletions of ALTO entity property types follow the same procedure.

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

- * Identifier: The identifier for the desired ALTO entity property type. The format MUST be as defined in Section 5.2.1 of this document.
- * 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.
- * Media type of defining information resource: when the property type allows values to be defined relative to a given information resource, the latter is referred to as the "defining information resource", see description in Section 4.7. For each property type, the potential defining information resources have one common media type. This unique common media type is specific to the property type and MUST be specified.
- * Security Considerations: ALTO entity properties expose information to ALTO clients. ALTO service providers should be cognizant of the security ramifications related to the exposure of an entity property.

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

This document requests registration of the identifier "pid", listed in Table 3. Semantics for this property are documented in Section 7.1.1 of [RFC7285]. No security issues related to the exposure of a "pid" identifier are considered, as it is exposed with the Network Map Service defined and mandated in [RFC7285].

13. Acknowledgments

The authors would like to thank Dawn Chen, and Shenshen Chen for their contributions to earlier drafts. Thank you also to Qiao Xiang, Shawn Lin, Xin Wang and Vijay Gurbani for fruitful discussions. Last, big thanks to Danny Perez and Luis Contreras for their substantial Working Group review feedback and suggestions to improve this document, to Vijay Gurbani, ALTO WG Chair and Martin Duke, Transport Area Director, for their thorough review, discussions, guidance and shepherding, that further helped to enrich this

document.

14. References

14.1. Normative References

- [ISO3166-1] ISO (International Organization for Standardization), ., "ISO 3166-1: Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", 2020.
- [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>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", February 2006.
- [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>>.
- [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>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <<https://www.rfc-editor.org/info/rfc8895>>.

14.2. Informative References

- [I-D.ietf-alto-cdni-request-routing-alto]
Seedorf, J., Yang, Y., Ma, K., Peterson, J., and J. Zhang, "Content Delivery Network Interconnection (CDNI) Request Routing: CDNI Footprint and Capabilities Advertisement using ALTO", Work in Progress, Internet-Draft, draft-ietf-alto-cdni-request-routing-alto-16, 12 January 2021, <<http://www.ietf.org/internet-drafts/draft-ietf-alto-cdni-request-routing-alto-16.txt>>.
- [I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector", Work in Progress, Internet-Draft, draft-ietf-alto-path-vector-13, 20 November 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-alto-path-vector-13.txt>>.
- [RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", July 2004.
- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", April 2009.
- [RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", January 2010.

- [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>>.
- [RFC8896] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", RFC 8896, DOI 10.17487/RFC8896, November 2020, <<https://www.rfc-editor.org/info/rfc8896>>.

Appendix A. Features introduced with the Entity Property Maps extension

The Entity Property Maps extension described in this document introduces a number of features that are summarized in table below. The first column provides the name of the feature. The second column provides the section number of this document that gives a high level description of the feature. The third column provides the section number of this document that gives a normative description relating to the feature, when applicable.

Feature	High-level description	Related normative description
Entity	Section 3.1	Section 5.1.3
Entity domain (ED)	Section 3.2	
Entity domain type	Section 3.2.1	Section 5.1.1
Entity domain name	Section 3.2.2	Section 5.1.2
Entity property (EP) type	Section 3.3	Section 5.2, Section 5.2.1, Section 5.2.2, Section 5.2.3
Entity property map	Section 3.4	Section 7, Section 8
Resource-specific ED name	Section 4.2	Section 5.1.2, Section 5.1.2.1
Resource-specific EP value	Section 4.3	Section 5.2.3
Entity Hierarchy and property inheritance	Section 4.4	Section 5.1.4
Defining information resource	Section 4.6, Section 4.7	Section 12.3.2, Section 12.4

Table 4: Features introduced with ALTO Entity Property Maps

Authors' Addresses

Wendy Roome
 Nokia Bell Labs (Retired)
 124 Burlington Rd
 Murray Hill, NJ 07974
 United States of America
 Phone: +1-908-464-6975
 Email: wendy@wdroome.com

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
91460 NOZAY
France
Email: Sabine.Randriamasy@nokia-bell-labs.com

Y. Richard Yang
Yale University
51 Prospect Street
New Haven, CT 06511
United States of America
Phone: +1-203-432-6400
Email: yry@cs.yale.edu

Jingxuan Jensen Zhang
Tongji University
4800 Cao'An Hwy
Shanghai
201804
China
Email: jingxuan.n.zhang@gmail.com

Kai Gao
Sichuan University
No.24 South Section 1, Yihuan Road
Chengdu
610000
China
Email: kaigao@scu.edu.cn

ALTO WG
Internet-Draft
Intended status: Informational
Expires: January 14, 2021

D. Lachos
C. Rothenberg
Unicamp
July 13, 2020

Multi-domain E2E Network Services
draft-lachosrothenberg-alto-md-e2e-ns-02

Abstract

Evolving networking scenarios (e.g., 5G) are considering the provision of value-added and on-demand end-to-end (E2E) network services in multi-domain (multi-operator/multi-technology) environments. This document presents different initiatives, mainly within standardization efforts and research projects, working on E2E network services across multiple domains. Problem statement and a layered network model are also described. In addition, this document raises an initial proposal towards a new ALTO service in support of E2E network service requirements. Finally, another important objective of this document is to begin a discussion about motivating use cases in scope of the ALTO WG after the re-chartering process.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Context and Motivation	3
2.1. Standardization Activities	3
2.1.1. IETF	3
2.1.2. ETSI	4
2.1.3. MEF	5
2.2. Research projects	5
3. Problem Statement	6
3.1. Network Function Placement Decisions	6
3.2. Network Inventory	6
3.3. Publishing Information	6
4. Network Function Virtualization Architectures and Infrastructures	7
4.1. Layered Network Model	8
5. ALTO Extension: E2E Network Service Requirements Representation	10
6. IANA Considerations	12
7. Security Considerations	12
8. Acknowledgments	12
9. References	12
9.1. Normative References	12
9.2. Informative References	12
Authors' Addresses	14

1. Introduction

The fifth generation (5G) of cellular networks is not only considered an evolution but a revolution in the field of information and communication technologies [WHITE-PAPER-5G]. 5G will support the creation of new and novel End-to-End (E2E) services, applications and complex use case scenarios, such as massive Internet of Things, extreme real-time communications, broadband access everywhere, higher user mobility. All these scenarios and services are triggering a modification in the way telecommunications sector deploy new network services, shifting from a commonly manual and long process to a flexible and programmable process.

In this context, cloud computing , Software Defined Networking (SDN), and Network Function Virtualization (NFV) arise as technological

pillars to achieve the necessary function programmability, network programmability, and resource virtualization during the provision of E2E network services.

The delivery of an E2E network service, or simply E2E service, often requires VNFs and their specific order [RFC7665]. Network operators start offering to their customers the possibility of configuring network services with specific requirements in terms of resources (e.g., cpu, memory, hard-disk) and performance objectives (e.g., bandwidth, latency) [VNF-PLAC]. Such demands are usually composed by distributed resources which are expected to be available across multiple domains with different technology and/or administration.

This document offers an overview of standardization activities and research projects, including problem statement, behind building E2E services traversing different domains (technological and/or administrative). Moreover, from a layered network model, it is proposed a potential ALTO extension related to E2E Network Service requirements representation based on the ETSI NFV MANO data model.

The overall rationale of this document is to arouse discussions into the ALTO WG concerning potential new items to be considered for the re-charter.

2. Context and Motivation

Different standardization efforts (e.g., IETF, MEF, ETSI) and research projects activities (e.g., 5GEX [H2020.5GEX], 5G-Transformer [H2020-5G-TRANSFORMER], T-NOVA [T-NOVA]) have been focused on multi-domain network service chaining. Standardization is essential to provide recommendations to create interoperable architectures with standardized protocols, and solutions (being developed by different projects) are addressing a diverse range of requirements to provide network services provided using multiple domains.

This section briefly describes, on the one hand, main standardization efforts delivering collections of norms and recommendations, while on the other hand it also provides an overview of several projects formed to develop network services across multiple domains.

2.1. Standardization Activities

2.1.1. IETF

SFC that span domains owned by single or multiple administrative entities are being proposed. The Hierarchical Service Function Chaining (hSFC) [RFC8459], for example, defines an architecture to deploy SFC in large networks. This RFC proposes to decompose the

network into smaller domains (domains under the control of a single organization). Another proposed initiative is [DRAFT-HH-MDSFC] that describes SFC crossing different domains owned by various organizations (e.g., ISPs) or by a single organization with administration partitions. The proposed architecture uses a SFC eXchange Platform (SXP) to collect and exchange information (topology, service states, policies, etc.) between different organizations and it works both in centralized (Multiple SFC domains connected by a logical SXP) and distributed (SXP server as a broker) environments.

More recently, the IETF ALTO WG started to discuss the uses of ALTO as an information model for representing network resource and services in multi-domain scenarios:

- o [DRAFT-ALTO-BROKER-MDO] proposes an ALTO-based Broker-assisted architecture where a broker plane works as a coordinator between a set of top-level control planes, i.e., Domain Orchestrators (DOs) and Multi-Domain Orchestrators (MdOs). The ALTO services (with the proposed extensions) provides abstract maps with a simplified, yet enough information view about MdOs involved in the federation. This information includes the abstract network topology, resource availability (e.g., CPUs, Memory, and Storage) and capabilities (e.g., supported network functions).
- o [DRAFT-ALTO-UNICORN] presents Unicorn, a resource orchestration framework for multi-domain, geo-distributed data analytics. This work resorts in ALTO as the information model to support the accurate, yet privacy-preserving resource discovery across different domains. The key information to be provided by the use of ALTO including different types of resources, e.g., the computing, storage, and networking resources.
- o [DRAFT-MD-SFC-ALTO] describes different standardization activities and research projects addressing the challenges posed by Service Function Chaining (SFC) across multiple domains (specifically, multiple administrative domains). In addition, this document presents an initial approach to realize inter-domain service chaining leveraging the ALTO protocol. Finally, another important concern of this document is to initiate a discussion (ALTO, SFC as well as 9other WGs) regarding if, how, and under what conditions ALTO can be useful to improve the multi-domain SFC process.

2.1.2. ETSI

The ETSI NFV ISG is paving the way toward viable architectural options supporting the efficient placement of functions in different administrative domains. More specifically, the document

[ETSI-NFV-IFA028] reports different NFV MANO architectural approaches with use cases related to network services provided using multiple administrative domains. Besides, it gives a non-exhaustive list of key information to be exchanged between administrative domains (monitoring parameters, topology view, resource capabilities, etc.) and recommendations related to security to permit the correct and proper operation of the final service.

2.1.3. MEF

With its work on the Service Operations Specification MEF 55 [MEF-SOE-MEF55], MEF has defined a reference architecture and framework for describing functional management entities (and interfaces between them) needed to support Lifecycle Service Orchestration (LSO). This LSO architecture enables automated management and control of E2E connectivity services across multiple operator networks. The automated service management includes fulfillment, control, performance, assurance, usage, security, analytics, and policy capabilities that make it possible, for example, expanding the footprint of service providers to interact with potentially several operators to manage and control the access portions of E2E services.

2.2. Research projects

Several projects include an architectural model integrating NFV management with SDN control capabilities to address the challenges towards flexible, dynamic, cost-effective, and on-demand service chaining.

[H2020.5GEX] aims to integrate multiple administrations and technologies through the collaboration between operators in the context of emerging 5G networking. [VITAL][T-NOVA] follow a centralized approach where each domain advertises its capabilities to a federation layer which will act as a broker. In order to avoid one network operator per country or regions, [H2020-5G-NORMA] proposes the use of management and control into a single virtual domain. Also, the 5G-Transformer project [H2020-5G-TRANSFORMER] is defining flexible slicing and federation of transport networking and computing resources across multiple domains. The NECOS project [H2020-NECOS], focused on the realization of E2E multi-domain cloud network slicing, proposes an architectural approach with slice information interfaces for resource exposure and resource discovery during slice provisioning. In addition, the architecture includes a slice marketplace interface between domain orchestrators and a marketplace broker.

3. Problem Statement

3.1. Network Function Placement Decisions

An E2E service request specify virtual nodes (a set of required VNFs) as well as virtual links (the order in which they must be executed). Virtual nodes are deployed in virtual machines hosted by different physical servers, and virtual links correspond to physical paths that connect those servers hosting VNFs. Both virtual nodes and virtual links are limited resources and both may also be located on different technological domains in a single administration and even crossing multiple administrations [VNF-MOB][SFC-ORC]. So that the placement decision problem involves to discover "best" candidate resources and "best" feasible paths between such resources.

3.2. Network Inventory

Placement decisions are a fundamental step for the management and orchestration of network services. Management systems (e.g., DOs, MdOs) need to maintain an inventory of the network providing a real-time representation or view of available infrastructure resources, software resources, and their relationships. However, The size of a network inventory can be very large in scenarios, such as distributed cloud and edge computing. As a result, management systems experiment scalability problems processing large amounts of data to decide where to instantiate a service or part of the service. Therefore, building a network inventory, under these circumstances, needs aggregation mechanisms to reduce time for discovery of resources and to simplify and optimize management of them.

3.3. Publishing Information

Once a network inventory is built, a mechanism for publishing information is also necessary so that the network inventory can provide a simplified, yet enough network information view to management systems. In order to retrieve such information to perform placement decisions, a communication protocol between management systems and network inventory is also necessary.

Therefore, on the one hand, network information (e.g., network locations, costs between them, endhost properties) needs to be advertised to the network applications and, on the other hand, network applications (e.g., DOs, MdOs) needs to describe their requirements and obtain information about resources that suit such requirements.

4. Network Function Virtualization Architectures and Infrastructures

With the introduction of NFV, network functions (e.g., switches, routers, firewalls), and also complex network functions (e.g., EPC) are able to be virtualized and implemented as a collection of virtual machines (VMs) deployed over the virtualized infrastructure. In turn, the virtualized infrastructure is instantiated on a substrate network.

In this context, one of the most accepted NFV architectural frameworks is the proposed by the ETSI ISG NFV working group [ETSI-NFV-WHITEPAPER]. Figure 1 [DRAFT-MD-VIRT] shows this NFV reference architecture. On the left, we can see the data plane: NFVIs hardware/software, VNFs, and optional element management systems. On the right, we see the control plane: VIM which is something like Openstack or Kubernetes, virtual network function managers, and the NFV Orchestrator on the top.

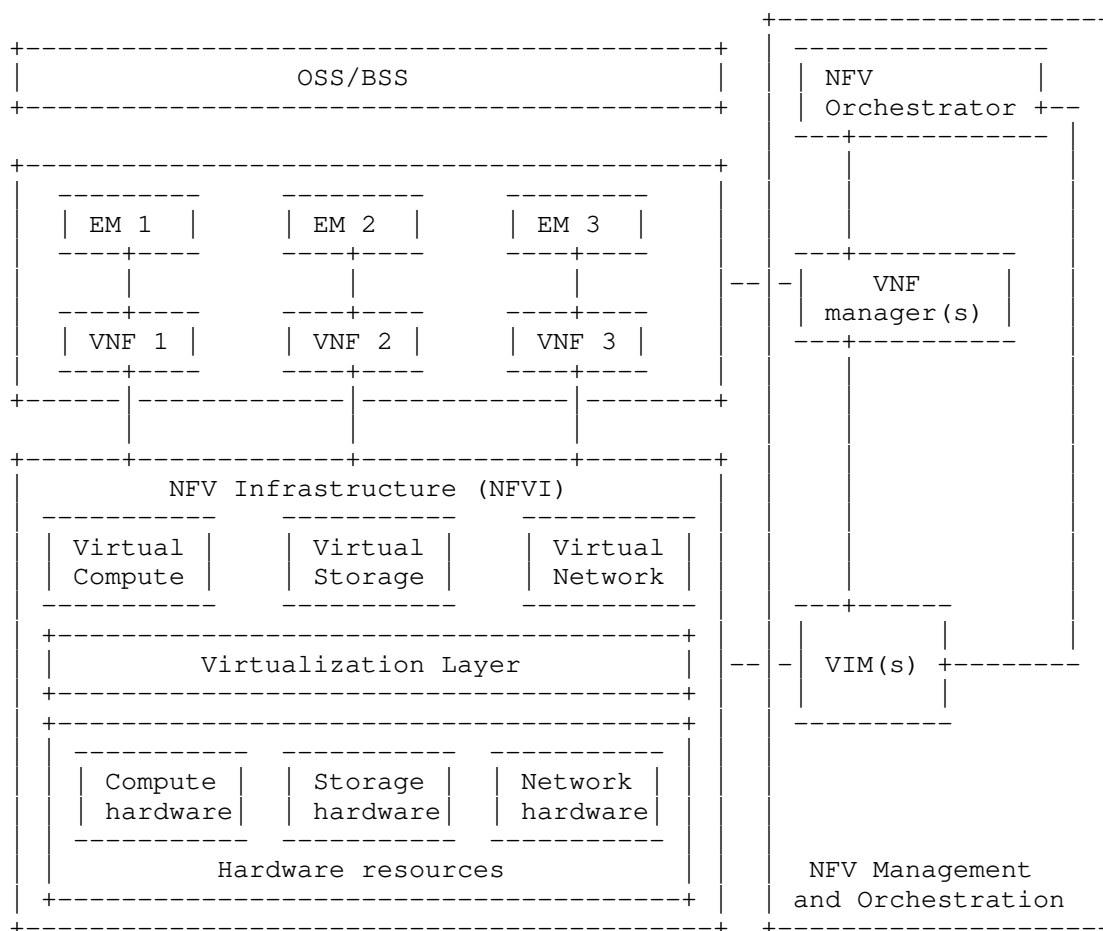


Figure 1: ETSI MANO Reference Architecture

4.1. Layered Network Model

Based on the ETSI NFV reference architecture, a layered network model is identified: Network Service layer, VNF layer, and Resource Layer. This model allows a separation of network relationships in different levels of abstraction. For example, network services can be queried at different levels of abstraction or we can map service paths in different layers (from an abstract to a more concrete layer).

In Figure 2, we have a network service with a set of interconnected VNFs. This network service topology is represented by the Network Service layer.

A VNF is typically divided into a set of virtual function components (VFCs) which comprise the VNF Layer. Each VFC is an application running within a single VM or container.

In case of the resource layer, we have virtual layer and physical layer. The virtual layer represents the virtual overlay network and the physical layer represents the substrate network. Virtualized infrastructures (e.g., VMs, virtual routers) are instantiated on a physical infrastructure.

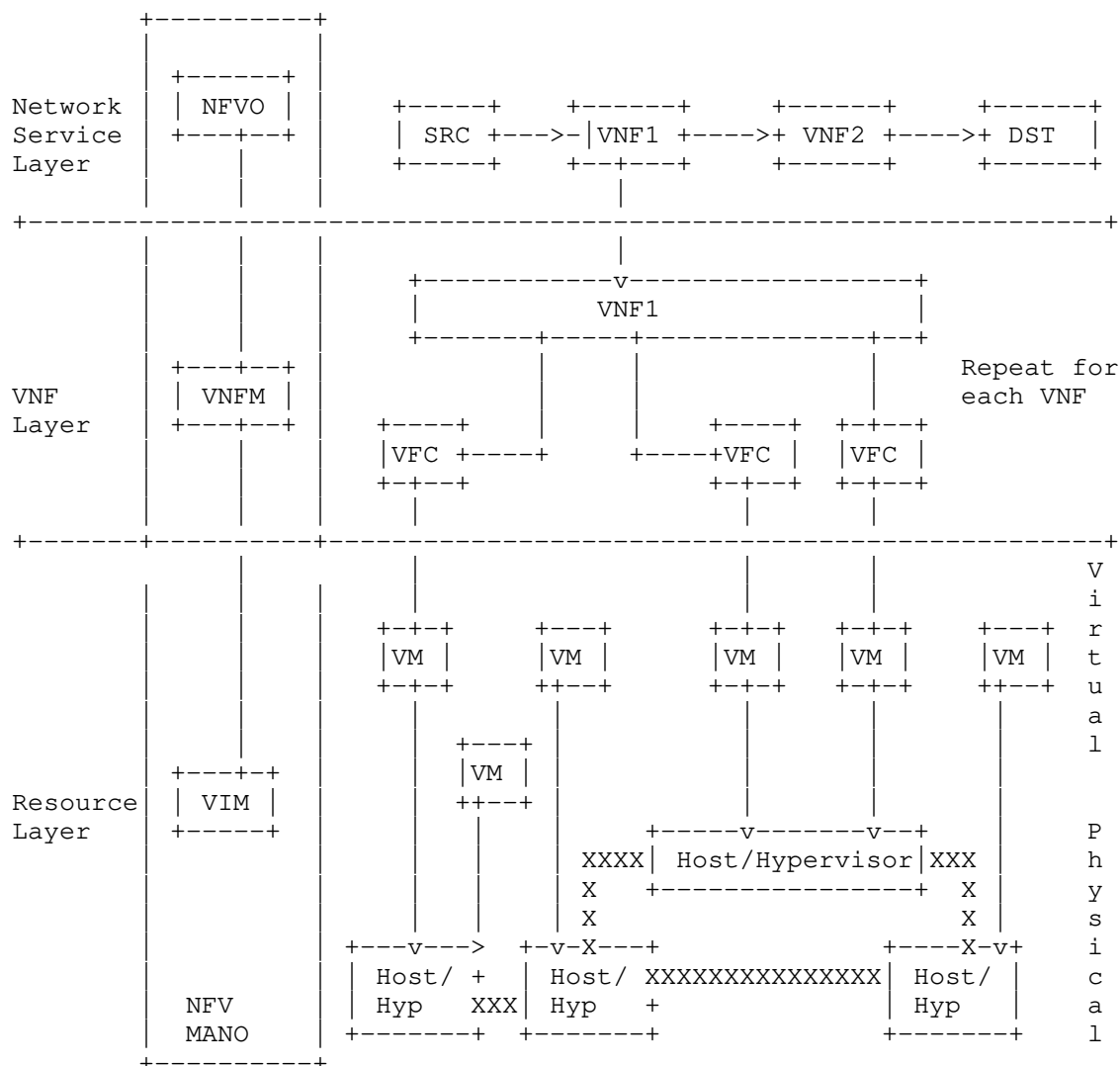


Figure 2: ETSI MANO Reference Architecture

5. ALTO Extension: E2E Network Service Requirements Representation

From the layered network model described in the previous section, we are considering an ALTO extension related to E2E Network Service requirements representation. An initial proposal has been presented in the ALTO-based Broker-assisted Mdo draft [DRAFT-ALTO-BROKER-MDO] where network applications (as ALTO clients) can specify a set of

basic E2E service requirements to an ALTO server in order to obtain candidate resources (domains) and candidate paths.

This initial E2E service requirement representation is inspired on the ETSI NFV MANO data model [ETSI-NFV-MAN001]. This model defines network services as a composition of network functions including the specification of deployment and operational requirements. Such specifications are captured in templates called Network Service Descriptor (NSD) and Virtual Network Function Descriptor (VNFD) that contain (relatively) static information used in the process of on-boarding network services and VNFs, respectively.

- o High level objects in a NSD include (among others) [ETSI-NFV-MAN001][OSM-DM]:
 - * **Constituent VNFs:** List of VNFDs that are part of the network service.
 - * **VNF Dependencies:** This describes dependencies between VNFs. For example, the order in which the VNFs inside a network service should be started.
 - * **Network service Connection Points:** Each network service has one or more external connection points (which act as endpoints) used to link two network services or to link external networks.
 - * **Virtual Links:** List of Virtual Link Descriptors (VLDs) that describe how VNFs (in the NSD) are connected.
- o High level objects in a VNFD include (among others) [ETSI-NFV-MAN001][OSM-DM]:
 - * **Constituent VDUs:** List of virtual deployment units (VDUs) in a specific VNF. Each VDU (also referred to VFC) describes the VM/Container capabilities (e.g., CPU, RAM, disks).
 - * **VDU Dependencies:** List of VDU dependencies used for determining the order of startup for VDUs.
 - * **VNF Connection Points:** List of external connection points used for connecting a VNF to other VNFs or to external networks.
 - * **Internal VLDs:** List of internal virtual links to connect various VDUs/VFCs.

6. IANA Considerations

This document includes no request to IANA.

7. Security Considerations

TBD.

8. Acknowledgments

This work is supported by the Innovation Center of Ericsson S.A., Brazil (grant agreement UNI.64). This document however does not necessary represent Ericsson's official viewpoint.

9. References

9.1. Normative References

- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC8459] Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", RFC 8459, DOI 10.17487/RFC8459, September 2018, <<https://www.rfc-editor.org/info/rfc8459>>.

9.2. Informative References

- [DRAFT-ALTO-BROKER-MDO] Perez, D. and C. Rothenberg, "ALTO-based Broker-assisted Multi-domain Orchestration", draft-lachosrothenberg-alto-brokermdo-03 (work in progress), March 2020.
- [DRAFT-ALTO-UNICORN] Xiang, Q., Zhang, J., Le, F., Yang, Y., and H. Newman, "Resource Orchestration for Multi-Domain, Exascale, Geo-Distributed Data Analytics", draft-xiang-alto-multidomain-analytics-03 (work in progress), March 2020.
- [DRAFT-HH-MDSFC] Li, G., Li, G., Xu, Q., Zhou, H., and B. Feng, "Hybrid Hierarchical Multi-Domain Service Function chaining", draft-li-sfc-hhsfc-08 (work in progress), March 2020.

[DRAFT-MD-SFC-ALTO]

Perez, D., Xiang, Q., Rothenberg, C., and Y. Yang, "Multi-domain Service Function Chaining with ALTO", draft-lachos-multi-domain-sfc-alto-01 (work in progress), March 2020.

[DRAFT-MD-VIRT]

Bernardos, C., Contreras, L., Vaishnavi, I., Szabo, R., Li, X., Paolucci, F., Sgambelluri, A., Martini, B., Valcarenghi, L., Landi, G., Andrushko, D., and A. Mourad, "Multi-domain Network Virtualization", draft-bernardos-nfvrg-multidomain-05 (work in progress), September 2018.

[ETSI-NFV-IFA028]

ETSI, "Report on architecture options to support multiple administrative domains V3.1.1", Jan 2018, <http://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/028/03.01.01_60/gr_NFV-IFA028v030101p.pdf>.

[ETSI-NFV-MAN001]

ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", Dec 2014, <https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf>.

[ETSI-NFV-WHITEPAPER]

ETSI, "Network Functions Virtualisation - White Paper 2", Oct 2013, <https://portal.etsi.org/NFV/NFV_White_Paper2.pdf>.

[H2020-5G-NORMA]

H2020, "5G-NORMA -- 5G Novel Radio Multiservice adaptive network Architecture", 2015, <<https://5gnorma.5g-ppp.eu/>>.

[H2020-5G-TRANSFORMER]

H2020, "5G-Transformer -- 5G Mobile Transport Platform for Vertical", 2017, <<http://5g-transformer.eu/>>.

[H2020-NECOS]

H2020 EU-Brazil, "NECOS -- Novel Enablers for Cloud Slicing", 2018, <<http://www.h2020-necos.eu/>>.

[H2020.5GEX]

Bernardos, C., Dugeon, O., Galis, A., Morris, D., Simon, C., and R. Szabo, "5G Exchange (5GEx)-- Multi-domain Orchestration for Software Defined Infrastructures", focus vol. 4, no.5, p.2, 2015.

- [MEF-SOE-MEF55] Metro Ethernet Forum, "Lifecycle Service Orchestration (LSO): Reference Architecture and Framework", Mar 2016, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_55.pdf>.
- [OSM-DM] Open Source MANO, "OSM - Data Model", 2016, <https://osm.etsi.org/wikipub/index.php/Release_0_Data_Model_Details>.
- [SFC-ORC] Sun, G., Li, Y., Liao, D., and V. Chang, "Service Function Chain Orchestration across Multiple domains: A Full Mesh Aggregation Approach", IEEE Transactions on Network and Service Management 1175--1191, 2018.
- [T-NOVA] FP7 project T-NOVA, "T-NOVA Project, Network Functions as a Service over Virtualised Infrastructures", 2014, <<http://www.t-nova.eu/>>.
- [VITAL] VITAL PROJECT H2020, "VITAL -- Virtualized hybrid satellite-Terrestrial systems for resilient and flexible future networks", 2015, <<http://www.ict-vital.eu/>>.
- [VNF-MOB] Patel, A., Vutukuru, M., and D. Krishnaswamy, "Mobility-aware VNF placement in the LTE EPC", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) 1--7, 2017.
- [VNF-PLAC] Slim, F., Guillemin, F., Gravey, A., and Y. Hadjadj-Aoul, "Towards a dynamic adaptive placement of virtual network functions under ONAP", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) 210--215, 2017.
- [WHITE-PAPER-5G] NetWorld2020, ETP, "5g: Challenges, research priorities, and recommendations", Journal: Joint White Paper September, 2014.

Authors' Addresses

Danny Alex Lachos Perez
University of Campinas
Av. Albert Einstein 400
Campinas, Sao Paulo 13083-970
Brazil

Email: dlachosp@dca.fee.unicamp.br
URI: <https://intrig.dca.fee.unicamp.br/danny-lachos/>

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein 400
Campinas, Sao Paulo 13083-970
Brazil

Email: chesteve@dca.fee.unicamp.br
URI: <https://intrig.dca.fee.unicamp.br/christian/>

ALTO WG
Internet-Draft
Intended status: Informational
Expires: September 12, 2019

Q. Xiang
Yale University
F. Le
IBM
Y. Yang
Yale University
March 11, 2019

ALTO for Multi-Domain Applications: A Review of Use Cases and Design
Requirements
draft-xiang-alto-multidomain-usecases-00.txt

Abstract

With the development of novel network technology, such as software defined networking and network function virtualization, many novel multi-domain applications, such as flexible interdomain routing, distributed, federated machine learning and multi-domain collaborative dataset transfer, have been deployed. These applications can benefit substantially from the ALTO protocol [RFC7285], through which the information of multiple networks can be provided to applications. This document first introduces several multi-domain applications and how they can benefit from ALTO. It then describes a generic framework for multi-domain applications to use ALTO to improve the performance, followed by a discussion on new requirements and challenges for ALTO to better support these applications.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Review of Multi-Domain Applications	3
3.1. Flexible Interdomain Routing	3
3.1.1. How flexible interdomain routing can benefit from ALTO?	3
3.1.2. Example	4
3.2. Resource Orchestration for Collaborative Data Sciences	4
3.2.1. How multi-domain resource orchestration can benefit from ALTO	4
3.2.2. Example	5
3.3. Federated Machine Learning	6
3.3.1. How federated machine learning can benefit from ALTO	6
3.3.2. Example	6
4. A Generic Framework	7
4.1. Workflow	8
5. Requirements of ALTO in Multi-Domain Applications	9
5.1. Design Requirements	9
5.2. Existing Efforts in the ALTO Working Group	10
6. Summary	10
7. References	10
7.1. Normative References	10
7.2. Informative References	11
Authors' Addresses	12

1. Introduction

The ALTO protocol [RFC7285] provides network information to applications so that applications can make network informed decisions to improve the performance. Not only traditional applications such peer-to-peer systems, many recent, novel multi-domain applications,

which orchestrate resources across multiple networks, can also benefit substantially from ALTO.

The goal of this document is to explore how ALTO can help improve the performance of novel multi-domain applications, what ALTO extension services are needed, and what are the corresponding requirements and challenges for designing such extensions. To this end, this document first give a case-by-case review of emerging multi-domain applications and how they can benefit from ALTO. It then describes a generic framework for multi-domain applications to use ALTO to improve the performance, followed by a discussion on the need of new ALTO services and the corresponding requirements and challenges for these extensions to better support these applications.

2. Requirements Language

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

3. Review of Multi-Domain Applications

3.1. Flexible Interdomain Routing

Flexible interdomain routing can be a highly valuable service for network providers. Specifically, an autonomous system (AS) providing such a service (the provider) allows other ASes (clients) to specify routing actions at the provider based on flexible matching conditions (e.g., match on TCP/IP 5-tuple). In this way, a client AS using the flexible interdomain routing service can offload access and traffic control to provider ASes, leading to a simpler client network configuration while giving the provider ASes additional business opportunities.

3.1.1. How flexible interdomain routing can benefit from ALTO?

ALTO provides provider ASes a standardized approach to expose its routing capability to client ASes. Traditional interdomain routing protocols such as BGP are not good options because they only expose the currently used routes, limiting client ASes' choices to specify flexible routes. In contrast, ALTO and its extensions provide interfaces for provider ASes to expose not only currently used routes, but also available yet unused routes, to client ASes so that they can have the flexibility to specify different routes for different data traffic.

3.1.2. Example

Consider the example in Figure 1. AS A is compromised and being used to send DDoS traffic to AS E. Without flexible interdomain routing, AS E can setup a firewall locally, but normal traffic from B to E will still be congested at C-D-E due to the existence of malicious traffic from A to E. If AS C provides flexible interdomain routing service, AS E can specify such a firewall at AS C to block DDoS traffic from A, and at the same time avoid the congestion of normal traffic from B to E.

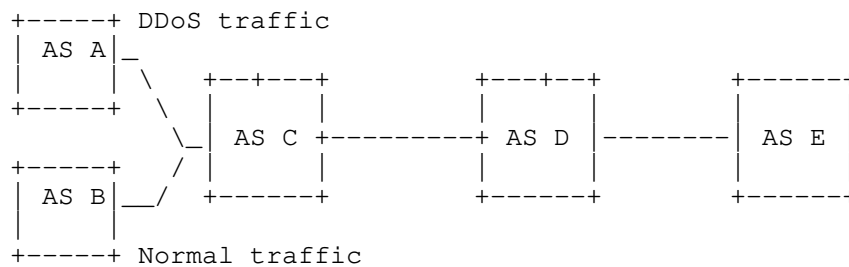


Figure 1: Flexible interdomain routing for DDoS mitigation.

3.2. Resource Orchestration for Collaborative Data Sciences

As the data volume increases exponentially over time, data analytics is transiting from a single-domain network to a multi-domain, geo-distributed network, where different member networks contribute various resources, e.g., computation, storage and networking resources, to collaboratively collect, share and analyze extremely large amounts of data. Such a paradigm calls for a unified resource orchestration framework to manage a large set of distributively-owned, heterogeneous resources, with the objective of efficient resource utilization, following the autonomy and privacy of different domains.

3.2.1. How multi-domain resource orchestration can benefit from ALTO

One key design challenge for multi-domain resource orchestration is its resource information model. Existing design options such as resource graph and ClassAds are inadequate because they cannot simultaneously (1) allow member networks to provide accurate information on different types of resource, (2) avoid the exposure of private information of member networks such as topology, and (3) allow data analytics jobs to accurately describe their requirements of different types of resources. In contrast, the section 7.1 of

Figure 2 discusses the advantages of choosing ALTO as the resource information model for multi-domain resource orchestration, and how ALTO can simultaneously satisfy the aforementioned design requirements.

3.2.2. Example

Consider an example of three member networks in Figure 2, where $s1$ and $s2$ are storage endpoints and $d1$ and $d2$ are computation endpoints. Assume a data analytics job is composed of two parallel tasks T1 and T2. T1 needs dataset X as input and T2 needs dataset Y as input.

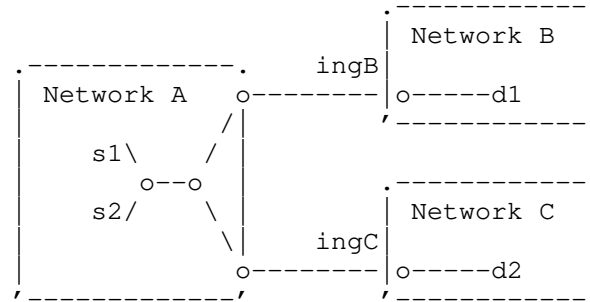


Figure 2: Multi-domain resource orchestration.

Using the ALTO endpoint property service, an ALTO client in the resource orchestrator can discover that $d1$ satisfies the computing requirements of T1 and $d2$ satisfies the computing requirements of T2. Hence there are only two candidate endpoint pairs: $(s1, d1)$ and $(s2, d2)$.

Afterwards, using the ALTO path vector extension, the ALTO client can retrieve the bandwidth sharing information of task T1 and T2, denoted as $x1$ and $x2$, respectively, as follows.

A: $x1 + x2 \leq 10\text{Mbps}$
 B: $x1 \leq 3\text{Mbps}$
 C: $x2 \leq 3\text{Mbps}$

With such information, the resource orchestrator can make the optimal resource orchestration decision to reserve 3 Mbps bandwidth for task T1, and 3 Mbps bandwidth for task T2.

3.3. Federated Machine Learning

Instead of moving large-scale datasets from multiple devices / networks to a centralized location for training, federated learning, is a distributed machine learning approach which enables training on distributed datasets residing on different autonomous systems (devices or networks). In this way, only updates on the training model need to be communicated between networks, leading to substantial reduction of networking resource consumption (e.g., saving bandwidth).

3.3.1. How federated machine learning can benefit from ALTO

Federated machine learning requires efficient scheduling algorithms to decide how networking resources should be allocated to transmit training model updates between different ASes. Similar as moving large-scale datasets between multiple ASes, moving updates of training model between ASes can also benefit from the availability of networking information, such as the AS-path and bandwidth sharing. ALTO provides a standardized approach for federated machine learning schedulers to retrieve such information from networks so that adaptive scheduling decisions can be made.

3.3.2. Example

Consider the example in Figure 3, where machine learning workers are located in AS A and D, while AS B and C are transit networks for data traffic transmitted between A and D. When AS A has a large, critical training model update to send to D. It first queries the ALTO servers at B and C for the endpoint cost (e.g., bandwidth) to transmit data from A to D. Suppose the ALTO server at AS B returns an endpoint cost of 10Mbps, while the ALTO server at AS C returns an endpoint cost of 100 Mbps. AS A can then use such information to make the optimal model update scheduling algorithm to send the training model update to AS D via AS C, instead of AS B.

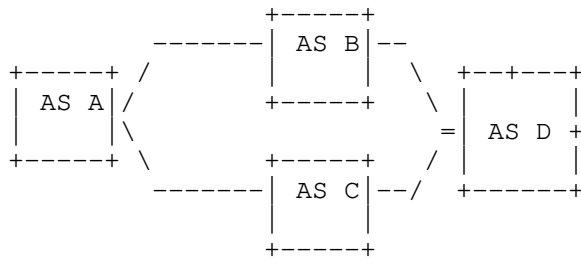


Figure 3: Federated machine learning.

4. A Generic Framework

After reviewing several important, novel multi-domain applications that can benefit substantially from ALTO, this document describes a generic framework for such applications to use ALTO to retrieve information from networks to improve their performance. The high-level architecture of this framework is given in Figure 4.

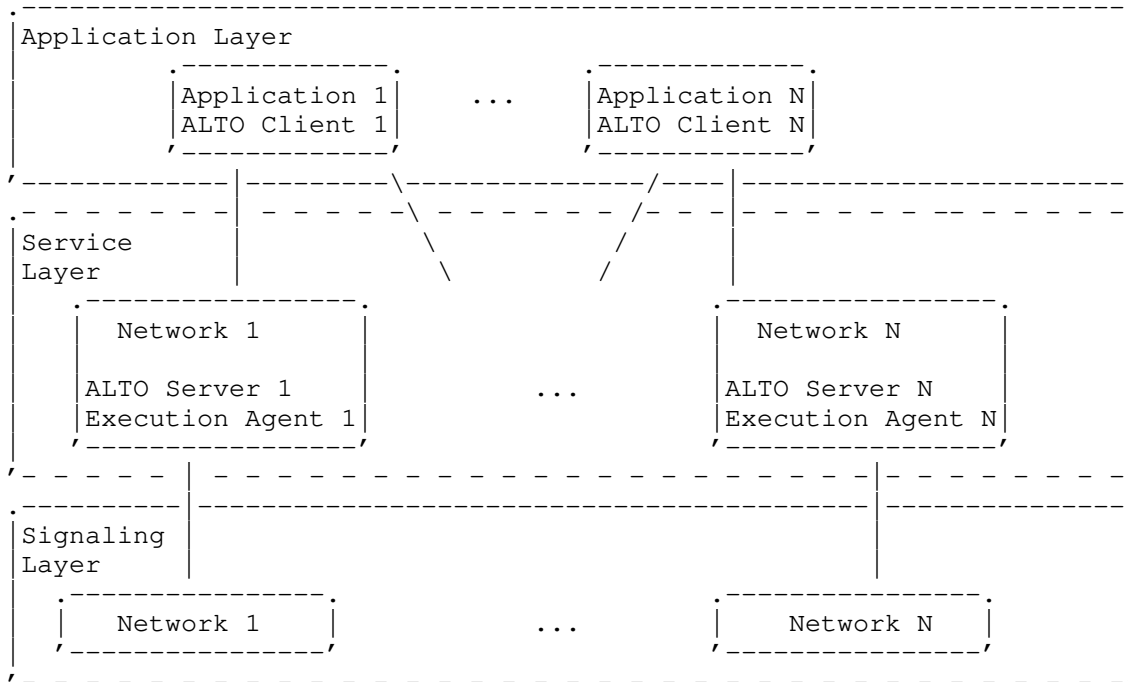


Figure 4: Generic framework of using ALTO in multi-domain applications.

The top layer of this framework is the application layer, in which each application deploy one or more ALTO clients to query for information provided by networks. The middle layer is the service layer. In this layer, each network deploys one more more ALTO servers to respond the queries sent by the ALTO clients from applications, and deploys one or more execution agents to respond to the applications' resource consumption actions. The bottom layer is the signaling layer, in which each network deploys interdomain protocols / systems, such as routing protocol BGP and resource reservation system OSCARS.

4.1. Workflow

The basic workflow of this framework is as follows.

- o An application identifies the networks whose resources (e.g., networking, computation and storage) it may want to consume, and invokes its ALTO clients to query the ALTO servers deployed in those networks for detailed resource information using base ALTO

protocol and its extension services (e.g., path vector, cost calendar and so on);

- o Upon receiving a query from an ALTO client, an ALTO server checks its local information, contacts the underlying signaling layer protocol / system of its residing network if local information is outdated, and returns the latest resource information to the querying ALTO client;
- o The applications uses the resource information collected from ALTO servers to make resource allocation decisions (e.g., route selection, resource reservation, etc.), and send such decisions to corresponding execute agents in the corresponding networks (e.g., the simple-reservation-interface of OSCARS).

5. Requirements of ALTO in Multi-Domain Applications

Using ALTO to improve the performance of recent novel multi-domain applications poses several new design requirements. This section discusses these requirements and briefly review existing efforts in the ALTO working group aiming to satisfy them.

5.1. Design Requirements

- o Exposing information of alternative resources. Current ALTO protocols and its extensions only provide information of currently used resources (e.g., currently used interdomain route). However, exposing information of alternative resources (e.g., available but not used interdomain routes) may provide the users of new multi-domain applications (e.g., flexible interdomain routing) more flexibility on choosing different resources, giving networks that provide such applications additional business opportunities.
- o Providing a unified, accurate representation of multiple types of resources. Current ALTO protocols and its extensions mainly focus on providing network information to applications, with the exception of endpoint property service. However, as new multi-domain applications often consume multiple types of resources across multiple networks, encoding such information accurately in a unified approach is crucial for deploying ALTO to improve such applications' performance.
- o Providing interfaces for more flexible query. Current ALTO protocol and its extensions allows applications to query resource information by specifying IP addresses of endpoints and simple filters. However, with the emerging of new networking architecture (e.g., software defined networking and network function virtualization) and the fine-grained resource requirement

of applications (e.g., link-disjoint paths and endpoint precedence), applications need a more flexible interface to specify queries of resource information.

5.2. Existing Efforts in the ALTO Working Group

Several documents have been submitted to the ALTO working group, with the aim to satisfy one or more of the design requirements discussed above. For example, [DRAFT-PV], [DRAFT-RSA], [DRAFT-UNICORN-INFO] and several other documents propose and apply the ALTO path vector extension to provide accurate networking resource information to support multi-domain resource orchestration. [DRAFT-NFCHAIN] proposes to use ALTO to support resource orchestration for multi-domain service function chaining, and proposes a new ALTO extension to retrieve AS path of network functions across different networks. [DRAFT-CONTEXT] proposes to extend cost information specified in RFC7285 by providing several possible cost values for the same cost metric where each value depends on qualitative criteria as opposed to quantitative criteria such as time. [DRAFT-UR] makes a proposal to use mathematical programming constraint as a generic representation of multiple resources. [DRAFT-FCS] proposes a flexible flow query extension service to allow applications to specify query entities based on flexible matching conditions (e.g., TCP/IP 5-tuple) instead of IP addresses only.

6. Summary

This document reviews several emerging multi-domain applications and how they can benefit from ALTO. It then describes a generic framework for multi-domain applications to use ALTO to improve the performance. In addition, several design requirements are discussed. Though different drafts in the working group have been trying to address one or more these design requirements, a systematic investigation of these issues is still missing. The authors of this document plan to perform such an investigation and make a unified design proposal in the next version of this document.

7. References

7.1. Normative References

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

7.2. Informative References

[DRAFT-CONTEXT]

Randriamasy, S., "ALTO Contextual Cost Values", 2017, <<https://datatracker.ietf.org/doc/draft-randriamasy-alto-cost-context/>>.

[DRAFT-FCS]

Zhang, J., Gao, K., Wang, J., Xiang, Q., and Y. Yang, "ALTO Extension: Flow-based Cost Query", 2017, <<https://datatracker.ietf.org/doc/draft-gao-alto-fcs/>>.

[DRAFT-NFCHAIN]

Perez, D. and C. Rothenberg, "ALTO-based Broker-assisted Multi-domain Orchestration", 2018, <<https://datatracker.ietf.org/doc/html/draft-lachosrothenberg-alto-brokermdo-01>>.

[DRAFT-PV]

Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Abstract Path Vector as a Cost Mode", 2015, <<https://tools.ietf.org/html/draft-yang-alto-path-vector-01>>.

[DRAFT-RSA]

Gao, K., Wang, X., Xiang, Q., Gu, C., Yang, Y., and G. Chen, "A Recommendation for Compressing ALTO Path Vectors", 2017, <<https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>>.

[DRAFT-UNICORN-INFO]

Xiang, Q., Newman, H., Bernstein, G., Du, H., Gao, K., Mughal, A., Balcas, J., Zhang, J., and Y. Yang, "Implementation and Deployment of A Resource Orchestration System for Multi-Domain Data Analytics", 2017, <<https://datatracker.ietf.org/doc/draft-xiang-alto-exascale-network-optimization/>>.

[DRAFT-UP]

Roome, W., Chen, S., Randriamasy, S., Yang, Y., and J. Zhang, "Unified Properties for the ALTO Protocol", 2015, <<https://datatracker.ietf.org/doc/draft-ietf-alto-unified-props-new/>>.

[DRAFT-UR]

Xiang, Q., Le, F., and Y. Yang, "ALTO Extension: Unified Resource Representation", 2018,
<<https://datatracker.ietf.org/doc/draft-xiang-alto-exascale-network-optimization/>>.

[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>>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/info/rfc8189>>.

Authors' Addresses

Qiao Xiang
Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

Franck Le
IBM
Thomas J. Watson Research Center
Yorktown Heights, NY
USA

Email: fle@us.ibm.com

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

Email: yry@cs.yale.edu

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

Q. Xiang
Yale University
J. Zhang
Tongji/Yale University
F. Le
IBM
Y. Yang
Yale University
July 13, 2020

ALTO Extension: Unified Resource Representation
draft-xiang-alto-unified-representation-03.txt

Abstract

The ALTO protocol [RFC7285] provides network information to applications so that applications can make network informed decisions to improve the performance. However, the base ALTO protocol only provides coarse-grained end-to-end metrics, which are insufficient to satisfy the demands of applications to solve more complex network optimization problems. The ALTO Path Vector extension [DRAFT-PV] has been introduced to allow ALTO clients to query information such as capacity regions for a given set of flows. However, the current design of this extension has a limited expressiveness. The goal of this document is to introduce a unified resource representation service as an extension of ALTO (ALTO-UR), which allows the ALTO clients to query and get the capacity regions of more complex resource information, such as Shared-Risk-Link-Group (SRLG), multi-path routing, multicast and on-demand routing, for a given set of flows.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Changes Since Version -02	3
4. Limitations of the ALTO Path Vector Extension	3
5. Overview of the Unified Representation Extension	5
5.1. Basic idea	5
5.2. New Cost Type to Encode Mathematical Programming Variables	6
5.2.1. Cost Mode: array	6
5.2.2. Cost Metric: variable-list	7
5.3. New Entity Domain to Provide Mathematical Programming Constraints	7
5.4. Multipart Response to Provide the Unified Representation	7
5.5. Designing New Interfaces for More Flexible Queries	7
6. Example	8
6.1. Protocol Extension	8
6.2. Workflow	9
6.3. Information Resource Directory Example	9
6.4. Cost Map Service Example	10
7. Use Case: Programmable End-to-End Multi-Domain Route Control	12
8. Ongoing Design Update: A Language-Based Resource Discovery Framework	12
9. Security and Privacy Considerations	13
10. References	13
10.1. Normative References	13
10.2. Informative References	13
Authors' Addresses	14

1. Introduction

As discussed in [DRAFT-PV], the "one-big-switch" abstraction used in the ALTO base protocol lacks the ability to support emerging use cases, such as inter-datacenter data transfers, because this abstraction cannot reveal the resource sharing, i.e., the capacity region, for a set of flows. The ALTO Path Vector extension addresses this insufficiency by using the path vector abstraction to express the capacity region in a set of linear inequalities. However, in an internal discussion with the leading persons of several important ALTO use cases, it is revealed that the expressiveness of the ALTO Path Vector extension is limited in three aspects:

- o It cannot provide compact encoding of the SRLG for a set of flows;
- o It assumes that each flow in the client's query will use a single-path route, and hence cannot encode the resource sharing for flows that are forwarded along multi-path routes or multicast flows;
- o It assumes that the route of each flow in the client's query is pre-computed, and hence cannot encode the resource sharing for flows that use on-demand routing, e.g., the path computation element (PCE) protocol.

To cope with these issues, this document introduces a new ALTO extension, the unified representation (ALTO-UR). This extension expands the linear inequality encoding of capacity regions used in ALTO-PV, to a generic, complete encoding, which uses mathematical programming constraints to represent the capacity regions for a set of flows.

2. Requirements Language

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

3. Changes Since Version -02

- o Add a discussion of an ongoing investigation of a language-based resource discovery framework utilizing the unified resource representation in Section 8.

4. Limitations of the ALTO Path Vector Extension

The limitations of the ALTO-PV extension are illustrated with the same dumbbell topology used in [DRAFT-PV]. Assume that the bandwidth of every link is 100 Mbps, and that the SRLG of each link is shown in

Figure 1. Consider an application overlay (e.g., a large data analytics system) which wants to schedule the traffic among a set of end host source-destination pairs, say eh1 → eh2 and eh1 → eh4.

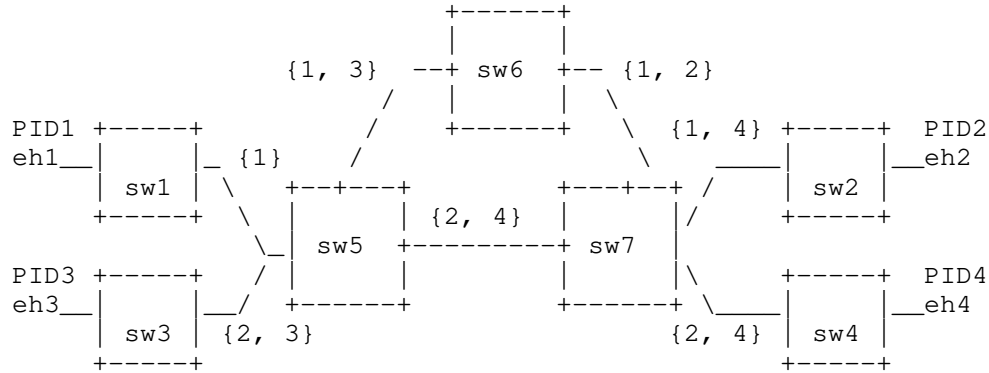


Figure 1: A Dumbbell Network Topology

- o Assume the application is only interested in the SRLG of both flows, not the bandwidth. The route of eh1 → eh2 is eh1 → sw1 → sw5 → sw6 → sw7 → sw2 → eh2, and the route of eh3 → eh4 is eh3 → sw1 → sw5 → sw7 → sw2 → eh4. The minimal yet accurate information returned to the application should be {2, 3, 4}, the SRLG of both flows, since flow 1 has a set of SRLG {1, 2, 3, 4} and flow 2 has a set of SRLG {2, 3, 4}. In contrast, in the current ALTO-PV service, the ALTO server needs to return the ane-path of each flow and the SRLG of every ane, where ane and ane-path are defined in [DRAFT-PV]. This response is redundant and causes unnecessary information exposure to the application, e.g., the information of flow has an SRLG 1 should not be returned to the application.
- o Assume the application is only interested in the bandwidth of both flows. The route of eh1 → eh2 is eh1 → sw1 → sw5 → sw6 → sw7 → sw2 → eh2, and the route of eh3 → eh4 is a multi-path route, i.e., {eh3 → sw1 → sw5 → sw7 → sw2 → eh4, eh3 → sw1 → sw5 → sw6 → sw7 → sw2 → eh4}, where each path would forward 50 percent of the traffic for eh3 → eh4. The ALTO-PV service cannot reveal the traffic split of the multi-path route for eh3 → eh4, or the bandwidth sharing for both flows on link sw5 → sw6.

- o Assume the network has a PCE sever, through which the application can reserve bandwidth for both flows. Before the application makes the reservation request, the application queries the ALTO server to get the bandwidth capacity region of both flows, which it wants to use to decide how much bandwidth to reserve for each flow. Suppose when the ALTO server receives a query, the network only has two precomputed routes for both flows: eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4. Through the ALTO-PV service, the application receives the information that the total bandwidth it can reserve for both flows cannot exceed 100 Mbps. However, one important feature of the PCE server is that it can compute the route for reservation request on-demand, and hence it can find routes with a larger bandwidth. In this example, if the application submits a request to reserve 100 Mbps bandwidth for each flow, the PCE server can compute two on-demand routes, i.e., eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2 and eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, and still return a success signal to the application. This shows that the ALTO-PV service cannot encode the capacity region for flows who use on-demand routing.

5. Overview of the Unified Representation Extension

Although different patches and extensions can be introduced to address the aforementioned insufficiencies of the ALTO-PV service, it is desirable to design a service that provides a generic solution that can encode different types of resource sharing for a set of flows. To this end, this document introduces the ALTO Unified Representation (ALTO-UR) service.

5.1. Basic idea

The basic idea of the ALTO-UR service is to use mathematical programming constraints to represent the capacity region for a set of flows. Different from linear inequalities used in the ALTO-PV service, mathematical programming constraints can represent a much wider range of resource information. To illustrate the expressiveness of mathematical programming constraints, we revisit the examples in Figure 1.

Assume the route of eh1 -> eh2 is eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and the route of eh3 -> eh4 is eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4. Denote the SRLG of flow eh1 -> eh2 as f1:SRLG, and that of flow eh3->eh4 as f2:SRLG. Then the SRLG of both flows can be represented as

$$f1:SRLG \text{ intersect } f2:SRLG = \{2, 3, 4\}$$

Assume the route of eh1 -> eh2 is eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and the route of eh3 -> eh4 is a multi-path route, i.e., {eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4}, where each path would forward 50 percent of the traffic for eh3 -> eh4. Denote the available bandwidth of eh1 -> eh2 as f1-bw and those of eh3 -> eh4 along two paths as f2-bw-p1 and f2-bw-p2. The bandwidth sharing of two flows can be represented as:

```
f1:bw <= 100 Mbps, for (sw1, sw5), (sw7, sw2)
f2:bw:p1 + f2:bw:p2 <= 100 Mbps, for (sw3, sw5), (sw7, sw4)
f1:bw + f2:bw:p1 <= 100 Mbps, for (sw5, sw6), (sw6, sw7)
f2:bw:p2 <= 100 Mbps, for (sw5, sw7)
f2:bw:p1 = f2:bw:p2
```

Assume the routes for both flows are computed on demand and each flow can only use a single path. For eh1 -> eh2, use f1-bw-p1 and f1-bw-p2 to represent the available bandwidth of routes eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2 and eh1 -> sw1 -> sw5 -> sw7 -> sw2 -> eh2, respectively. For eh3 -> eh4, use f2-bw-p1 and f2-bw-p2 to represent the available bandwidth of routes eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4 and eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, respectively. The bandwidth capacity region of both flows can be represented as:

```
f1:bw:p1 + f1:bw:p2 <= 100 Mbps, for (sw1, sw5), (sw7, sw2)
f2:bw:p1 + f2:bw:p2 <= 100 Mbps, for (sw3, sw5), (sw7, sw4)
f1:bw:p1 + f2:bw:p1 <= 100 Mbps, for (sw5, sw6), (sw6, sw7)
f1:bw:p2 + f2:bw:p2 <= 100 Mbps, for (sw5, sw7)
f1:bw:p1 = 0 or f1:bw:p2 = 0
f2:bw:p1 = 0 or f2:bw:p2 = 0
```

The next few subsections present the approaches adopted by the ALTO unified representation extension.

5.2. New Cost Type to Encode Mathematical Programming Variables

This document introduces the unified representation cost type, with the following cost mode and cost metric.

5.2.1. Cost Mode: array

The cost mode of the notation cost type is "array", which is defined in [DRAFT-PV]. The values are arrays of JSONValue. The specific type of each element in the array depends on the cost metric.

5.2.2. Cost Metric: variable-list

This document specifies a new cost metric called "variable-list". This cost metric indicates that the cost value is a list of variables that will be used in mathematical programming constraints.

5.3. New Entity Domain to Provide Mathematical Programming Constraints

This document adopts the property map defined in [DRAFT-UP] to encode the properties of abstract network elements. A new domain "cstr" (short for constraint) is registered in the property map. Each entity in the "cstr" domain has an identifier of an CSTR. Each CSTR has one property, which represents the semantics of this constraint, e.g., a "bw-cstr" property indicates that this constraint represents the bandwidth sharing among flows. This property is provided in information resources called "Property Map Resource" and "Filtered Property Map Resource". The "Filtered Property Map" resource which supports the "cstr" domain is used to encode the properties of cstr entities, and it is called a cstr Property Map in this document.

5.4. Multipart Response to Provide the Unified Representation

To ensure the consistency between the unified representation cost map and the corresponding CSTR property map, this document adopts the design of [DRAFT-PV] to allow a response to contain both the unified representation in a filtered cost map and the associated CSTR property map.

5.5. Designing New Interfaces for More Flexible Queries

Current ALTO protocol and its extensions allow applications to query resource information by specifying IP addresses of endpoints and simple filters. However, with the emerging of new networking architecture (e.g., software defined networking and network function virtualization) and the fine-grained resource requirement of applications (e.g., link-disjoint paths and endpoint precedence), applications need a more flexible interface to specify queries of resource information.

[DRAFT-FCS] proposes a flexible flow query extension service to allow applications to specify query entities based on flexible matching conditions (e.g., TCP/IP 5-tuple) instead of IP addresses only. However, it still does not allow an application to specify more fine-grained resource requirements. For example, a multi-domain service function chaining application may want to get the endpoint cost information of a chain of endpoints in the order of firewall, load balancer and data analyzer. The endpoint cost information of a chain

of endpoints not in this order is of no interest to the application and should not be returned to the application.

As such, this document makes an initial proposal to design new interfaces for application to express fine-grained requirements of resource in the query. In addition to allowing the ALTO client to specify endpoints using flexible matching conditions (e.g., TCP/IP 5 tuple), one key idea in this proposal is to use a resource filter design

Specifically, two types of resource properties are defined. The first type is value property, which are typical quality of services metrics for a given flow. Examples of the value property include the bandwidth, delay, loss rate and so on. The second type is set property. Examples of such a property include the forwarding and processing devices used by a flow (denoted as nodes), the physical links used by a flow (denoted as links) and the shared risk link group (SRLG) of all the devices and links used by a flow.

With these two types of resource properties, the atomic resource requirement predicates in resource filters are the comparison expressions on value properties and set properties. Resource requirement predicates can be composited using conjunction, disjunction and negation. The ALTO client can send resource query with composed resource requirement predicates to ALTO server, which only returns the information of resources that satisfy such predicates to ALTO client.

The details of this new interface will be fully specified in the next version of this document.

6. Example

6.1. Protocol Extension

To allow the ALTO client to query and receive the mathematical programming constraints for a set of flows, the Filtered Cost Map and Endpoint Cost Service of the ALTO protocol need to be extended. The current design adopted in this document uses a similar approach as the ALTO-PV extension does in [DRAFT-PV]: (1) extending the FilteredCostMapCapabilities object with a new member "property-map" and (2) using a multipart service to send both the unified representation cost map and the CSTR property map together. This design is illustrated in the next few subsections.

However, for the ALTO-UR service, this is still an early stage design. As a major next step for ALTO-UR service, other design options are being investigated with the aim of enabling better

modularity and extensibility, and the protocol extension will be updated in the next version accordingly.

6.2. Workflow

A typical workflow of an ALTO client using the unified representation extension is as follows:

1. Send a GET request for the whole Information Resource Directory.
2. Look for the resource of the Cost Map Service which contains the unified representation cost type and get the resource ID of the dependent cstr property map.
3. Check whether the capabilities of the property map includes the desired "prop-types".
4. Send a unified-representation request which accepts "multipart/related" media type following "application/alto-costmap+json" or "application/endpointcost+json"

6.3. Information Resource Directory Example

An example of an Information Resource Directory is as follows. In this example, filtered cost map "cost-map-ur" supports the unified-representation extension. The property map "propmap-cstr" support two properties, "bw-cstr" and "srlg-cstr", representing bandwidth constraint and SRLG constraint, respectively.

```

{
  "meta": {
    "cost-types": {
      "ur": {
        "cost-mode": "array",
        "cost-metric": "variable-list"
      }
    }
  }
  "resources": {
    "my-default-networkmap": {
      "uri" : "http://alto.example.com/networkmap",
      "media-type" : "application/alto-networkmap+json"
    }
    "cost-map-ur" : {
      "uri": "http://alto.example.com/costmap/ur",
      "media-type": "application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "capabilities": {
        "cost-type-names": [ "ur" ]
      },
      "property-map": "propmap-cstr",
      "uses": [ "my-default-networkmap" ]
    },
    "propmap-cstr" : {
      "uri": "http://alto.exmaple.com/propmap/cstr",
      "media-type": "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "capabilities": {
        "domain-types": [ "cstr" ],
        "prop-types": [ "bw-cstr", "srlg-cstr" ]
      }
    }
  }
}

```

6.4. Cost Map Service Example

The following is an example of the cost map service in the ALTO-UR extension. In the returned cost map, flow 1 has two bandwidth variables, f1:bw:p1 and f2:bw:p2, and one SRLG variable, f1:srlg. And flow 2 has one bandwidth variable, f2:bw, and one SRLG variable, f2:srlg. Four mathematical programming constraints are returned in the property map. The first three are bandwidth sharing constraints, and the fourth is an SRLG constraint.

```

POST /costmap/pv HTTP/1.1
Host: alto.example.com

```

```
Accept: multipart/related, application/alto-costmap+json,
application/alto-propmap+json, application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "variable-list"
  },
  "pids": {
    "srcs": [ "PID1" ],
    "dsts": [ "PID2", "PID3" ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=42
```

```
--42
```

```
Content-Type: application/alto-costmap+json
```

```
{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "default-network-map",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ],
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "variable-list"
    },
  },

  "cost-map": {
    "PID1": {
      "PID2": [ "f1:bw:p1", "bw:p2", "f1:srlg" ]
      "PID3": [ "f2:bw:p1", "f2:srlg" ]
    }
  }
}
```

```
--42
```

```
Content-Type: application/alto-propmap+json
```

```
{
```

```

"property-map": {
  "cstr:001": { "bw-cstr": "[0][0] add [0][1] leq 100"},
  "cstr:002": { "bw-cstr": "[0][0] eq [0][1]"},
  "cstr:003": { "bw-cstr": "[1][0] add [0][0] leq 100"},
  "cstr:004": { "srlg-cstr": "[0][2] intersect [1][1] eq {2, 3, 4}"},
}
}

```

7. Use Case: Programmable End-to-End Multi-Domain Route Control

This section describes a multi-domain use case that can benefit from a unified resource presentation extension of ALTO. Specifically, a novel multi-domain network programming framework is recently proposed to achieve programmable, end-to-end interdomain route control. Specifically, in this framework, each autonomous system (AS) deploys an ALTO server to provide a programming abstraction. Collectively, these ALTO servers provide the client a single, abstract, programmable network spanning multiple individual networks. Unlike existing SDN abstractions (e.g., OpenFlow and P4), it includes a built-in layer to extract and learn the interactions of interdomain policies of individual networks (e.g., route selection preference), providing a unified abstraction.

In particular, given a destination IP prefix p specified by a client, each autonomous system (AS) exposes its routing information base (RIB), i.e., all available routes it has to reach p , and the corresponding price for the client to use each route, by deploying an ALTO server. A client queries the available routes and the corresponding prices at different ASes. With the retrieved information, it then iteratively selects the best route based on its internal objective function and budget, and interacts with different ASes to check if the selected route is policy compliant, i.e., whether all ASes along this route will export the preceding segment to its upstream neighbor. After finding the best policy compliant route, the client then interacts with ASes of the route in a backward order along the route to setup the AS path. A blackbox based optimization algorithm has been developed to allow the client to swiftly find the best policy compliant route. A paper describing this framework has been accepted by IEEE INFOCOM 2020.

8. Ongoing Design Update: A Language-Based Resource Discovery Framework

To further investigate the feasibility, challenges and benefits of using a unified resource representation to facilitate application-network integration, the authors of this draft are investigating a language-based resource discovery framework. This framework consists of three key design points. First, this framework uses generic

mathematical programming constraints as a unified, compact representation of network information, as proposed in this draft. Second, it utilizes the equivalence between relational algebra and first order logic to provide a SQL-style language for application / ALTO client to express their intents on discovering resources in the network. Third, the framework develops a compiler to translate an application's resource discovery intent into a constraint programming problem with a set of logical constraints, whose feasible solutions correspond to qualified resources for application. In addition, an algorithm is also designed to improve the network's efficiency in finding qualified resources. Details of this framework and its early evaluation results have been accepted by ACM SIGCOMM NAI 2020 Workshop, with title "'Towards Deep Network & Application Integration: Possibilities, Challenges, and Research Directions'".

9. Security and Privacy Considerations

The unified representation extension may expose more private information to applications than the ALTO base protocol does. However, as shown in the motivating example of providing SRLG information for a set of flows, this extension has the capability of exposing less private information than the ALTO-PV extension does, while having a better expressiveness on providing fine-grained resource information to applications. A systematic study on the security and privacy issues of the ALTO-UR extension is one of the major next steps.

10. References

10.1. Normative References

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

10.2. Informative References

[DRAFT-FCS]
Zhang, J., Gao, K., Wang, J., Xiang, Q., and Y. Yang, "ALTO Extension: Flow-based Cost Query", 2017, <<https://datatracker.ietf.org/doc/draft-gao-alto-fcs/>>.

[DRAFT-NFCHAIN]
Perez, D. and C. Rothenberg, "ALTO-based Broker-assisted Multi-domain Orchestration", 2018, <<https://datatracker.ietf.org/doc/html/draft-lachosrothenberg-alto-brokermdo-01>>.

[DRAFT-PV]

Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Abstract Path Vector as a Cost Mode", 2015, <<https://tools.ietf.org/html/draft-yang-alto-path-vector-01>>.

[DRAFT-RSA]

Gao, K., Wang, X., Xiang, Q., Gu, C., Yang, Y., and G. Chen, "A Recommendation for Compressing ALTO Path Vectors", 2017, <<https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>>.

[DRAFT-UP]

Roome, W., Chen, S., Randriamasy, S., Yang, Y., and J. Zhang, "Unified Properties for the ALTO Protocol", 2015, <<https://datatracker.ietf.org/doc/draft-ietf-alto-unified-props-new/>>.

[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>>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/info/rfc8189>>.

Authors' Addresses

Qiao Xiang
Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

J. Jensen Zhang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: jingxuan.zhang@yale.edu

Franck Le
IBM
Thomas J. Watson Research Center
Yorktown Heights, NY
USA

Email: fle@us.ibm.com

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

Email: yry@cs.yale.edu

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2022

J. Zhang
Tongji University
Y.R. Yang
Yale University
12 July 2021

Multiple ALTO Resources Query Using Multipart Message
draft-zhang-alto-multipart-06

Abstract

Many ALTO use cases involve multiple ALTO information resources like different network maps, cost maps and property maps to achieve their own specific goals. To make the ALTO client query them one by one is not only inefficient but also error-prone. The inconsistent responses can be performed because of the unstable communication environment, and finally conduct the unexpected traffic optimization. Further more, some ALTO information resources may have correlation, which means one's input parameters may depends on another one's response. To address those issues, some advanced query schema is required. This document proposes an ALTO extension to support the multiple ALTO resources query in the single request using the HTTP multipart message and the existing JSON query languages.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 13 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminologies	4
2.1. Resource Query Entry	4
2.2. Resource Response Entry	4
2.3. Resource Response Entry Body	4
3. Use Cases	4
3.1. Simple Batch Query	4
3.2. Properties Constrained Query	5
3.3. Path Vector Query	5
4. Requirements	5
5. Design Space of Multipart Resource in ALTO	6
6. Multipart Query Resource	7
6.1. Media Type	7
6.2. HTTP Method	7
6.3. Capabilities	7
6.4. Accept Input Parameters	7
6.5. Uses	8
6.6. Response	8
7. Protocol Errors	8
7.1. Partial Error	9
7.2. Entire Error	10
8. Incremental Update Integration	10
9. Examples	10
9.1. IRD Example	10
9.2. Example 1: Simple Batch Query	12
9.3. Example 2: Properties Constrained Query	14
9.4. Example 3: Path Vector Query	17
9.5. Example 4: Incremental Updates	18
10. Compatibility	20
10.1. Compatibility with Legacy ALTO Clients/Servers	20
10.2. Compatibility with Existing Protocol Extensions	20
11. Misc Considerations	21

11.1.	Return ALTO Information Resources over HTTP/2	21
11.2.	Support Incremental Update	21
11.3.	Anonymous Resources	21
12.	Security Considerations	21
13.	IANA Considerations	22
13.1.	application/alto-* Media Types	22
14.	Acknowledgements	23
15.	References	23
15.1.	Normative References	23
15.2.	Informative References	24
Appendix A.	Figures	25
Authors'	Addresses	25

1. Introduction

Application-Layer Traffic Optimization (ALTO) protocol [RFC7285] and its extensions already define several types of information resources, like Network Map, Cost Map and Property Map, to expose useful network information to applications. However, many applications do not only use a single information resource to perform their traffic optimization. Retrieving multiple ALTO information resources is very common in many ALTO use cases.

Using the current ALTO framework defined in [RFC7285], the ALTO client can only query multiple ALTO information resources one by one. It is not only inefficient but also error-prone. Because of the network delay between different requests and the frequent change of ALTO information resources, the responses received by the ALTO client may be inconsistent.

Furthermore, some ALTO information resources have known dependencies, which means the ALTO client may need one's response to decide another one's query input parameters.

To be summarized, we need the multipart query service for three reasons:

- * Clients may want to query multiple ALTO information resources in a single request to reduce the network latency.
- * Clients may want to query multiple ALTO resources consistently, which means the server should guarantee the responses of all resources are generated at the same time.
- * Some use cases need to query multiple ALTO resources with a joint relationship.

This document defines a new ALTO service for: (1) querying multiple ALTO resources in a single request/response, and (2) supporting general-purpose JSON query languages to resolve the relational query.

2. Terminologies

Besides the terms defined in [RFC2045], [RFC2046], [RFC2387], and [RFC7285], this document also uses the following additional terms:

2.1. Resource Query Entry

A Resource Query Entry indicates the ResourceQuery object (see Section 6.4) for an individual resource in the accept input parameters of the Multipart Query resource.

2.2. Resource Response Entry

A Resource Response Entry indicates the entry of an individual part of the multipart response message, including the MIME headers and the body content.

2.3. Resource Response Entry Body

A Resource Response Entry Body indicates the body content of a Resource Response Entry.

3. Use Cases

The following use cases can benefit from the multipart query service.

3.1. Simple Batch Query

The simplest use case is to query a batch of ALTO resources in a single request.

Although the ALTO client can perform ALTO requests for multiple times, it is not only inefficient but also inconsistent.

For example, the ALTO server provides a network map resource A and a dependent cost map resource B. Both resources may change frequently. Assume the ALTO client queries the network map first, and it gets the revision A1. When the client queries the cost map, the network map may be already changed from A1 to A2, and the client receives cost map B2 which depends on A2 not A1. So the responded cost map B2 is not consistent with the previous network map A1.

This case requires the ALTO server to provide a way for the ALTO client to query multiple ALTO resources in a single transaction.

3.2. Properties Constrained Query

Beyond the simple batch query, there are also some another use cases requiring a new service for relational query. For example, Some clients may need to query an endpoint property map first, and find endpoints with some properties fitting some conditions. And then they query the endpoint cost of these endpoints.

In this case, the endpoint cost query depends on the result of the property map query. Although the ALTO client can cache the whole property map in its local storage, it is still not efficient and may conduct the consistency issue if the property map changes frequently. So it requires a new service to provide multiple dependent resources efficiently and consistently.

A general multipart query service benefits the ALTO client in two aspects:

- * It allows the ALTO client to specify the boolean test to reduce the transmission of the useless data from the ALTO server.
- * It compounds multiple ALTO information resources in a single response to reduce the communication times. Thus, the transmission latency can be reduced.

3.3. Path Vector Query

Another use case requiring the multiple resource query is the relational query between the on-demand generated resources. A straightforward example is the path vector query demonstrated in [I-D.ietf-alto-path-vector].

[I-D.ietf-alto-path-vector] introduces an extension of ALTO to provide path vector information by cost map and unified property map [I-D.ietf-alto-unified-props-new]. The client using path vector extension will usually query cost map and a dynamically generated property map sequentially. It is even hard to cache the full data of resources, because both the cost map and the property map are on-demand generated by the query input here. Thus, the only way to reduce the time consumption is to compound the two resources.

4. Requirements

From the use cases described in Section 3, there are three additional requirements for ALTO protocol:

MPQ-Req1: The ALTO protocol SHOULD allow the client to query

multiple ALTO resources in a single request, and return the result in a single response.

It is the basic requirement to provide the query for the compound resources. Even simple cases can benefit if this requirement is realized.

MPQ-Req2: The ALTO protocol SHOULD provide general filter schema for any ALTO resources.

Current filter schema in ALTO protocol only supports the simple boolean test of numerical comparison. And the boolean filtered query is only supported by the cost map and the endpoint cost resource. It is not enough for the general cases. Even simple property map may require more general filter schema.

MPQ-Req3: The ALTO protocol SHOULD support relational query for multiple joint resources.

Some ALTO resources are relational and cannot be used individually. The path vector query is such an example. In these use cases, the support of relational query for multiple joint resources is very helpful.

5. Design Space of Multipart Resource in ALTO

This document discusses the solution of how to apply "multipart/*" (see [RFC2045] and [RFC2046]) response to the ALTO protocol.

There are three cases applying Multipart response to ALTO:

Multipart Request and Multipart Response: In this case, an ALTO client can start a single request using Multipart encoding to query a batch of resources.

Single Request and Fixed-Layout Multipart Response: In this case, an ALTO server receives a non-Multipart request, e.g., the filtered costmap request or the endpoint cost request, but returns a Multipart response. The ALTO server MUST export the layout of the Multipart response in the IRD. A special example can be found in [I-D.ietf-alto-path-vector].

Single Request and Flexible-Layout Multipart Response: This case

extends the previous case to allows the Multipart response with flexible layout. The ALTO server receives the unified query request and generate the layout of the response based on the request. The ALTO client can even use general-purpose query language like XQuery [W3CXQUERY] and JSONiq [JSONIQ] for general query process and relational joint query.

The application about Multipart request to the single object response is out of the scope of this document.

6. Multipart Query Resource

6.1. Media Type

"multipart/related" [RFC2387].

6.2. HTTP Method

An ALTO Multipart Query resource is requested using the HTTP POST method.

6.3. Capabilities

The capabilities are defined by an object of type MultipartQueryCapabilities:

```
object {  
  JSONString query-langs<0..*>;  
} MultipartQueryCapabilities;
```

where "query-langs" is an array of JSONString to indicate which query languages are supported by this resource.

6.4. Accept Input Parameters

The input parameters for a Multipart Query 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-multipartquery+alto", which is a JSON object of type ReqMultipartQuery:

```
object {  
  ResourceQuery  resources<1..*>;  
  [JsonString    query-lang;]  
} ReqMultipartQuery;  
  
object {  
  JsonString      resource-id;  
  [JsonValue      input;]  
} ResourceQuery;
```

with fields:

resources: List of ResourceQuery objects for which resources are to be queried and how to query them. Each ResourceQuery object MUST include the "resource-id" field to indicate which resource is to be queried. If the queried resource requires the POST method, the "input" field MUST be specified. The value of the "input" field MUST be either a JSONString or a JSONObject. When its value is a JSONObject, its format MUST be as the accept input parameters of its resource. When its value is a JSONString, it MUST be a program written in the query language specified by the "query-lang" field.

query-lang: Optional. The value of the "query-lang" field MUST be one of values in the "query-langs" capability. If this field is not specified in the request, the ALTO client SHOULD NOT use any query language in the "input" field.

6.5. Uses

An array with the resource ID(s) of resource(s) which this multipart query resource can compound. The used resource can be any available ALTO resources except for the multipart query resource. If the "uses" field is not specified, all the available ALTO resources can be queried except for the multipart query resource.

6.6. Response

The response of multipart query resource is a multipart message. Each part of this multipart message is the response of a queried resource in the request.

7. Protocol Errors

At the top level, the request of ALTO Multipart Query resource may conduct two types of errors: Partial Error and Entire Error.

7.1. Partial Error

The Partial Error only occurs when the value of the "resource-id" field or the "input" field is invalid.

When the Partial Error occurs, the ALTO server MUST still return the response in the media type "multipart/related". For the resource query entry with an error, the ALTO server MUST specify the "Content-Type" of its resource response entry as "application/alto-error+json", and include the ALTO error message in its resource response entry body. For the resource query entry without any error, the ALTO server MUST perform its query request normally.

The value of the "resource-id" field is invalid when this resource id is not defined by the Information Resource Directory. In this case, the ALTO server MUST return the E_INVALID_FIELD_VALUE error.

The validation of each "input" field of the multipart query input parameters depends on the queried resource:

- * If the "input" field of the multipart query input parameters is neither a JSONObject nor a JSONString, the ALTO server SHOULD return the E_INVALID_FIELD_TYPE error, unless a future protocol extension supports the non-JSONObject input parameters.
- * If the "input" field of the multipart query input parameters is a JSONObject, the ALTO server MUST validate the value using its queried resource and return the corresponding error if it has.
- * If the "input" field of the multipart query input parameters is a JSONString:
 - If the "query-lang" is not specified, the ALTO server MUST return the E_INVALID_FIELD_TYPE error.
 - If the "query-lang" is specified, the ALTO server MUST execute this JSONString as a program written in the "query-lang". If the execution failed, the ALTO server MUST return the E_INVALID_FIELD_VALUE error. If the execution succeeds but the result fails to pass the validation of the queried resource, the ALTO server MUST return the E_INVALID_FIELD_VALUE error and attach the error message returned by the queried resource into the "message" field of the ALTO error message.

The syntax error is an Entire Error.

7.2. Entire Error

Any other invalid request will conduct the Entire Error.

When the Entire Error occurs, the ALTO server MUST return the error response in the media type "application/alto-error+json" instead of "multipart/related". The process of the Entire Error is as defined in Section 8.5 of [RFC7285].

8. Incremental Update Integration

This document defines a compatible incremental update process for Multipart Query resource with [RFC8895].

An ALTO server's IRD can export an Update Stream service defined in [RFC8895] including the Resource ID of a Multipart Query resource in the "uses" field. When an ALTO client subscribes the incremental update for this Multipart Query resource, the ALTO server sends the whole Multipart response message back at the first data update message. Then the ALTO server subscribes all nodes in this multipart resource tree automatically. Once data updated later, the ALTO server publishes the update for each node individually.

9. Examples

9.1. IRD Example

Assume the root IRD is like the following:

```
{
  "meta": {
    "path-vector": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    },
    "num-routingcost": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    },
    "num-hopcount": {
      "cost-mode": "numerical",
      "cost-metric": "hopcount"
    }
  },
  "resources": {
    "my-default-networkmap": {
      "uri": "https://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    }
  }
}
```

```
    },
    "my-default-costmap": {
      "uri": "https://alto.example.com/costmap",
      "media-type": "application/alto-costmap+json",
      "capabilities": {
        "cost-type-names": [ "num-routingcost" ]
      },
      "uses": [ "my-default-networkmap" ]
    },
    "my-filtered-costmap": {
      "uri": "https://alto.example.com/costmap/filtered",
      "media-type": "application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "capabilities": {
        "cost-type-names": [ "num-hopcount" ]
      },
      "uses": [ "my-default-networkmap" ]
    },
    "endpoint-path-vector": {
      "uri": "https://alto.exmaple.com/endpointcost",
      "media-type": "application/alto-endpointcost+json",
      "accepts": "application/alto-endpointcostparams+json",
      "capabilities": {
        "cost-constraints": true,
        "cost-type-names": [ "path-vector" ],
      },
      "property-map": "propmap-availbw"
    },
    "propmap-availbw-delay": {
      "uri": "https://alto.exmaple.com/propmap/availbw",
      "media-type": "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "uses": [ "endpoint-path-vector" ],
      "capabilities": {
        "mappings": {
          "endpoint-path-vector.ane": [ "availbw" ]
        }
      }
    },
    "propmap-location": {
      "uri": "https://alto.exmaple.com/propmap/location",
      "media-type": "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "uses": [ "my-default-networkmap" ],
      "capabilities": {
        "mappings": {
          "my-default-networkmap.pid": [ "country", "state" ]
        }
      }
    }
  },
  "propmap-location": {
    "uri": "https://alto.exmaple.com/propmap/location",
    "media-type": "application/alto-propmap+json",
    "accepts": "application/alto-propmapparams+json",
    "uses": [ "my-default-networkmap" ],
    "capabilities": {
      "mappings": {
        "my-default-networkmap.pid": [ "country", "state" ]
      }
    }
  }
}
```

```
    }
  },
  "multipart-query": {
    "uri": "https://alto.example.com/multipart",
    "media-type": "multipart/related",
    "accepts": "application/alto-multipartquery+json",
    "capabilities": {
      "query-langs": [ "xquery", "jsoniq" ]
    }
  },
  "update-multipart-query": {
    "uri": "https://alto.example.com/updates/multipart",
    "media-type": "text/event-stream",
    "uses": [ "multipart-query" ],
    "accepts": "application/alto-updatestreamparams+json",
    "capabilities": {
      "incremental-change-media-types": {
        "multipart-query": "application/merge-patch+json"
      },
      "support-stream-control": true
    }
  }
}
```

9.2. Example 1: Simple Batch Query

```
POST /multipart HTTP/1.1
Host: alto.example.com
Accept: multipart/related, application/alto-error+json
Content-Length: TBD
Content-Type: application/alto-multipartquery+json
```

```
{
  "resources": [
    {
      "resource-id": "my-default-networkmap"
    },
    {
      "resource-id": "my-default-costmap"
    }
  ]
}
```

```
HTTP/1.1 200 OK
Content-Lenght: TBD
Content-Type: multipart/related; boundary=simple-batch-query

--simple-batch-query
Content-ID: my-default-networkmap
Content-Type: application/alto-networkmap+json
```

```
{
  "meta": {
    "vtag": {
      "resource-id": "my-default-networkmap",
      "tag": "75ed013b3cb58f896e839582504f622838ce670f"
    }
  },
  "network-map": {
    "PID1" : {
      "ipv4" : [
        "192.0.2.0/24",
        "198.51.100.0/25"
      ]
    },
    "PID2" : {
      "ipv4" : [
        "198.51.100.128/25"
      ]
    },
    "PID3" : {
      "ipv4" : [
        "0.0.0.0/0"
      ],
      "ipv6" : [
        "::/0"
      ]
    }
  }
}
```

```
--simple-batch-query
Content-ID: my-default-costmap
Content-Type: application/alto-costmap+json
```

```
{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ]
  }
}
```

```
    }  
  },  
  "cost-type": {  
    "cost-mode": "numerical",  
    "cost-metric": "routingcost"  
  }  
},  
"cost-map": {  
  "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },  
  "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },  
  "PID3": { "PID1": 20, "PID2": 15 }  
}  
}  
--simple-batch-query
```

9.3. Example 2: Properties Constrained Query

NOTE: In this example, we use the `"\"` block to express the raw string with unescaped characters like `"\n"` and `"\""`. It is not a valid HTTP body, but only used to better present. When the request is sent to the ALTO server, the `"\"` block should be escaped.

```
POST /multipart HTTP/1.1
Host: alto.example.com
Accept: multipart/related, application/alto-error+json
Content-Lenght: TBD
Content-Type: application/alto-multipartquery+json

{
  "query-lang": "jsoniq",
  "resources": [
    {
      "resource-id": "propmap-location"
    },
    {
      "resource-id": "my-default-costmap",
      "input": `
        let $propmap := collection("propmap-location")
                          .("property-map")
        return {
          "cost-type": {
            "cost-mode": "numerical",
            "cost-metric": "hopcount"
          },
          "pids": {
            "srcs": [
              for $pid in keys($propmap)
              where $propmap.$pid.country eq "US"
              return substring-after($pid, "PID:")
            ],
            "dsts": [
              for $pid in keys($propmap)
              where $propmap.$pid.country eq "CA"
              return substring-after($pid, "PID:")
            ]
          }
        }
      `
    }
  ]
}

HTTP/1.1 200 OK
Content-Lenght: TBD
Content-Type: multipart/related; boundary=prop-const-query

--prop-const-query
Content-ID: propmap-location
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "pid:PID1": {
      "country": "US",
      "state": "CA"
    },
    "pid:PID2": {
      "country": "US",
      "state": "CT"
    },
    "pid:PID3": {
      "country": "CA",
      "state": "QC"
    },
    "pid:PID4": {
      "country": "CA",
      "state": "NT"
    },
    "pid:PID5": {
      "country": "FR"
    }
  }
}

--prop-const-query
Content-ID: my-default-costmap
Content-Type: application/alto-costmap+json

{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "hopcount"
    }
  },
  "cost-map": {
    "PID1": {
      "PID3": 5,
      "PID4": 7
    },
    "PID2": {
      "PID3": 8,
      "PID4": 4
    }
  }
}

--prop-const-query
```

9.4. Example 3: Path Vector Query

```

POST /multipart HTTP/1.1
Host: alto.example.com
Accept: multipart/related, application/alto-error+json
Content-Lenght: TBD
Content-Type: application/alto-multipartquery+json

{
  "query-lang": "jsoniq",
  "resources": [
    {
      "resource-id": "endpoint-path-vector",
      "input": {
        "cost-type": {
          "cost-mode": "array",
          "cost-metric": "ane-path"
        },
        "endpoints": {
          "srcs": [ "ipv4:192.0.2.2" ],
          "dsts": [ "ipv4:192.0.2.89",
                    "ipv4:203.0.113.45" ]
        }
      }
    },
    {
      "resource-id": "propmap-availbw",
      "input": `
        let $propmap := collection("endpiont-path-vector")
                           .("endpoint-cost-map")

        return {
          "entities": [
            distinct-values(flatten(
              for $src in keys($propmap)
              let $dsts := $propmap.$src
              return flatten(
                for $dst in keys($dsts)
                return $dsts.$dst
              )
            )
          ],
          "properties": [ "availbw" ]
        }
      `
    }
  ]
}

```



```
HTTP/1.1 200 OK
Content-Length: TBD
Content-Type: multipart/related; boundary=path-vector-query

--path-vector-query
Content-ID: endpoint-path-vector
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89": [ "ane:L001", "ane:L003", "ane:L004" ],
      "ipv4:203.0.113.45": [ "ane:L001", "ane:L004", "ane:L005" ],
      "ipv6:2001:db8::10": [ "ane:L001", "ane:L005", "ane:L007" ]
    }
  }
}
```

```
--path-vector-query
Content-ID: propmap-availbw
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "ane:L001": { "availbw": 50 },
    "ane:L003": { "availbw": 48 },
    "ane:L004": { "availbw": 55 },
    "ane:L005": { "availbw": 60 },
    "ane:L007": { "availbw": 35 }
  }
}

--path-vector-query
```

9.5. Example 4: Incremental Updates

```
POST /updates/multipart
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Length: TBD
Content-Type: application/alto-updatestreamparams+json
```

```
{
  "add": {
    "multipart-query": {
      "resource-id": "multipart-query",
      "input": {
        "resources": [
          {
            "resource-id": "my-default-networkmap"
          },
          {
            "resource-id": "my-default-costmap"
          }
        ]
      }
    }
  }
}
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data:  "https://alto.example.com/updates/streams/1414"}

event: multipart/related;boundary=example-update,multipart-query
data: --simple-batch-query
data: Content-ID: netmap
data: Content-Type: application/alto-networkmap+json
data:
data: { ... data (object) of network map ... }
data:
data: --simple-batch-query
data: Content-ID: costmap
data: Content-Type: application/alto-costmap+json
data:
data: { .. data (object) of cost map ... }
data: --simple-batch-query

      (pause)

event: application/merge-patch+json,multipart-query.netmap
data: { ... merge patch for updates of network map ... }

event: application/merge-patch+json,multipart-query.costmap
data: { ... merge patch for updates of cost map ... }
```

10. Compatibility

10.1. Compatibility with Legacy ALTO Clients/Servers

The multipart query service is a new ALTO service using the new media type. So the legacy ALTO client cannot identify this service from the IRD of the ALTO server supporting it. And the legacy ALTO server also cannot interpret the request of a multipart query service sent by the ALTO client.

10.2. Compatibility with Existing Protocol Extensions

The multipart query service can use any ALTO resources exchanging JSON data in request/response mechanism. So all the known ALTO extensions like ALTO Calendar [RFC8896], Multi-Cost [RFC8189] and the Path Vector [I-D.ietf-alto-path-vector] extension, which does not change the request/response mechanism, are compatible with the multipart query service.

11. Misc Considerations

11.1. Return ALTO Information Resources over HTTP/2

HTTP/2 [RFC7540] provides new features like streams and multiplexing that can essentially reduce the web interface communication latency. As the growth of deployment of HTTP/2, it is valuable to consider how to transit the ALTO information resources over HTTP/2.

The multipart query service defined in this document includes two parts: the multiple-resource query schema and the multipart response schema.

By leveraging HTTP/2 multiplexing in the scope of this document, the multipart response schema can be replaced with sending multiple HTTP/2 streams using HTTP/2 server push. Each stream only needs to include a single ALTO information resource. The benefit is that the Server can include additional meta information in the HTTP HEADERS frame of each stream. And the Client can parse each ALTO information resource in parallel. However, the multiple-resource query schema is required to be reused to keep the consistent request semantics. If the Client wants to receive consistent query results, it should send a single multiple-resource query request over the HTTP/2 stream to enforce the Server to generate the response to different ALTO information resources based on the same database snapshot.

11.2. Support Incremental Update

According to Section 5.2 of [RFC8895], the update stream service can use concatenation of the substream-id, the '.' separator and a Content-ID to identify the update to each part of a multipart response. Thus, each part of a multiple-resource query response MUST include a Content-ID, if the Server provides an update stream service defined in [RFC8895] for this multiple-resource query service.

11.3. Anonymous Resources

Some use cases may need the server to generate "anonymous" ALTO resources for the on-demand information. The "anonymous" ALTO resources usually cannot appear alone but need to bind with some "non-anonymous" ALTO resources.

12. Security Considerations

Allow the ALTO clients to upload the query language script may not be safe. The code injection and many potential attacks can be conducted. The security issue should be discussed and considered.

To avoid the attacks like the code injection, this document recommends the following approaches:

Database Isolation: Some clients may attempt to access the secure database inside the server. Isolate the data into the different databases can reduce the risk of the information leak.

Application Container Isolation: Attackers may inject harmful code into the input query programs to attempt to access the system control. To avoid this, each query process is recommended to be isolated using the application container.

Resource Limit: Even attackers cannot get the permission to crack the data or the system, they can still inject some heavy-load programs to consume the server resources. Thus, limiting the memory usage and execution time of each query process is highly recommended.

13. IANA Considerations

13.1. application/alto-* Media Types

This document registers an additional ALTO media type, listed in Table 1.

Type	Subtype	Specification
application	alto-multipartquery+json	Section 6.4

Table 1: Additional ALTO Media Type.

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 [RFC8259].

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).

14. Acknowledgements

15. References

15.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <<https://www.rfc-editor.org/info/rfc2045>>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <<https://www.rfc-editor.org/info/rfc2046>>.

- [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>>.
- [RFC2387] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387, DOI 10.17487/RFC2387, August 1998, <<https://www.rfc-editor.org/info/rfc2387>>.
- [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>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <<https://www.rfc-editor.org/info/rfc8895>>.

15.2. Informative References

- [I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y. R., and J. J. Zhang, "ALTO Extension: Path Vector", Work in Progress, Internet-Draft, draft-ietf-alto-path-vector-14, 22 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-alto-path-vector-14.txt>>.
- [I-D.ietf-alto-unified-props-new]
Roome, W., Randriamasy, S., Yang, Y. R., Zhang, J. J., and K. Gao, "ALTO Extension: Entity Property Maps", Work in Progress, Internet-Draft, draft-ietf-alto-unified-props-new-17, 16 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-alto-unified-props-new-17.txt>>.
- [JSONIQ] Robie, J., Fourny, G., Brantner, M., Florescu, D., Westmann, T., and M. Zaharioudakis, "JSONiq - the SQL of NoSQL 1.0", JSONiq , 2015.

- [RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/info/rfc8189>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8896] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", RFC 8896, DOI 10.17487/RFC8896, November 2020, <<https://www.rfc-editor.org/info/rfc8896>>.
- [W3CXQUERY] Robie, J., Chamberlin, D., Dyck, M., and J. Snelson, "XQuery 3.0: An XML query language", W3C Recommendation, W3C, 2014.

Appendix A. Figures

TODO: Put additional figures here if we have.

Authors' Addresses

Jingxuan Jensen Zhang
Tongji University
4800 Cao'An Hwy
Shanghai
201804
China

Email: jingxuan.n.zhang@gmail.com

Yang Richard Yang
Yale University
51 Prospect Street
New Haven, CT
United States of America

Email: yry@cs.yale.edu