Unified Properties for ALTO

-04 updates and discussion

ALTO WG meeting IETF102 – July 16, 2018

W. Roome

S. Chen

S. Randriamasy

Y. Yang

J. Zhang

Main Motivation for Changes

- The goal of -04
  - To address WG request during IETF 101
  - Specify a consistency procedure between ALTO Address Type (AAT) and ALTO Entity Domain (AED) registries
  - To clarify text and names
- Two technical issues are remaining in -04
  - Entities for address blocks may be decomposed in Filtered Property Map
  - Resource dependencies in "uses" may have ambiguity
- Next version -05 planned after IETF 102

Main Updates and Discussion

- Updates from -03 to -04
  - Registry consistency: between AAT and AED registries
    - Option chosen: “manual” method
    - Detailed in section on IANA considerations
    - Upon IANA guidance at IETF 101
  - Update on error handling on entities and properties
  - Text and terms updates
    - Adoption of term “Entity Domain“ instead of “Domain” to avoid ambiguities with "network domain"
- Discussion on proposed further changes in -05
  - Systematic specification of requesting entities for address blocks
  - Systematic specification of uses for resource-specific properties

Manual Consistency between AAT and AED Registries

- New Section 9.2.1 “Consistency Procedure between ALTO Address Type Registry and ALTO Entity Domain Registry”
  - **Defines** 2 conditions at which both registries are consistent
  - **Rule**: if an ALTO domain has the same identifier as an ALTO address type, their addresses encoding MUST be compatible
- Consistency procedure when a new ALTO domain is registered
  - Do corresponding entity addresses match a known "network" address type?
  - If YES: is such an address type present in the ALTO Address Type Registry?
    - If YES: new ALTO domain identifier = found ALTO address type identifier
    - If NO
      - Define new ALTO domain identifier and use it to register a new address type in the AAT Registry following Section 14.4 of [RFC7285]
    - Register the new ALTO domain in the AAD
  - If NO: register the new ALTO domain in the AAD

  } **Can be simplified in 1 instruction**
- Domain name registration process in AAD specified in section 9.2.2

Manual Consistency between AAT and AED Registries

- Example 1: “ipv4” and “ipv6” entity domains proposed in section 9.2
  - Entity addresses match a known "network" address type
  - Entity addresses already specified in AAT registry

  ➔ domain name MUST be the same = “ipv4” and “ipv6”
- Example 2: entity domain “cell”
  - Entity addresses match a known "network" address type, e.g. ECGI type
  - Not yet specified in AAT
    - New ALTO Domain ID = “Cell”
    - New AAT created = “Cell” + registered in AAT registry
    - Domain ID “Cell” registered in AED registry
- Example 3: entity domain “pid” proposed in section 9.2
  - Entity addresses does not match a known "network" address type

  ➔ New ALTO Domain ID = “pid” + registered in AED registry

Error Handling on Entities and Properties

- In section « 5.6. Response » - to Filtered Property Map requests
- ALTO server MUST return an "E_INVALID_FIELD_VALUE" error
  - When member « entities » of request is invalid
  - When requested property not defined in IRD for this service/resource
- Section 5.6 defines when member “entities” is invalid
  - Invalid address format
  - Entity address is an invalid address of the entity domain (to be re-phrased)
- If Server does not define a value for a requested property on an entity
  - It MUST omit that property from the response for only that entity
  - Discussion: or put a “null” value?

Other Updates

- Adopted usage of expression "ALTO Entity Domain" throughout the document
- Section 1. “Introduction”: paragraph introducing ALTO Entity domains
- Section 6.3 “Impact on the pid Property”
  - some rewording to clarify between "pid" and "PID" and avoid headaches,
- Section 10 References: updates and reformatting

Updates in Examples

- 7.3 example IRD: name update for the Endpoint property resource
  - Removed "availbw-property-map" for "ane" entities
  - Added "location-property-map" for "pid" entities
  - Resource name of legacy Endpoint property "pid" changed to "legacy-pid-property"

- 7.8. Filtered Property Map Example #4
- Example response for
  - with Entity “ane” replaced by example by entity “pid”

"pid:pid5": {

"country": "ca",

"state": "QC"

Remaining Issues

- Issue 1: Entity decomposition
  - Section 5.6: "... it only includes the entities and properties requested by the client."
  - Consider a request for property P of entity A=ipv4:192.0.2.0/31. Assume that P has value v0 for A, and has value v1 for A1=ipv4:192.0.2.0/32.
  - If response only includes A (as defined in Section 5.6), the client gets wrong P for all addresses in A1.
  - To be fixed: Specify how the server responds the Filtered Property Map request correctly.
- Issue 2: Resource-specific properties
  - The ALTO client SHOULD be able to resolve the dependencies of resource-specific properties without ambiguity.
  - To be fixed: Specify how the client interprets the "uses" field correctly.

Next Steps

- WG discussion on proposed solutions to issues 1 and 2
- Submit version -05 with upon WG agreement
- WGLC on -05

THANK YOU

Backup Slides:
Open Discussions

Issue 1: Entity Decomposition

- Filtered Property Map allows the client to request properties for an address block.
- **Basic principle**: The client SHOULD be able to get or derive correct properties for each address in the requested address block.
- Current revision (-04) cannot guarantee this principle.

EXAMPLE: Assume the full property map:

```
"property-map": {
  "ipv4:192.0.2.0/23": {"pid": "pid0"},
  "ipv4:192.0.2.0/28": {"pid": "pid1"},
  "ipv4:192.0.2.16/28": {"pid": "pid2"}
}
```

If the client requests "ipv4:192.0.2.0/27", it will get the following response from the inheritance rule:

```
"property-map": {
  "ipv4:192.0.2.0/27": {"pid": "pid0"}
}
```

From the response, the client will interpret the "pid" property of "ipv4:192.0.2.1" as "pid0", but it should be "pid1" from the full property map.

Solution for Entity Decomposition

- Proposed solution (in the revision -05):
  - **Rule 1**: If a property for a requested entity is inherited from another entity not included in the request, the response SHOULD include this property for the requested entity.
  - **Rule 2**: If there are entities covered by a requested entity but having different values for the requested properties, the response SHOULD include all those entities and the different property values for them.
  - **Rule 3**: If an entity in the response is already covered by some other entities in the same response, it SHOULD be removed from the response for compactness.
- Considering the same full property map, if the client requests "ipv4:192.0.2.0/27", the response SHOULD be:

```
"property-map": {
  "ipv4:192.0.2.0/27": {"pid": "pid0"},
  "ipv4:192.0.2.0/28": {"pid": "pid1"},
  "ipv4:192.0.2.0/28": {"pid": "pid2"}
}
```

From Rule 1, inherit "pid0" from "ipv4:192.0.2.0/23

From Rule 3, "ipv4:192.0.2.0/27" SHOULD be removed

From Rule 2, include "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28"

- Involved sections: Sec 4.6, Sec 5.6 and Sec 6.3

Issue 2: Resource-Specific Properties

- **Basic principle**: The "uses" field should have no ambiguity in specifying dependencies.
- Previous version (-04 and before) does not handle generic dependencies well.
  - Example 1 (<"pid", "region"> depends on "net1" or "net2"?)

```
"uses": ["net1", "net2"],
"capabilities": {
  "entity-domain-types": ["pid"],
  "property-types": ["region", "center"]
}
```

  - Example 2 (who depends on "pv1"?)

```
"uses": ["net1", "pv1"],
"capabilities": {
  "entity-domain-types": ["ipv4", "ipv6", "ane"],
  "property-types": ["pid"]
}
```

- How does the ALTO client resolve the relationship between each dependent Resource in "uses" and each resource-specific (Domain, Property) pair in "capabilities"?

Solution for Resource-Specific Properties

- Proposed solution (in the revision -05):

  the ALTO server MUST ensure the ALTO client can interpret the resource dependencies by the following **"uses" rule**:

  ➜ For each domain in "entity-domains", take the full "uses" list, and then,

  ➜ go over each property in "properties" capability in array order:

  ➜ If the property is a resource-specific property for the current domain, and it needs a sequence of S resources, then

  ➜ take the S resource ID(s) at the beginning of "uses" to interpret the property;

  ➜ and remove the S resource ID(s) from "uses".

- Two examples show the interpretation process in the client side:

Interpretation of the IRD in Example 1:

```
// uses=["net1", "net2"];
pid_uses = uses.copy();
DepRes ["pid"]["region"] = pid_uses.pop(); // net1
DepRes ["pid"]["center"] = pid_uses.pop(); // net2
```

Interpretation of the IRD in Example 2:

```
// uses=["net1", "pv1"];
ipv4_uses = uses.copy();
DepRes["ipv4"]["pid"] = ipv4_uses.pop(); // net1
ipv6_uses = uses.copy();
DepRes["ipv6"]["pid"] = ipv6_uses.pop(); // net1
ane_uses = uses.copy();
DepRes["ane"]["pid"] = ane_uses.pop(2); // net1 and pv1
```

- Involved sections: Sec 2.7, Sec 4.5, Sec 4.6, Sec 5.5 and Sec 5.6