Push-MDQ

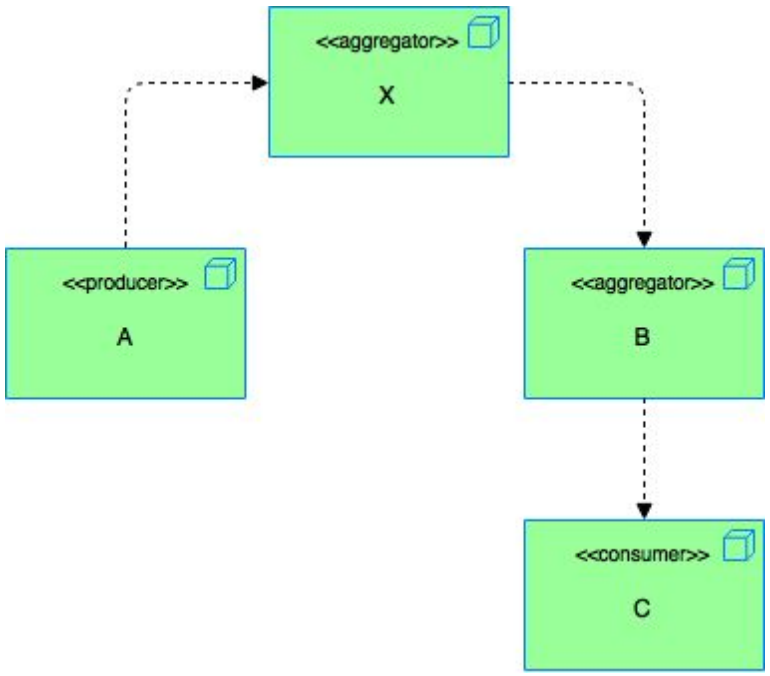
Faster metadata update

Abstract

Consider the following aggregator configuration: A provides an upstream to B which provides downstreams to C and D. Changes at A take X hours to reach B and even more hours to reach C and D. We need this to go faster. One approach is to just update more often but this is wasteful and scales poorly. Instead we propose a push-based notification mechanism for metadata changes based on ATOM and PuSH

Problem Statement

Consider the following configuration of metadata producers, aggregators and consumers:



A produces metadata to aggregator X which in turn produces a “downstream” metadata aggregate to B which in turn feeds C. Lets assume that all feeds (A->X, X->B and B->C) has validUntil set to 7 days in the future and cacheDuration set to 12 hours (PT12H). A change in A->X will require up to 36 hours to traverse to C provided that A, X and B all update metadata as often as possible.

There are several situations where waiting 36 hours is an unacceptably long time for a change to propagate from A to C. Notable examples include emergency key rollovers (for instance in the case of a key compromise) where waiting 36 hours for a compromised key to become invalid may very well be 35 hours too long.

To make matters worse, when end-users are involved even a couple of hours of downtime for a service may be unacceptable.

This document proposes a mechanism for adding push-notifications to the metadata query protocol (MDQ) based on ATOM and PuSH (pubsubhububb). The design goal of this approach is to enable changes to propagate through multiple hops of interconnected metadata aggregators and consumers within seconds allowing for fast convergence of large-scale metadata deployments. Another goal is to avoid hard dependencies on SAML metadata, allowing the same approach to be reused for other applications of MDQ such as U2F attestations.

Strawman approach

- Each endpoint at an aggregator supporting PMDQ indicates this by permitting content negotiation for `application/atom+xml` in addition of the default `application/samlmetadata+xml`
- When requesting /foo with Content-Type: `application/atom+xml` the aggregator will return the changes associated with /foo (subject to some reasonable ttl based on `validUntil` etc).
- The /entities change-feed is the “all in” change feed - contains all changes
- Each object in the ATOM feed should contain a reference to signed metadata that is at least as fresh as the one that was current at the time of the change.
- This ‘change feed’ could also implement PuSH (pubsubhubbub) in the normal way
- An aggregator could implement its own PuSH hub or use an existing hub (superfeedr etc)

This leads to the following behaviour associated with the example

- Aggregator B fetches the ATOM change feed associated with /upstream-to-B and parses the Link header to find the pubsubhubbub H
- B registers as a consumer of http://A/upstream-to-B at H
- When something happens at aggregator A to the entities in /upstream-to-B the aggregator updates the /upstream-to-B ATOM change feed and notifies H
- H notifies B which can download the change feed and and fetch metadata

Example message?


```
{
  "messages": [
    {
      "entityData": {
        "entityID": "https://login.aaiedu.hr/edugain/saml2/idp/metadata.php",
        "registrationAuthority": "http://www.srce.hr"
      },
      "metadataUpdate":
"http://www.srce.hr/pmdq/https%3A%2F%2Flogin.aaiedu.hr%2Fedugain%2Fsaml2%2Fidp%2F
metadata.php/v2.5"
      “fingerprint”: "SGVsbG8gQ2xvdWQgUHViL1N1YiEgSGVyZSBpcyBteSBtZXNzYWdlIQ=="
    }
  ]
}
```