Updates in -02

• We assume HTTP+JSON as base protocol (ALTO Network & Cost Maps)

• Determine Map Version
  – Conclusion: Existing HTTP/1.1 mechanisms useful for conditional requests, but not for incremental updates
  • To be discussed: HTTP delta encoding [RFC 3229] as potential option
  – Proposal: (cost)-map-vtag plus update-interval

• Incremental Update Options
  – New text: HTTP Byte range
  – Detailed specification of ALTO extension

• New section: Numerical evaluation
  – Cost Map size
  – JSON Patch vs. ALTO extension
Incremental Update

Request:
  – New MIME Type: "application/alto-update-param+json"
  – Input parameter: {"reference-tag": "1266506140"}

Response (options):
• JSON Patch (draft-ietf-appsawg-json-patch-02)
  – Supports "add", "remove", "replace", "move", "copy" or "test" operations
  – Example:

    { "replace": "meta/data/map/SRC-PID/DEST-PID", "value": 123 }

• ALTO Extension
  – Syntactically equal to Filtered Map services
  – Contains only changed values (or -1 for delete)
  – Example:

  "SRC-PID": {"DEST-PID": 123}
Numerical Evaluation

Full Cost Map Size

<table>
<thead>
<tr>
<th>np</th>
<th>cost-map size [bytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1,640</td>
</tr>
<tr>
<td>25</td>
<td>9,725</td>
</tr>
<tr>
<td>50</td>
<td>38,200</td>
</tr>
<tr>
<td>100</td>
<td>151,400</td>
</tr>
<tr>
<td>250</td>
<td>941,000</td>
</tr>
<tr>
<td>500</td>
<td>3,757,000</td>
</tr>
<tr>
<td>1000</td>
<td>15,014,000</td>
</tr>
<tr>
<td>2500</td>
<td>93,785,000</td>
</tr>
<tr>
<td>5000</td>
<td>375,070,000</td>
</tr>
</tbody>
</table>

→ For less than 100 PIDs incr. updates are not needed

→ For maps that require greater accuracy incr. updates are required

JSON Patch vs. ALTO Extension

<table>
<thead>
<tr>
<th>np</th>
<th>JSON patch [bytes]</th>
<th>ALTO Cost Map message [bytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>220</td>
<td>95</td>
</tr>
<tr>
<td>50</td>
<td>5,500</td>
<td>2,375</td>
</tr>
<tr>
<td>100</td>
<td>22,000</td>
<td>9,500</td>
</tr>
<tr>
<td>250</td>
<td>137,500</td>
<td>59,375</td>
</tr>
<tr>
<td>500</td>
<td>550,000</td>
<td>237,500</td>
</tr>
<tr>
<td>1000</td>
<td>2,200,000</td>
<td>950,000</td>
</tr>
<tr>
<td>2500</td>
<td>13,750,000</td>
<td>5,937,500</td>
</tr>
<tr>
<td>5000</td>
<td>55,000,000</td>
<td>23,750,000</td>
</tr>
</tbody>
</table>

→ ALTO extension has a higher efficiency in terms of encoded bytes

np: Number of PIDS
Average length of PID names = 8 characters
Average number of digits in costs = 3 digits
Fraction of cost values changed = 5%
Numerical Evaluation

Note: When several destination costs change for the same source PID, the ALTO extension can be much more efficient than JSON Patch.

• JSON Patch

```json
{ "replace": "meta/data/map/SRC-PID/DEST-1", "value": 123 }
{ "replace": "meta/data/map/SRC-PID/DEST-2", "value": 321 }
{ "replace": "meta/data/map/SRC-PID/DEST-3", "value": 456 }
{ "replace": "meta/data/map/SRC-PID/DEST-4", "value": 654 }
```

• ALTO Extension

```
"SRC-PID": {"DEST-1": 123, "DEST-2": 321, "DEST-3": 456, "DEST-4": 654}
```
Architectural Considerations

- **JSON Patch**
  - Upcoming Standard
  - To be supported in libraries

- **ALTO Extension**
  - Syntactically equal to Filtered Maps
Architectural Considerations

- **JSON Patch**
  - Upcoming Standard
  - To be supported in libraries

- **ALTO Extension**
  - Syntactically equal to Filtered Maps

Diagram:

- Full Map JSON -> JSON Parser -> ALTO Client DB
- Full Map stored in internal DB
- Incr. Update JSON Patch
- Incr. Update ALTO Ext.
Architectural Considerations

- **JSON Patch**
  - Upcoming Standard
  - To be supported in libraries

- **ALTO Extension**
  - Syntactically equal to Filtered Maps

---

**Diagram:**
- Full Map JSON
  - JSON Parser
    - (+ JSON Patch)
  - ALTO Client DB
    - Full Map stored in internal DB
      - Standard JSON lib requires original resource to create updated full map
        - Original resource needs to be cached
        - Internal DB refreshed entirely

- Incremental Update JSON Patch
- Incremental Update ALTO Ext.
Architectural Considerations

- **JSON Patch**
  - Upcoming Standard
  - To be supported in libraries

- **ALTO Extension**
  - Syntactically equal to Filtered Maps

Full Map stored in internal DB

Standard JSON lib requires original resource to create updated *full map*

- Original resource needs to be cached
- Internal DB *refreshed entirely*

ALTO extension can be based on Filtered Maps implementation and work on DB directly
Architectural Considerations

- **JSON Patch**
  - Upcoming Standard
  - To be supported in libraries

- **ALTO Extension**
  - Syntactically equal to Filtered Maps

Diagram:

- Full Map JSON
- Incremental Update JSON Patch
- Incremental Update ALTO Ext.

Flow:

1. Full Map JSON → JSON Parser
2. Incremental Update JSON Patch → JSON Patch Parser
3. Incremental Update ALTO Ext. → JSON Parser

→ ALTO Client DB

- Full Map stored in internal DB
- Standard JSON lib requires original resource to create updated full map
  → Original resource needs to be cached
  → Internal DB refreshed entirely
- Probably not supported by standard libs
- ALTO extension can be based on Filtered Maps implementation and work on DB directly
Summary

• Current draft identifies incremental update options
  – Determine Map Version
  – Incremental Update Options
  – Numerical evaluation

• JSON Patch vs. ALTO Extension
  – No clear winner
    • JSON Patch: Upcoming standard
    • ALTO Extension: Higher efficiency

• Opinions?