Extensible Property Maps for the ALTO Protocol

draft-roome-alto-unified-props-new-01

Y. Richard Yang

IETF 98
March 31, 2017

Updates

- Two versions submitted before WG meeting
  - Mar. 11: Revise of structure from early version
  - Mar. 28: A lot of small edits from earlier versions, according to feedback.
    - Clarified that LPM will not lead to multiple inheritance
    - …
- Still several remaining (TODO) issues according to individual feedback (see end of slides)
- Key design pointed posted on mailing list (Mar. 28) to seek feedback

Key Design Points (1/2)

- D1. The goal is to provide properties to entities.
- D2. Each entity must have an entity name to be identified. An entity name is a typed (domained) string, in a format of <domain>:<name>, e.g., "ipv4:192.1.1.1", "pid:myid1", "ane:myane111". The <domain> provides essentially the type of the name.
- D3. There are essentially three types of domains: global, per-resource, per-query (dynamic):
  - D3.1: For example, ipv4 and ipv6 defines **global entities**, in that they are not dependent on particular resources;
  - **D3.2 For example, pid defines per-resource entities, for example, “pid:pid1” may refer to one PID in one network map, and another PID in another network map;**
  - **D3.3 For example, a general design of “ane” (abstract network element) may generated dynamic entities. Hence, knowing the resource is still not enough to identify the entity.**

Key Design Points (2/2)

- D4. Aggregation of entities is allowed, to improve scalability. Hence, an entity name may be either an individual entity or a set. An example is an IP prefix.
  - D4.1 An implication of D4. is that we need to handle property inheritance. Multi-inheritance is tricky, as OOP multi-inheritance demonstrated. So far longest prefix matching (LPM) avoids the problem. But we need to decide if we want to have a spec on future design of this aspect.
- D5. Property names are in a global namespace, to enforce global, consistent usage of property names.

D3.2: Entity Conflict

- Issue: If not specified, an entity property resource (application/alto-propmap+json) may have ambiguity in query and/or response, e.g.,

```
"pid-property-map" : {
   "uri" : "http://alto.example.com/propmap/lookup/pid",
   "media-type" : "application/alto-propmap+json",
   "accepts" : "application/alto-propmapparams+json",
   "uses" : [ "default-network-map”, "network-map-2” ]
   "capabilities" : {
        "domain-types": [ "ipv4", "ipv6" ],
        "prop-types" : [ "pid" ]
```

- Solution: Specification makes clear the condition that each such resource MUST lead to conflict free entity identification.

D3.3: Dynamic Entity

- Issue: A path vector abstraction can be query dependent.
- Potential solutions:
  - Disallow
  - Using session HANDLER
    - Path vector returns a session HANDLE ID
    - Extend the current design to allow query (application/alto-propmapparams+json) to include the HANDLE (Sec. 5.3)

```
 object {
   EntityAddr     entities<1..*>
   PropertyName   properties<1..*>;
} ReqFilteredPropertyMap;
```

```
 object {
   EntityAddr            entities<1..*>;
   PropertyName   properties<1..*>;
   DynamicDomainUUID   vag;
} ReqFilteredPropertyMap;
```

TODO

- 2.5
  - Uniform property names (i.e., property names are not scoped by domain) single property name space
- 2.6 revision
  - Keep at current place
  - Generalize to general case, beyond network maps
  - Move to later
- 3.1.3
  - Clarified that LPM will not lead to multiple inheritance
- 3.1.4
  - Revise the setting on Relationships to Network Maps
- 3.1 vs 3.2
  - PID EntityAddr and address EntityAddr: ipv4:xxxxx vs just pid name

TODO

- 4.4
  - Multiple domains
- 4.5
  - Multiple uses
- 4.6
  - Clarify defined as no value vs null
- 5.5
  - Handle multiple resources

Backup Slides

TODO

- Mixed ”uses” and specific domains, e.g.,

"pid-property-map" : { "uri" : "http://alto.example.com/propmap/lookup/pid", "media-type" : "application/alto-propmap+json", "accepts" : "application/alto-propmapparams+json", "uses" : ["default-network-map"] "capabilities" : { "domain-types": ["ipv4", "ipv6"], "prop-types" : ["pid"]

Motivation

- In the beginning there were Endpoint Properties (EPs).
- EPs were independent of the Network Map, but there was only one Network Map, so it was moot.
- And then we added multiple Network Maps, and “resource-specific” EPs vs. “global” EPs, and EPs became more complicated.
- And then we proposed PID Properties.
- And Abstract Network Element Properties (topology draft).
- And Foo Properties, and Bar Properties, and ….

Let’s unify all those Property Services into a common framework that can be extended for new entity classes

Entity Naming

- Extend typed endpoint addresses:

```
entity-name  :=  entity-class  :  entity-specific-name
entity-class  :=  ipv4  |  ipv6
                  cidrv4  | cidrv6  |
                  mac48  |
                  pid  |
                  ane  | ….
```

- Examples:

```
ipv4:1.2.3.4
cidrv4:1.2.0.0/16
pid:mypid1
ane:link42
ane:datacenter-14.rack-37.tor-router
```

Property Naming

- Common property name space, independent of entity type
  - Values should have same format for all entity types
  - Interpretation may vary, but basic meaning should be the same
  - If a property does not make sense for an entity type, skip it!
- Good example:
  - geo-location property is “latitude longitude [height]”
  - For PIDs, it’s the centroid of endpoints in PID
- Bad example:
  - For endpoints, geo-location is “lat long [height]”
  - For PIDs, geo-location is “nw-lat nw-long se-lat se-long”
- Only applies to IANA registered properties. For “priv:” properties, do whatever you want.

Property Map Services

- Two new services, modeled on Full & Filtered Network Maps:
  - GET-mode Full Property Map
  - POST-mode Filtered Property Map
- IRD gives property names and entity types each map returns
  - Implicit cross product of entity types & property names
  - Server omits meaningless combinations
  - Server can define multiple maps to avoid meaningless combinations
- A Full Property Map for Endpoint Properties???
  - Yes, there are billions of endpoints, but the server might only define properties for a few thousand
  - And if a Full Map would be too big, provide a Filtered Map instead

Property Maps & Network Maps

- In RFC 7285, Endpoint Properties were independent of Network Maps
  - Holdover from early single Network Map versions of the protocol
  - Illusion, because the “pid” property depends on the Network Map
  - Led to “resource-specific property” kludge (mea culpa!)
- Conceptual change:

  Each Property Map resource depends on <u>one</u> Network Map
- Many entity types are defined by the Network Map, so this provides necessary context
- Use the default Network Map for any properties that really are independent of the network

IRD Entries: Full Property Maps

```
"full-property-1" : {
    "uri" : "http://----------",
    "media-type" : "application/alto-propmap+json",    (new type)
    "uses" : "my-default-network-map",
    "capabilities" : {
      "prop-types" : [ "geo-location", "asn" ],
      "entity-types" : [ "pid" ]
    }
 },
"full-property-2" : {
    "uri" : "http://----------",
    "media-type" : "application/alto-propmap+json",
    "uses" : "my-default-network-map",
    "capabilities" : {
      "prop-types" : [ "bandwidth", "type" ],
      "entity-types" : [ "ane" ]
    }
 }
```

IRD Entries: Filtered Property Maps

```
  "filtered-property-1" : {
     "uri" : "http://----------",
     "media-type" : "application/alto-propmap+json",
     "accepts" : "application/alto-propmapfilter+json",    (new type)
     "uses" : "my-default-network-map",
     "capabilities" : {
       "prop-types" : [ "pid", "location", "asn" ]
       "entity-types" : [ "ipv4", "ipv6", "pid" ]
     },
  },
 "filtered-property-2" : {
     "uri" : "http://----------",
     "media-type" : "application/alto-propmap+json",
     "accepts" : "application/alto-propmapfilter+json",
     "uses" : "my-default-network-map",
     "capabilities" : {
       "prop-types" : [ "bandwidth", "type" ]
       "entity-types" : [ "ane" ]
     },
  }
```

Filtered Request

Client gives property names & entity names:

```
POST /---- HTTP/1.1
Host: alto.example.com
Content-Length: ###
Content-Type: application/alto-propmapfilter+json
Accept: application/alto-propmap+json,application/alto-error+json

{
  "properties" : [ "geo-location", "asn" ],
  "entities" : [ "ipv4:1.2.3.4", "pid:mypid2" ]
}
```

Response

Similar to current Endpoint Property service:

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"
      }
    ]
  },
  "property-map": {
    "ipv4:1.2.3.4" : { "geo-location": "40.1205,-74.2519",
                       "asn": 65000 }
    "pid:mypid2" :   { "geo-location": "40.0,-74.0",
                       "asn": 65000 }
  }
```

ALTO Properties Simplify Access To …

DNS:

- Properties for (say) “dns:ietf.org”:
  - “address” is preferred address
  - “addresses” is list of alternate addresses
  - Properties for the various DNS resource records?
  - Resolved at ALTO server

WHOIS:

- Properties for (say) “whois:ietf.org”:
  - “registrant”, “admin” and “tech” could be JSON dictionaries
  - “name-servers” could be list of registered name servers

Effect On Current Documents

RFC 7285:

- Deprecate the current Endpoint Property Service
- Do not define any new resource-specific properties

PID Properties Draft:

- Extend this Property Map service
- Define the “pid” and “cidr” entity types
- Define inheritance between pids, cidrs and endpoints

New Properties Drafts:

- Define the entity types for those properties