Cisco Notes

Yann Esposito

Entitlements (Monetization in IROH) [2023-07-12 Wed]

Intro

What?

- **Entitlements**: What the customer is paying for.
- **Access Rules**: What service should provide/restrict.

Example

Entitlements:

- Tier: Essentials for 1000 *users* (number of Lees).
- Extra Data Retention "add-on": 180 *days*
- Extra Ingest "add-on": 2/GB/

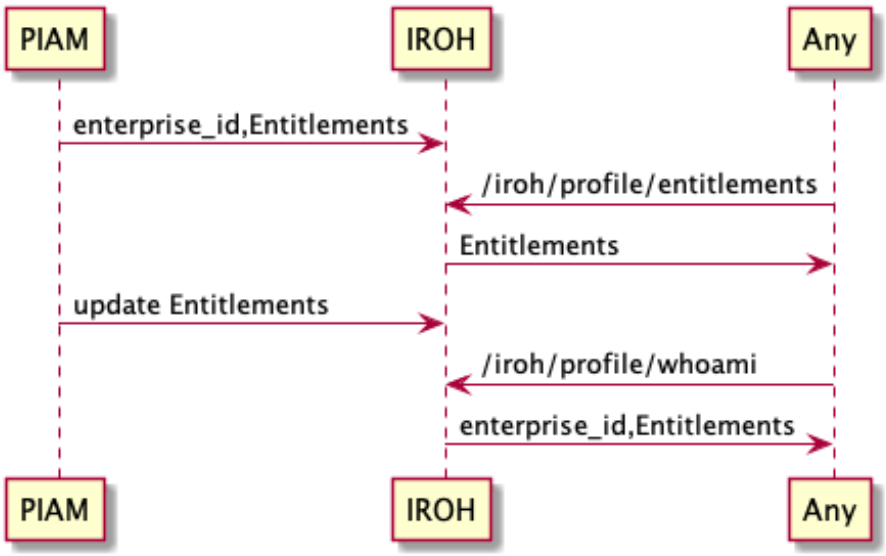
One Access Rule example:

- **Total Ingest**: 4000GB (1000 user × (2GB + 2GB))
- **Time to Keep Data**: 180 days (yes, **extra** might not mean what we could expect)

ref: https://wwwin-github.cisco.com/cisco-sbgidm/docs/blob/master/provisioning/xdr/xdr-ga.md#entitlements

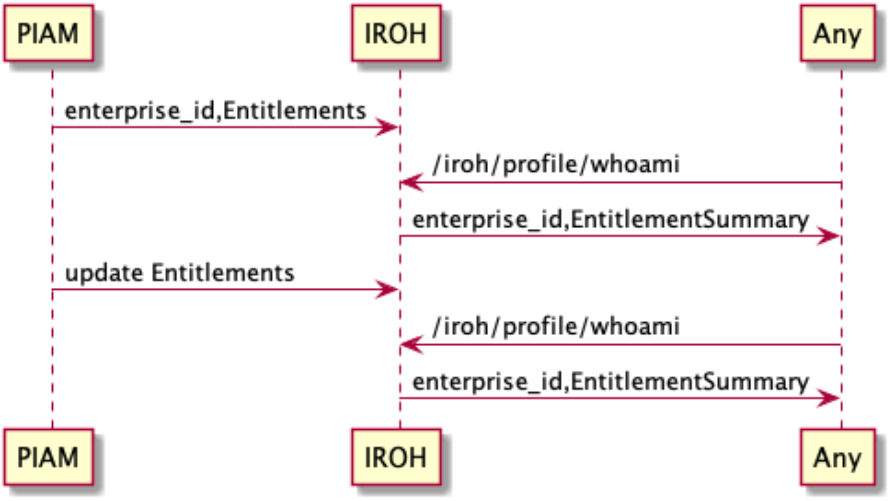
How?

Entitlement represent what the customer is paying for. PIAM is in charge of creating and updating them.



Also Entitlement Summary

IROH exposes an API to retrieve an `EntitlementSummary` which is a data structure easier to consume than the list of entitlements.



Entitlements (technically)

Here is an example of a list of `Entitlements` PIAM can send to IROH:

```
[{"name": "tier",
  "value": "essentials",
```

```
  "quantity": {"value": 1000, "unit": "users"},
  "enforce_quantity": true},
 {"name": "extra_data_retention",
  "value": "",
  "quantity": {"value": 2, "unit": "days"},
  "enforce_quantity": true}]
```

PIAM Doc

From Paul Chichonski's doc

https://wwwin-github.cisco.com/cisco-sbgidm/docs/blob/master/provisioning/product-spec.md

Entitlements

```
[{"name": "tier",
  "value": "essentials",
  "quantity": {"value": 1000, "unit": "users"},
  "enforce_quantity": true},
 {"name": "extra_data_retention",
  "value": "",
  "quantity": {"value": 2, "unit": "days"},
  "enforce_quantity": true}]
```

entitlements: A list of entitlements the tenant is allowed to use. Each item in the list is an object with the following fields:

1. name
    - name - The name of the entitlement (defined as part of the entitlement controlled vocabulary between PIAM and the product)
2. value
    - value - Some entitlements will have a string value that serves to qualify the entitlement, for example an entitlement with name=tier may have three different manifestations if there are three different tiers (e.g., {"name": "tier", "value": "essentials"}, {"name": "tier", "value": "premier"}, {"name": "tier", "value": "advantage"})
3. quantity
    - quantity - Some entitlements will have numeric quantity associated with the entitlement, this represents the amount of this entitlement the tenant is permitted to consume. Each quantity field will contain an object with the following values:
        - value - The number holding the actual quantity.

- unit - A string representing what unit to use when interpreting the quantity.

4. $\text{quantity}_{\text{enforced}}$

   - $\text{quantity}_{\text{enforced}}$ - A boolean field, if true it means that the product should enforce the allocated quantity of the entitlement for this tenant. It is up to the product to determine how to do this. Cases where this will be false are if the customer purchased via a buying program that supports a "pay as you go" pricing model.

A few other examples

1. Just the Tier, no add-on:

```
[ {"name" : "tier",
   "title" : "advantage",
   "quantity" : {
       "value" : 10,
       "unit" : "users"
   },
   "enforce-quantity" : true
  }]
```

2. Tier with add-ons

```
[ {"name" : "tier",
   "title" : "essentials",
   "quantity" : {
       "value" : 1,
       "unit" : "users"},
   "enforce-quantity" : true},
  {"name" : "extra_ingest",
   "quantity" : {
       "value" : 10,
       "unit" : "GB"},
   "enforce-quantity" : true},
  {"name" : "extra_data_retention",
   "quantity" : {
       "value" : 90,
       "unit" : "days"
   },
   "enforce-quantity" : true}]
```

Entitlement Summary

The Entitlement Summary goal is to provide an easier to consume data-structure to read the entitlements list.

So instead of being a list it is a single JSON Object. We also plan to add additional technically useful entries.

The main structure of the `EntitlementSummary` is:

```
{<entitlement-name>: <entitlement-details>}
```

Where `<entitlement-details>` looks like:

```
{"title": "something", // <- optional
 "quantity": Integer,
 "unit": "human-readable-unit",
 "enforce?": Boolean}
```

A few examples

1. An `EntitlementSummary` for just a Tier Entitlement

   When PIAM send this list of `Entitlements`:

   ```
   [{"name" : "tier",
     "title" : "advantage",
     "quantity" : {
         "value" : 32000,
         "unit" : "users"},
     "enforce-quantity" : true}]
   ```

   The `EntitlementSummary` will look like this:

   ```
   {"tier" : {"unit" : "users",
              "title" : "advantage",
              "enforce?" : true,
              "quantity" : 32000}}
   ```

2. With Add-ons

   If PIAM send a list of `Entitlements` with add-ons:

   ```
   [ {"name" : "tier",
      "title" : "premier",
      "quantity" : {
          "value" : 1000,
          "unit" : "users"
      },
      "enforce-quantity" : true},
     {"name" : "extra_ingest",
      "quantity" : {
          "value" : 2,
          "unit" : "GB"},
      "enforce-quantity" : true},
     {"name" : "extra_data_retention",
   ```

```
   "quantity" : {
       "value" : 180,
       "unit" : "days"},
   "enforce-quantity" : true}
]
```

The `EntitlementSummary` will be:

```
{"tier":
     {"title": "premier",
      "quantity": 1000,
      "unit": "users",
      "enforce?": true},
 "extra_data_retention":
     {"quantity": 180,
      "unit": "days",
      "enforce?": true},
 "extra_ingest":
     {"quantity": 2,
      "unit": "GB",
      "enforce?": true}}
```

3. `Entitlements` vs `EntitlementSummary` consumption

   Now, compare how to retrieve the Entitlement Tier in javascript from the entitlements list:

   ```
   function get_entitlement_tier (entitlements) {
       for (entitlement in org.entitlements) {
           if (entitlement.name == "tier") {
               return entitlement.title;
           }
       }
   }
   let tier =  get_entitlement_tier (entitlements);
   ```

   As compared to the `EntitlementSummary`

   ```
   let tier = whoami.org["entitlement-summary"].tier.title;
   ```

4. More to come

   (a) IROH Internal

   But we plan to add more technical specific values so it helps every Entitlement consumer. That way it would make possible to share between product specific technical values.

   For example, we plan to add:

   - a list of allowed modules.
   - an optional list of additional scopes

- rate limits

(b) XDR global values

If you want us to add some information, so we could centralize some logic related to entitlement into IROH just ask us to add it. Ideally, this should only contain data that could be shared between different modules. For example:

- allowed workflows, or allowed properties for workflows
- specific limitations for a specific module (read-only, etc. . .)

(c) Example

So we could imagine that it will look like this:

```
{"tier": {"title": "premier",
          "quantity": 1000,
          "unit": "users",
          "enforce?": true},
 "extra_data_retention": {"quantity": 180,
                          "unit": "days",
                          "enforce?": true},
 "extra_ingest": {"quantity": 2,
                  "unit": "GB",
                  "enforce?": true},
 // ---- SUMMARY OF TECHNICAL LIMITS
 "summary" {
     // PIAM Logic
     "data-retention-in-days": 180, // use extra_data_retention + tier
     "data-maximal-size-in-GB": 4000, // use extra_ingest + tier quantity
     // IROH Internal
     "additional-scopes": [ ... ],  // depends on the tier
     "allowed-modules": [ ... ],    // depends on the tier
     // XDR Shared Global Rules
     "restricted-workflows": [...], // depends on the tier (or something else)
     "rate-limits": // can change depending on the tier
         {"sca": {"queries-per-minutes": "100"},
          "sxo": {"queries-per-minutes": "80"},
          "csc": ...},
     ...
 }
}
```

Today, there is not yet a `summary` field. But there will be for Q1 and monetization. So instead of letting everyone have an internal value for "default" everyone could look at these values, and the default will be kept in a single place and could be changed.