

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: March 11, 2013

R. Alimi, Ed.
Google
R. Penno, Ed.
Cisco Systems
Y. Yang, Ed.
Yale University
Sept 7, 2012

ALTO Protocol
draft-ietf-alto-protocol-13.txt

Abstract

Networking applications today already have access to a great amount of Inter-Provider network topology information. For example, views of the Internet routing table are easily available at looking glass servers and entirely practical to be downloaded by clients. What is missing is knowledge of the underlying network topology from the ISP or Content Provider (henceforth referred as Provider) point of view. In other words, what a Provider prefers in terms of traffic optimization -- and a way to distribute it.

The ALTO Service provides network information (e.g., basic network location structure, preferences of network paths) with the goal of modifying network resource consumption patterns while maintaining or improving application performance. The basic information of ALTO is based on abstract maps of a network. These maps provide a simplified view, yet enough information about a network for applications to effectively utilize them. Additional services are built on top the maps.

This document describes a protocol implementing the ALTO Service. Although the ALTO service would primarily be provided by the network operator (e.g., an ISP), content providers and third parties could also operate this service. Applications that could use this service are those that have a choice in connection endpoints. Examples of such applications are peer-to-peer (P2P) and content delivery networks.

Requirements Language

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on March 11, 2013.

Copyright Notice

Copyright (c) 2012 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	6
1.1.	Background and Problem Statement	6
1.2.	Design History and Merged Proposals	6
1.3.	Solution Benefits	7
1.3.1.	Service Providers	7
1.3.2.	Applications	7
2.	Architecture	7
2.1.	Terminology	8
2.1.1.	Endpoint	8
2.1.2.	Endpoint Address	8
2.1.3.	ASN	8
2.1.4.	Network Location	8
2.1.5.	ALTO Information	8
2.1.6.	ALTO Information Base	9
2.2.	ALTO Service and Protocol Scope	9
2.3.	ALTO Information Reuse and Redistribution	10
3.	Service Framework	11
3.1.	ALTO Information Services	12
3.1.1.	Map Service	12
3.1.2.	Map Filtering Service	12
3.1.3.	Endpoint Property Service	12
3.1.4.	Endpoint Cost Service	12
4.	Network Map	12
4.1.	PID	13
4.2.	Endpoint Addresses	14
4.2.1.	IP Addresses	14
4.3.	Example Network Map	14
5.	Cost Map	15
5.1.	Cost Attributes	16
5.1.1.	Cost Type	16
5.1.2.	Cost Mode	16
5.2.	Cost Map Structure	17
5.3.	Network Map and Cost Map Dependency	18
6.	Protocol Specification	18
6.1.	Overall Design	18
6.2.	Notation	19
6.3.	Basic Operation	19
6.3.1.	Discovering Information Resources	19
6.3.2.	Requesting Information Resources	19
6.3.3.	Response	20
6.3.4.	Client Behavior	21
6.3.5.	Authentication and Encryption	21
6.3.6.	HTTP Cookies	21
6.3.7.	Parsing	21
6.4.	Information Resource	21
6.4.1.	Capabilities	22

6.4.2.	Input Parameters Media Type	22
6.4.3.	Media Type	22
6.4.4.	Encoding	22
6.5.	ALTO Errors	23
6.5.1.	Media Type	24
6.5.2.	Resource Format	24
6.5.3.	Error Codes	24
6.5.4.	Overload Conditions and Server Unavailability	25
6.6.	ALTO Types	25
6.6.1.	PID Name	26
6.6.2.	Version Tag	26
6.6.3.	Endpoints	26
6.6.4.	Cost Mode	28
6.6.5.	Cost Type	29
6.6.6.	Endpoint Property	29
6.7.	Information Resource Directory	29
6.7.1.	Media Type	30
6.7.2.	Encoding	30
6.7.3.	Example	31
6.7.4.	Usage Considerations	34
6.8.	Information Resources	35
6.8.1.	Map Service	35
6.8.2.	Map Filtering Service	40
6.8.3.	Endpoint Property Service	46
6.8.4.	Endpoint Cost Service	49
7.	Use Cases	53
7.1.	ALTO Client Embedded in P2P Tracker	54
7.2.	ALTO Client Embedded in P2P Client: Numerical Costs	55
7.3.	ALTO Client Embedded in P2P Client: Ranking	56
8.	Discussions	57
8.1.	Discovery	57
8.2.	Hosts with Multiple Endpoint Addresses	58
8.3.	Network Address Translation Considerations	58
8.4.	Endpoint and Path Properties	59
9.	IANA Considerations	59
9.1.	application/alto-* Media Types	59
9.2.	ALTO Cost Type Registry	61
9.3.	ALTO Endpoint Property Registry	62
9.4.	ALTO Address Type Registry	63
10.	Security Considerations	64
10.1.	Privacy Considerations for ISPs	64
10.2.	ALTO Clients	65
10.3.	Authentication, Integrity Protection, and Encryption	65
10.4.	ALTO Information Redistribution	66
10.5.	Denial of Service	66
10.6.	ALTO Server Access Control	67
11.	Manageability Considerations	67
11.1.	Operations	67

11.1.1.	Installation and Initial Setup	67
11.1.2.	Migration Path	68
11.1.3.	Requirements on Other Protocols and Functional Components	68
11.1.4.	Impact on Network Operation	68
11.1.5.	Verifying Correct Operation	69
11.2.	Management	69
11.2.1.	Management Interoperability	69
11.2.2.	Management Information	69
11.2.3.	Fault Management	69
11.2.4.	Configuration Management	69
11.2.5.	Accounting Management	70
11.2.6.	Performance Management	70
11.2.7.	Security Management	70
12.	References	71
12.1.	Normative References	71
12.2.	Informative References	72
Appendix A.	Acknowledgments	74
Appendix B.	Authors	75
	Authors' Addresses	75

1. Introduction

1.1. Background and Problem Statement

Today, network information available to applications is mostly from the view of endhosts. There is no clear mechanism to convey information about the network infrastructure (e.g., preferences or topological properties) to applications, forcing applications to make approximations using data sources such as BGP Looking Glass or their own measurements, which can be misleading or inaccurate. On the other hand, modern network applications can be adaptive, with the potential to become more network-efficient (e.g., reduce network resource consumption) and achieve better application performance (e.g., accelerated download rate), by leveraging better network-provided information.

The ALTO Service provides a simple mechanism to convey network information to applications. Its objective is to provide basic, abstract but useful network information to applications. The mechanism includes abstractions to achieve concise, flexible network information expression.

The goal of this document is to specify a simple and unified protocol that meets the ALTO requirements [[I-D.ietf-alto-reqs](#)] while providing a migration path for Internet Service Providers (ISP), Content Providers, and clients that have deployed protocols with similar intentions (see [Section 1.2](#)).

The ALTO Protocol design uses a REST-ful design with the goal of leveraging current HTTP [[RFC2616](#)] implementations and infrastructure. The REST-ful design supports flexible deployment strategies and provides extensibility. ALTO requests and responses are encoded with JSON [[RFC4627](#)].

1.2. Design History and Merged Proposals

The protocol specified here consists of contributions from

- o P4P [[I-D.p4p-framework](#)], [[P4P-SIGCOMM08](#)], [[I-D.wang-alto-p4p-specification](#)];
- o ALTO Info-Export [[I-D.shalunov-alto-infoexport](#)];
- o Query/Response [[I-D.saumitra-alto-queryresponse](#)], [[I-D.saumitra-alto-multi-ps](#)];
- o ATTP [[ATTP](#)];

- o Proxidor [[I-D.akonjang-alto-proxidor](#)].

See [Appendix A](#) for a list of people that have contributed significantly to this effort and the projects and proposals listed above.

[1.3.](#) Solution Benefits

At a high level, the ALTO Service allows a Service Provider (e.g., an ISP) to publish information about network locations and costs between them at configurable granularities.

The ALTO Service offers many benefits to both end-users (consumers of the service) and Internet Service Providers (providers of the service).

[1.3.1.](#) Service Providers

The ALTO Service enables Service Providers to influence the peer or resource selection process in distributed applications in order to increase locality of traffic, improve user-experience, amongst others. It also helps ISPs to efficiently manage traffic that traverses more expensive links such as transit and backup links, thus allowing a better provisioning of the networking infrastructure.

[1.3.2.](#) Applications

Applications that use the ALTO Service can benefit in multiple ways. For example, they may no longer need to infer topology information, and some applications can reduce reliance on measuring path performance metrics themselves. They can take advantage of the ISP's knowledge to avoid bottlenecks and boost performance.

An example type of application is a Peer-to-Peer overlay where peer selection can be improved by including ALTO information in the selection process.

[2.](#) Architecture

Two key design objectives of the ALTO Protocol are simplicity and extensibility. At the same time, it introduces additional techniques to address potential scalability and privacy issues. This section first introduces the terminology, and then defines the ALTO architecture and the ALTO Protocol's place in the overall architecture.

2.1. Terminology

We use the following terms defined in [[RFC5693](#)]: Application, Overlay Network, Peer, Resource, Resource Identifier, Resource Provider, Resource Consumer, Resource Directory, Transport Address, Host Location Attribute, ALTO Service, ALTO Server, ALTO Client, ALTO Query, ALTO Reply, ALTO Transaction, Local Traffic, Peering Traffic, Transit Traffic.

We also use the following additional terms: Endpoint Address, Autonomous System Number (ASN), and Network Location.

2.1.1. Endpoint

An endpoint is an entity capable of communicating (sending and/or receiving messages) on a network.

An Endpoint is typically either a Resource Provider or Resource Consumer.

2.1.2. Endpoint Address

An Endpoint Address represents the communication address of an endpoint. An Endpoint Address can be network-attachment based (IP address) or network-attachment agnostic. Common forms of Endpoint Addresses include IP address, MAC address, overlay ID, and phone number.

Each Endpoint Address has an associated Address Type, which indicates both its syntax and semantics.

2.1.3. ASN

An Autonomous System Number.

2.1.4. Network Location

Network Location is a generic term denoting a single endpoint or group of endpoints.

2.1.5. ALTO Information

ALTO Information is a generic term referring to the network information sent by an ALTO Server.

2.1.6. ALTO Information Base

Internal representation of the ALTO Information maintained by the ALTO Server. Note that the structure of this internal representation is not defined by this document.

2.2. ALTO Service and Protocol Scope

An ALTO Server conveys the network information from the perspective of a network region; the ALTO Server presents its "my-Internet View" of the network region. In particular, an ALTO Server defines network Endpoints (and aggregations thereof) and generic costs amongst them from the network region's own perspective. A network region in this context can be an Autonomous System, an ISP, or perhaps a smaller region or set of ISPs; the details depend on the ALTO deployment scenario and ALTO service discovery mechanism.

To better understand the ALTO Service and the role of the ALTO Protocol, we show in Figure 1 the overall ALTO system architecture. In this architecture, an ALTO Server prepares ALTO Information; an ALTO Client uses ALTO Service Discovery to identify an appropriate ALTO Server; and the ALTO Client requests available ALTO Information from the ALTO Server using the ALTO Protocol.

The ALTO Information provided by the ALTO Server can be updated dynamically based on network conditions, or can be seen as a policy which is updated at a larger time-scale.

More specifically, the ALTO Information provided by an ALTO Server may be influenced (at the operator's discretion) by other systems. The ALTO Server aggregates information from multiple systems to provide an abstract, unified, useful network view to applications. Examples of other systems include (but are not limited to) static network configuration databases, dynamic network information, routing protocols, provisioning policies, and interfaces to outside parties. These components are shown in the figure for completeness but are outside the scope of this specification. Recall that while ALTO may convey dynamic network information, it is not intended to replace near-real-time congestion protocols.

It may also be possible for ALTO Servers to exchange network information with other ALTO Servers (either within the same administrative domain or another administrative domain with the consent of both parties) in order to adjust exported ALTO Information. Such a protocol is also outside the scope of this specification.

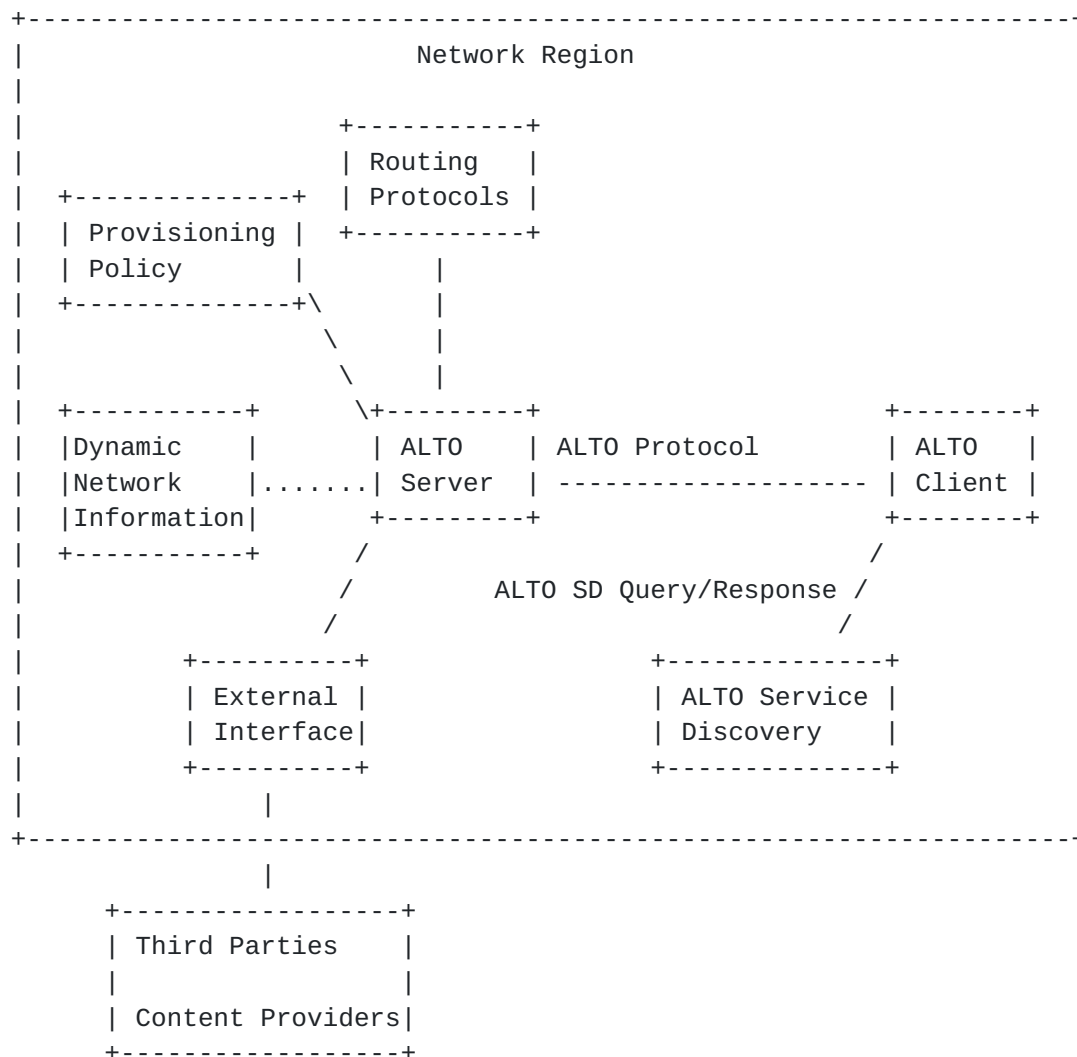


Figure 1: Basic ALTO Architecture.

2.3. ALTO Information Reuse and Redistribution

ALTO information may be useful to a large number of applications and users. At the same time, distributing ALTO information must be efficient and not become a bottleneck.

Beyond integration with existing HTTP caching infrastructure, ALTO information may also be cached or redistributed using application-dependent mechanisms, such as P2P DHTs or P2P file-sharing. This document does not define particular mechanisms for such redistribution. See [[I-D.gu-alto-redistribution](#)] for further discussion.

Additional protocol mechanisms (e.g., expiration times and digital signatures for returned ALTO information) are left for extension

documents.

If caching or redistribution is used, the response message may be returned from another (possibly third-party) entity.

3. Service Framework

The ALTO Protocol uses a simple extensible framework to convey network information. In the general framework, the ALTO protocol will convey properties on both Network Locations and the paths between Network Locations.

In this document, we focus on a particular Endpoint property to denote the location of an endpoint, and provider-defined costs for paths between pairs of Network Locations.

The ALTO Protocol is built on a common transport protocol, messaging structure and encoding, and transaction model. The protocol is subdivided into services of related functionality. The Map Service provides the core ALTO information to clients. Other ALTO Information services provide additional functionalities. There are three such services defined in this document: the Map Filtering Service, Endpoint Property Service, and Endpoint Cost Service. Additional services may be defined in companion documents. Functionalities offered in different services may overlap (e.g., the Map Service and Map Filtering Service).

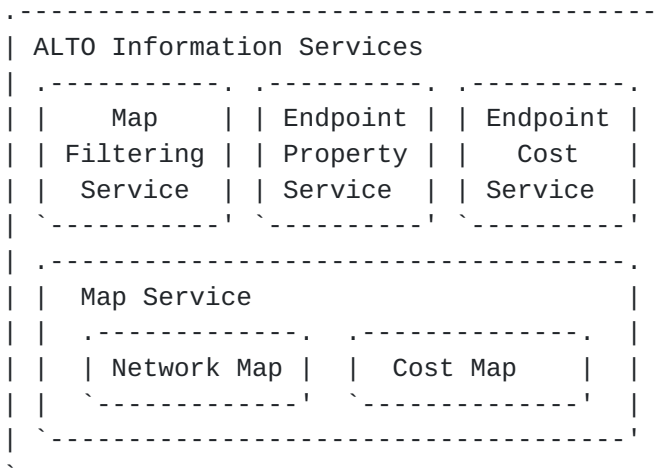


Figure 2: ALTO Service Framework

[3.1.](#) ALTO Information Services

Multiple, distinct services are defined to allow ALTO Clients to query ALTO Information from an ALTO Server. The ALTO Server internally maintains an ALTO Information Base that encodes the network provider's preferences. The ALTO Information Base encodes the Network Locations defined by the ALTO Server (and their corresponding properties), as well as the provider-defined costs between pairs of Network Locations.

[3.1.1.](#) Map Service

The Map Service provides batch information to ALTO Clients in the form of Network Map and Cost Map. The Network Map (See [Section 4](#)) provides the full set of Network Location groupings defined by the ALTO Server and the Endpoints contained with each grouping. The Cost Map (see [Section 5](#)) provides costs between the defined groupings.

These two maps can be thought of (and implemented as) as simple files with appropriate encoding provided by the ALTO Server.

[3.1.2.](#) Map Filtering Service

Resource constrained ALTO Clients may benefit from query results being filtered at the ALTO Server. This avoids an ALTO Client spending network bandwidth or CPU collecting results and performing client-side filtering. The Map Filtering Service allows ALTO Clients to query for the ALTO Server Network Map and Cost Map based on additional parameters.

[3.1.3.](#) Endpoint Property Service

This service allows ALTO Clients to look up properties for individual Endpoints. An example endpoint property is its Network Location (its grouping defined by the ALTO Server) or connectivity type (e.g., ADSL, Cable, or FTTH).

[3.1.4.](#) Endpoint Cost Service

Some ALTO Clients may also benefit from querying for costs and rankings based on Endpoints. The Endpoint Cost Service allows an ALTO Server to return either numerical costs or ordinal costs (rankings) directly amongst Endpoints.

[4.](#) Network Map

The Network Location Endpoint Property allows an ALTO Server to group

endpoints together to indicate their proximity. The resulting set of groupings is called the ALTO Network Map.

In reality, many endpoints are very close to one another in terms of network connectivity, for example, endpoints on the same site of an enterprise. By treating a group of endpoints together as a single entity in ALTO, we can achieve much greater scalability without losing critical information.

The definition of proximity varies depending on the granularity of the ALTO information configured by the provider. In one deployment, endpoints on the same subnet may be considered close; while in another deployment, endpoints connected to the same PoP may be considered close.

As used in this document, the Network Map refers to the syntax and semantics of the information distributed by the ALTO Server. This document does not discuss the internal representation of this data structure within the ALTO Server.

4.1. PID

Each group of Endpoints is identified by a provider-defined Network Location identifier called a PID. A PID is a US-ASCII string of type PIDName (see [Section 6.6.1](#)) and its associated set of Endpoint Addresses. There can be many different ways of grouping the endpoints and assigning PIDs.

A PID is an identifier that provides an indirect and network-agnostic way to specify an aggregation of network endpoints that may be treated similarly, based on network topology, type, or other properties. For example, a PID may be defined by the ALTO service provider to denote a subnet, a set of subnets, a metropolitan area, a PoP, an autonomous system, or a set of autonomous systems. Aggregation of endpoints into PIDs can indicate proximity and can improve scalability. In particular, network preferences (costs) may be specified between PIDs, allowing cost information to be more compactly represented and updated at a faster time scale than the network aggregations themselves.

Using PIDs, the Network Map may also be used to communicate simple preferences with only minimal information from the Cost Map. For example, an ISP may prefer that endpoints associated with the same PoP (Point-of-Presence) in a P2P application communicate locally instead of communicating with endpoints in other PoPs. The ISP may aggregate endhosts within a PoP into a single PID in the Network Map. The Cost Map may be encoded to indicate that Network Locations within the same PID are preferred; for example, $\text{cost}(\text{PID}_i, \text{PID}_i) == c^*$ and

$\text{cost}(\text{PID}_i, \text{PID}_j) > c^*$ for $i \neq j$. [Section 5](#) provides further details about Cost Map structure.

[4.2.](#) Endpoint Addresses

Communicating endpoints may have many types of addresses, such as IP addresses, MAC addresses, or overlay IDs. The current specification only considers IP addresses.

[4.2.1.](#) IP Addresses

The endpoints aggregated into a PID are denoted by a list of IP prefixes. When either an ALTO Client or ALTO Server needs to determine which PID in a Network Map contains a particular IP address, longest-prefix matching **MUST** be used.

A Network Map **MUST** define a PID for each possible address in the IP address space for all of the address types contained in the map. A **RECOMMENDED** way to satisfy this property is to define a PID with the shortest enclosing prefix of the addresses provided in the map. For a map with full IPv4 reachability, this would mean including the 0.0.0.0/0 prefix in a PID; for full IPv6 reachability, this would be the ::/0 prefix.

Each endpoint **MUST** map into exactly one PID. Since longest-prefix matching is used to map an endpoint to a PID, this can be accomplished by ensuring that no two PIDs contain an identical IP prefix.

[4.3.](#) Example Network Map

Figure 3 illustrates an example Network Map. PIDs are used to identify network-agnostic aggregations.

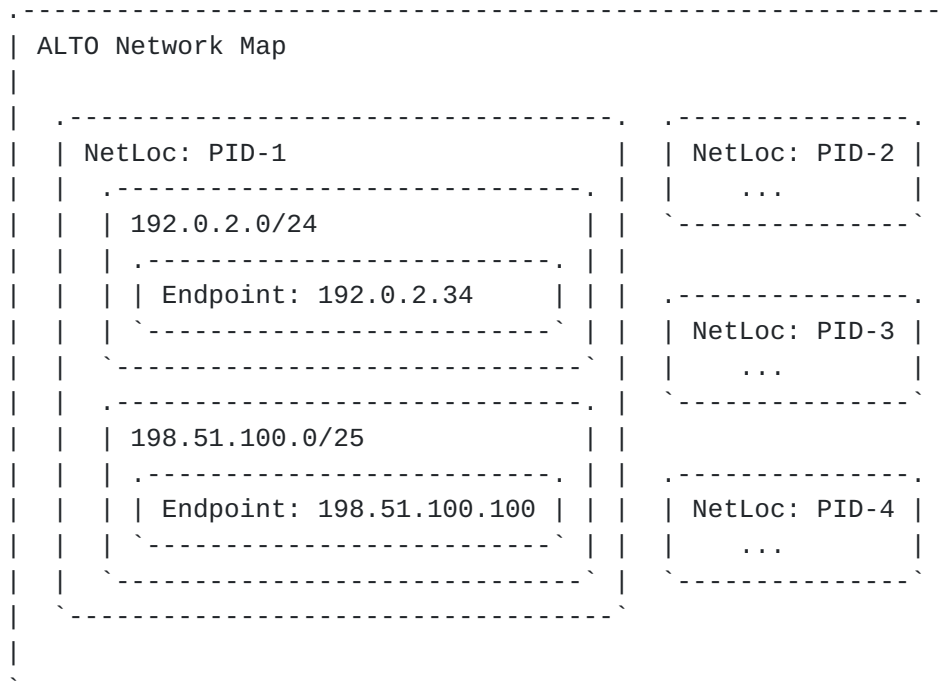


Figure 3: Example Network Map

5. Cost Map

An ALTO Server indicates preferences amongst network locations in the form of Path Costs. Path Costs are generic costs and can be internally computed by a network provider according to its own needs.

An ALTO Cost Map defines Path Costs pairwise amongst sets of source and destination Network Locations. Each Path Cost is the end-to-end cost from the source to the destination.

Each application may independently determine how the Resource Consumer and Resource Provider are designated as the source or destination, and hence how to utilize the Path Cost provided by ALTO. For example, if the cost is expected to be correlated with throughput, a typical application concerned with bulk data retrieval may use the Resource Provider as the source, and Resource Consumer as the destination.

One advantage of separating ALTO information into a Network Map and a Cost Map is that the two components can be updated at different time scales. For example, Network Maps may be stable for a longer time while Cost Maps may be updated to reflect dynamic network conditions.

As used in this document, the Cost Map refers to the syntax and

semantics of the information distributed by the ALTO Server. This document does not discuss the internal representation of this data structure within the ALTO Server.

5.1. Cost Attributes

Path Costs have attributes:

- o Type: identifies what the costs represent;
- o Mode: identifies how the costs should be interpreted.

Certain queries for Cost Maps allow the ALTO Client to indicate the desired Type and Mode.

5.1.1. Cost Type

The Type attribute indicates what the cost represents. For example, an ALTO Server could define costs representing air-miles, hop-counts, or generic routing costs.

Cost types are indicated in protocol messages as strings.

5.1.1.1. Cost Type: routingcost

An ALTO Server MUST define the 'routingcost' Cost Type.

This Cost Type conveys a generic measure for the cost of routing traffic from a source to a destination. Lower values indicate a higher preference for traffic to be sent from a source to a destination.

Note that an ISP may internally compute routing cost using any method it chooses (e.g., air-miles or hop-count) as long as it conforms to these semantics.

5.1.2. Cost Mode

The Mode attribute indicates how costs should be interpreted. Specifically, the Mode attribute indicates whether returned costs should be interpreted as numerical values or ordinal rankings.

It is important to communicate such information to ALTO Clients, as certain operations may not be valid on certain costs returned by an ALTO Server. For example, it is possible for an ALTO Server to return a set of IP addresses with costs indicating a ranking of the IP addresses. Arithmetic operations that would make sense for numerical values, do not make sense for ordinal rankings. ALTO

Clients may handle such costs differently.

Cost Modes are indicated in protocol messages as strings.

An ALTO Server MUST support at least one of 'numerical' and 'ordinal' costs. ALTO Clients SHOULD be cognizant of operations when a desired cost mode is not supported. For example, an ALTO Client desiring numerical costs may adjust behavior if only the ordinal Cost Mode is available. Alternatively, an ALTO Client desiring ordinal costs may construct ordinal costs given numerical values if only the numerical Cost Mode is available.

5.1.2.1. Cost Mode: numerical

This Cost Mode is indicated by the string 'numerical'. This mode indicates that it is safe to perform numerical operations (e.g. normalization or computing ratios for weighted load-balancing) on the returned costs. The values are floating-point numbers.

5.1.2.2. Cost Mode: ordinal

This Cost Mode is indicated by the string 'ordinal'. This mode indicates that the costs values in a Cost Map are a ranking (relative to all other values in the Cost Map), with lower values indicating a higher preference. The values are non-negative integers. Ordinal cost values in a Cost Map need not be unique nor contiguous. In particular, it is possible that two entries in a map have an identical rank (ordinal cost value). This document does not specify any behavior by an ALTO Client in this case; an ALTO Client may decide to break ties by random selection, other application knowledge, or some other means.

It is important to note that the values in the Cost Map provided with the ordinal Cost Mode are not necessarily the actual cost known to the ALTO Server.

5.2. Cost Map Structure

A query for a Cost Map either explicitly or implicitly includes a list of Source Network Locations and a list of Destination Network Locations. (Recall that a Network Location can be an endpoint address or a PID.)

Specifically, assume that a query has a list of multiple Source Network Locations, say [Src_1, Src_2, ..., Src_m], and a list of multiple Destination Network Locations, say [Dst_1, Dst_2, ..., Dst_n].

The ALTO Server will return the Path Cost for each communicating pair (i.e., Src_1 -> Dst_1, ..., Src_1 -> Dst_n, ..., Src_m -> Dst_1, ..., Src_m -> Dst_n). If the ALTO Server does not define a Path Cost for a particular pair, it may be omitted. We refer to this structure as a Cost Map.

If the Cost Mode is 'ordinal', the Path Cost of each communicating pair is relative to the m*n entries.

5.3. Network Map and Cost Map Dependency

If a Cost Map contains PIDs in the list of Source Network Locations or the list of Destination Network Locations, the Path Costs are generated based on a particular Network Map (which defines the PIDs). Version Tags are introduced to ensure that ALTO Clients are able to use consistent information even though the information is provided in two maps.

A Version Tag is an opaque string associated with a Network Map maintained by the ALTO Server. When the Network Map changes, the Version Tag MUST also be changed. (Thus, the Version Tag is defined similarly to HTTP's Entity Tags; see [Section 3.11 of \[RFC2616\]](#).) Possibilities for generating a Version Tag include the last-modified timestamp for the Network Map, or a hash of its contents.

A Network Map distributed by the ALTO Server includes its Version Tag. A Cost Map referring to PIDs also includes the Version Tag of the Network Map on which it is based.

6. Protocol Specification

This section first specifies general client and server processing, followed by a detailed specification for each ALTO Information Resource.

6.1. Overall Design

The ALTO Protocol uses a REST-ful design. There are two primary components to this design:

- o Information Resources: Each service provides network information as a set of resources, which are distinguished by their media types [\[RFC2046\]](#). An ALTO Client may construct an HTTP request for a particular resource (including any parameters, if necessary), and an ALTO Server returns the requested resource in an HTTP response.

- o Information Resource Directory: An ALTO Server provides to ALTO Clients a list of available resources and the URI at which each is provided. This document refers to this list as the Information Resource Directory. This directory is the single entry point to an ALTO Service. ALTO Clients consult the directory to determine the services provided by an ALTO Server.

6.2. Notation

This document uses an adaptation of the C-style struct notation to define the required and optional members of JSON objects. Unless explicitly noted, each member of a struct is REQUIRED.

The types 'JSONString', 'JSONNumber', 'JSONBool' indicate the JSON string, number, and boolean types, respectively. 'JSONValue' indicates a JSON value, as specified in [Section 2.1 of \[RFC4627\]](#).

Note that no standard, machine-readable interface definition or schema is provided. Extension documents may document these as necessary.

6.3. Basic Operation

The ALTO Protocol employs standard HTTP [[RFC2616](#)]. It is used for discovering available Information Resources at an ALTO Server and retrieving Information Resources. ALTO Clients and ALTO Servers use HTTP requests and responses carrying ALTO-specific content with encoding as specified in this document, and MUST be compliant with [[RFC2616](#)].

6.3.1. Discovering Information Resources

To discover available resources, an ALTO Client requests the Information Resource Directory, which an ALTO Server provides at the URI found by the ALTO Discovery protocol.

Informally, an Information Resource Directory enumerates URIs at which an ALTO Server offers Information Resources. Each entry in the directory indicates a URI at which an ALTO Server accepts requests, and returns either the requested Information Resource or an Information Resource Directory that references additional Information Resources. See [Section 6.7](#) for a detailed specification.

6.3.2. Requesting Information Resources

Through the retrieved Information Resource Directories, an ALTO Client can determine whether an ALTO Server supports the desired Information Resource, and if it is supported, the URI at which it is

available.

Where possible, the ALTO Protocol uses the HTTP GET method to request resources. However, some ALTO services provide Information Resources that are the function of one or more input parameters. Input parameters are encoded in the HTTP request's entity body, and the request uses the HTTP POST method.

It is possible for an ALTO Server to leverage caching HTTP intermediaries for responses to both GET and POST requests by including explicit freshness information (see [Section 14 of \[RFC2616\]](#)). Caching of POST requests is not widely implemented by HTTP intermediaries, however an alternative approach is for an ALTO Server, in response to POST requests, to return an HTTP 303 status code ("See Other") indicating to the ALTO Client that the resulting Information Resource is available via a GET request to an alternate URL. HTTP intermediaries that do not support caching of POST requests could then cache the response to the GET request from the ALTO Client following the alternate URL in the 303 response if the response to the subsequent GET request contains explicit freshness information.

When requesting an ALTO Information Resource that requires input parameters specified in a HTTP POST request, an ALTO Client MUST set the Content-Type HTTP header to the media type corresponding to the format of the supplied input parameters.

[6.3.3. Response](#)

Upon receiving a request, an ALTO server either returns the requested resource, provides the ALTO Client an Information Resource Directory indicating how to reach the desired resource, or returns an error.

The type of response MUST be indicated by the media type attached to the response (the Content-Type HTTP header). If an ALTO Client receives an Information Resource Directory, it can consult the received directory to determine if any of the offered URIs contain the desired Information Resource.

The generic encoding for an Information Resource is specified in [Section 6.4](#).

Errors are indicated via either ALTO-level error codes, or via HTTP status codes; see [Section 6.5](#).

6.3.4. Client Behavior

6.3.4.1. Using Information Resources

This specification does not indicate any required actions taken by ALTO Clients upon successfully receiving an Information Resource from an ALTO Server. Although ALTO Clients are suggested to interpret the received ALTO Information and adapt application behavior, ALTO Clients are not required to do so.

6.3.4.2. Error Conditions

If an ALTO Client does not successfully receive a desired Information Resource from a particular ALTO Server, it can either choose another server (if one is available) or fall back to a default behavior (e.g., perform peer selection without the use of ALTO information). An ALTO Client may also retry the request at a later time.

6.3.5. Authentication and Encryption

An ALTO Server MAY support SSL/TLS [[RFC5246](#)] to implement server and/or client authentication, encryption, and/or integrity protection. See [[RFC6125](#)] for considerations regarding verification of server identity.

6.3.6. HTTP Cookies

If cookies are included in an HTTP request received by an ALTO Server, they MUST be ignored.

6.3.7. Parsing

This document only details object members used by this specification. Extensions may include additional members within JSON objects defined in this document. ALTO implementations MUST ignore such unknown fields when processing ALTO messages.

6.4. Information Resource

An Information Resource is an HTTP entity body received by an ALTO Server that encodes the ALTO Information desired by an ALTO Client.

This document specifies multiple Information Resources that can be provided by an ALTO Server. Each Information Resource has certain attributes associated with it, indicating its data format, the input parameters it supports, and format of the input parameters.

6.4.1. Capabilities

An ALTO Server may advertise to an ALTO Client that it supports certain capabilities in requests for an Information Resource. For example, if an ALTO Server allows requests for a Cost Map to include constraints, it may advertise that it supports this capability.

6.4.2. Input Parameters Media Type

An ALTO Server may allow an ALTO Client to supply input parameters when requesting certain Information Resources. The format of the input parameters (i.e., as contained in the entity body of the HTTP POST request) is indicated by the media type [[RFC2046](#)].

6.4.3. Media Type

The media type [[RFC2046](#)] uniquely indicates the data format of the Information Resource as returned by an ALTO Server in the HTTP entity body.

6.4.4. Encoding

Though each Information Resource may have a distinct syntax, they are designed to have a common structure containing generic ALTO-layer metadata about the resource, as well as data itself.

An Information Resource has a single top-level JSON object of type InfoResourceEntity:

```
object {  
  InfoResourceMetaData  meta;    [OPTIONAL]  
  [InfoResourceDataType] data;  
} InfoResourceEntity;
```

with members:

meta meta-information pertaining to the Information Resource

data the data contained in the Information Resource

6.4.4.1. Meta Information

Meta information is encoded as a JSON object. This document does not specify any members, but it is defined here as a standard container for extensibility. Specifically, InfoResourceMetaData is defined as:


```
object {  
  } InfoResourceMetaData;
```

6.4.4.2. ALTO Information

The "data" member of the InfoResourceEntity encodes the resource-specific data; the structure of this member is detailed later in this section for each particular Information Resource.

6.4.4.3. Example

The following is an example of the encoding for an Information Resource:

```
HTTP/1.1 200 OK  
Content-Length: 40  
Content-Type: application/alto-costmap+json
```

```
{  
  "meta" : {},  
  "data" : {  
    ...  
  }  
}
```

6.5. ALTO Errors

If there is an error processing a request, an ALTO Server SHOULD return additional ALTO-layer information, if it is available, in the form of an ALTO Error Resource encoded in the HTTP response's entity body.

If no ALTO-layer information is available, an ALTO Server may omit an ALTO Error resource from the response. An appropriate HTTP status code MUST be set.

It is important to note that the HTTP Status Code and ALTO Error Code have distinct roles. An ALTO Error Code provides detailed information about why a particular request for an ALTO Resource was not successful. The HTTP status code indicates to HTTP processing elements (e.g., intermediaries and clients) how the response should be treated.

6.5.1. Media Type

The media type for an ALTO Error Resource is "application/alto-error+json".

6.5.2. Resource Format

An ALTO Error Resource has the format:

```
object {  
  JSONString code;  
} ErrorResourceEntity;
```

where:

code An ALTO Error Code defined in Table 1

6.5.3. Error Codes

This document defines ALTO Error Codes to support the error conditions needed for purposes of this document. Additional status codes may be defined in companion or extension documents.

The HTTP status codes corresponding to each ALTO Error Code are defined to provide correct behavior with HTTP intermediaries and clients. When an ALTO Server returns a particular ALTO Error Code, it **MUST** indicate one of the corresponding HTTP status codes in Table 1 in the HTTP response.

If multiple errors are present in a single request (e.g., a request uses a JSONString when a JSONInteger is expected and a required field is missing), then the ALTO Server **MUST** return exactly one of the detected errors. However, the reported error is implementation defined, since specifying a particular order for message processing encroaches needlessly on implementation technique.

ALTO Error Code	HTTP Status Code(s)	Description
E_SYNTAX	400	Parsing error in request (including identifiers)
E_JSON_FIELD_MISSING	400	Required field missing
E_JSON_VALUE_TYPE	400	JSON Value of unexpected type
E_INVALID_COST_MODE	400	Invalid cost mode
E_INVALID_COST_TYPE	400	Invalid cost type
E_INVALID_PROPERTY_TYPE	400	Invalid property type

Table 1: Defined ALTO Error Codes

6.5.4. Overload Conditions and Server Unavailability

If an ALTO Server detects that it cannot handle a request from an ALTO Client due to excessive load, technical problems, or system maintenance, it SHOULD do one of the following:

- o Return an HTTP 503 ("Service Unavailable") status code to the ALTO Client. As indicated by [[RFC2616](#)], a the Retry-After HTTP header may be used to indicate when the ALTO Client should retry the request.
- o Return an HTTP 307 ("Temporary Redirect") status code indicating an alternate ALTO Server that may be able to satisfy the request.

The ALTO Server MAY also terminate the connection with the ALTO Client.

The particular policy applied by an ALTO Server to determine that it cannot service a request is outside of the scope of this document.

6.6. ALTO Types

This section details the format for particular data values used in the ALTO Protocol.

6.6.1. PID Name

A PID Name is encoded as a US-ASCII string. The string MUST be no more than 64 characters, and MUST NOT contain any ASCII character below 0x21 or above 0x7E or the '.' separator (0x2E). The '.' separator is reserved for future use and MUST NOT be used unless specifically indicated by a companion or extension document.

The type 'PIDName' is used in this document to indicate a string of this format.

6.6.2. Version Tag

A Version Tag is encoded as a US-ASCII string. The string MUST be no more than 64 characters, and MUST NOT contain any ASCII character below 0x21 or above 0x7E.

The type 'VersionTag' is used in this document to indicate a string of this type.

6.6.3. Endpoints

This section defines formats used to encode addresses for Endpoints. In a case that multiple textual representations encode the same Endpoint address or prefix (within the guidelines outlined in this document), the ALTO Protocol does not require ALTO Clients or ALTO Servers to use a particular textual representation, nor does it require that ALTO Servers reply to requests using the same textual representation used by requesting ALTO Clients. ALTO Clients must be cognizant of this.

6.6.3.1. Address Type

Address Types are encoded as US-ASCII strings consisting of only alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A). This document defines the address type 'ipv4' to refer to IPv4 addresses, and 'ipv6' to refer to IPv6 addresses. All Address Type identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Address Type registry [Section 9.4](#).

The type 'AddressType' is used in this document to indicate a string of this format.

6.6.3.2. Endpoint Address

Endpoint Addresses are encoded as US-ASCII strings. The exact characters and format depend on the type of endpoint address.

The type 'EndpointAddr' is used in this document to indicate a string of this format.

[6.6.3.2.1.](#) IPv4

IPv4 Endpoint Addresses are encoded as specified by the 'IPv4address' rule in [Section 3.2.2 of \[RFC3986\]](#).

[6.6.3.2.2.](#) IPv6

IPv6 Endpoint Addresses are encoded as specified in [Section 4 of \[RFC5952\]](#).

[6.6.3.2.3.](#) Typed Endpoint Addresses

When an Endpoint Address is used, an ALTO implementation must be able to determine its type. For this purpose, the ALTO Protocol allows endpoint addresses to also explicitly indicate their type.

Typed Endpoint Addresses are encoded as US-ASCII strings of the format 'AddressType:EndpointAddr' (with the ':' character as a separator). The type 'TypedEndpointAddr' is used to indicate a string of this format.

[6.6.3.3.](#) Endpoint Prefixes

For efficiency, it is useful to denote a set of Endpoint Addresses using a special notation (if one exists). This specification makes use of the prefix notations for both IPv4 and IPv6 for this purpose.

Endpoint Prefixes are encoded as US-ASCII strings. The exact characters and format depend on the type of endpoint address.

The type 'EndpointPrefix' is used in this document to indicate a string of this format.

[6.6.3.3.1.](#) IPv4

IPv4 Endpoint Prefixes are encoded as specified in [Section 3.1 of \[RFC4632\]](#).

[6.6.3.3.2.](#) IPv6

IPv6 Endpoint Prefixes are encoded as specified in [Section 7 of \[RFC5952\]](#).

6.6.3.4. Endpoint Address Group

The ALTO Protocol includes messages that specify potentially large sets of endpoint addresses. Endpoint Address Groups provide a more efficient way to encode such sets, even when the set contains endpoint addresses of different types.

An Endpoint Address Group is defined as:

```
object {  
  EndpointPrefix [AddressType]<0..*>;  
  ...  
} EndpointAddrGroup;
```

In particular, an Endpoint Address Group is a JSON object with the name of each member being the string corresponding to the address type, and the member's corresponding value being a list of prefixes of addresses of that type.

The following is an example with both IPv4 and IPv6 endpoint addresses:

```
{  
  "ipv4": [  
    "192.0.2.0/24",  
    "198.51.100.0/25"  
  ],  
  "ipv6": [  
    "2001:db8:0:1::/64",  
    "2001:db8:0:2::/64"  
  ]  
}
```

6.6.4. Cost Mode

A Cost Mode is encoded as a US-ASCII string. The string **MUST** either have the value 'numerical' or 'ordinal'.

The type 'CostMode' is used in this document to indicate a string of this format.

6.6.5. Cost Type

A Cost Type is encoded as a US-ASCII string. The string MUST be no more than 32 characters, and MUST NOT contain characters other than alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A), the hyphen ('-', code point 0x2D), or the colon (':', code point 0x3A).

Identifiers prefixed with 'priv:' are reserved for Private Use [RFC5226]. Identifiers prefixed with 'exp:' are reserved for Experimental use. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Cost Types registry [Section 9.2](#).

The type 'CostType' is used in this document to indicate a string of this format.

6.6.6. Endpoint Property

An Endpoint Property is encoded as a US-ASCII string. The string MUST be no more than 32 characters, and MUST NOT contain characters other than alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A), the hyphen ('-', code point 0x2D), or the colon (':', code point 0x3A).

Identifiers prefixed with 'priv:' are reserved for Private Use [RFC5226]. Identifiers prefixed with 'exp:' are reserved for Experimental use. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Endpoint Property registry [Section 9.3](#).

The type 'EndpointProperty' is used in this document to indicate a string of this format.

6.7. Information Resource Directory

An Information Resource Directory indicates to ALTO Clients which Information Resources are made available by an ALTO Server.

Since resource selection happens after consumption of the Information Resource Directory, the format of the Information Resource Directory is designed to be simple with the intention of future ALTO Protocol versions maintaining backwards compatibility. Future extensions or versions of the ALTO Protocol SHOULD be accomplished by extending existing media types or adding new media types, but retaining the same format for the Information Resource Directory.

An ALTO Server MUST make an Information Resource Directory available

via the HTTP GET method to a URI discoverable by an ALTO Client. Discovery of this URI is out of scope of this document, but could be accomplished by manual configuration or by returning the URI of an Information Resource Directory from the ALTO Discovery Protocol [[I-D.ietf-alto-server-discovery](#)].

6.7.1. Media Type

The media type is "application/alto-directory+json".

6.7.2. Encoding

An Information Resource Directory is a JSON object of type InfoResourceDirectory:

```
object {  
    ...  
} Capabilities;  
  
object {  
    JSONString    uri;  
    JSONString    media-types<1..*>;  
    JSONString    accepts<0..*>;           [OPTIONAL]  
    Capabilities  capabilities;             [OPTIONAL]  
} ResourceEntry;  
  
object {  
    ResourceEntry resources<0..*>;  
} InfoResourceDirectory;
```

where the "resources" array indicates a list of Information Resources provided by an ALTO Server. Note that the list of available resources is enclosed in a JSON object for extensibility; future protocol versions may specify additional members in the InfoResourceDirectory object.

Any URI endpoint indicated in an Information Resource Directory MAY provide a response to an OPTIONS request that is in the format of an Information Resource Directory response. This provides ALTO Clients a means to discover resources and capabilities offered by that URI endpoint. ALTO Servers that reply with an HTTP 300 status code ("Multiple Choices") SHOULD use the Information Resource Directory format in the reply.

Each entry in the directory specifies:

uri A URI at which the ALTO Server provides one or more Information Resources, or an Information Resource Directory indicating additional Information Resources.

media-types The list of all media types of Information Resources (see [Section 6.4.3](#)) available via GET or POST requests to the corresponding URI or URIs discoverable via the URI.

accepts The list of all media types of input parameters (see [Section 6.4.2](#)) accepted by POST requests to the corresponding URI or URIs discoverable via the URI. If this member is not present, it MUST be assumed to be an empty array.

capabilities A JSON Object enumerating capabilities of an ALTO Server in providing the Information Resource at the corresponding URI and Information Resources discoverable via the URI. If this member is not present, it MUST be assumed to be an empty object. If a capability for one of the offered Information Resources is not explicitly listed here, an ALTO Client may either issue an OPTIONS HTTP request to the corresponding URI to determine if the capability is supported, or assume its default value documented in this specification or an extension document describing the capability.

If an entry has an empty list for "accepts", then the corresponding URI MUST support GET requests. If an entry has a non-empty list for "accepts", then the corresponding URI MUST support POST requests. If an ALTO Server wishes to support both GET and POST on a single URI, it MUST specify two entries in the Information Resource Directory.

[6.7.3](#). Example

The following is an example Information Resource Directory returned by an ALTO Server. In this example, the ALTO Server provides additional Network and Cost Maps via a separate subdomain, "custom.alto.example.com". The maps available via this subdomain are Filtered Network and Cost Maps as well as pre-generated maps for the "hopcount" and "routingcost" Cost Types in the "ordinal" Cost Mode.

An ALTO Client can discover the maps available by "custom.alto.example.com" by successfully performing an OPTIONS request to "http://custom.alto.example.com/maps".

In this example, the ALTO server provides the Endpoint Cost Service for Cost Types 'routingcost' and 'hopcount', each available for both 'numerical' and 'ordinal' mode".


```
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: 1472
Content-Type: application/alto-directory+json
```

```
{
  "resources" : [
    {
      "uri" : "http://alto.example.com/networkmap",
      "media-types" : [ "application/alto-networkmap+json" ]
    }, {
      "uri" : "http://alto.example.com/costmap/num/routingcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "routingcost" ]
      }
    }, {
      "uri" : "http://alto.example.com/costmap/num/hopcount",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "hopcount" ]
      }
    }, {
      "uri" : "http://custom.alto.example.com/maps",
      "media-types" : [
        "application/alto-networkmap+json",
        "application/alto-costmap+json"
      ],
      "accepts" : [
        "application/alto-networkmapfilter+json",
        "application/alto-costmapfilter+json"
      ]
    }, {
      "uri" : "http://alto.example.com/endpointprop/lookup",
      "media-types" : [ "application/alto-endpointprop+json" ],
      "accepts" : [ "application/alto-endpointpropparams+json" ],
      "capabilities" : {
        "prop-types" : [ "pid" ]
      }
    }
  ]
}
```



```
    "uri" : "http://alto.example.com/endpointcost/lookup",
    "media-types" : [ "application/alto-endpointcost+json" ],
    "accepts" : [ "application/alto-endpointcostparams+json" ],
    "capabilities" : {
      "cost-constraints" : true,
      "cost-modes" : [ "ordinal", "numerical" ],
      "cost-types" : [ "routingcost", "hopcount" ]
    }
  ]
}
```

OPTIONS /maps HTTP/1.1

Host: custom.alto.example.com

Accept: application/alto-directory+json,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: 1001
Content-Type: application/alto-directory+json

```
{
  "resources" : [
    {
      "uri" : "http://custom.alto.example.com/networkmap/filtered",
      "media-types" : [ "application/alto-networkmap+json" ],
      "accepts" : [ "application/alto-networkmapfilter+json" ]
    }, {
      "uri" : "http://custom.alto.example.com/costmap/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "cost-modes" : [ "ordinal", "numerical" ],
        "cost-types" : [ "routingcost", "hopcount" ]
      }
    }, {
      "uri" : "http://custom.alto.example.com/ord/routingcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "ordinal" ],
        "cost-types" : [ "routingcost" ]
      }
    }, {
      "uri" : "http://custom.alto.example.com/ord/hopcount",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "ordinal" ],
        "cost-types" : [ "hopcount" ]
      }
    }
  ]
}
```

[6.7.4.](#) Usage Considerations

[6.7.4.1.](#) ALTO Client

This document specifies no requirements or constraints on ALTO Clients with regards to how they process an Information Resource Directory to identify the URI corresponding to a desired Information Resource. However, some advice is provided for implementors.

It is possible that multiple entries in the directory match a desired

Information Resource. For instance, in the example in [Section 6.7.3](#), a full Cost Map with "numerical" Cost Mode and "routingcost" Cost Type could be retrieved via a GET request to "http://alto.example.com/costmap/num/routingcost", or via a POST request to "http://custom.alto.example.com/costmap/filtered".

In general, it is preferred for ALTO Clients to use GET requests where appropriate, since it is more likely for responses to be cacheable.

[6.7.4.2](#). ALTO Server

This document indicates that an ALTO Server may or may not provide the Information Resources specified in the Map Filtering Service. If these resources are not provided, it is indicated to an ALTO Client by the absence of a Network Map or Cost Map with any media types listed under "accepts".

[6.8](#). Information Resources

This section documents the individual Information Resources defined in the ALTO Protocol.

[6.8.1](#). Map Service

The Map Service provides batch information to ALTO Clients in the form of two types of maps: a Network Map and Cost Map.

[6.8.1.1](#). Network Map

The Network Map Information Resource lists for each PID, the network locations (endpoints) within the PID. It MUST be provided by an ALTO Server.

[6.8.1.1.1](#). Media Type

The media type is "application/alto-networkmap+json".

[6.8.1.1.2](#). HTTP Method

This resource is requested using the HTTP GET method.

[6.8.1.1.3](#). Input Parameters

None.

6.8.1.1.4. Capabilities

None.

6.8.1.1.5. Response

The returned InfoResourceEntity object "data" member of type InfoResourceNetworkMap:

```
object {  
  EndpointAddrGroup [pidname]<0..*>;  
  ...  
} NetworkMapData;  
  
object {  
  VersionTag      map-vtag;  
  NetworkMapData map;  
} InfoResourceNetworkMap;
```

with members:

map-vtag The Version Tag ([Section 5.3](#)) of the Network Map.

map The Network Map data itself.

NetworkMapData is a JSON object with each member representing a single PID and its associated set of endpoint addresses. A member's name is a string of type PIDName.

The returned Network Map MUST include all PIDs known to the ALTO Server.

6.8.1.1.6. Example

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



```
HTTP/1.1 200 OK
Content-Length: 370
Content-Type: application/alto-networkmap+json
```

```
{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "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"
        ]
      }
    }
  }
}
```

6.8.1.2. Cost Map

The Cost Map resource lists the Path Cost for each pair of source/destination PID defined by the ALTO Server for a given Cost Type and Cost Mode. This resource MUST be provided for at least the 'routingcost' Cost Type and 'numerical' Cost Mode.

Note that since this resource, an unfiltered Cost Map requested by an HTTP GET, does not indicate the desired Cost Mode or Cost Type as input parameters, an ALTO Server MUST indicate in an Information Resource Directory a unfiltered Cost Map Information Resource by specifying the capabilities ([Section 6.8.1.2.4](#)) with "cost-types" and "cost-modes" members each having a single element. This technique will allow an ALTO Client to determine a URI for an unfiltered Cost Map of the desired Cost Mode and Cost Type.

[6.8.1.2.1.](#) Media Type

The media type is "application/alto-costmap+json".

[6.8.1.2.2.](#) HTTP Method

This resource is requested using the HTTP GET method.

[6.8.1.2.3.](#) Input Parameters

None.

[6.8.1.2.4.](#) Capabilities

This resource may be defined for across multiple Cost Types and Cost Modes. The capabilities of an ALTO Server URI providing this resource are defined by a JSON Object of type CostMapCapability:

```
object {  
  CostMode cost-modes<0..*>;  
  CostType cost-types<0..*>;  
} CostMapCapability;
```

with members:

cost-modes The Cost Modes ([Section 5.1.2](#)) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

cost-types The Cost Types ([Section 5.1.1](#)) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

An ALTO Server MUST support all of the Cost Types listed here for each of the listed Cost Modes. Note that an ALTO Server may provide multiple Cost Map Information Resources, each with different capabilities.

[6.8.1.2.5.](#) Response

The returned InfoResourceEntity object has "data" member of type InfoResourceCostMap:


```
object DstCosts {
  JSONValue [PIDName];
  ...
};

object {
  DstCosts [PIDName]<0..*>;
  ...
} CostMapData;

object {
  CostMode      cost-mode;
  CostType      cost-type;
  VersionTag    map-vtag;
  CostMapData   map;
} InfoResourceCostMap;
```

with members:

cost-mode Cost Mode ([Section 5.1.2](#)) used in the Cost Map.

cost-type Cost Type ([Section 5.1.1](#)) used in the Cost Map.

map-vtag The Version Tag ([Section 5.3](#)) of the Network Map used to generate the Cost Map.

map The Cost Map data itself.

CostMapData is a JSON object with each member representing a single Source PID; the name for a member is the PIDName string identifying the corresponding Source PID. For each Source PID, a DstCosts object denotes the associated cost to a set of destination PIDs ([Section 5.2](#)); the name for each member in the object is the PIDName string identifying the corresponding Destination PID. An implementation of the protocol in this document 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 returned Cost Map MUST include the Path Cost for each (Source PID, Destination PID) pair for which a Path Cost is defined. An ALTO Server MAY omit entries for which a Path Cost is not defined (e.g., both the Source and Destination PIDs contain addresses outside of the Network Provider's administrative domain).

[6.8.1.2.6](#). Example

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

```
HTTP/1.1 200 OK
Content-Length: 262
Content-Type: application/alto-costmap+json
```

```
{
  "meta" : {},
  "data" : {
    "cost-mode" : "numerical",
    "cost-type" : "routingcost",
    "map-vtag" : "1266506139",
    "map" : {
      "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
      "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
      "PID3": { "PID1": 20, "PID2": 15 }
    }
  }
}
```

[6.8.2](#). Map Filtering Service

The Map Filtering Service allows ALTO Clients to specify filtering criteria to return a subset of the full maps available in the Map Service.

[6.8.2.1](#). Filtered Network Map

A Filtered Network Map is a Network Map Information Resource ([Section 6.8.1.1](#)) for which an ALTO Client may supply a list of PIDs to be included. A Filtered Network Map MAY be provided by an ALTO Server.

[6.8.2.1.1](#). Media Type

See [Section 6.8.1.1.1](#).

[6.8.2.1.2.](#) HTTP Method

This resource is requested using the HTTP POST method.

[6.8.2.1.3.](#) Input Parameters

Input parameters 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-networkmapfilter+json", which is a JSON Object of type ReqFilteredNetworkMap, where:

```
object {  
  PIDName pids<0..*>;  
  AddressType address-types<0..*>;  
} ReqFilteredNetworkMap;
```

with members:

pids Specifies list of PIDs to be included in the returned Filtered Network Map. If the list of PIDs is empty, the ALTO Server MUST interpret the list as if it contained a list of all currently-defined PIDs. The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

address-types Specifies list of address types to be included in the returned Filtered Network Map. If the list of address types is empty, the ALTO Server MUST interpret the list as if it contained a list of all address types known to the ALTO Server. The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

[6.8.2.1.4.](#) Capabilities

None.

[6.8.2.1.5.](#) Response

See [Section 6.8.1.1.5](#) for the format.

The ALTO Server MUST only include PIDs in the response that were specified (implicitly or explicitly) in the request. If the input parameters contain a PID name that is not currently defined by the ALTO Server, the ALTO Server MUST behave as if the PID did not appear in the input parameters. Similarly, the ALTO Server MUST only enumerate addresses within each PID that have types which were specified (implicitly or explicitly) in the request. If the input

parameters contain an address type that is not currently known to the ALTO Server, the ALTO Server MUST behave as if the address type did not appear in the input parameters.

6.8.2.1.6. Example

```
POST /networkmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Length: 27
Content-Type: application/alto-networkmapfilter+json
Accept: application/alto-networkmap+json,application/alto-error+json
```

```
{
  "pids": [ "PID1", "PID2" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: 255
Content-Type: application/alto-networkmap+json
```

```
{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "map" : {
      "PID1" : {
        "ipv4" : [
          "192.0.2.0/24",
          "198.51.100.0/24"
        ]
      },
      "PID2" : {
        "ipv4": [
          "198.51.100.128/24"
        ]
      }
    }
  }
}
```

6.8.2.2. Filtered Cost Map

A Filtered Cost Map is a Cost Map Information Resource ([Section 6.8.1.2](#)) for which an ALTO Client may supply additional parameters limiting the scope of the resulting Cost Map. A Filtered

Cost Map MAY be provided by an ALTO Server.

[6.8.2.2.1](#). Media Type

See [Section 6.8.1.2.1](#).

[6.8.2.2.2](#). HTTP Method

This resource is requested using the HTTP POST method.

[6.8.2.2.3](#). Input Parameters

Input parameters 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-costmapfilter+json", which is a JSON Object of type ReqFilteredCostMap, where:

```
object {
  PIDName srcs<0..*>;
  PIDName dsts<0..*>;
} PIDFilter;

object {
  CostMode    cost-mode;
  CostType    cost-type;
  JSONString constraints<0..*>;  [OPTIONAL]
  PIDFilter   pids;               [OPTIONAL]
} ReqFilteredCostMap;
```

with members:

cost-type The Cost Type ([Section 5.1.1](#)) for the returned costs. This MUST be one of the supported Cost Types indicated in this resource's capabilities ([Section 6.8.2.2.4](#)).

cost-mode The Cost Mode ([Section 5.1.2](#)) for the returned costs. This MUST be one of the supported Cost Modes indicated in this resource's capabilities ([Section 6.8.2.2.4](#)).

constraints Defines a list of additional constraints on which elements of the Cost Map are returned. This parameter MUST NOT be specified if this resource's capabilities ([Section 6.8.2.2.4](#)) indicate that constraint support is not available. A constraint contains two entities separated by whitespace: (1) an operator, 'gt' for greater than, 'lt' for less than, 'ge' for greater than or equal to, 'le' for less than or equal to, or 'eq' for equal to

(2) a target cost value. The cost value is a number that MUST be defined in the same units as the Cost Type indicated by the cost-type parameter. ALTO Servers SHOULD use at least IEEE 754 double-precision floating point [[IEEE.754.2008](#)] to store the cost value, and SHOULD perform internal computations using double-precision floating-point arithmetic. If multiple 'constraint' parameters are specified, they are interpreted as being related to each other with a logical AND.

pids A list of Source PIDs and a list of Destination PIDs for which Path Costs are to be returned. If a list is empty, the ALTO Server MUST interpret it as the full set of currently-defined PIDs. The ALTO Server MUST interpret entries appearing in a list multiple times as if they appeared only once. If the "pids" member is not present, both lists MUST be interpreted by the ALTO Server as containing the full set of currently-defined PIDs.

[6.8.2.2.4](#). Capabilities

The URI providing this resource supports all capabilities documented in [Section 6.8.1.2.4](#) (with identical semantics), plus additional capabilities. In particular, the capabilities are defined by a JSON object of type FilteredCostMapCapability:

```
object {  
  CostMode cost-modes<0..*>;  
  CostType cost-types<0..*>;  
  JSONBool cost-constraints;  
} FilteredCostMapCapability;
```

with members:

cost-modes See [Section 6.8.1.2.4](#).

cost-types See [Section 6.8.1.2.4](#).

cost-constraints If true, then the ALTO Server allows cost constraints to be included in requests to the corresponding URI. If not present, this member MUST be interpreted as if it specified false. ALTO Clients should be aware that constraints may not have the intended effect for cost maps with the 'ordinal' Cost Mode since ordinal costs are not restricted to being sequential integers.

6.8.2.2.5. Response

See [Section 6.8.1.2.5](#) for the format.

The returned Cost Map MUST NOT contain any source/destination pair that was not indicated (implicitly or explicitly) in the input parameters. If the input parameters contain a PID name that is not currently defined by the ALTO Server, the ALTO Server MUST behave as if the PID did not appear in the input parameters.

If any constraints are specified, Source/Destination pairs for which the Path Costs do not meet the constraints MUST NOT be included in the returned Cost Map. If no constraints were specified, then all Path Costs are assumed to meet the constraints.

Note that ALTO Clients should verify that the Version Tag included in the response is consistent with the Version Tag of the Network Map used to generate the request (if applicable). If it is not, the ALTO Client may wish to request an updated Network Map, identify changes, and consider requesting a new Filtered Cost Map.

[6.8.2.2.6.](#) Example

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

```
{
  "cost-mode" : "numerical",
  "cost-type" : "routingcost",
  "pids" : {
    "srcs" : [ "PID1" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 177
Content-Type: application/alto-costmap+json
```

```
{
  "meta" : {},
  "data" : {
    "cost-mode" : "numerical",
    "cost-type" : "routingcost",
    "map-vtag" : "1266506139",
    "map" : {
      "PID1": { "PID1": 0, "PID2": 1, "PID3": 2 }
    }
  }
}
```

[6.8.3.](#) Endpoint Property Service

The Endpoint Property Service provides information about Endpoint properties to ALTO Clients.

[6.8.3.1.](#) Endpoint Property

The Endpoint Property resource provides information about properties for individual endpoints. It MAY be provided by an ALTO Server. If an ALTO Server provides one or more Endpoint Property resources, then at least one MUST provide the 'pid' property.

[6.8.3.1.1.](#) Media Type

The media type is "application/alto-endpointprop+json".

[6.8.3.1.2.](#) HTTP Method

This resource is requested using the HTTP POST method.

[6.8.3.1.3.](#) Input Parameters

Input parameters are supplied in the entity body of the POST request. This document specifies the data format of input parameters with the media type "application/alto-endpointpropparams+json", which is a JSON Object of type ReqEndpointProp:

```
object {  
  EndpointProperty  properties<1..*>;  
  TypedEndpointAddr endpoints<1..*>;  
} ReqEndpointProp;
```

with members:

properties List of endpoint properties to be returned for each endpoint. Each specified property MUST be included in the list of supported properties indicated by this resource's capabilities ([Section 6.8.3.1.4](#)). The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

endpoints List of endpoint addresses for which the specified properties are to be returned. The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

[6.8.3.1.4.](#) Capabilities

This resource may be defined across multiple types of endpoint properties. The capabilities of an ALTO Server URI providing Endpoint Properties are defined by a JSON Object of type EndpointPropertyCapability:

```
object {  
  EndpointProperty prop-types<0..*>;  
} EndpointPropertyCapability;
```

with members:

prop-types The Endpoint Property Types (see [Section 6.6.6](#)) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

6.8.3.1.5. Response

The returned InfoResourceEntity object has "data" member of type InfoResourceEndpointProperty, where:

```
object {
  JSONValue [EndpointProperty];
  ...
} EndpointProps;

object {
  EndpointProps [TypedEndpointAddr]<0..*>;
  ...
} EndpointPropertyMapData;

object {
  VersionTag          map-vtag;          [OPTIONAL]
  EndpointPropertyMapData map;
} InfoResourceEndpointProperty;
```

EndpointPropertyMapData has one member for each endpoint indicated in the input parameters (with the name being the endpoint encoded as a TypedEndpointAddr). The requested properties for each endpoint are encoded in a corresponding EndpointProps object, which encodes one name/value pair for each requested property, where the property names are encoded as strings of type EndpointProperty. An implementation of the protocol in this document SHOULD assume that the property value is a JSONString 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.

The ALTO Server returns the value for each of the requested endpoint properties for each of the endpoints listed in the input parameters.

If the ALTO Server does not define a requested property's value for a particular endpoint, then it MUST omit that property from the response for only that endpoint.

The ALTO Server MAY include the Version Tag ([Section 5.3](#)) of the Network Map used to generate the response (if desired and applicable) as the 'map-vtag' member in the response. If the 'pid' property is returned for any endpoints in the response, the 'map-vtag' member is

REQUIRED instead of OPTIONAL.

6.8.3.1.6. Example

```
POST /endpointprop/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 96
Content-Type: application/alto-endpointpropparams+json
Accept: application/alto-endpointprop+json,application/alto-error+json
```

```
{
  "properties" : [ "pid", "example-prop" ],
  "endpoints" : [ "ipv4:192.0.2.34", "ipv4:203.0.113.129" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: 149
Content-Type: application/alto-endpointprop+json
```

```
{
  "meta" : {},
  "data": {
    "map" : {
      "ipv4:192.0.2.34" : { "pid": "PID1", "example-prop": "1" },
      "ipv4:203.0.113.129" : { "pid": "PID3" }
    }
  }
}
```

6.8.4. Endpoint Cost Service

The Endpoint Cost Service provides information about costs between individual endpoints.

In particular, this service allows lists of Endpoint prefixes (and addresses, as a special case) to be ranked (ordered) by an ALTO Server.

6.8.4.1. Endpoint Cost

The Endpoint Cost resource provides information about costs between individual endpoints. It MAY be provided by an ALTO Server.

It is important to note that although this resource allows an ALTO Server to reveal costs between individual endpoints, an ALTO Server

is not required to do so. A simple alternative would be to compute the cost between two endpoints as the cost between the PIDs corresponding to the endpoints. See [Section 10.1](#) for additional details.

[6.8.4.1.1](#). Media Type

The media type is "application/alto-endpointcost+json".

[6.8.4.1.2](#). HTTP Method

This resource is requested using the HTTP POST method.

[6.8.4.1.3](#). Input Parameters

Input parameters are supplied in the entity body of the POST request. This document specifies input parameters with a data format indicated by media type "application/alto-endpointcostparams+json", which is a JSON Object of type ReqEndpointCostMap:

```
object {
  TypedEndpointAddr srcs<0..*>;           [OPTIONAL]
  TypedEndpointAddr dsts<1..*>;
} EndpointFilter;

object {
  CostMode          cost-mode;
  CostType          cost-type;
  JSONString        constraints<0..*>;    [OPTIONAL]
  EndpointFilter    endpoints;
} ReqEndpointCostMap;
```

with members:

cost-mode The Cost Mode ([Section 5.1.2](#)) to use for returned costs. This MUST be one of the Cost Modes indicated in this resource's capabilities ([Section 6.8.4.1.4](#)).

cost-type The Cost Type ([Section 5.1.1](#)) to use for returned costs. This MUST be one of the Cost Types indicated in this resource's capabilities ([Section 6.8.4.1.4](#)).

constraints Defined equivalently to the "constraints" input parameter of a Filtered Cost Map (see [Section 6.8.2.2](#)).

endpoints A list of Source Endpoints and Destination Endpoints for which Path Costs are to be returned. If the list of Source Endpoints is empty (or not included), the ALTO Server MUST interpret it as if it contained the Endpoint Address corresponding to the client IP address from the incoming connection (see [Section 8.3](#) for discussion and considerations regarding this mode). The list of destination Endpoints MUST NOT be empty. The ALTO Server MUST interpret entries appearing multiple times in a list as if they appeared only once.

[6.8.4.1.4.](#) Capabilities

See [Section 6.8.2.2.4.](#)

[6.8.4.1.5.](#) Response

The returned InfoResourceEntity object has "data" member equal to InfoResourceEndpointCostMap, where:

```
object EndpointDstCosts {
  JSONValue [TypedEndpointAddr];
  ...
};

object {
  EndpointDstCosts [TypedEndpointAddr]<0..*>;
  ...
} EndpointCostMapData;

object {
  CostMode          cost-mode;
  CostType          cost-type;
  EndpointCostMapData map;
} InfoResourceEndpointCostMap;
```

InfoResourceEndpointCostMap has members:

cost-mode The Cost Mode used in the returned Cost Map.

cost-type The Cost Type used in the returned Cost Map.

map The Endpoint Cost Map data itself.

EndpointCostMapData is a JSON object with each member representing a single Source Endpoint specified in the input parameters; the name for a member is the TypedEndpointAddr string identifying the

corresponding Source Endpoint. For each Source Endpoint, a `EndpointDstCosts` object denotes the associated cost to each Destination Endpoint specified in the input parameters; the name for each member in the object is the `TypedEndpointAddr` string identifying the corresponding Destination Endpoint. An implementation of the protocol in this document SHOULD assume that the cost value is a `JSONNumber` and fail to parse if it is not, unless the implementation is using an extensions to this document that indicates when and how costs of other data types are signaled. If the ALTO Server does not define a cost value from a Source Endpoint to a particular Destination Endpoint, it MAY be omitted from the response.

[6.8.4.1.6](#). Example

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 195
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json
```

```
{
  "cost-mode" : "ordinal",
  "cost-type" : "routingcost",
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45"
    ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 231
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta" : {},
  "data" : {
    "cost-mode" : "ordinal",
    "cost-type" : "routingcost",
    "map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89" : 1,
        "ipv4:198.51.100.34" : 2,
        "ipv4:203.0.113.45" : 3
      }
    }
  }
}
```

[7](#). Use Cases

The sections below depict typical use cases. While these use cases focus on peer-to-peer applications, ALTO can be applied to ther

environments such as CDNs [[I-D.jenkins-alto-cdn-use-cases](#)].

7.1. ALTO Client Embedded in P2P Tracker

Many currently-deployed P2P systems use a Tracker to manage swarms and perform peer selection. P2P trackers may currently use a variety of information to perform peer selection to meet application-specific goals. By acting as an ALTO Client, an P2P tracker can use ALTO information as an additional information source to enable more network-efficient traffic patterns and improve application performance.

A particular requirement of many P2P trackers is that they must handle a large number of P2P clients. A P2P tracker can obtain and locally store ALTO information (the Network Map and Cost Map) from the ISPs containing the P2P clients, and benefit from the same aggregation of network locations done by ALTO Servers.

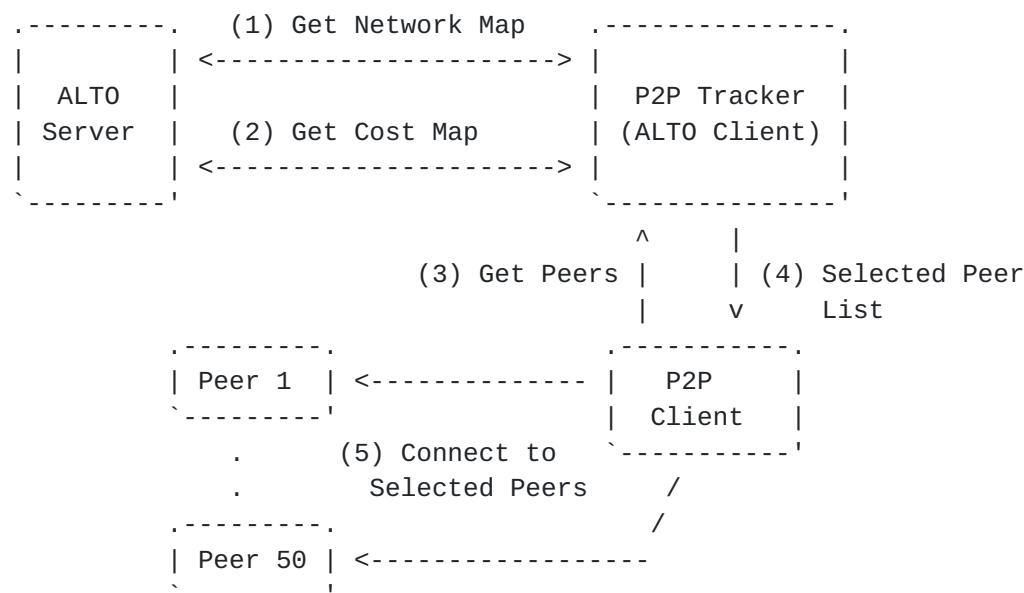


Figure 4: ALTO Client Embedded in P2P Tracker

Figure 4 shows an example use case where a P2P tracker is an ALTO Client and applies ALTO information when selecting peers for its P2P clients. The example proceeds as follows:

1. The P2P Tracker requests the Network Map covering all PIDs from the ALTO Server using the Network Map query. The Network Map includes the IP prefixes contained in each PID, allowing the P2P tracker to locally map P2P clients into a PIDs.

2. The P2P Tracker requests the Cost Map amongst all PIDs from the ALTO Server.
3. A P2P Client joins the swarm, and requests a peer list from the P2P Tracker.
4. The P2P Tracker returns a peer list to the P2P client. The returned peer list is computed based on the Network Map and Cost Map returned by the ALTO Server, and possibly other information sources. Note that it is possible that a tracker may use only the Network Map to implement hierarchical peer selection by preferring peers within the same PID and ISP.
5. The P2P Client connects to the selected peers.

Note that the P2P tracker may provide peer lists to P2P clients distributed across multiple ISPs. In such a case, the P2P tracker may communicate with multiple ALTO Servers.

7.2. ALTO Client Embedded in P2P Client: Numerical Costs

P2P clients may also utilize ALTO information themselves when selecting from available peers. It is important to note that not all P2P systems use a P2P tracker for peer discovery and selection. Furthermore, even when a P2P tracker is used, the P2P clients may rely on other sources, such as peer exchange and DHTs, to discover peers.

When an P2P Client uses ALTO information, it typically queries only the ALTO Server servicing its own ISP. The my-Internet view provided by its ISP's ALTO Server can include preferences to all potential peers.

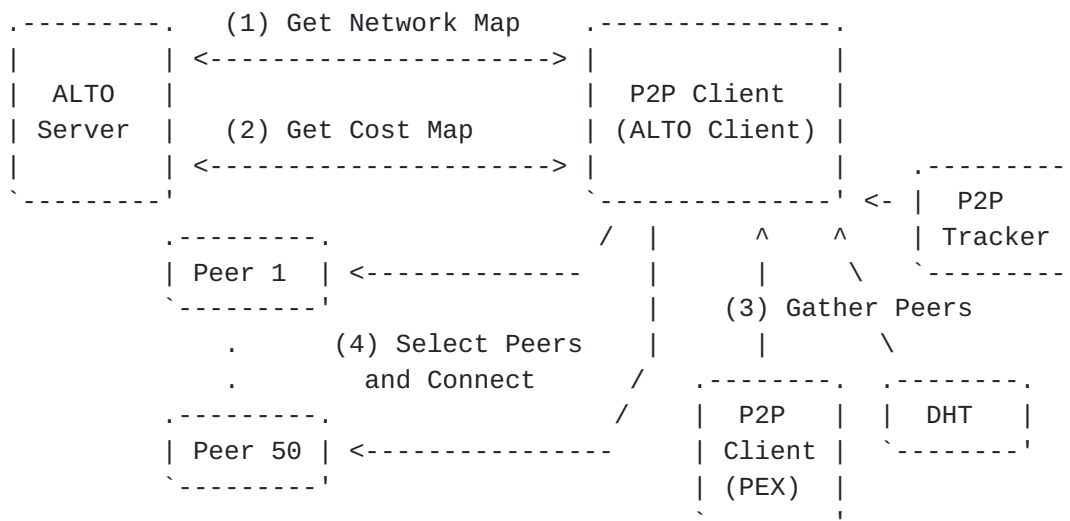


Figure 5: ALTO Client Embedded in P2P Client

Figure 5 shows an example use case where a P2P Client locally applies ALTO information to select peers. The use case proceeds as follows:

1. The P2P Client requests the Network Map covering all PIDs from the ALTO Server servicing its own ISP.
2. The P2P Client requests the Cost Map amongst all PIDs from the ALTO Server. The Cost Map by default specifies numerical costs.
3. The P2P Client discovers peers from sources such as Peer Exchange (PEX) from other P2P Clients, Distributed Hash Tables (DHT), and P2P Trackers.
4. The P2P Client uses ALTO information as part of the algorithm for selecting new peers, and connects to the selected peers.

7.3. ALTO Client Embedded in P2P Client: Ranking

It is also possible for a P2P Client to offload the selection and ranking process to an ALTO Server. In this use case, the ALTO Client gathers a list of known peers in the swarm, and asks the ALTO Server to rank them.

As in the use case using numerical costs, the P2P Client typically only queries the ALTO Server servicing its own ISP.

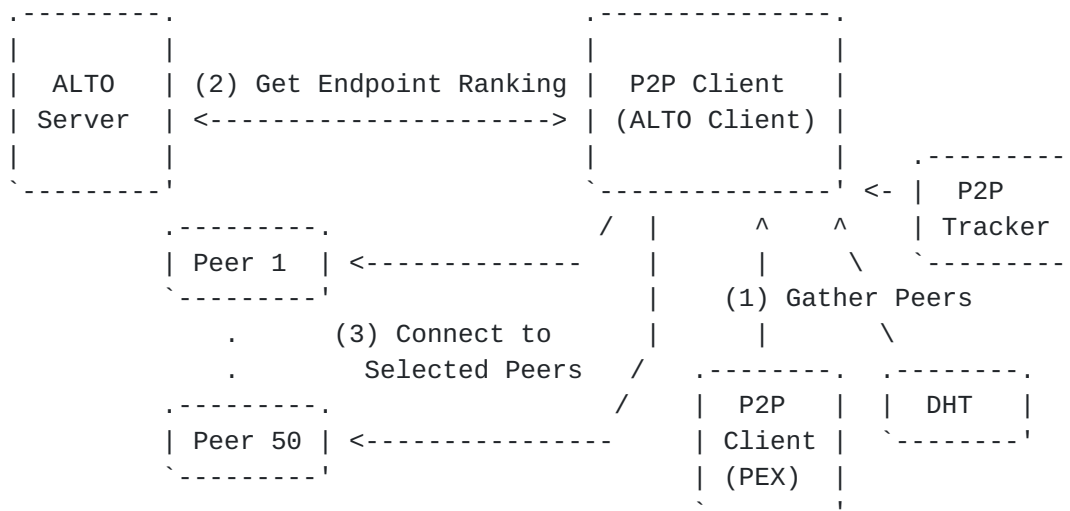


Figure 6: ALTO Client Embedded in P2P Client: Ranking

Figure 6 shows an example of this scenario. The use case proceeds as follows:

1. The P2P Client discovers peers from sources such as Peer Exchange (PEX) from other P2P Clients, Distributed Hash Tables (DHT), and P2P Trackers.
2. The P2P Client queries the ALTO Server's Ranking Service, including discovered peers as the set of Destination Endpoints, and indicates the 'ordinal' Cost Mode. The response indicates the ranking of the candidate peers.
3. The P2P Client connects to the peers in the order specified in the ranking.

8. Discussions

8.1. Discovery

The discovery mechanism by which an ALTO Client locates an appropriate ALTO Server is out of scope for this document. This document assumes that an ALTO Client can discover an appropriate ALTO Server. Once it has done so, the ALTO Client may use the Information Resource Directory (see [Section 6.7](#)) to locate an Information Resource with the desired ALTO Information.

8.2. Hosts with Multiple Endpoint Addresses

In practical deployments, especially during the transition from IPv4 to IPv6, a particular host may be reachable using multiple addresses. Furthermore, the particular network path followed when sending packets to the host may differ based on the address that is used. Network providers may prefer one path over another (e.g., one path may have a NAT64 middlebox). An additional consideration may be how to handle private address spaces (e.g., behind carrier-grade NATs).

To support such behavior, this document allows multiple types of endpoint addresses. In supporting multiple address types, the ALTO Protocol also allows ALTO Service Provider the flexibility to indicate preferences for paths from an endpoint address of one type to an endpoint address of a different type. Note that in general, the path through the network may differ dependent on the types of addresses that are used.

Note that there are limitations as to what information ALTO can provide in this regard. In particular, a particular ALTO Service provider may not be able to determine if connectivity with a particular endhost will succeed over IPv4 or IPv6, as this may depend upon information unknown to the ISP such as particular application implementations.

8.3. Network Address Translation Considerations

At this day and age of NAT v4<->v4, v4<->v6 [[RFC6144](#)], and possibly v6<->v6[I-D.mrw-nat66], a protocol should strive to be NAT friendly and minimize carrying IP addresses in the payload, or provide a mode of operation where the source IP address provide the information necessary to the server.

The protocol specified in this document provides a mode of operation where the source network location is computed by the ALTO Server (i.e., the the Endpoint Cost Service) from the source IP address found in the ALTO Client query packets. This is similar to how some P2P Trackers (e.g., BitTorrent Trackers - see "Tracker HTTP/HTTPS Protocol" in [[BitTorrent](#)]) operate.

There may be cases where an ALTO Client needs to determine its own IP address, such as when specifying a source Endpoint Address in the Endpoint Cost Service. It is possible that an ALTO Client has multiple network interface addresses, and that some or all of them may require NAT for connectivity to the public Internet.

If a public IP address is required for a network interface, the ALTO client SHOULD use the Session Traversal Utilities for NAT (STUN)

[[RFC5389](#)]. If using this method, the host MUST use the "Binding Request" message and the resulting "XOR-MAPPED-ADDRESS" parameter that is returned in the response. Using STUN requires cooperation from a publicly accessible STUN server. Thus, the ALTO client also requires configuration information that identifies the STUN server, or a domain name that can be used for STUN server discovery. To be selected for this purpose, the STUN server needs to provide the public reflexive transport address of the host.

ALTO Clients should be cognizant that the network path between Endpoints can be dependent on the network interfaces, source address, and destination address used for communication. An ALTO Server provides information based on Endpoint Addresses (more generally, Network Locations), but the mechanisms used for determining existence of connectivity or usage of NAT between Endpoints are out of scope of this document.

[8.4.](#) Endpoint and Path Properties

An ALTO Server could make available many properties about Endpoints beyond their network location or grouping. For example, connection type, geographical location, and others may be useful to applications. This specification focuses on network location and grouping, but the protocol may be extended to handle other Endpoint properties.

[9.](#) IANA Considerations

[9.1.](#) application/alto-* Media Types

This document requests the registration of multiple media types, listed in Table 2.

Type	Subtype	Specification
application	alto-directory+json	Section 6.7
application	alto-networkmap+json	Section 6.8.1.1
application	alto-networkmapfilter+json	Section 6.8.2.1
application	alto-costmap+json	Section 6.8.1.2
application	alto-costmapfilter+json	Section 6.8.2.2
application	alto-endpointprop+json	Section 6.8.3.1
application	alto-endpointpropparams+json	Section 6.8.3.1
application	alto-endpointcost+json	Section 6.8.4.1
application	alto-endpointcostparams+json	Section 6.8.4.1
application	alto-error+json	Section 6.5

Table 2: ALTO Protocol Media Types

Type name: application

Subtype name: This documents requests the registration of multiple subtypes, as listed in Table 2.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the 'application/json' media type. See [\[RFC4627\]](#).

Security considerations: Security considerations relating to the generation and consumption of ALTO protocol messages are discussed in [Section 10](#).

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

Published specification: This document is the specification for these media types; see Table 2 for the section documenting each media type.

Applications that use this media type: ALTO Servers and ALTO Clients either standalone or embedded within other applications.

Additional information:

Magic number(s): n/a

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

Macintosh file type code(s): n/a

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

Intended usage: COMMON

Restrictions on usage: n/a

Author: See "Authors' Addresses" section.

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

9.2. ALTO Cost Type Registry

This document requests the creation of an ALTO Cost Type registry to be maintained by IANA.

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO Cost Types. Second, it provides references to particular semantics of allocated Cost Types to be applied by both ALTO Servers and applications utilizing ALTO Clients.

New ALTO Cost Types are assigned after Expert Review [[RFC5226](#)]. The Expert Reviewer will generally consult the ALTO Working Group or its successor. Expert Review is used to ensure that proper documentation regarding ALTO Cost Type semantics and security considerations has been provided. The provided documentation should be detailed enough to provide guidance to both ALTO Service Providers and applications utilizing ALTO Clients as to how values of the registered ALTO Cost Type should be interpreted. Updates and deletions of ALTO Cost Types follow the same procedure.

Registered ALTO Cost Type identifiers MUST conform to the syntactical requirements specified in [Section 6.6.5](#). Identifiers are to be recorded and displayed as ASCII strings.

Identifiers prefixed with 'priv:' are reserved for Private Use.
Identifiers prefixed with 'exp:' are reserved for Experimental use.

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

- o Identifier: The name of the desired ALTO Cost Type.
- o Intended Semantics: ALTO Costs carry with them semantics to guide their usage by ALTO Clients. For example, if a value refers to a measurement, the measurement units must be documented. For proper implementation of the ordinal Cost Mode (e.g., by a third-party service), it should be documented whether higher or lower values of the cost are more preferred.
- o Security Considerations: ALTO Costs expose information to ALTO Clients. As such, proper usage of a particular Cost Type may require certain information to be exposed by an ALTO Service Provider. Since network information is frequently regarded as proprietary or confidential, ALTO Service Providers should be made aware of the security ramifications related to usage of a Cost Type.

This specification requests registration of the identifier 'routingcost'. Semantics for the this Cost Type are documented in [Section 5.1.1.1](#), and security considerations are documented in [Section 10.1](#).

[9.3.](#) ALTO Endpoint Property Registry

This document requests the creation of an ALTO Endpoint Property registry to be maintained by IANA.

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO Endpoint Properties. Second, it provides references to particular semantics of allocated Endpoint Properties to be applied by both ALTO Servers and applications utilizing ALTO Clients.

New ALTO Endpoint Properties are assigned after Expert Review [[RFC5226](#)]. The Expert Reviewer will generally consult the ALTO Working Group or its successor. Expert Review is used to ensure that proper documentation regarding ALTO Endpoint Property semantics and security considerations has been provided. The provided documentation should be detailed enough to provide guidance to both ALTO Service Providers and applications utilizing ALTO Clients as to how values of the registered ALTO Endpoint Properties should be interpreted. Updates and deletions of ALTO Endpoint Properties follow the same procedure.

Registered ALTO Endpoint Property identifiers MUST conform to the syntactical requirements specified in [Section 6.6.6](#). Identifiers are to be recorded and displayed as ASCII strings.

Identifiers prefixed with 'priv:' are reserved for Private Use.
Identifiers prefixed with 'exp:' are reserved for Experimental use.

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

- o Identifier: The name of the desired ALTO Endpoint Property.
- o Intended Semantics: ALTO Endpoint Properties carry with them semantics to guide their usage by ALTO Clients. For example, if a value refers to a measurement, the measurement units must be documented.
- o Security Considerations: ALTO Endpoint Properties expose information to ALTO Clients. As such, proper usage of a particular Endpoint Properties may require certain information to be exposed by an ALTO Service Provider. Since network information is frequently regarded as proprietary or confidential, ALTO Service Providers should be made aware of the security ramifications related to usage of an Endpoint Property.

This specification requests registration of the identifier 'pid'. Semantics for the this Endpoint Property are documented in [Section 4.1](#), and security considerations are documented in [Section 10.1](#).

[9.4.](#) ALTO Address Type Registry

This document requests the creation of an ALTO Address Type registry to be maintained by IANA.

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

New ALTO Address Types are assigned after Expert Review [[RFC5226](#)]. The Expert Reviewer will generally consult the ALTO Working Group or its successor. Expert Review is used to ensure that proper documentation regarding the new ALTO Address Types and their security considerations has been provided. The provided documentation should indicate how an address of a registered type is encoded as an EndpointAddr and, if possible, a compact method (e.g., IPv4 and IPv6 prefixes) for encoding a set of addresses as an EndpointPrefix. Updates and deletions of ALTO Address Types follow the same procedure.

Registered ALTO Address Type identifiers MUST conform to the syntactical requirements specified in [Section 6.6.3.1](#). Identifiers

are to be recorded and displayed as ASCII strings.

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

- o Identifier: The name of the desired ALTO Address Type.
- o Endpoint Address Encoding: The procedure for encoding an address of the registered type as an EndpointAddr (see [Section 6.6.3.2](#)).
- o Endpoint Prefix Encoding: The procedure for encoding a set of addresses of the registered type as an EndpointPrefix (see [Section 6.6.3.3](#)). If no such compact encoding is available, the same encoding used for a singular address may be used. In such a case, it must be documented that sets of addresses of this type always have exactly one element.
- o Mapping to/from IPv4/IPv6 Addresses: If possible, a mechanism to map addresses of the registered type to and from IPv4 or IPv6 addresses should be specified.
- o Security Considerations: In some usage scenarios, Endpoint Addresses carried in ALTO Protocol messages may reveal information about an ALTO Client or an ALTO Service Provider. Applications and ALTO Service Providers using addresses of the registered type should be made aware of how (or if) the addressing scheme relates to private information and network proximity.

This specification requests registration of the identifiers 'ipv4' and 'ipv6'. Endpoint Address Encoding is documented in [Section 6.6.3.2.1](#) and [Section 6.6.3.2.2](#). Endpoint Prefix Encoding is documented in [Section 6.6.3.3.1](#) and [Section 6.6.3.3.2](#). Since these are registrations for IPv4 and IPv6 address types, no mapping is required. Security considerations are documented in [Section 10.1](#) and [Section 10.2](#).

10. Security Considerations

10.1. Privacy Considerations for ISPs

ISPs must be cognizant of the network topology and provisioning information provided through ALTO Interfaces. ISPs should evaluate how much information is revealed and the associated risks. On the one hand, providing overly fine-grained information may make it easier for attackers to infer network topology. In particular, attackers may try to infer details regarding ISPs' operational policies or inter-ISP business relationships by intentionally posting

a multitude of selective queries to an ALTO server and analyzing the responses. Such sophisticated attacks may reveal more information than an ISP hosting an ALTO server intends to disclose. On the other hand, revealing overly coarse-grained information may not provide benefits to network efficiency or performance improvements to ALTO Clients.

It is possible that one or multiple ALTO Clients issue queries in an effort to reverse-engineer specific details (e.g., network topology) that was used to produce ALTO information. Operators should have security policies in place such that confidential information or information that could be reverse-engineered to reveal confidential information is not sent to unauthorized ALTO Clients.

10.2. ALTO Clients

Applications using the information must be cognizant of the possibility that the information is malformed or incorrect. Even if an ALTO Server has been properly authenticated by the ALTO Client, the information provided may be malicious because the ALTO Server and its credentials have been compromised (e.g., through malware). Other considerations (e.g., relating to application performance) can be found in [Section 6 of \[RFC5693\]](#).

ALTO Clients should also be cognizant of revealing Network Location Identifiers (IP addresses or fine-grained PIDs) to the ALTO Server, as doing so may allow the ALTO Server to infer communication patterns. One possibility is for the ALTO Client to only rely on Network Map for PIDs and Cost Map amongst PIDs to avoid passing IP addresses of other endpoints (e.g., peers) to the ALTO Server.

In addition, ALTO clients should be cautious not to unintentionally or indirectly disclose the resource identifier (of which they try to improve the retrieval through ALTO-guidance), e.g., the name/identifier of a certain video stream in P2P live streaming, to the ALTO server. Note that the ALTO Protocol specified in this document does not explicitly reveal any resource identifier to the ALTO Server. However, for instance, depending on the popularity or other specifics (such as language) of the resource, an ALTO server could potentially deduce information about the desired resource from information such as the Network Locations the client sends as part of its request to the server.

10.3. Authentication, Integrity Protection, and Encryption

SSL/TLS [\[RFC5246\]](#) can provide encryption and integrity protection of transmitted messages as well as authentication of the ALTO Client and Server. HTTP Basic or Digest authentication can provide

authentication of the client (combined with SSL/TLS, it can additionally provide encryption, integrity protection and server authentication).

Issues resulting from an attacker controlling the data received by an ALTO Client are discussed in [Section 10.2](#).

An ALTO Server may optionally use authentication (and potentially encryption) to limit the parties with whom ALTO information is directly shared. There may be special use cases where encryption of ALTO information is desirable. In many cases, however, information sent out by an ALTO Server may be regarded as non-confidential information.

ISPs should be cognizant that encryption only protects ALTO information until it is decrypted by the intended ALTO Client. Digital Rights Management (DRM) techniques and legal agreements protecting ALTO information are outside of the scope of this document.

[10.4.](#) ALTO Information Redistribution

It is possible for applications to redistribute ALTO information to improve scalability. Even with such a distribution scheme, ALTO Clients obtaining ALTO information must be able to validate the received ALTO information to ensure that it was generated by an appropriate ALTO Server. Support for this validation is not provided in this document, but may be provided by extension documents.

[10.5.](#) Denial of Service

ISPs should be cognizant of the workload at the ALTO Server generated by certain ALTO Queries, such as certain queries to the Map Filtering Service and Ranking Service. In particular, queries which can be generated with low effort but result in expensive workloads at the ALTO Server could be exploited for Denial-of-Service attacks. For instance, a simple ALTO query with n Source Network Locations and m Destination Network Locations can be generated fairly easily but results in the computation of $n*m$ Path Costs between pairs by the ALTO Server (see [Section 5.2](#)). One way to limit Denial-of-Service attacks is to employ access control to the ALTO server. Another possible mechanism for an ALTO Server to protect itself against a multitude of computationally expensive bogus requests is to demand that each ALTO Client to solve a computational puzzle first before allocating resources for answering a request (see, e.g., [\[I-D.jennings-sip-hashcash\]](#)). The current specification does not use such computational puzzles, and discussion regarding tradeoffs of such an approach would be needed before including such a technique in

the ALTO Protocol.

ISPs should also leverage the fact that the the Map Service allows ALTO Servers to pre-generate maps that can be useful to many ALTO Clients.

10.6. ALTO Server Access Control

In order to limit access to an ALTO server (e.g., for an ISP to only allow its users to access its ALTO server, or to prevent Denial-of-Service attacks by arbitrary hosts from the Internet), an ALTO server may employ access control policies. Depending on the use-case and scenario, an ALTO server may restrict access to its services more strictly or rather openly (see [[I-D.ietf-alto-deployments](#)] for a more detailed discussion on this issue).

11. Manageability Considerations

This section details operations and management considerations based on existing deployments and discussions during protocol development. It also indicates where extension documents are expected to provide appropriate functionality discussed in [[RFC5706](#)] as additional deployment experience becomes available.

11.1. Operations

11.1.1. Installation and Initial Setup

The ALTO Protocol is based on HTTP. Thus, configuring an ALTO Server may require configuring the underlying HTTP server implementation to define appropriate security policies, caching policies, performance settings, etc.

Additionally, an operator of an ALTO Server will need to configure the ALTO information to be provided by the ALTO Server. The granularity of the topological map and the cost map is left to the specific policies of the operator of the ALTO Server. However, a reasonable default may include two PIDs, one to hold the endpoints in the operator's network and the second PID to represent full IPv4 and IPv4 reachability (see [Section 4.2.1](#)), with the cost between each source/destination PID set to 1.

Implementers employing an ALTO Client should attempt to automatically discover an appropriate ALTO Server. Manual configuration of the ALTO Server location may be used where automatic discovery is not appropriate. Methods for automatic discovery and manual configuration are discussed in [[I-D.ietf-alto-server-discovery](#)].

Specifications for underlying protocols (e.g., TCP, HTTP, SSL/TLS) should be consulted for their available settings and proposed default configurations.

11.1.2. Migration Path

This document does not detail a migration path for ALTO Servers since there is no previous standard protocol providing the similar functionality.

There are existing applications making use of network information discovered from other entities such as whois, geo-location databases, or round-trip time measurements, etc. Such applications should consider using ALTO as an additional source of information; ALTO need not be the sole source of network information.

11.1.3. Requirements on Other Protocols and Functional Components

The ALTO Protocol assumes that HTTP client and server implementations exist. It also assumes that JSON encoder and decoder implementations exist.

An ALTO Server assumes that it can gather sufficient information to populate Network and Cost maps. "Sufficient information" is dependent on the information being exposed, but likely includes information gathered from protocols such as IGP and EGP Routing Information Bases (see Figure 1). Specific mechanisms have been proposed (e.g., [[I-D.medved-alto-svr-apis](#)]) and are expected to be provided in extension documents.

11.1.4. Impact on Network Operation

ALTO presents a new opportunity for managing network traffic by providing additional information to clients. The potential impact to network operation is large.

Deployment of an ALTO Server may shift network traffic patterns. Thus, operators should consider impacts on (or integration with) traffic engineering.

Operators providing an ALTO Server should ensure that appropriate information is being exposed. Privacy implications for ISPs are discussed in [Section 10.1](#). Both operators and ALTO Servers and those using ALTO Clients should be aware of the impact of incorrect or faked guidance (see Section 10.3 of [[I-D.ietf-alto-deployments](#)] and future versions of that document).

11.1.5. Verifying Correct Operation

ALTO-specific monitoring and metrics are discussed in 6.3 of [\[I-D.ietf-alto-deployments\]](#) and future versions of that document.

11.2. Management

11.2.1. Management Interoperability

A common management API would be desirable given that ALTO Servers may typically be configured with dynamic data from various sources, and ALTO Servers are intended to scale horizontally for fault-tolerance and reliability. A specific API or protocol is outside the scope of this document, but may be provided by an extension document.

Logging is an important functionality for ALTO Servers and, depending on the deployment, ALTO Clients. Logging should be done via syslog [\[RFC5424\]](#).

11.2.2. Management Information

A Management Information Model (see [Section 3.2 of \[RFC5706\]](#)) is not provided by this document, but should be included or referenced by any extension documenting an ALTO-related management API or protocol.

11.2.3. Fault Management

Monitoring ALTO Servers and Clients is described in Section 6.3 of [\[I-D.ietf-alto-deployments\]](#) and future versions of that document.

11.2.4. Configuration Management

Standardized approaches and protocols to configuration management for ALTO are outside the scope of this document, but this document does outline high-level principles suggested for future standardization efforts.

An ALTO Server requires at least the following logical inputs:

- o Data sources from which ALTO Information is derived. This can either be raw network information (e.g., from routing elements) or pre-processed ALTO-level information in the form of a Network Map, Cost Map, etc.
- o Algorithms for computing the ALTO information returned to clients. These could either return information from a database, or information customized for each client.

- o Security policies mapping potential clients to the information that they have privilege to access.

Multiple ALTO Servers may be deployed for horizontal scalability. A centralized configuration database may be used to ensure they are providing the desired ALTO information with appropriate security controls. The ALTO information (e.g., Network Maps and Cost Maps) being served by each ALTO Server, as well as security policies (HTTP authentication, SSL/TLS client and server authentication, SSL/TLS encryption parameters) intended to serve the same information should be monitored for consistency.

11.2.5. Accounting Management

Monitoring ALTO Servers and Clients is described in Section 6.3 of [[I-D.ietf-alto-deployments](#)] and future versions of that document.

11.2.6. Performance Management

An exhaustive list of desirable performance information from a ALTO Servers and ALTO Clients are outside of the scope of this document. The following is a list of suggested ALTO-specific to be monitored based on the existing deployment and protocol development experience:

- o Requests and responses for each service listed in a Information Directory (total counts and size in bytes).
- o CPU and memory utilization
- o ALTO map updates
- o Number of PIDs
- o ALTO map sizes (in-memory size, encoded size, number of entries)

Additional information for monitoring performance of ALTO Servers and Clients is described in Section 6.3 of [[I-D.ietf-alto-deployments](#)] and future versions of that document.

11.2.7. Security Management

[Section 10](#) documents ALTO-specific security considerations. Operators should configure security policies with those in mind. Readers should refer to HTTP [[RFC2616](#)] and SSL/TLS [[RFC5246](#)] and related documents for mechanisms available for configuring security policies. Other appropriate security mechanisms (e.g., physical security, firewalls, etc) should also be considered.

12. References

12.1. Normative References

- [IEEE.754.2008]
Institute of Electrical and Electronics Engineers,
"Standard for Binary Floating-Point Arithmetic", IEEE
Standard 754, August 2008.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
Extensions (MIME) Part Two: Media Types", [RFC 2046](#),
November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
Transfer Protocol -- HTTP/1.1", [RFC 2616](#), June 1999.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
Resource Identifier (URI): Generic Syntax", STD 66,
[RFC 3986](#), January 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for
JavaScript Object Notation (JSON)", [RFC 4627](#), July 2006.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
(CIDR): The Internet Address Assignment and Aggregation
Plan", [BCP 122](#), [RFC 4632](#), August 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#),
May 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
(TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
"Session Traversal Utilities for NAT (STUN)", [RFC 5389](#),
October 2008.
- [RFC5424] Gerhards, R., "The Syslog Protocol", [RFC 5424](#), March 2009.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
Address Text Representation", [RFC 5952](#), August 2010.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and

Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", [RFC 6125](#), March 2011.

12.2. Informative References

[BitTorrent]

"Bittorrent Protocol Specification v1.0",
<<http://wiki.theory.org/BitTorrentSpecification>>.

[I-D.akonjang-alto-proxidor]

Akonjang, O., Feldmann, A., Previdi, S., Davie, B., and D. Saucez, "The PROXIDOR Service",
[draft-akonjang-alto-proxidor-00](#) (work in progress),
March 2009.

[I-D.gu-alto-redistribution]

Yingjie, G., Alimi, R., and R. Even, "ALTO Information Redistribution", [draft-gu-alto-redistribution-03](#) (work in progress), July 2010.

[I-D.ietf-alto-deployments]

Stiemerling, M., Kiesel, S., and S. Previdi, "ALTO Deployment Considerations", [draft-ietf-alto-deployments-04](#) (work in progress), March 2012.

[I-D.ietf-alto-reqs]

Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", [draft-ietf-alto-reqs-08](#) (work in progress), March 2011.

[I-D.ietf-alto-server-discovery]

Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery",
[draft-ietf-alto-server-discovery-04](#) (work in progress),
July 2012.

[I-D.jenkins-alto-cdn-use-cases]

Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs",
[draft-jenkins-alto-cdn-use-cases-03](#) (work in progress),
June 2012.

[I-D.jennings-sip-hashcash]

Jennings, C., "Computational Puzzles for SPAM Reduction in SIP", [draft-jennings-sip-hashcash-06](#) (work in progress),

July 2007.

[I-D.medved-alto-svr-apis]

Medved, J., Ward, D., Peterson, J., Woundy, R., and D. McDysan, "ALTO Network-Server and Server-Server APIs", [draft-medved-alto-svr-apis-00](#) (work in progress), March 2011.

[I-D.mrw-nat66]

Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", [draft-mrw-nat66-16](#) (work in progress), April 2011.

[I-D.p4p-framework]

Alimi, R., Pasko, D., Popkin, L., Wang, Y., and Y. Yang, "P4P: Provider Portal for P2P Applications", [draft-p4p-framework-00](#) (work in progress), November 2008.

[I-D.saumitra-alto-multi-ps]

Das, S., Narayanan, V., and L. Dondeti, "ALTO: A Multi Dimensional Peer Selection Problem", [draft-saumitra-alto-multi-ps-00](#) (work in progress), October 2008.

[I-D.saumitra-alto-queryresponse]

Das, S. and V. Narayanan, "A Client to Service Query Response Protocol for ALTO", [draft-saumitra-alto-queryresponse-00](#) (work in progress), March 2009.

[I-D.shalunov-alto-infoexport]

Shalunov, S., Penno, R., and R. Woundy, "ALTO Information Export Service", [draft-shalunov-alto-infoexport-00](#) (work in progress), October 2008.

[I-D.wang-alto-p4p-specification]

Wang, Y., Alimi, R., Pasko, D., Popkin, L., and Y. Yang, "P4P Protocol Specification", [draft-wang-alto-p4p-specification-00](#) (work in progress), March 2009.

[P4P-SIGCOMM08]

Xie, H., Yang, Y., Krishnamurthy, A., Liu, Y., and A. Silberschatz, "P4P: Provider Portal for (P2P) Applications", SIGCOMM 2008, August 2008.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", [RFC 5693](#),

October 2009.

- [RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", [RFC 5706](#), November 2009.
- [RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", [RFC 6144](#), April 2011.

[Appendix A](#). Acknowledgments

Thank you to Jan Seedorf for contributions to the Security Considerations section.

We would like to thank the following people whose input and involvement was indispensable in achieving this merged proposal:

Obi Akonjang (DT Labs/TU Berlin),
Saumitra M. Das (Qualcomm Inc.),
Syon Ding (China Telecom),
Doug Pasko (Verizon),
Laird Popkin (Pando Networks),
Satish Raghunath (Juniper Networks),
Albert Tian (Ericsson/Redback),
Yu-Shun Wang (Microsoft),
David Zhang (PPLive),
Yunfei Zhang (China Mobile).

We would also like to thank the following additional people who were involved in the projects that contributed to this merged document:

Alex Gerber (AT&T), Chris Griffiths (Comcast), Ramit Hora (Pando Networks), Arvind Krishnamurthy (University of Washington), Marty Lafferty (DCIA), Erran Li (Bell Labs), Jin Li (Microsoft), Y. Grace Liu (IBM Watson), Jason Livingood (Comcast), Michael Merritt (AT&T), Ingmar Poesche (DT Labs/TU Berlin), James Royalty (Pando Networks), Damien Saucez (UCL) Thomas Scholl (AT&T), Emilio Sepulveda (Telefonica), Avi Silberschatz (Yale University), Hassan Sipra (Bell Canada), Georgios Smaragdakis (DT Labs/TU Berlin), Haibin Song

(Huawei), Oliver Spatscheck (AT&T), See-Mong Tang (Microsoft), Jia Wang (AT&T), Hao Wang (Yale University), Ye Wang (Yale University), Haiyong Xie (Yale University).

[Appendix B](#). Authors

[[CmtAuthors: RFC Editor: Please move information in this section to the Authors' Addresses section at publication time.]]

Stefano Previdi
Cisco

Email: sprevidi@cisco.com

Stanislav Shalunov
BitTorrent

Email: shalunov@bittorrent.com

Richard Woundy
Comcast

Richard_Woundy@cable.comcast.com

Authors' Addresses

Richard Alimi (editor)
Google
1600 Amphitheatre Parkway
Mountain View CA
USA

Email: ralimi@google.com

Reinaldo Penno (editor)
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: repenno@cisco.com

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

Email: yry@cs.yale.edu