                             Multi-Cost ALTO
                     draft-ietf-alto-multi-cost-00

Abstract

   The ALTO (Application Layer-Traffic Optimization) Protocol
   ([RFC7285]) defines several services that return various metrics
   describing the costs between network endpoints. For example, when
   downloading a file that is mirrored on several sites, a user
   application may use these ALTO cost metrics to determine the most
   efficient mirror site.

   An ALTO Server may offer a variety of cost metrics, based on latency,
   bandwidth, hop count, jitter, or whatever else the ALTO Server deems
   useful. When selecting a mirror site, a client may consider more
   than one metric, perhaps trading bandwidth for latency. While the
   base ALTO Protocol allows a client to use more than one cost metric,
   to do so, the client must request each metric separately. This
   document defines a new service that allows a client to retrieve
   several cost metrics with one request, which is considerably more
   efficient. In addition, this document extends the ALTO constraint
   tests to allow a user to specify an arbitrary logical combination of
   tests on several cost metrics.

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].

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 November 23, 2015.

Copyright Notice

Table of Contents

1. Introduction

   IETF has designed a new service called ALTO that provides guidance to
   overlay applications, which have to select one or several hosts from
   a set of candidates that are able to provide a desired resource.
   This guidance is based on parameters that affect performance and
   efficiency of the data transmission between the hosts, e.g., the
   topological distance. The purpose of ALTO is to improve Quality of
   Experience (QoE) in the application while reducing resource
   consumption in the underlying network infrastructure. The ALTO
   protocol conveys the Internet View from the perspective of a Provider
   Network region that spans from a region to one or more Autonomous
   System (AS). Together with this Network Map, it provides the
   Provider determined Cost Map between locations of the Network Map.
   Last, it provides the Ranking of Endpoints w.r.t. their routing cost.

   Current ALTO Costs and their modes provide values that are seen to be
   stable over a longer period of time, such as hopcount and
   administrative routing cost to reflect ISP routing preferences.
   Recently, new use cases have extended the usage scope of ALTO to
   Content Delivery Networks, Data Centers and applications that need
   additional information to select their Endpoints or handle their
   PIDs.

   Thus a multitude of new Cost Types that better reflect the
   requirements of these applications are expected to be specified, in
   particular cost values that change more frequently than previously
   assumed.

   The ALTO protocol [RFC7285] restricts ALTO Cost Maps and Endpoint
   Cost services to only one Cost Type and Cost Mode per ALTO request.
   To retrieve information for several Cost Types, an ALTO client must
   send several separate requests to the server.

   It would be far more efficient, in terms of RTT, traffic, and
   processing load on the ALTO client and server, to get all costs with
   a single query/response transaction. Vector costs provide a robust
   and natural input to multi-variate path computation as well as robust
   multi-variate selection of multiple Endpoints. In particular, one
   Cost Map reporting on N Cost Types is less bulky than N Cost Maps
   containing one Cost Type each. This is valuable for both the storage
   of these maps and their transmission. Additionally, for many
   emerging applications that need information on several Cost Types,
   having them gathered in one map will save time.

   Along with multi-cost values queries, the filtering capabilities need
   to be extended to allow constraints on multiple metrics. The base
   protocol allows a client to provide optional constraint tests for a

Filtered Cost Map or the Endpoint Cost Service. In the base
protocol, the constraint tests are limited to the AND-combination of
simple comparison tests on the value of the (single) requested Cost
Type. It is therefore necessary to allow constraints on multiple
metrics. Beyond that, applications that are sensitive to several
metrics and struggle with complicated network conditions may need to
arbitrate between conflicting objectives such as routing cost and
network performance. To address this issue, this document proposes
to extend the base protocol by extending constraints to test multiple
metrics, and by allowing these constraints to be combined with
logical 'ORs' as well as logical 'ANDs'. This allows an application
to make requests such as: "select solutions with either (moderate
"hopcount" AND high "routingcost") OR (higher "hopcount" AND moderate
"routingcost")".

This document is organized as follows: Section 2 defines terminology
used in this document. Section 3 gives a non-normative overview of
the multi-cost extensions, and Section 4 gives their formal
definition. Section 5 gives several complete examples. The
remaining sections describe the IANA and privacy considerations.

2. Terminology

This document uses terms defined as follows:

o {1.2.3}: References of this form are to sections in the ALTO
   protocol specification [RFC7285].

o Endpoint (EP): can be a Peer, a CDN storage location, a physical
   server involved in a virtual server-supported application, a Party
   in a resource sharing swarm such as a computation Grid or an
   online multi-party game.

o Endpoint Discovery (EP Discovery) : this term covers the different
   types of processes used to discover the eligible endpoints.

o Network Service Provider (NSP): includes both ISPs, who provide
   means to transport the data, and Content Delivery Networks (CDNs)
   who care for the dissemination, persistent storage and possibly
   identification of the best/closest content copy.

o ALTO transaction: a request/response exchange between an ALTO
   Client and an ALTO Server.

o Application Client (AC): this term generalizes the case of a P2P
   client to include the case of a CDN client, a client of an
   application running on a virtual server, a GRID application client

       and any Client having the choice in several connection points for
       data or resource exchange.

3. Overview Of Approach

   The following is a non-normative overview of the multi-cost
   extensions defined in this document. It assumes the reader is
   familiar with Cost Map resources in the ALTO Protocol ([RFC7285]).

3.1. Multi-Cost Data Format

   Formally, the cost entries in an ALTO Cost Map can be any type of
   JSON value (see the DstCosts object in {11.2.3.6}). However, that
   section also says that an implementation may assume costs are JSON
   numbers, unless the implementation is using an extension which
   signals a different data type.

   Therefore this document extends the definition of a Cost Map to allow
   a cost to be an array of costs, one per metric, instead of just one
   number. For example, here is a Cost Map with the "routingcost" and
   "hopcount" metrics. Note that this is identical to a regular ALTO
   Cost Map, except that the values are arrays instead of numbers.

```
{
 "meta" : {
   "dependent-vtags" : [ ... ],
   "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
   ]
 }
 "cost-map" : {
   "PID1": { "PID1":[1,0],  "PID2":[5,23],  "PID3":[10,5] },
   ...
 }
}
```

3.2. Compatibility With Legacy Clients

   The multi-cost extensions defined in this document should not break
   legacy implementations (that is, clients and servers which are not
   aware of these extensions). One way to achieve that would be to
   define a new media type for an array-valued Multi Cost Map. However,
   as indicated above, an array-valued Multi Cost Map is almost
   identical to a single-valued Cost Map, so it should be simple to
   write a parser which handles either type of cost map. Hence defining
   a new media type could result in a lot of wasteful duplication.

Therefore this document does not define any new media types.
Instead, as described below, it extends the specifications in the
ALTO Server's Information Resource Directory (IRD) so that legacy
clients will not request array-valued Cost Map resources. This
relies on the requirement that implementations MUST ignore unknown
fields ({8.3.7} in [RFC7285]).

3.3. Filtered Multi Cost Map Resources

   This document extends the Filtered Cost Map service to allow the same
   resource to return either a single-valued Cost Map, as defined in
   [RFC7285], or an array-valued Multi Cost Map, as defined in this
   document. An extended Filtered Cost Map resource has a new
   capability, "max-cost-types". The value is the maximum number of
   cost types this resource can return for one request. The existence
   of this capability means the resource understands the extensions in
   this document.

   For example, the following fragment from an IRD defines an extended
   Filtered Cost Map resource:

```
    "filtered-multicost-map" : {
      "uri" : "http://alto.example.com/multi/costmap/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "max-cost-types" : 3,
        "cost-type-names" : [ "num-routingcost",
                              "num-hopcount" ],
      ...
    }
```

   A legacy client will ignore the "max-cost-types" capability, and will
   send a request with the input parameter "cost-type" describing the
   desired cost metric, as defined in [RFC7285]. The ALTO Server will
   return a single-valued legacy Cost Map.

   However, a multi-cost-aware client will realize that this resource
   supports the multi-cost extensions, and can send a POST request with
   the new input parameter "multi-cost-types", whose value is an array
   of cost types. Because the request has the "multi-cost-types"
   parameter (rather than the "cost-type" parameter defined in the base
   protocol), the server realizes that the client also supports the
   extensions in this document, and hence responds with a Multi Cost
   Map, with the costs in the order listed in "multi-cost-types".

3.4. Endpoint Cost Service Resources

   This document uses the technique described above to extend Endpoint
   Cost Service to return array-valued costs to clients who also are
   aware of these extensions.

3.5. Full Cost Map Resources

   Because Full Cost Map resources are GET-mode requests, with no
   capabilities other than the name of the cost type they return, it is
   not possible to define an array-valued Full Cost Map resource so that
   multi-cost-aware clients can recognize it, but legacy clients will
   ignore it.

   However {11.3.2.3} of [RFC7285] requires a Filtered Cost Map to
   return the entire Cost Map if the client omits the source and
   destination PIDs. Hence a client can use an extended Filtered Cost
   Map resource to get a full Multi Cost Map.

3.6. Extended Constraint Tests

   [RFC7285] defines a simple constraint test capability for Filtered
   Cost Maps and Endpoint Cost Services. If a resource supports
   constraints, the server restricts the response to costs that satisfy
   a list of simple predicates provided by the client. For example, if
   the client gives the constraints

        "constraints": ["ge 10", "le 20"]

   Then the server only returns costs in the range [10,20].

   To be useful with multi-cost requests, the constraint tests require
   several extensions. First, because a multi-cost request involves
   more than one cost metric, the simple predicates must be extended to
   specify the metric to test. Therefore we extend the predicate syntax
   to "[##] op value", where "##" is the index of a cost metric in this
   multi cost request.

   Second, the "AND" of simple predicates is not sufficient; to be
   useful, clients must be able to express "OR" tests. Hence we add a
   new field, "or-constraints", to the client request. The value is an
   array of arrays of simple predicates, and represents the OR of ANDs
   of those predicates.

   Thus the following request tells the server to limit its response to
   cost points with "routingcost" <= 100 AND "hopcount" <= 2, OR else
   "routingcost" <= 10 AND "hopcount" <= 6:

```
     {
       "multi-cost-types": [
           {"cost-metric": "routingcost", "cost-mode": "numerical"},
           {"cost-metric": "hopcount",    "cost-mode": "numerical"}
       ],
       "or-constraints": [
           ["[0] le 100", "[1] le 2"],
           ["[0] le 10",  "[1] le 6"]
       ],
       "pids": {...}
     }
```

   Finally, a client might want to test a cost type whose actual value
   is irrelevant, as long as it satisfies the tests. For example, the
   following request tells the server to return just "routingcost" for
   those source and destination pairs for which "hopcount" is <= 6:

```
     {
       "multi-cost-types": [
           {"cost-metric": "routingcost", "cost-mode": "numerical"},
       ],
       "testable-cost-types": [
           {"cost-metric": "hopcount", "cost-mode": "numerical"},
       ],
       "constraints": ["[0] le 6"],
       "pids": {...}
     }
```

   In this example, "[0]" means the constraint applies to "hopcount"
   because that is the first cost type in the "testable-cost-types"
   parameter.

4. Protocol Extensions for Multi-Cost ALTO Transactions

4.1. Filtered Cost Map Extensions

   This document extends Filtered Cost Maps, as defined in {11.3.2} of
   [RFC7285], by adding new input parameters and capabilities, and by
   returning JSONArrays instead of JSONNumbers as the cost values.

   The media type (11.3.2.1}, HTTP method (11.3.2.2} and "uses"
   specifications (11.3.2.5} are unchanged.

4.1.1. Accept Input Parameters

   The ReqFilteredCostMap object in {11.3.2.3} of [RFC7285] is extended
   as follows:

```
      object {
         [CostType cost-type;]
         [CostType multi-cost-types<1..*>;]
         [CostType testable-cost-types<1..*>;]
         [JSONString constraints<0..*>;]
         [JSONString or-constraints<0..*>;]
          PIDFilter pids;
      } ReqFilteredCostMap;

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

cost-type: If present, as defined in {11.3.2.3} of [RFC7285], with
   the additional requirement that the client MUST provide either
   "cost-type" or "multi-cost-types", but MUST NOT provide both.

multi-cost-types: If present, the ALTO Server MUST return array-
   valued costs for the cost types in thls list. For each entry, the
   "cost-metric" and "cost-mode" fields MUST match one of the
   supported cost types indicated in this resource's "capabilities"
   field (Section 4.1.2). The client MUST NOT use this field unless
   this resource's "max-cost-types" capability exists and has a value
   greater than 0. The client MUST specify either "cost-type" or
   "multi-cost-types", but MUST NOT specify both.

testable-cost-types: A list of cost types for extended constraint
   tests, as described for the "constraints" parameter. If present,
   the cost types must be a subset of the cost types in the
   resource's "testable-cost-type-names" capability (Section 4.1.2).

constraints: Unless this resource's "max-cost-types" capability
   (Section 4.1.2) is defined with a value greater than 0, this
   parameter is an array of constraint tests as defined in {11.3.2.3}
   of [RFC7285].

   If this resource's "max-cost-types" capability is greater than 0,
   then this parameter MUST be an array of extended constraint tests,
   where each test consists of two or three entities separated by
   white space: (1) an optional cost type index, of the form "[#]",
   with default value "[0]", (2) a required operator, and (3) a
   required target value. The operator and target value are as
   defined in {11.3.2.3} of [RFC7285]. The cost type index specifies
   the cost type to test. If the "testable-cost-types" parameter is
   present, assuming the index is "i", the test applies to the i'th
   cost type in "testable-cost-types" (starting with index 0).
   Otherwise, if the "multi-cost-types" parameter is present, the

test applies to the i'th cost type in "multi-cost-types". If
neither of those parameters are present, the test applies to the
cost type in the "cost-type" parameter. In this case, the index
MUST be 0. Regardless of how the tested cost type is selected, it
MUST be a cost type in the resource's "testable-cost-type-names"
capability, or, if omitted, the resource's "cost-type-names"
capability.

This parameter MUST NOT be specified if the "or-constraints"
parameter is specified, or if the resource's "cost-constraints"
capability is false.

Note that this feature allows a client to test cost types which
the server does not return. For example, suppose "multi-cost-
types" has the single element "routingcost", "testable-cost-types"
has the single element "hopcount", and "constraints" has the
single element "[0] le 5". This is equivalent to the database
query "SELECT routingcost WHERE hopcount <= 5".

Also note that as long as this resource's "max-cost-types"
capability is greater than 0, a client may use the extended
constraint tests even on single-valued cost map requests, that is,
requests with the "cost-type" parameter rather than "multi-cost-
types".

or-constraints: A JSONArray of JSONArrays of JSONStrings, where each
string is a constraint test as defined for the "constraints"
parameter. The constraint tests are interpreted as the logical OR
of ANDs. That is, the ALTO Server should return a cost point only
if it satisfies all constraints in any one of the sub-arrays.
This parameter MUST NOT be specified unless this resource's "cost-
constraints" capability is "true" and its "max-cost-types"
capability is defined with a value greater than 0 (Section 4.1.2).

This parameter MUST NOT be specified if the "constraints"
parameter is specified.

Note that if the "max-cost-types" capability has a value greater
than 0, a client MAY use the "or-constraints" parameter together
with the "cost-type" parameter. That is, if the client and server
are both aware of the extensions in this document, a client MAY
use an "OR" test for a single-valued cost request.

pids, srcs, dsts: As defined in {11.3.2.3} of [RFC7285].

4.1.2. Capabilities

   The FilteredCostMapCapabilities object in {11.3.2.4} is extended as
   follows:

       object {
          JSONString cost-type-names<1..*>;
          [JSONBool cost-constraints;]
          [JSONNumber max-cost-types;]
          [JSONString testable-cost-type-names<0..*>;]
       } FilteredCostMapCapabilities;

   max-cost-types: If present with value N greater than 0, this
      resource understands the multi-cost extensions in this document,
      and can return a Multi Cost Map with any combination of N or fewer
      cost types in the "cost-type-names" list. If omitted, the default
      value is 0.

   testable-cost-type-names: If present, and if "cost-constraints" is
      true, the resource only allows constraint tests on the cost type
      names in this array. Each name in "testable-cost-type-names" MUST
      be in "cost-type-names". If omitted or empty, the default is the
      value of the "cost-type-names" capability.

   cost-type-names and cost-constraints: As defined in {11.3.2.4} of
      [RFC7285].

   Note that "testable-cost-type-names" allows an ALTO Server to provide
   constraint tests on some, but not all, cost types.

4.1.3. Response

   If the client specifies the "cost-type" input parameter, the response
   is exactly as defined in {11.2.3.6} of [RFC7285]. If the client
   provides the "multi-cost-types" instead, then the response is changed
   as follows:

   o In "meta", the field "cost-type" is replaced with the field
      "multi-cost-types", with the same value as the "multi-cost-types"
      input parameter.

   o The costs are JSONArrays, instead of JSONNumbers. All arrays have
      the same cardinality as the "multi-cost-types" input parameter,
      and contain the cost type values in that order. If a cost type is
      not available for a particular source and destination, the ALTO
      Server MUST use the JSON null value for that array element. If
      none of the cost types are available for a particular source and
      destination, the ALTO Server MAY omit the entry for that source

      and destination.

4.2. Endpoint Cost Service Extensions

   This document extends the Endpoint Cost Service, as defined in
   {11.5.1} of [RFC7285], by adding new input parameters and
   capabilities, and by returning JSONArrays instead of JSONNumbers as
   the cost values.

   The media type (11.5.1.1}, HTTP method (11.5.1.2} and "uses"
   specifications (11.5.1.5} are unchanged.

4.2.1. Accept Input Parameters

   The ReqEndpointCostMap object in {11.5.1.3} of [RFC7285] is extended
   as follows:

      object {
         [CostType cost-type;]
         [CostType multi-cost-types<1..*>;]
         [CostType testable-cost-types<1..*>;]
         [JSONString constraints<0..*>;]
         [JSONString or-constraints<0..*>;]
          EndpointFilter endpoints;
      } ReqFilteredCostMap;

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

   cost-type: As defined in {11.5.1.3} of [RFC7285], with the
      additional requirement that the client MUST specify either "cost-
      type" or "multi-cost-types", but not both.

   multi-cost-types: If present, the ALTO Server MUST return array-
      valued costs for the cost types in this list. For each entry, the
      "cost-metric" and "cost-mode" fields MUST match one of the
      supported cost types indicated in this resource's "capabilities"
      field (Section 4.2.2). The client MUST NOT use this field unless
      this resource's "max-cost-types" capability exists and has a value
      greater than 0. Although optional, the client MUST specify either
      "cost-type" or "multi-cost-types". The client MUST NOT specify
      both.

testable-cost-types, constraints, or-constraints: Defined
   equivalently to the corresponding input parameters for an extended
   Filtered Cost Map (Section 4.1.1).

endpoints, srcs, dsts: As defined in {11.5.1.3} of [RFC7285].

4.2.2. Capabilities

The extensions to the Endpoint Cost Service capabilities are
identical to the extensions to the Filtered Cost Map (see
Section 4.1.2).

4.2.3. Response

The extensions to the Endpoint Cost Service response are similar to
the extensions to the Filtered Cost Map response (Section 4.1.3).
Specifically, if the client specifies the "cost-type" input
parameter, the response is exactly as defined in {11.5.1.6} of
[RFC7285]. If the client provides the "multi-cost-types" instead,
then the response is changed as follows:

o In "meta", the field "cost-type" is replaced with the field
   "multi-cost-types", with the same value as the "multi-cost-types"
   input parameter.

o The costs are JSONArrays, instead of JSONNumbers. All arrays have
   the same cardinality as the "multi-cost-types" input parameter,
   and contain the cost type values in that order. If a cost type is
   not available for a particular source and destination, the ALTO
   Server MUST use the JSON null value for that array element. If
   none of the cost types are available for a particular source and
   destination, the ALTO Server MAY omit the entry for that source
   and destination.

5. Examples

5.1. Information Resource Directory

The following is an example of an ALTO Server's Information Resource
Directory. In addition to Network and Cost Map resources, it defines
a Filtered Cost Map and an Endpoint Cost Service, both which
understand the multi-cost extensions.

```
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: [TODO]
   Content-Type: application/alto-directory+json

   {
     "meta" : {
       "default-alto-network-map" : "my-default-network-map",
       "cost-types" : {
          "num-routing" : {
            "cost-mode" : "numerical",
            "cost-metric" : "routingcost"
          },
          "num-hopcount" : {
            "cost-mode" : "numerical",
            "cost-metric" : "hopcount"
          },
            .....
            Other ALTO cost types as described
            in current ALTO Protocol
            .....
          }
     },
     "resources" : {
          "my-default-network-map" : {
            "uri" : "http://alto.example.com/networkmap",
            "media-type" : "application/alto-networkmap+json"
          },
          "numerical-routing-cost-map" : {
            "uri" : "http://alto.example.com/costmap/num-routing",
            "media-types" : [ "application/alto-costmap+json" ],
            "uses" : [ "my-default-network-map" ],
            "capabilities" : {
              "cost-type-names" : [ "num-routing" ]
            }
          },
          "numerical-hopcount-cost-map" : {
            "uri" : "http://alto.example.com/costmap/num-hopcount",
            "media-types" : [ "application/alto-costmap+json" ],
            "uses" : [ "my-default-network-map" ],
            "capabilities" : {
              "cost-type-names" : [ "num-hopcount" ]
            }
          },
          .........
          And other information resources as described in RFC7285
          .........
          "filtered-multicost-map" : {
            "uri" : "http://alto.example.com/multi/costmap/filtered",
```

```
            "media-types" : [ "application/alto-costmap+json" ],
            "accepts" : [ "application/alto-costmapfilter+json" ],
            "uses" : [ "my-default-network-map" ],
            "capabilities" : {
              "cost-constraints" : true,
              "max-cost-types" : 2,
              "cost-type-names" : [ "num-routingcost",
                                    "num-hopcount" ],
              "testable-cost-type-names" : [ "num-routingcost",
                                             "num-hopcount" ]
            }
          },
          "endpoint-multicost-map" : {
            "uri" : "http://alto.example.com/multi/endpointcost/lookup",
            "media-types" : [ "application/alto-endpointcost+json" ],
            "accepts" : [ "application/alto-endpointcostparams+json" ],
            "uses" : [ "my-default-network-map" ],
            "capabilities" : {
              "cost-constraints" : true,
              "max-cost-types" : 2,
              "cost-type-names" : [ "num-routingcost",
                                    "num-hopcount" ],
              "testable-cost-type-names" : [ "num-routingcost",
                                             "num-hopcount" ]
            }
          }
        }
      }
    }
```

5.2. Multi-Cost Filtered Cost Map: Example #1

   This example illustrates a static multi-cost ALTO transaction, where
   the utilized Cost Types all have static values. We assume that the
   Cost Types available at the ALTO Server are "routingcost" and
   "hopcount" and the "numerical" mode is available for both of them.
   The "routingcost" may be based on monetary considerations where as
   the "hopcount" is used to report on the path delay. We also assume
   that ALTO server does not know the value of the "routingcost" between
   PID2 and PID3, and hence uses 'null' for those costs.

```
    POST /multi/costmap/filtered" HTTP/1.1
    Host: alto.example.com
    Accept: application/alto-costmap+json,application/alto-error+json

    {
      "multi-cost-types": [
        {"cost-mode": "numerical", "cost-metric": "routingcost"},
        {"cost-mode": "numerical", "cost-metric": "hopcount"}
      ],
      "pids" : {
        "srcs" : [ ],
        "dsts" : [ ]
      }
    }


    HTTP/1.1 200 OK
    Content-Length: [TODO]
    Content-Type: application/alto-costmap+json

    {
     "meta" : {
       "dependent-vtags" : [
         {"resource-id": "my-default-network-map",
          "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
         }
       ],
       "multi-cost-types" : [
         {"cost-mode": "numerical", "cost-metric": "routingcost"},
         {"cost-mode": "numerical", "cost-metric": "hopcount"}
       ]
      }
     }
     "cost-map" : {
       "PID1": { "PID1":[1,0],    "PID2":[4,23],   "PID3":[10,5]    },
       "PID2": { "PID1":[15,5],  "PID2":[1,0],     "PID3":[null,9] },
       "PID3": { "PID1":[20,12], "PID2":[null,1], "PID3":[1,0]     }
     }
    }
```

5.3. Multi-Cost Filtered Cost Map: Example #2

   This is an example of using constraints to restrict the returned
   source/destination PID pairs to those with "routingcost" between 5
   and 10, or "hopcount" equal to 0.

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

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints" : [ ["[0] ge 5", "[0] le 10"],
                       ["[1] eq 0"] ]
  "pids" : {
    "srcs" : [ "PID1", "PID2" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}


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

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
    ]
  }
  "cost-map" : {
    "PID1": { "PID1": [1,0], "PID3": [10,5] },
    "PID2": { "PID2": [1,0]                 }
  }
}
```

5.4. Multi-Cost Filtered Cost Map: Example #3

   This example uses extended constraints to limit the response to cost
   points with ("routingcost" <= 10 and "hopcount" <= 2), or else
   ("routingcost" <= 2 and "hopcount" <= 6). Unlike the previous
   example, the client is only interested in the "routingcost" cost
   type, and uses the "cost-type" parameter instead of "multi-cost-

types" to tell the server to return scalar costs instead of array
costs:

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

{
  "cost-type" : {
    "cost-mode": "numerical", "cost-metric": "routingcost"
  },
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints": [
        ["[0] le 10", "[1] le 2"],
        ["[0] le 3",  "[1] le 6"]
  ],
  "pids" : {
    "srcs" : [ ],
    "dsts" : [ ]
  }
}


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

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

5.5. Endpoint Cost Service

   This example uses the Endpoint Cost Service to retrieve the
   "routingcost" and "hopcount" for selected endpoints, limiting the
   response to costs with either low hopcount and reasonable routingcost
   (hopcount <= 2 and routingcost <= 10), or else low routingcost and
   reasonable hopcount (routingcost <= 3 and hopcount <= 6).

```
   POST /multi/endpointcost/lookup HTTP/1.1
   Host: alto.example.com
   Content-Length: [TODO]
   Content-Type: application/alto-endpoincostparams+json
   Accept: application/alto-endpointcost+json,
           application/alto-error+json

   {
     "multi-cost-types" : [
       {"cost-mode": "numerical", "cost-metric": "routingcost"},
       {"cost-mode": "numerical", "cost-metric": "hopcount"}
     ],
     "or-constraints": [
           ["[0] le 10", "[1] le 2"],
           ["[0] le 3",  "[1] le 6"]
     ],
     "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: [TODO]
   Content-Type: application/alto-endpointcost+json

   {
     "meta" : {
       "multi-cost-types" : [
         {"cost-mode": "numerical", "cost-metric": "routingcost"},
         {"cost-mode": "numerical", "cost-metric": "hopcount"}
       ]
     }
     "endpoint-cost-map" : {
       "ipv4:192.0.2.2": {
         "ipv4:192.0.2.89":   [15, 5],
         "ipv4:203.0.113.45": [4, 23]
       }
     }
   }
```

6. IANA Considerations

   This document does define any new media types or introduce any new
   IANA considerations.

7. Privacy And Security Considerations

   This document does introduce any privacy or security issues not
   already present in the ALTO protocol.

8. Acknowledgements

   The authors would like to thank Vijay Gurbani, Dave Mac Dysan, Dhruv
   Dhodi and Young Lee for fruitful discussions and comments on this
   document and previous versions.

9. References

9.1. Normative References

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

   [RFC5693] "Application Layer Traffic Optimization (ALTO) Problem
              Statement", October 2009.

   [RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S.,
              Roome, W., Shalunov, S., and R. Woundy, "Application-Layer
              Traffic Optimization (ALTO) Protocol", RFC 7285,
              September 2014.

9.2. Informative References

   [RFC6708] "Application-Layer Traffic Optimization (ALTO)
              Requirements", February 2012.

   [draft-jenkins-alto-cdn-use-cases-01]
              ""Use Cases for ALTO within CDNs"
              draft-jenkins-alto-cdn-use-cases-01", June 2011.

   [draft-randriamasy-alto-cost-schedule-01]
              "ALTO Cost Schedule", July 2012.

Authors' Addresses

    Sabine Randriamasy (editor)
    Alcatel-Lucent/Bell Labs
    Route de Villejust
    NOZAY   91460
    FRANCE

    Email: Sabine.Randriamasy@alcatel-lucent.com

    Wendy Roome (editor)
    Alcatel-Lucent/Bell Labs
    600 Mountain Ave, Rm 3B-324
    Murray Hill, NJ   07974
    USA

    Phone: +1-908-582-7974
    Email: w.roome@alcatel-lucent.com

    Nico Schwan
    Thales Deutschland
    Lorenzstrasse 10
    Stuttgart   70435
    Germany

    Email: nico.schwan@thalesgroup.com