Internet Engineering Task Force (IETF)
Request for Comments: 8189
Category: Standards Track
ISSN: 2070-1721

S. Randriamasy
W. Roome
Nokia Bell Labs
N. Schwan
Thales Deutschland

October 2017

Multi-Cost Application-Layer Traffic Optimization (ALTO)

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol, specified in RFC 7285, defines several services that return various metrics describing the costs between network endpoints.

This document defines a new service that allows an ALTO Client to retrieve several cost metrics in a single request for an ALTO filtered cost map and endpoint cost map. In addition, it extends the constraints to further filter those maps by allowing an ALTO Client to specify a logical combination of tests on several cost metrics.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8189.

Table of Contents

1. Introduction

IETF has defined ALTO services in [RFC7285] to provide 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 such as the topological distance that affect performance of the data transmission between the hosts. 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 a view of the Internet called a Network Map, which is composed of provider-defined locations spanning from subnets to several Autonomous Systems (ASes). ALTO may also convey the provider-determined costs between Network Map locations or between groups of individual endpoints.

Current ALTO cost types provide values such as "hopcount" and administrative "routingcost" to reflect ISP routing preferences. Recently, new use cases have extended the usage scope of ALTO to Content Delivery Networks (CDNs), data centers, and applications that need additional information to select their endpoints or network locations. Thus, a multitude of new cost types that better reflect the requirements of these applications are expected to be specified.

The ALTO protocol [RFC7285], which this document refers to as the base protocol, restricts ALTO cost maps and Endpoint Cost Services to only one cost type per ALTO request. To retrieve information for several cost types, an ALTO Client must send several separate requests to the Server.

It is far more efficient, in terms of Round-Trip Time (RTT), traffic, and processing load on the ALTO Client and Server, to get all costs with a single query/response transaction. 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. Another advantage is consistency: providing values for several cost types in one single batch is useful for ALTO Clients needing synchronized ALTO information updates. This document defines how to retrieve multiple cost metrics in a single request for ALTO filtered cost maps and endpoint cost maps. To ensure compatibility with legacy ALTO Clients, only the Filtered Cost Map and Endpoint Cost Map Services are extended to return multi-cost values.

Along with multi-cost values queries, the filtering capabilities need to be extended to allow constraints on multiple metrics. The base protocol allows an ALTO Client to provide optional constraint tests for a Filtered Cost Map Service or the Endpoint Cost Service, where

the constraint tests are limited to the AND combination of comparison tests on the value of the (single) requested cost type. However, 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 this end, this document extends the base protocol with constraints that may test multiple metrics and may 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 the formal definitions. Section 5 gives several complete examples. The remaining sections describe the IANA, privacy, and security considerations.

1.1. Requirements Language

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

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

2. Terminology

- ALTO transaction: A request/response exchange between an ALTO Client and an ALTO Server.

- Client: When used with a capital "C", this term refers to an ALTO Client.

- Endpoint (EP): An endpoint is defined as in Section 2.1 of [RFC7285]. It can be, for example, a peer, a CDN storage location, a physical server involved in a virtual server-supported application, a party in a resource-sharing swarm such as a computation grid, or an online multi-party game.

- Server: When used with a capital "S", this term refers to an ALTO Server.

3. Overview Of Approach

The following is a non-normative overview of the multi-cost ALTO 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 [RFC7159] (see the DstCosts object in Section 11.2.3.6 of [RFC7285]). However, that section also says that an implementation may assume costs are JSON numbers, unless the implementation is using an extension that 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. The multiple metrics are listed in member "multi-cost-types", indicating to the Client how to map values in the array to cost metrics.

```
{
 "meta" : {
   "dependent-vtags" : [ ... ],
   "cost-type" : {},
   "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] },
   ...
 }
}
```

Note also the presence of member '"cost-type" : {}' to maintain backwards compatibility with [RFC7285].

3.2. Compatibility with Legacy ALTO Clients

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 multi-cost map resources. This relies on the requirement that ALTO Clients MUST ignore unknown fields (Section 8.3.7 of [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-type" : "application/alto-costmap+json",
     "accepts" : "application/alto-costmapfilter+json",
     "uses" : [ "my-default-network-map" ],
     "capabilities" : {
       "max-cost-types" : 2,
       "cost-type-names" : [ "num-routingcost",
                             "num-hopcount" ],
       ...
     }
```

A legacy ALTO 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 ALTO 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 ALTO 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

Section 4.1.4 of [RFC7285] specifies that "The Endpoint Cost Service allows an ALTO server to return costs directly amongst endpoints", whereas the Filtered Cost Map Service returns costs amongst Provider-defined Identifiers (PIDs). This document uses the technique described in Section 3.3 to extend the Endpoint Cost Service to return array-valued costs to ALTO Clients who also are aware of these extensions.

3.5. Full Cost Map Resources

Section 11.3.2.3 of [RFC7285] requires a filtered cost map to return the entire cost map if the ALTO Client omits the source and destination PIDs. Hence, a multi-cost-aware ALTO Client can use an extended Filtered Cost Map resource to get a full multi-cost map.

Full cost map resources are GET-mode requests. The response for a full cost map conveying multiple cost types would include a "meta" field that would itself include a "cost-type" field that would list several values corresponding to the cost types of the cost map. A legacy ALTO Client would not be able to understand this list. Neither would it be able to interpret the cost values array provided by a full multi-cost map.

3.6. Extended Constraint Tests

[RFC7285] defines a simple constraint test capability for Filtered Cost Map 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 ALTO Client. For example, if the ALTO Client gives the following 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.

3.6.1. Extended Constraint Predicates

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.

3.6.2. Extended Logical Combination of Predicates

Second, once multiple cost metrics are involved, the "AND" of simple predicates is no longer 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": {...}
}
```

Note that a "constraints" parameter with the array of predicates [P1, P2, ...] is equivalent to an "or-constraints" parameter with one array of value [[P1, P2, ...]]. A Client is therefore allowed to express either "constraints" or "or-constraints" but not both.

3.6.3. Testable Cost Types in Constraints

Finally, a Client may want to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a Client may want the value of the cost metric "routingcost" for all PID pairs that satisfy constraints on the metric "hopcount", without needing the actual value of "hopcount".

To this end, we add a specific parameter named "testable-cost-types" that does not contain the same cost types as parameter "multi-cost-types". The Client can express constraints only on cost types listed in "testable-cost-types".

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": {...}
}
```

3.6.4. Testable Cost Type Names in IRD Capabilities

In [RFC7285], when a resource's capability "constraints" is true, the Server accepts constraints on all the cost types listed in the "cost-type-names" capability. However, some ALTO Servers may not be willing to allow constraint tests on all available cost metrics. Therefore, the multi-cost ALTO protocol extension defines the capability field "testable-cost-type-names". Like "cost-type-names", it is an array of cost type names. If present, that resource only allows constraint tests on the cost types in that list. "testable-cost-type-names" must be a subset of "cost-type-names".

3.6.5. Legacy ALTO Client Issues

While a multi-cost-aware Client will recognize the "testable-cost-type-names" field and will honor those restrictions, a legacy Client will not. Hence, when "constraints" has the value 'true', a legacy Client may send a request with a constraint test on any of the cost types listed in "cost-type-names".

To avoid that problem, the "testable-cost-type-names" and "cost-constraints" fields are mutually exclusive: a resource may define one or the other capability but MUST NOT define both. Thus, a resource that does not allow constraint tests on all cost metrics will set "testable-cost-type-names" to the testable metrics and will set "cost-constraints" to 'false'. A multi-cost-aware Client will recognize the "testable-cost-type-names" field and will realize that its existence means the resource does allow (limited) constraint

tests, while a legacy Client will think that resource does not allow constraint tests at all. To allow legacy Clients to use constraint tests, the ALTO Server can define an additional resource with "cost-constraints" set to ’true’ and "cost-type-names" set to the metrics that can be tested.

In the IRD example below, the resource "filtered-cost-map-extended" provides values for three metrics: "num-routingcost", "num-hopcount", and "num-bwscore". The capability "testable-cost-type-names" indicates that the Server only allows constraints on "routingcost" and "hopcount". A multi-cost-capable Client will see this capability and will limit its constraint tests to those metrics. Because capability "cost-constraints" is false (by default), a legacy Client will not use constraint tests on this resource at all.

The second resource, "filtered-multicost-map", is similar to the first, except that all the metrics it returns are testable. Therefore, it sets "cost-constraints" to ’true’ and does not set the "testable-cost-type-names" field. A legacy Client that needs a constraint test will use this resource rather than the first. A multi-cost-aware Client that does not need to retrieve the "num-bwscore" metric may use either resource.

Note that if a multi-cost Server specifies a "filtered-cost-map-extended", it will most likely not specify an "filtered-multicost-map" if the capabilities of the latter are covered by the capabilities of the former or unless the "filtered-multicost-map" resource is also intended for legacy Clients.

```
   "filtered-cost-map-extended" : {
      "uri" : "http://alto.example.com/multi/extn/costmap/filtered",
      "media-type" : "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",
                               "num-bwscore"],
         "testable-cost-type-names" : [ "num-routingcost",
                                        "num-hopcount" ]
      }
   },

   "filtered-multicost-map" : {
      "uri" : "http://alto.example.com/multi/costmap/filtered",
      "media-type" : "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"],
      }
   }
```

4. Protocol Extensions for Multi-Cost ALTO Transactions

This section formally specifies the extensions to [RFC7285] to support multi-cost ALTO transactions.

This document uses the notation rules specified in Section 8.2 of [RFC7285]. In particular, an optional field is enclosed by [ ]. In the definitions, the JSON names of the fields are case sensitive. An array is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of values and n is the maximum. When this document uses * for n, it means no upper bound.

4.1. Filtered Cost Map Extensions

This document extends Filtered Cost Maps, as defined in Section 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, HTTP method, and "uses" specifications (described in Sections 11.3.2.1, 11.3.2.2, and 11.3.2.5 of [RFC7285], respectively) are unchanged.

4.1.1. Capabilities

The filtered cost map capabilities are extended with two new members:

- max-cost-types
- testable-cost-type-names

The capability "max-cost-types" indicates whether this resource supports the multi-cost ALTO extensions, and the capability "testable-cost-type-names" allows the resource to restrict constraint tests to a subset of the available cost types. With these two additional members, the FilteredCostMapCapabilities object in Section 11.3.2.4 of [RFC7285] is structured as follows:

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

cost-type-names: As defined in Section 11.3.2.4 of [RFC7285].

cost-constraints: As defined in Section 11.3.2.4 of [RFC7285]. Thus, if "cost-constraints" is true, the resource MUST accept constraint tests on any cost type in "cost-type-names". In addition, note that if "cost-constraints" is true, the "testable-cost-type-names" capability MUST NOT be present.

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, the resource allows constraint tests, but only on the cost type names in this array. Each name in "testable-cost-type-names" MUST also be in "cost-type-names". If "testable-cost-type-names" is present, the "cost-constraints" capability MUST NOT be true.

As discussed in Section 3.6.4, this capability is useful when a Server is unable or unwilling to implement constraint tests on all cost types. As discussed in Section 3.6.5, "testable-cost-type-names" and "cost-constraints" are mutually exclusive to prevent legacy Clients from issuing constraint tests on untestable cost types.

4.1.2. Accept Input Parameters

The ReqFilteredCostMap object in Section 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<1..*><1..*>;]
   [PIDFilter pids];
} ReqFilteredCostMap;
```

cost-type: As defined in Section 11.3.2.3 of [RFC7285], with the additional requirement that the Client MUST specify either "cost-type" or "multi-cost-types" but MUST NOT specify both. Therefore, this field is made optional. When placing a single cost request as specified in [RFC7285], a Client MUST use "cost-type".

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 member "cost-type-names" of this resource's "capabilities" field (Section 4.1.1). The Client MUST NOT use this field unless this resource's "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The Client MUST specify either "cost-type" or "multi-cost-types" but MUST NOT specify both.

Note that if "multi-cost-types" has one cost type, the values in the cost map will be arrays with one value.

testable-cost-types: A list of cost types used for extended constraint tests, as described for the "constraints" and "or-constraints" parameters. These cost types must either be a subset of the cost types in the resource's "testable-cost-type-names" capability (Section 4.1.1), or else, if the resource's capability "cost-constraints" is true, a subset of the cost types in the resource's "cost-type-names" capability.

If "testable-cost-types" is omitted, it is assumed to have the cost types in "multi-cost-types" or "cost-type".

This feature is useful when a Client wants to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a Client may want the cost metric "routingcost" for those PID pairs whose "hopcount" is less than 10. The exact hop count does not matter.

constraints: If this resource's "max-cost-types" capability (Section 4.1.1) has the value 0 (or is not defined), this parameter is as defined in Section 11.3.2.3 of [RFC7285]: an array of constraint tests related to each other by a logical AND. In this case, it MUST NOT be specified unless the resource's "cost-constraints" capability is true.

If this resource's "max-cost-types" capability has a value greater than 0, then this parameter is an array of extended constraint predicates as defined below and related to each other by a logical AND. In this case, it MAY be specified if the resource allows constraint tests (the resource's "cost-constraints" capability is true, or its "testable-cost-type-names" capability is not empty).

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

An extended constraint predicate 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 Section 11.3.2.3 of [RFC7285]. The cost type index, i, specifies the cost type to test. If the "testable-cost-type" parameter is present, 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 that array. If neither parameter is present, the test applies to the cost type in the "cost-type" parameter, in which case the index MUST be 0. Regardless of how the tested cost type is selected, it MUST be in the resource's "testable-cost-type-names" capability or, if not present, in the "cost-type-names" capability.

As an 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 and provide routingcost WHERE hopcount <= 5".

Note that the index is optional, so a constraint test as defined in Section 11.3.2.3 of [RFC7285], such as "le 10", is equivalent to "[0] le 10". Thus, legacy constraint tests are also legal extended constraint tests.

Note that a "constraints" parameter with the array of extended predicates [P1, P2, ...] is equivalent to an "or-constraints" parameter as defined below with the value [[P1, P2, ...]].

or-constraints: A JSONArray of JSONArrays of JSONStrings, where each string is an extended constraint predicate as defined above. The "or-constraint" tests are interpreted as the logical OR of ANDs of predicates. 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 MAY be specified if this resource's "max-cost-types" capability is defined with a value greater than 0 (Section 4.1.1) and if the resource allows constraint tests (the resource's "cost-constraints" capability is true, or its "testable-cost-type-names" capability is not empty). Otherwise, this parameter MUST NOT be specified.

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

This parameter MUST NOT contain any empty array of AND predicates. An empty array would be equivalent to a constraint that is always true. An OR combination including such a constraint would be always true and thus useless.

As an example, suppose "multi-cost-types" has the two elements "routingcost" and "bandwidthscore", "testable-cost-types" has the two elements "routingcost" and "hopcount", and "or-constraints" has the two elements ["[0] le 100", "[1] le 2"] and ["[0] le 10", "[1] le 6"]. This is equivalent to the words: "SELECT and provide routingcost and bandwidthscore WHERE ("routingcost" <= 100 AND "hopcount" <= 2) OR ("routingcost" <= 10 AND "hopcount" <= 6)".

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: As defined in Section 11.3.2.3 of [RFC7285].

4.1.3. Response

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

- In "meta", the value of field "cost-type" will be ignored by the receiver and set to {}. Instead, the field "multi-cost-types" is added with the same value as the "multi-cost-types" input parameter.

- 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 Section 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, HTTP method, and "uses" specifications (described in Sections 11.5.1.1, 11.5.1.2, and 11.5.1.5 of [RFC7285], respectively) are unchanged.

4.2.1. Capabilities

The extensions to the Endpoint Cost Service capabilities are identical to the extensions to the Filtered Cost Map (see Section 4.1.1).

4.2.2. Accept Input Parameters

The ReqEndpointCostMap object in Section 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<1..*><1..*>;]
   EndpointFilter endpoints;
} ReqEndpointCostMap;
```

cost-type: As defined in Section 11.5.1.3 of [RFC7285], with the additional requirement that the Client MUST specify either "cost-type" or "multi-cost-types" but MUST NOT specify 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 4.2.1). The Client MUST NOT use this field unless this resource's "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The Client MUST specify either "cost-type" or "multi-cost-types" but MUST NOT specify both.

Note that if "multi-cost-types" has one cost type, the values in the cost map will be arrays with one value.

testable-cost-types, constraints, or-constraints: Defined equivalently to the corresponding input parameters for an extended filtered cost map (Section 4.1.2).

endpoints: As defined in Section 11.5.1.3 of [RFC7285].

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 Section 11.5.1.6 of [RFC7285]. If the Client provides the "multi-cost-types" instead, then the response is changed as follows:

- In "meta", the value of field "cost-type" will be ignored by the receiver and set to {}. Instead, the field "multi-cost-types" is added with the same value as the "multi-cost-types" input parameter.

- 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

This section provides examples of multi-cost ALTO transactions. It uses cost metrics, in addition to the mandatory legacy "routingcost", that are deliberately irrelevant and not registered with IANA.

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 two Filtered Cost Maps and an Endpoint Cost Service, which all 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: 2704
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-shoesize" : {
        "cost-mode" : "numerical",
        "cost-metric" : "shoesize"
      },
      "num-scenery" : {
        "cost-mode" : "numerical",
        "cost-metric" : "sceneryrate"
      }
    }
  },
  "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-type" : "application/alto-costmap+json",
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "cost-type-names" : [ "num-routing" ]
      }
    },
    "numerical-shoesize-cost-map" : {
      "uri" : "http://alto.example.com/costmap/num-shoesize",
      "media-type" : "application/alto-costmap+json",
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "cost-type-names" : [ "num-shoesize" ]
      }
    },
    "filtered-multicost-map" : {
      "uri" : "http://alto.example.com/multi/costmap/filtered",
      "media-type" : "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-shoesize" ]
        }
      },
      "filtered-cost-map-extended" : {
        "uri" : "http://alto.example.com/multi/extn/costmap/filtered",
        "media-type" : "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-shoesize",
                                "num-scenery"],
          "testable-cost-type-names" : [ "num-routingcost",
                                         "num-shoesize" ]
        }
      },
      "endpoint-multicost-map" : {
        "uri" : "http://alto.example.com/multi/endpointcost/lookup",
        "media-type" : "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-shoesize" ]
        }
      }
    }
  }
```

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

This example illustrates a simple multi-cost ALTO transaction. The ALTO Server provides two cost types, "routingcost" and "shoesize", both in "numerical" mode. The Client wants the entire multi-cost map. The Server does not know the value of "routingcost" between PID2 and PID3 and hence returns the value ’null’ for "routingcost" between PID2 and PID3.

```
POST /multi/costmap/filtered" HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 206

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


HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
Content-Length: 549

{
 "meta" : {
   "dependent-vtags" : [
     {"resource-id": "my-default-network-map",
      "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
     }
   ],
   "cost-type" : {},
   "multi-cost-types" : [
     {"cost-mode": "numerical", "cost-metric": "routingcost"},
     {"cost-mode": "numerical", "cost-metric": "shoesize"}
   ]
 }
 "cost-map" : {
   "PID1": { "PID1":[1,0],   "PID2":[4,3],    "PID3":[10,2]   },
   "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 example uses constraints to restrict the returned source/ destination PID pairs to those with "routingcost" between 5 and 10 or "shoesize" equal to 0.

```
POST /multi/costmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 333

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "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
Content-Length: 461

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "cost-type" : {},
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "shoesize"}
    ]
  }
  "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 "shoesize" <= 2), OR else ("routingcost" <= 3 AND "shoesize" <= 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.

In this example, "[0]" means the constraint applies to "routingcost" because that is the first cost type in the "testable-cost-types" parameter. (If "testable-cost-types" is omitted, it is assumed to be the same as "multi-cost-types".) The choice of using an index to refer to cost types aims at minimizing the length of the expression of constraints, especially for those combining several OR and AND expressions. It was also the shortest path from the constraints design in [RFC7285].

```
POST /multi/multicostmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 390

{
  "cost-type" : {
    "cost-mode": "numerical", "cost-metric": "routingcost"
  },
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "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
Content-Length: 368

{
  "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. Multi-Cost Filtered Cost Map: Example #4

This example uses extended constraints to limit the response to cost points with ("routingcost" <= 10 AND "shoesize" <= 2), OR else ("routingcost" <= 3 AND "shoesize" <= 6). In this example, the Client is interested in the "routingcost" and "sceneryrate" cost metrics but not in the "shoesize" metric:

```
POST /multi/extn/costmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 461

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "sceneryrate"}
  ],
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
```

```
    "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
  Content-Length: 481

  {
    "meta" : {
      "dependent-vtags" : [
        {"resource-id": "my-default-network-map",
         "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
        }
      ],
      "cost-type" : {},
      "multi-cost-types" : [
        {"cost-mode": "numerical", "cost-metric": "routingcost"},
        {"cost-mode": "numerical", "cost-metric": "sceneryrate"}
      ]
    }
    "cost-map" : {
      "PID1": { "PID1": [1,16] "PID3": [10,19] },
      "PID2": { "PID2": [1,8] },
      "PID3": { "PID3": [1,19] }
    }
  }
```

5.6. Endpoint Cost Service

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

```
POST /multi/endpointcost/lookup HTTP/1.1
Host: alto.example.com
Accept: application/alto-endpointcost+json,
        application/alto-error+json
```

```
Content-Type: application/alto-endpoincostparams+json
Content-Length: 455

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "or-constraints": [
        ["[0] le 10", "[1] le 2"],
        ["[0] le 3",  "[1] le 6"]
  ],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2", "ipv6:2001:db8::1:0 ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2001:db8::10"
    ]
  }
}


HTTP/1.1 200 OK
Content-Length: 419
Content-Type: application/alto-endpointcost+json

{
  "meta" : {
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "shoesize"}
    ]
  }
  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89":    [15, 5],
      "ipv4:203.0.113.45":  [4, 23]
    }
    "ipv6:2001:db8::1:0": {
      "ipv4:198.51.100.34": [16, 5],
      "ipv6:2001:db8::10":  [10, 2]
    }
  }
}
```

6. IANA Considerations

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

7. Privacy and Security Considerations

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

The multi-cost optimization even tends to reduce the on-the-wire data exchange volume compared to multiple single cost ALTO transactions. Likewise, the risk related to massive multi-cost requests is moderated by the fact that multi-cost constraints additionally filter ALTO Server responses and thus reduce their volume.

Note that, because queries for multiple metrics represent a stronger fingerprinting signal than queries for a single metric, implementations of this protocol may leak more information about the ALTO Client than would occur with a succession of individual queries. Though, in many cases, it would already be possible to link those queries by using the source IP address or other existing information.

8. References

8.1. Normative References

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

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

Acknowledgements

The authors would like to thank Richard Alimi, Fred Baker, Dhruv Dhodi, Vijay Gurbani, Dave Mac Dysan, Young Lee, and Richard Yang for fruitful discussions and feedback on this document and earlier draft versions. Gao Kai, Hans Seidel, Richard Yang, Qiao Xiang, and Wang Xin provided substantial review feedback and suggestions to the protocol design.

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
Nozay 91460
France

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

Wendy Roome
Nokia Bell Labs
124 Burlington Rd
Murray Hill, NJ 07974
United States of America

Email: ietf@wdroome.com

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com