Unified Properties for ALTO

draft-ietf-alto-unified-props-new-08

Wendy Roome
Sabine Randriamasy
Kai Gao
Y. Richard Yang
J. Jensen Zhang

Major Updates of Unified Properties -08

- Document organization
  - Separated concepts and data type encodings into different sections (Sec 2 and Sec 3)
- Requirements and concepts clarification
  - Clarified requirements of unified properties.
  - Clarified concepts and their motivation
  - Clarified the scope of the design
- Protocol specification update
  - Changed capabilities from *entity-domains x properties* to *{entity-domain -> properties}*
  - Updated IANA registries

Requirements for Unified Properties (UP) Extension

An ALTO client should be able to

- Req-1: Obtain properties of *generic entities*
  - examples of generic entities: endpoint, pid, AS, country, ...
- Req-2: Obtain requested properties of entities with no message redundancy
- Req-3: Obtain the full map of entity properties
  - Solution: Get-mode property map service

Requirements and Design

Req-1: Obtain properties of *generic entities*

- An ALTO client should have answers to the following questions
  - Q1.1: What entities can an ALTO client query?
  - Q1.2: For each entity, what properties can an ALTO client query?

Requirements and Design

Req-1: Obtain properties of *generic entities*

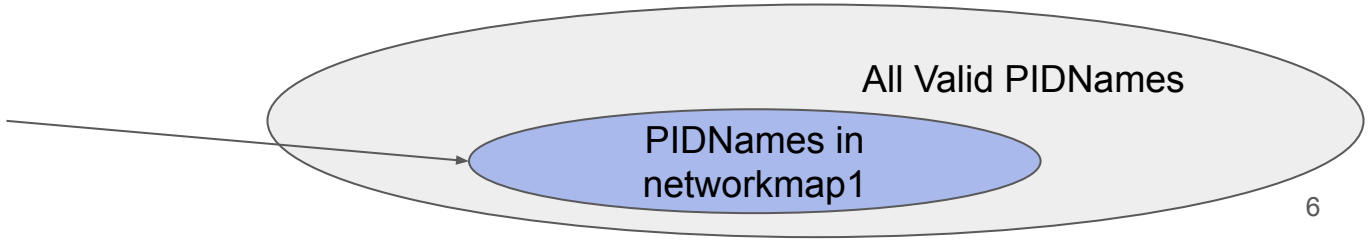
- An ALTO client should have answers to the following questions
  - Q1.1: What entities can an ALTO client query?
    - **Basic idea**: the ALTO server announces a set of types
    - Each type refers to entities of the same semantics and ID format
      - e.g., type=ipv4 -> {semantics: an ipv4 address of an endpoint, format: IPv4Addr}
      - An entity is valid if it has a valid entity ID
    - Each type defines a **complete entity set** which consists of **all** valid entities of this type
      - e.g., type=ipv4 represents the set of all valid IPv4Addr
    - An ALTO client can query entities in the complete entity set of any announced type

Design Update: Resource-Specific Entity Domain

Req-1: Obtain properties of *generic entities*

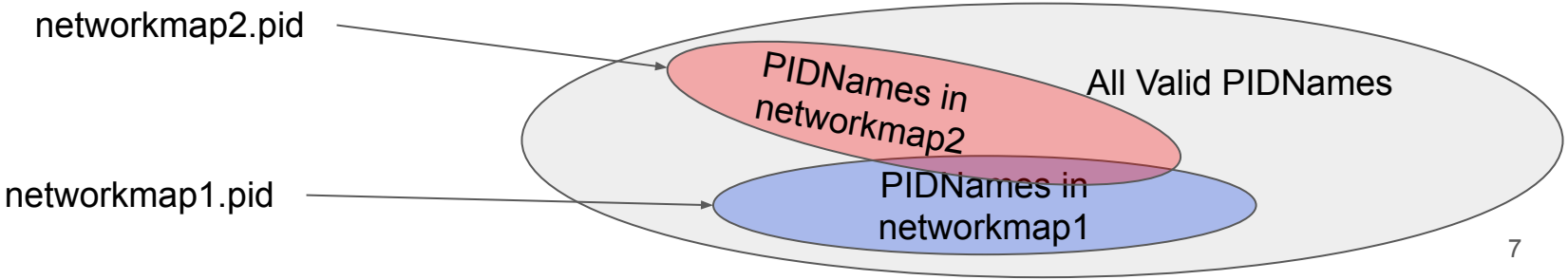
- An ALTO client should have answers to the following questions
  - Q1.1: What entities can an ALTO client query?
    - **Basic idea**: the ALTO server announces a set of types
    - **Issue**: Not all valid entities are meaningful
      - e.g., A PID is only meaningful in a network map
    - For a given entity type, an ALTO information resource only contains a subset of the complete entity set, which is referred to as a ***resource-specific entity domain***
    - Solution: the ALTO server announces a set of resource-specific entity domains

The ALTO server only defines properties for a subset of the complete PIDName set



Design Update: Resource-Specific Entity Domain

- **Issue**: Using entity ID to identify the entity may be ambiguous
  - clients/servers need to distinguish which entity domain an entity belongs to
  - e.g., two "PID1"s in `networkmap1` and `networkmap2` define two different PIDs
    - the ASN property of "PID1" in `networkmap1` is "1234"
    - the ASN property of "PID1" in `networkmap2` is "1235"
- **Solution**: Use `EntityDomainName ':' DomainSpecificEntityID` to identify entities
  - Entity Domain Names: `"networkmap1.pid"` and `"networkmap2.pid"`
  - Domain-Specific Entity ID: `"PID1"`
  - `"networkmap1.pid:PID1"` and `"networkmap2.pid:PID1"` identify two different PID entities



Requirements and Design

Req-1: Obtain properties of *generic entities*

- An ALTO client should have answers to the following questions
  - Q1.1: What types of entities can an ALTO client query?
    - Solution: the ALTO server announces a set of resource-specific entity domains
  - Q1.2: For each (~~entity~~) resource-specific entity domain, what properties can an ALTO client query?
    - Basic idea: The ALTO server announces a set of resource-specific entity domain -> property type mapping

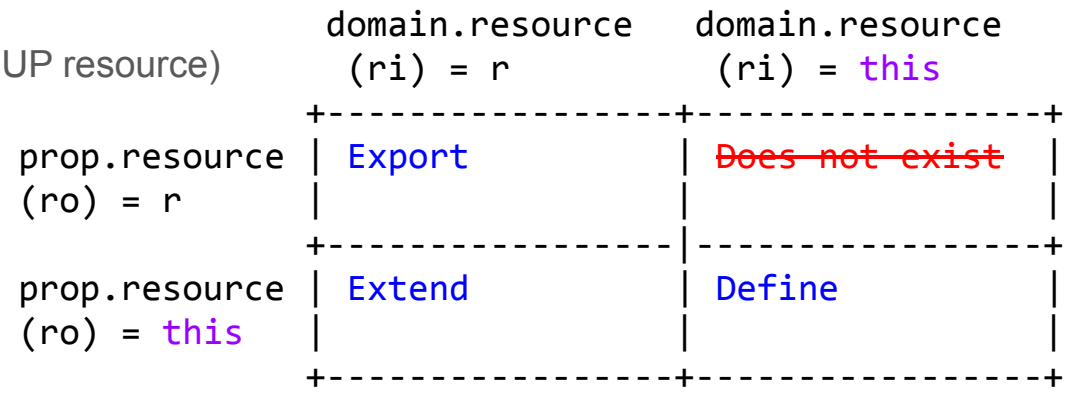
Design Update: Entity to Property Mapping

- **Issue**: For entities in a resource-specific entity domain `ri.di`, the mapping to a property of type `po` may be ambiguous
- Example: `ri.di (=networkmap1.ipv4) -> po (=pid)`
  - The value of property `pid` may be defined by either `networkmap1` or `this` property map itself
- **Solution**: To distinguish between the two cases, the UP design formulates the mapping as `ri.di -> ro.po`, where `ro` is either `ri` or `this`
  - The server announces the capabilities using the two mappings:
    - `networkmap1.ipv4 -> networkmap1.pid`
    - `networkmap1.ipv4 -> this.pid`
- In RFC 7285, `this` is not a reserved Resource ID. UP uses `.po` to represent `this.po`
  - To be discussed: or just use `po` without `'.'`?

Clarification of Design

- Each property map provides a set of *`ri.di -> ro.po`* mappings
- These mappings must be announced in the property map capabilities
- Three mapping modes are currently considered:
  - "Export": `networkmap1.ipv4 -> networkmap1.pid`
  - "Extend": `networkmap1.pid -> this.geolocation`
  - "Define": `this.ipv4 -> this.asn`

(`this` = the present UP resource)



Capability Announcement and Consistency

- Note that each announced *`ri.di -> ro.po`* mapping should be meaningful
  - Entity Domain Type *`di`* MUST be supported by *`ri`*
    - Bad announcement: `networkmap1.asn` is not meaningful
  - In *Export Mode*, Entity Property Mapping *`di -> po`* MUST be supported by *`ro`*
    - Bad announcement: `networkmap1.ipv4 -> networkmap1.asn` is not meaningful
- Solution: use additional IANA registries
- For each ALTO Information Resource Type (indicated by the Media Type):
  - Define its ***Resource-Specific Entity Domain Export Registry*** for each exportable *`Entity Domain Type`* *`di`* of it
  - Define its ***Entity Property Mapping Export Registry*** for each exportable *`EntityDomainType -> PropertyType`* mapping *`di -> po`* of it

Example of Additional IANA Registries

11.4.1. Network Map

Media-type: application/alto-networkmap+json

Entity Domain Type	Intended Semantics
ipv4	See Section 5.1.1
ipv6	See Section 5.1.1
pid	See Section 5.1.1

Table 4: ALTO Network Map Resource-Specific Entity Domain.

Mapping Descriptor	Entity Domain Type	Property Type	Intended Semantics
ipv4 -> pid	ipv4	pid	See Section 5.1.2
ipv6 -> pid	ipv6	pid	See Section 5.1.2

Table 6: ALTO Network Map Entity Property Mapping.

Protocol Specification Update: IRD

From *{di} x {po}* to *{ri.di -> {ro.po}}*

Old IRD

```
"filtered-property-map": {
  "uri": "http://alto.exmaple.com/propmap/region",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "uses" : [ "default-network-map" ],
  "capabilities": {
    "entity-domains": [ "pid" ],
    "properties": [ "region", "asn" ]
  }
}
```

New IRD

```
"filtered-property-map": {
  "uri": "http://alto.exmaple.com/propmap/region",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "uses" : [ "default-network-map",
             "alt-network-map" ],
  "capabilities": {
    "mappings": {
      "default-network-map.ipv4": [
        "default-network-map.pid" ],
      "alt-network-map.ipv4": [
        "alt-network-map.pid" ],
      "default-network-map.pid": [ ".region" ],
      "alt-network-map.pid": [ ".region", ".asn" ]
    }
  }
}
```

Request and Response of UP

- Query Request:
  - A set of entities in `ri.di` announced by IRD
  - A set of properties in `ro.po` announced by IRD
- Query Response:
  - Property Map for requested properties of requested entities whose mappings are announced by IRD.

```
{
  "entities" : [ "default-network-map.pid:pid1",
                 "alt-network-map.pid:pid1"],
  "properties" : [ ".region", ".asn" ]
}
```

Request

```
{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "default-network-map", "tag": "tag1"},
      {"resource-id": "alt-network-map", "tag": "tag2"} ]
  },
  "property-map": {
    "default-network-map.pid:pid1": {
      ".region": "us-west"
    },
    "alt-network-map.pid:pid1": {
      ".region": "us-east",
      ".asn": "3389"
    }
  }
}
```

Response

Requirements and Improvement Solutions

Req-2: Obtain requested properties of entities with no redundancy

- Current UP design may introduce redundancy in the following cases
  - C2.1: The requested entities may be redundant
  - C2.2: The returned entities may be redundant

Requirements and Improvement Solutions

Req-2: Obtain requested properties of entities with no redundancy

- Current UP design may introduce redundancy in the following cases
  - C2.1: The requested entities may be redundant
    - A property map may support multiple entity property mappings for different resource-specific entity domains of the same type:
      - `ri1.di -> ri1.po (e.g., networkmap1.ipv4 -> networkmap1.pid)`
      - `ri2.di -> ri2.po (e.g., networkmap2.ipv4 -> networkmap2.pid)`
    - An ALTO client has to query entities in `ri1.di` and `ri2.di` individually, which is redundant
      - If an ALTO client wants to query both `networkmap1.pid` and `networkmap2.pid` of `"192.0.1.1"`, it has to query both `"networkmap1.ipv4:192.0.1.1"` and `"networkmap2.ipv4:192.0.1.1"`

Improvement: Aggregated Entity Domain

- **Solution**: UP allows a property map to use `di -> {ri1.po, ri2.po}` to represent the equivalent entity property mappings to above
  - e.g., `ipv4 -> {networkmap1.pid, networkmap2.pid}`, where `ipv4` represents an entity domain including all domain-specific entity IDs in `networkmap1.ipv4` and `networkmap2.ipv4` (the union of the two resource-specific entity domains)
  - An ALTO client can query `"ipv4:192.0.1.1"` to get both `networkmap1.pid` and `networkmap2.pid`

Example of Aggregated Entity Domain in IRD

IRD without aggregated entity domain

```
"filtered-property-map": {
  "uri": "http://alto.exmaple.com/propmap/region",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "uses" : [ "default-network-map",
             "alt-network-map" ],
  "capabilities": {
    "mappings": {
      "default-network-map.ipv4": [
        "default-network-map.pid" ],
      "alt-network-map.ipv4": [
        "alt-network-map.pid" ],
      "default-network-map.pid": [ ".region" ],
      "alt-network-map.pid": [ ".region", ".asn" ]
    }
  }
}
```

IRD with aggregated entity domain

```
"filtered-property-map": {
  "uri": "http://alto.exmaple.com/propmap/region",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "uses" : [ "default-network-map",
             "alt-network-map" ],
  "capabilities": {
    "mappings": {
      "ipv4": [ "default-network-map.pid"
                "alt-network-map.pid" ],
      "default-network-map.pid": [ ".region" ],
      "alt-network-map.pid": [ ".region", ".asn" ]
    }
  }
}
```

Requirements and Improvement Solutions

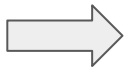
Req-2: Obtain requested properties of entities with no redundancy

- Current UP design may introduce message redundancy in the following cases
  - C2.1: The requested entities may be redundant
    - Improvement Solution: aggregated entity domain
  - C2.2: The returned entities may be redundant
    - For some types of entities, a group of entities may have the same properties
      - e.g., ipv4 entities in a cidr 192.0.1.0/24 may have the same ASN property value
    - Returning properties for each entities in such a group individually is redundant
      - e.g., To obtain the ASN property of all ipv4 entities in the cidr 192.0.1.0/24, the client has to enumerate all 256 individual ipv4 addresses

Revisit: Hierarchy and Inheritance

- **Current Design**:
  - Introduce the entity block to represent a set of individual entities, and allow the server to return properties for entity blocks
    - e.g., cidr as an entity block representation for ipv4 addresses
  - Define the property inheritance rule on the entity block representation
    - e.g., an ipv4 address / cidr inherits properties of the longest-match cidr

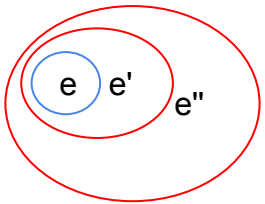
```
ipv4:192.0.2.0/26: P=v1
ipv4:192.0.2.0/28: P=v2
ipv4:192.0.2.0/30: P=v3
ipv4:192.0.2.0:    P=v4
```



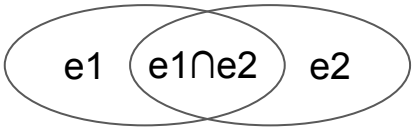
```
ipv4:192.0.2.16:   P=v1
ipv4:192.0.2.0/29: P=v2
ipv4:192.0.2.1:    P=v3
ipv4:192.0.2.64:   (not defined)
```

Clarification: Hierarchy and Inheritance

- **Requirement**: The property inheritance must not be ambiguous
- **Solution**: Claim the following constraint
  - For a given entity type, if its entity block representation is defined, the representation MUST be hierarchical
    - In other words, if there are two entity blocks including the same entity, they MUST NOT be partially intersected
- Note that this is a sufficient but not necessary condition to avoid ambiguity



Valid Entity Block Representation for Property Inheritance



Invalid Entity Block Representation for Property Inheritance

e1 = 192.*.*.1
e1 -> asn = 1234
e2 = 192.0.*.*
e2 -> asn = 1235
how about asn of 192.0.1.1?

(e, e', e'', e1, e2 are entity blocks)

Conclusions

- Each property map announces a set of *`ri.di -> ro.po`* mappings
  - Announce `di -> {ro.po}` to avoid potential message redundancy
- For each information resource, register supported `di` and `di -> po` in IANA registries
- An ALTO client requests a set of entities and properties and the ALTO server returns a property map
  - Entity block representation and inheritance rule can be applied to reduce the message size

Next Steps

Go to WGLC Request?

Backup

Requirement Space of Unified Properties

An ALTO client SHOULD be able to

- Req-1: Obtain properties of *generic entities*.
  - Req-1.1: What type of entities an ALTO client can query?
  - Req-1.2: For each entity, which properties an ALTO client can query?
  - Req-1.3: For each entity type, which entities the ALTO server MAY define certain properties for?

entity.property value	**entity is in domain**	**entity is not in domain**
(entity -> property) exists	*defined/undefined*	undefined
(entity -> property) doesn't exist	undefined	undefined

Motivation: Resource-Specific Entity Domain

When querying entity properties, the client should know which entities are ***valid*** and ***effective***:

- ***Valid***: each entity has a type to indicate its valid identifier encoding
  - Use specification/IANA registry to register types and their encoding format
- ***Effective***: the server should indicate an effective set of entities for client to query
  - Entity Domain: an effective set of entities with the same type
  - Effectiveness: somebody defines this entity to provide some properties for it
  - Enumerating all effective entities is not efficient
  - Instead of it, use existing ALTO information resources to export entity domains

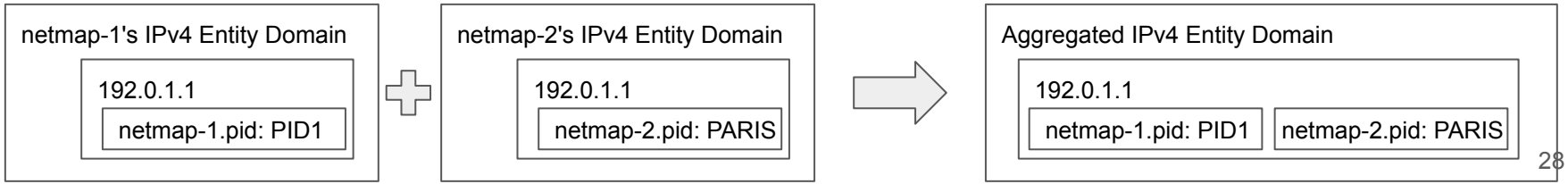
Requirement Space of Unified Properties

An ALTO client SHOULD be able to

- Req-1: Obtain properties of *generic entities*.
  - Req-1.1: ALTO clients need to know which entities may exist and may have properties to be queried.
    - If defined by an existing ALTO information resource, ALTO clients need to know *how this ALTO information resource exports all those entities*.
  - Req-1.2: ALTO clients need to know the semantics of each entity property.
    - If defined by an existing ALTO information resource, ALTO clients need to know *how this ALTO information resource maps an entity to its property*. (Refer to Sec 11.2.2 of [RFC7285])
- Req-2: Obtain *the full map* for properties of a given set of entities.

Design Point: Aggregated Entity Domain

- Entities in different resource-specific entity domains may have the same identifier but has different properties.
- Depending on the application, the server may regard them as the same object or not.
  - Two "PID1" associated to two different network maps should be different objects.
  - Two "192.0.1.1" associated to two different network maps may be the same object.
    - The client may want to know all the properties of "192.0.1.1" in two resource-specific entity domains.
    - The server can export the aggregated entity domain for several resource-specific entity domain with the same type so that the client can query the aggregated entity properties.



Syntax Sugar: Aggregated Entity Domain

- A property map MUST NOT use aggregated entity domain if it contains ambiguous mappings
  - For `di -> {ri1.po1, ..., riN.poN}`, a property map MUST either support `rik.di -> rik.pok` or `this.di -> rik.pok` but NOT BOTH
  - e.g., `networkmap1.pid -> this.geo-location`, `networkmap2.pid -> this.geo-location` CANNOT be aggregated as `pid -> geo-location`

IRD Design

```
# Representation of (ri, di)
EntityDomain := ResourceID '.' EntityDomainType  # e.g., netmap1.ipv4, costmap1.ane
              |            '.' EntityDomainType  # shortcut of (this, di)
              |                EntityDomainType  # shortcut of (*, di)

# Representation of (ro, po)
Property := ResourceID '.' PropertyType  # e.g., netmap1.pid
          |            '.' PropertyType  # shortcut of (this, po)

"uses" : [ ResourceID... ],  # all ro
"capabilities" : {
  "mappings" : {
    EntityDomain : [ Property... ],  # representation of (ri, di) -> (ro, po)
    ...
  }
}
```

Protocol Update: IANA Registry

- For each new *`Entity Domain Type`*, go to *ALTO Entity Domain Type Registry*.
- For each new *`Property Type`*, go to *ALTO Entity Property Type Registry*.
- For each ALTO Information Resource Type (indicated by the Media Type):
  - Define its *Resource-Specific Entity Domain Export Registry* for each exportable *`Entity Domain Type`* of it.
  - Define its *Entity Property Mapping Export Registry* for each exportable *`EntityDomainType -> PropertyType`* mapping of it.

Design Point: Entity Property

- Each entity property has a type which specifies its data format and semantics of the value.
  - Data format: the value of a "pid" property MUST be a PIDName format JSON string.
  - Semantics: the value of a "pid" property is an ALTO PID defined by a network map.

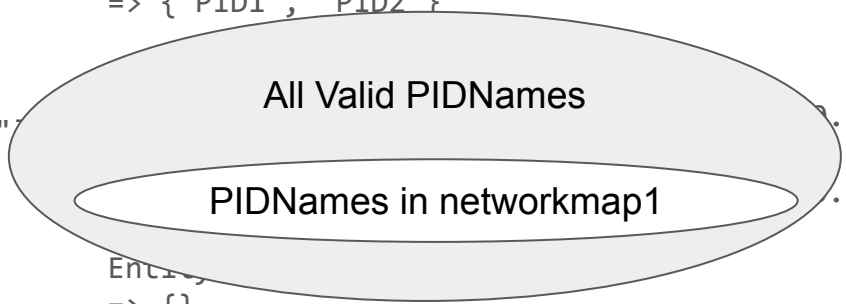
Design Update: Resource-Specific Entity Domain

- Each information resource exports several types of entities.
  - The entities of the same type define an entity domain, which is referred to as a ***resource-specific entity domain.***
    - e.g., a network map exports an ipv4 entity domain, an ipv6 entity domain and a pid entity domain.
  - (To be revised) By querying this information resource, the client can know the content of entity domains exported by it.

netmap-1:

```
{
  "PID1": {
    "ipv4": ["192.0.1.0/24", "192.0.2.0/24"]
  },
  "PID2": {
    "ipv4": ["192.0.3.0/24"]
  }
}
```

```
Entity Domain "netmap-1.pid"
=> {"PID1", "PID2"}
```




```
.1.255",
2.255",
.3.255"}
Enti
=> {}
```

Design Update: ...

- Problem: Only using property type cannot specify the entity to property mapping clearly.
  - Multiple ALTO information resources MAY have different values for the same type of property for an entity.
  - Property type specifies data format and semantics of property values, but does not specify which resource the value comes from.
  - e.g., Sec 11.2.2 of [RFC7285] defines how a network map maps an IP address to a PID property
- Solution:
  - Each ALTO information resource exports a set of entity -> property mappings in IANA registry.
  - UP announces all supported entity -> property mappings in its IRD capabilities

Design Update: Resource-Specific Entity Domain

- For a given entity type, an ALTO information resource only contains a given set of entities. This set is referred to as a ***resource-specific entity domain.***
  - This entity type is also called the type of this resource-specific entity domain.
- Two resource-specific entity domains MAY include the same domain-specific entity IDs. To distinguish which entity domain the entity belongs to, UP uses `EntityDomainName ':' DomainSpecificEntityID` to identify entities.
- The ALTO server does not guarantee the same domain-specific entity ID in two different resource-specific entity domain means the same physical/logical object.