internal services auth

WebRTC Virtual Classroom PLatform

Table of Contents

Overall design 2

ISA 2

Design 2

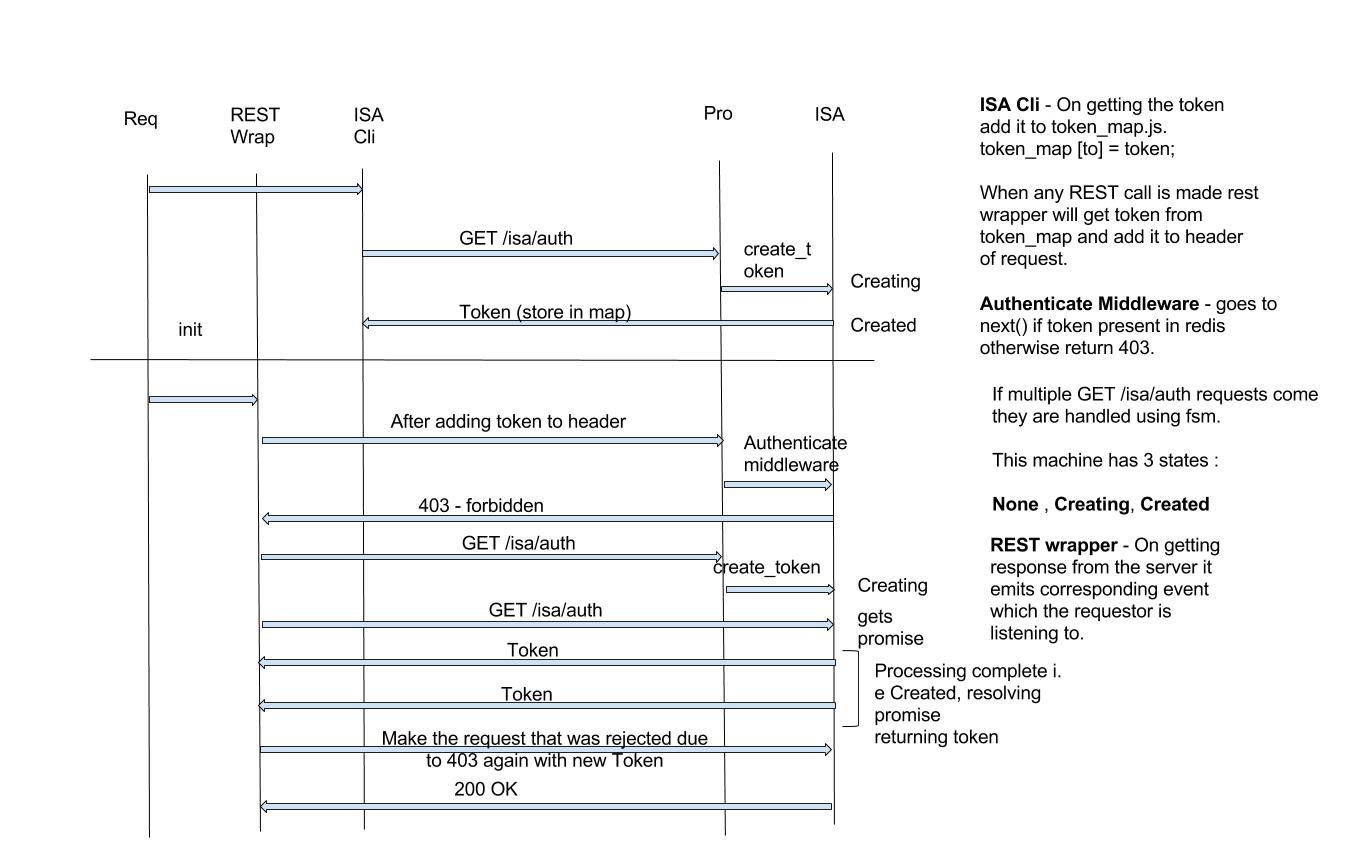
Provider 3

Reqestor 4

ISA API's 5

# Overall design

Insert design diagram here.



# isa

## Provider

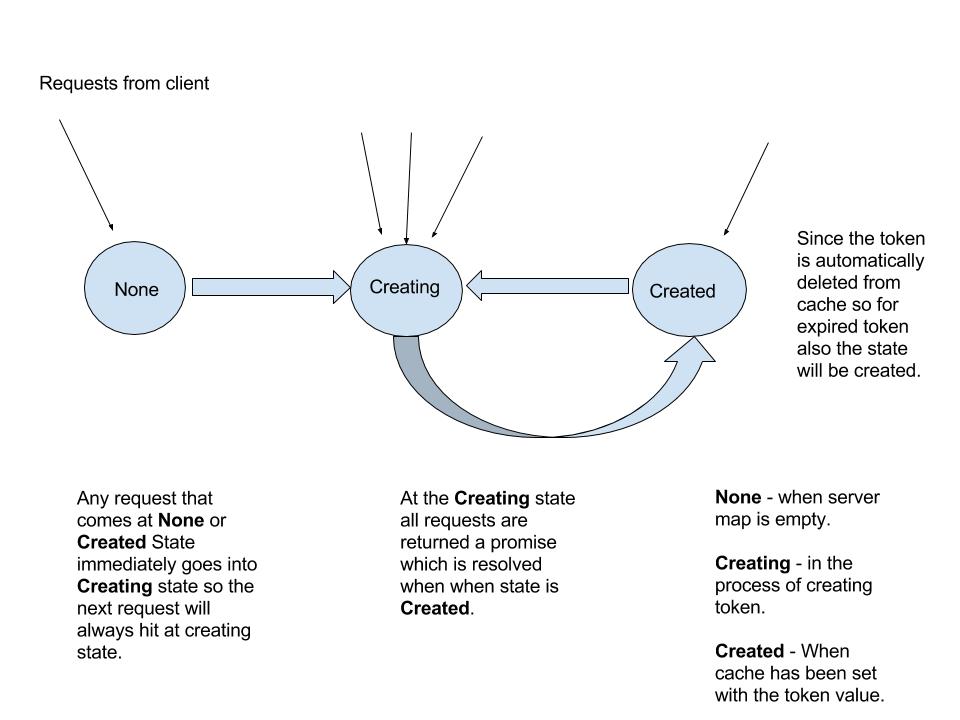
On the Provider End, Isa library is used as a middleware and as an end point.

**Middleware** – Used to verify the token validity. ( isa.authenticate )

* **Authenticate** – Looks for ‘x-access-token’ header in the request. If present tries to decode the value of the header (which is token). On decode success go next() otherwise return error.

**End point** – Returns token to the Requestor. ( isa.create\_token )

* **Create\_token** – Encoded data (created using jwt) which is sent from the Requestor to the Provider is decoded using the shared secret. If decode is successful token is created using that decoded data with some expiry using jwt.
* **State Machine for create\_token** – When multiple calls of create\_token are made from the same requestor instead of creating new token again we return existing token which is present in the cache. This machine ensures that at any point in time only 1 unique token is present between a requestor and provider.
  + **None –** When token\_map at provider end is empty. This state is immediately changed to “**creating**” so that if multiple requests are made only 1 call will hit at “none” state, remaining calls hit at the “**creating**” state and are returned a promise.
  + **Creating –** When the token is being created. All the requests made at this state are returned a promise which is resolved when token is successfully created and set in the cache.
  + **Created –** When token has been created and successfully stored in the cache. After updating the cache with the new token and updating the token\_map the state is changed to “**created**”. It is at this state that the promise is resolved with value of token.



## Requestor

ISA library provides a function – **init (url, service\_class, service\_id, to)**.

Note: Service\_class is provisioning or api-backend or other internal service.

**init (url, service\_class, service\_id, to) -** Makes the rest call to the specified url with encoded data. Data is encoded using the shared secret by jwt library.

Returns restler emitter.

Need to create a wrapper across restler to make get/post requests to the provider.

**Restler\_Client.js:** Wrapper for restler get/post methods. This is used to handle response from the Provider and also to attach token in the header of the rest calls made by the client.

How to handle token expired scenario:

When the rest calls are made with the expired token 403 is returned.

At the rest\_wrapper it is checked if the error is 403 then call the “init” to get new token from the provider. After getting the new token, the request that failed due to 403 is retried and its response is emitted to the client.

So the client will never get 403 as it is handled inside the rest\_wrapper itself.

## ISA API’s

The following operations need to be performed on the node:

P – Provider, R - Requestor

|  |  |  |
| --- | --- | --- |
| **Category** | **API** | **Comment** |
| **ISA(P)** | Create\_token | Generates the token |
| **ISA(P)** | Authenticate | Validate the token, if invalid returns 403 otherwise go next() (middleware) |
| **ISA(R)** | init | Make a rest call and send encoded info about requestor to the provider. Receives token as a response. |
| **REST(R)** | Rest.get/put/del/post  (to, url, options) | Need to specify the service whom you are making request to, this is needed for client side map. Parameters “url” and “options” as described in restler. |