ALTO Extension: Path Vector

draft-ietf-alto-path-vector-06

Kai Gao, Young Lee, Sabine Randriamasy
Y. Richard Yang, J. Jensen Zhang

IETF 104
March 26, 2019
Prague
Overview of Updates between IETF 102 and 104

• Main technical design
  • Finalized update on information structure: multipart/related

• Text updates
  • Clarify overview of approaches
  • Finalized update on "path-vector" cost-mode
  • Update entity domain and property type registry by following latest unified properties document
Recall: Information Structure of PV

- Fundamentally, path vector response structure consists of two maps
  - to remove redundancy; aka database normalized design should consist of two tables

```
"cost-map": {
  "PID1": { "PID2": ["ane:L001", "ane:L002"],
  ...
}

"prop-map": {
  "ane:L001": {"availbw": "100"},
  "ane:L002": {"availbw": "200"} ......
}
```

- Design I: Add “prop-map” in alto-costmap
  - Problem: break existing alto-costmap media type

- Design II: Send the two maps in two separate messages: path vector message and propmap message
  - Problem: Snapshot consistency
Recall: draft-ietf-alto-path-vector-03

- Introduce **MIME multipart/related** [RFC2387] to include two messages in a single response

### Request

```plaintext
POST /endpointcostmap/multicost HTTP/1.1
Host: alto.example.com
Accept: multipart/related, application/alto-costmap+json, application/alto-propmap+json, application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
    "multi-cost-types": [
        {
            "cost-mode": "...",
            "cost-metric": "..."
        },
        {
            "cost-mode": "numerical",
            "cost-metric": "routingcost"
        }
    ],
    "endpoints": {
        "srcs": ["ipv4:192.0.2.2"],
        "dsts": ["ipv4:192.0.2.89",
                  "ipv4:203.0.113.45",
                  "ipv6:2001:db8::10"
        ]
    }
}
```

### Response

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-2

```plaintext
--example-2
Content-Type: application/alto-endpointcost+json

{
    "meta": {
        ...
    },
    "endpoint-cost-map": ...
}
--example-2
Content-Type: application/alto-propmap+json

{
    "meta": {
        ...
    },
    "property-map": ...
}
--example-2--
```
Remaining Issue (1) of -03 multipart/related

- RFC7285 design: The media type of each IRD entry indicates the response type, which also indicates the type of service (information resource)
- "multipart/related" as media type will no longer indicate what the service is.
Remaining Issue(2) of -03 multipart/related

- Previous SSE only allows incremental update for ALTO response with only a **single** JSON object, but now PV multipart/related has two.

---

**Response**

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-2

```plaintext
--example-2
Content-Type: application/alto-networkmap+json

{ "data": { ... full network map message ... } }

--example-2
Content-Type: application/alto-costmap+json

{ "data": { ... full cost map message ... } }

--example-2
Content-Type: application/alto-merge-patch+json

{ "data": { ... JSON merge patch update for the cost map ... } }
```

---

**Figure 3: Examples of ALTO data update messages.**
**Idea**: consistent use of multipart/related media type grammar defined in RFC 2387:

- "multipart/related" has required "type=" parameter

**Solution**: add the required “type” parameter in media type in IRD, e.g.,

- “mediatype”
  =“multipart/related
type=application/alto-
costmap+json”
- “mediatype”
  =“multipart/related
type="application/alto-
endpointcostmap+json”
-06 Update Solving Remaining Issue (2)

• Idea: multipart/related body has content-id in response

• Solution:
  – A multipart/related PV message includes content-ids to identify the two objects
  – SSE uses a more generic data-id to identify the object to update
-06 Finalized Wording on PV Cost Type

- Apply the "consistency" principle (i.e., consistency with existing design)
  - Existing design
    - cost mode: numerical, ordinal
    - cost metric: routingcost, bw
  - Consistent PV cost type:
    - cost mode: path-vector
    - cost metric: indicates the metric using which the abstract network elements are computed

indicate data type of each cost map element: float/int respectively
indicate semantics
Put-it-together Example: PV IRD

```
"meta": {
  "cost-types": {
    "pv-bw": {
      "cost-mode": "path-vector",
      "cost-metric": "availbw"
    }
  }
},
"ecs-pv": {
  "uri": "http://alto.exmaple.com/endpointcost/pv",
  "media-type": "multipart/related; type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": ["pv-bw"]
  }
},
"cm-pv": {
  "uri": "http://alto.exmaple.com/costmap/pv",
  "media-type": "multipart/related; type=application/alto-costmap+json",
  "accepts": "application/alto-filteredcostparams+json",
  "capabilities": {
    "cost-type-names": ["pv-bw"]
  }
}
```
Put-it-together Example: PV Query and Response

**Request**

```plaintext
POST /endpointcost/pv HTTP/1.1  
Host: alto.example.com  
Accept: multipart/related;  
    type=application/alto-endpointcost+json,  
    application/alto-error+json  
Content-Length: [TBD]  
Content-Type:  
    application/alto-endpointcostparams+json  

{  
    "cost-type": {  
        "cost-mode": "path-vector",  
        "cost-metric": "availbw"  
    },  
    "endpoints": {  
        "srcs": [ "ipv4:192.0.2.2" ],  
        "dsts": [ "ipv4:192.0.2.89",  
                    "ipv4:203.0.113.45" ]  
    }  
}
```

**Response**

```plaintext
HTTP/1.1 200 OK  
Content-Length: [TBD]  
Content-Type: multipart/related;  
    boundary=example-2; start=pvmap;  
    type=application/alto-endpointcost+json  

--example-2  
Content-ID: pvmap  
Content-Type: application/alto-endpointcost+json  

{  
    "meta": {  
        "vtag": <vtag>,  
        "cost-type": ...  
    },  
    "endpoint-cost-map": <endpoint-cost-map>  
}  
--example-2  
Content-ID: nepmap  
Content-Type: application/alto-propmap+json  

{  
    "meta": {  
        "dependent-vtags": [ <vtag> ],  
        "property-map": <property-map>  
    }  
}
```
Get-it-together Example: PV Incremental Update (1)

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

{ "add": {
  "ecspvsub1": {
    "resource-id": "ecs-pv",
    "input": <ecs-input>
  } } }
Example: PV Incremental Update (2)

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

{ "add": {
  "ecspvsub1": {
    "resource-id": "ecs-pv",
    "input": <ecs-input>
  }
}}

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

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

data: --example-2
data: Content-ID: pvmap
data: Content-Type: application/alto-endpointcost+json
data:
data: <endpoint-cost-map-entry>
data: --example-2
data: Content-ID: nepmap
data: Content-Type: application/alto-propmap+json
data:
data: <property-map-entry>

data: --example-2--

event: multipart/related;boundary=example-2;start=pvmap;type=application/alto-endpointcost+json,ecspvsub1
data: Content-ID: pvmap
data: Content-Type: application/alto-endpointcost+json
data:
data: <endpoint-cost-map-entry>

...
Discussion and Next Steps

• The authors feel that they finally got it right, but appreciate any feedback

• Wait for SSE and UP finalization and then proceed with finalization
Q & A

Thanks