Coordinated Service Exposure Open Source enabler for R1

OSC NONRTRIC team (EST)

Objectives – Service Exposure

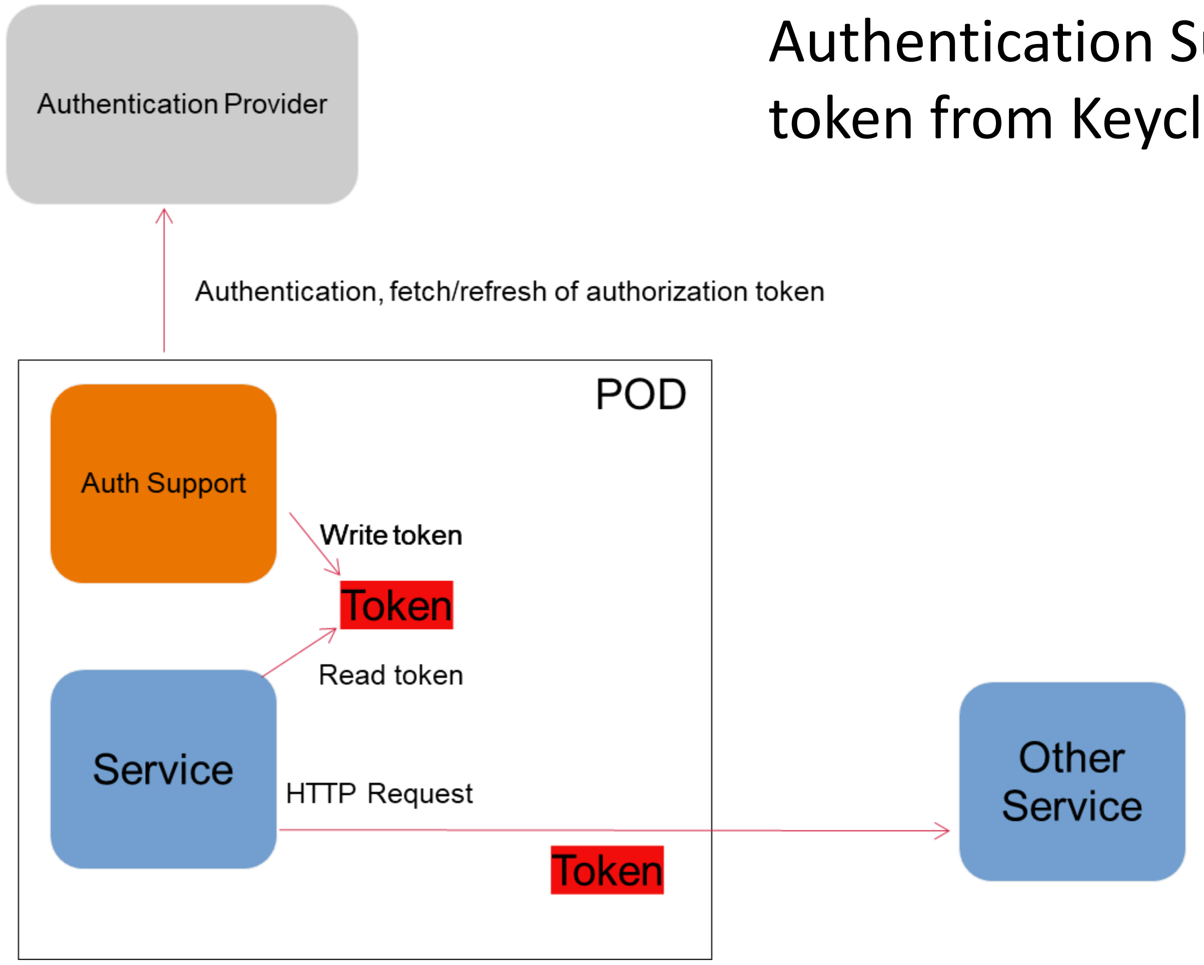
- Add a Service registry for SMO-/NonRTRIC-Platform-services & rApp-services to register/discover/access services
- Services/rApps can only see/use the service they are authorised to see/use

- K8s API gateway can be Kong or Istio Ingress gateway. We use Istio.
  - Can be layered on top of K8s Network Policies (e.g. Calico) to enforce isolation
- Leverage Istio & Authorisation policies to enable service exposure from rApps and Platform services in a K8s environment
- Investigate how to manually, declaratively, and programmatically easily implement exposure (authorisation) support in (K8s) platform services
- Investigate how to manually, declaratively, and programmatically apply exposure (authorisation) policies to rApp µservices during deployment
  - Enforced by Istio
  - Authenticated via Keycloak
- Use/Integrate 3GPP-spec’ed CAPIF core functionality for Registry/Discovery.
  - Investigate automated combined CAPIF Registration & Deployment

A little bit about Istio, Keycloak and helping K8s Services use authorisation tokens

Lightweight Service authentication & authorisation

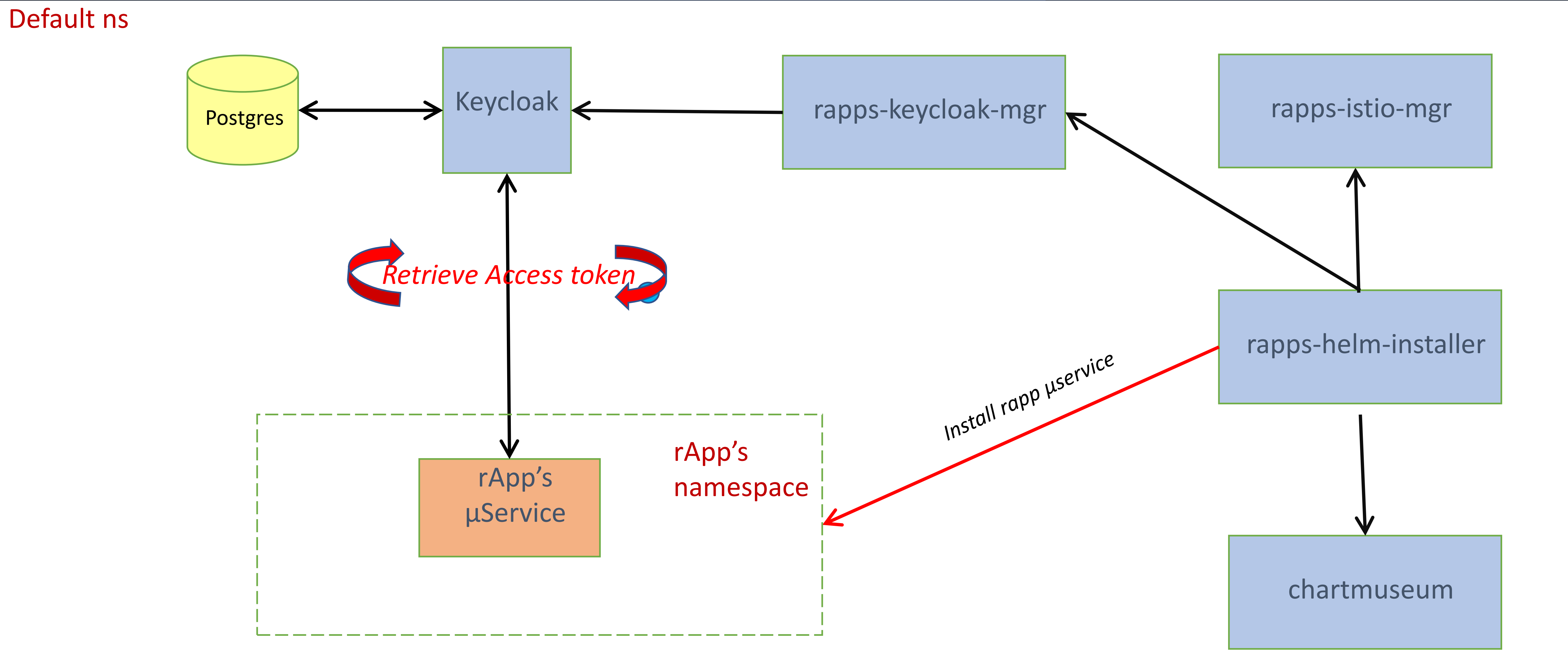
Authentication Support Service: sidecar to fetch authorization token from Keycloak & insert into request headers



How an rApp’s µService uses authorisation tokens – automatically enabled at instantiation time … a few different options

[OSC Investigation: "Coordinated Service Exposure" for Non-Realtime RIC - wiki.o-ran-sc.org]

rApps deployment & security model POC



rApp services

Service	Description
postgres	Keycloak database
keycloak	Token service
chartmuseum	Helm chart repository
rapps-helm-installer	Service to automate an rApp’s µService installation
rapps-istio-mgr	Service to automate Istio setup
rapps-keycloak-mgr	Service to automate keycloak client setup

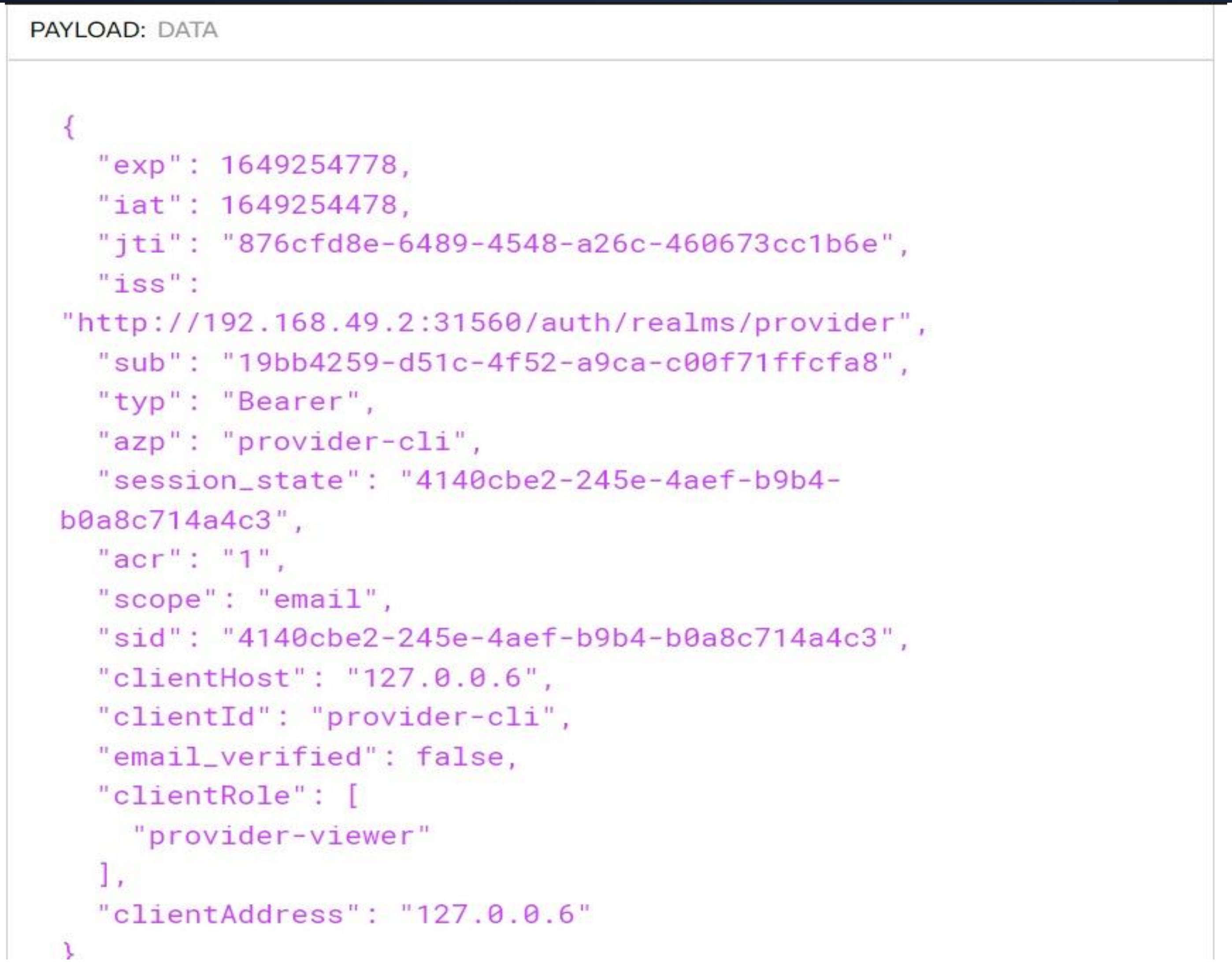
Keycloak Client Authenticator

4 Different Ways of using client authenticator

- Signed Jwt
- Client Id and Secret
- X509 Certificate
- Signed Jwt with Client Secret

Note: Signed JWT with client secret works the same way as Signed JWT except the JWT is signed with the client secret instead of the private key

Sample Token



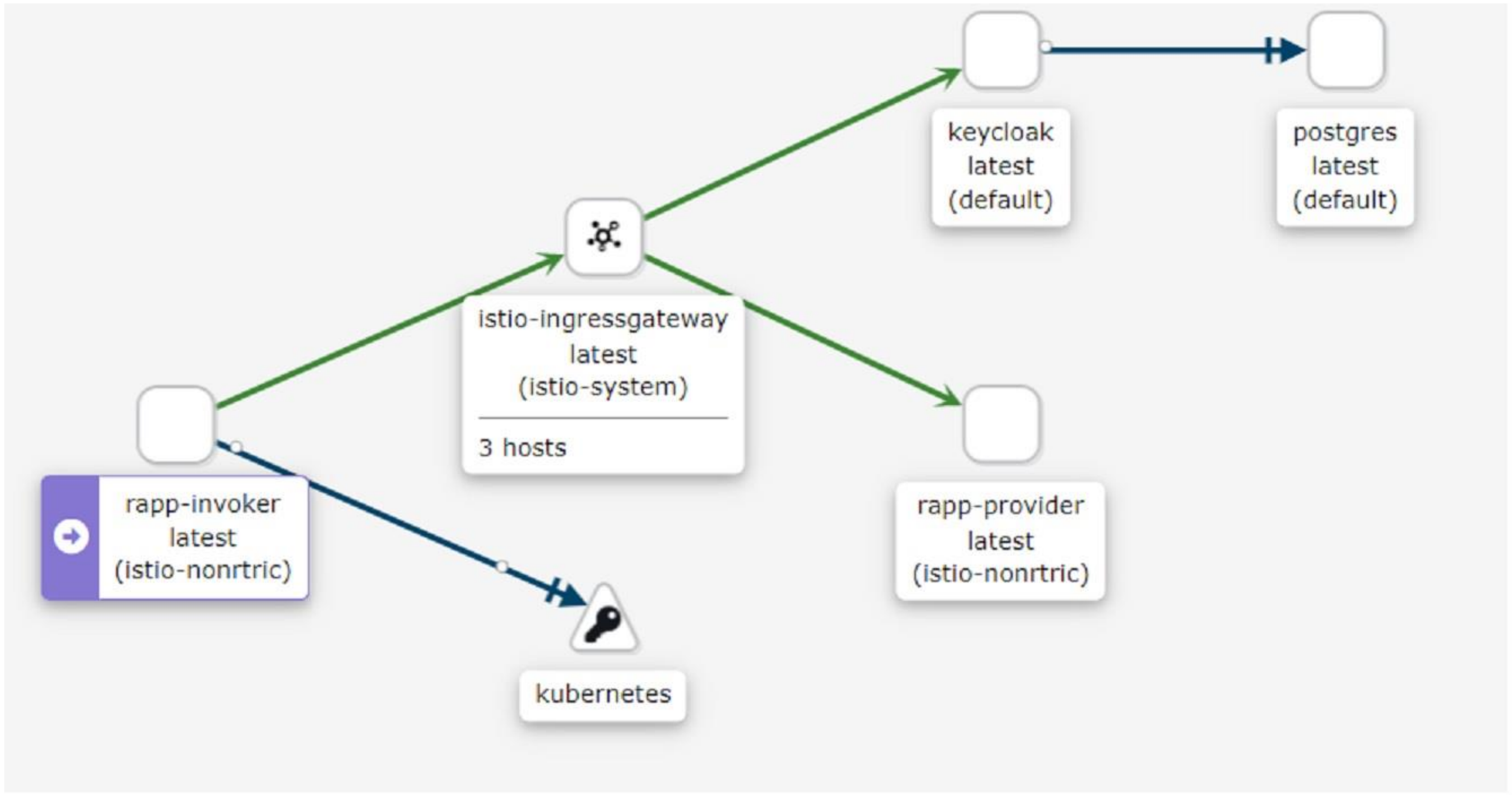
PAYLOAD: DATA

```
{
  "exp": 1649254778,
  "iat": 1649254478,
  "jti": "876cfd8e-6489-4548-a26c-460673cc1b6e",
  "iss":
"http://192.168.49.2:31560/auth/realms/provider",
  "sub": "19bb4259-d51c-4f52-a9ca-c00f71ffcfa8",
  "typ": "Bearer",
  "azp": "provider-cli",
  "session_state": "4140cbe2-245e-4aef-b9b4-
b0a8c714a4c3",
  "acr": "1",
  "scope": "email",
  "sid": "4140cbe2-245e-4aef-b9b4-b0a8c714a4c3",
  "clientHost": "127.0.0.6",
  "clientId": "provider-cli",
  "email_verified": false,
  "clientRole": [
    "provider-viewer"
  ],
  "clientAddress": "127.0.0.6"
}
```

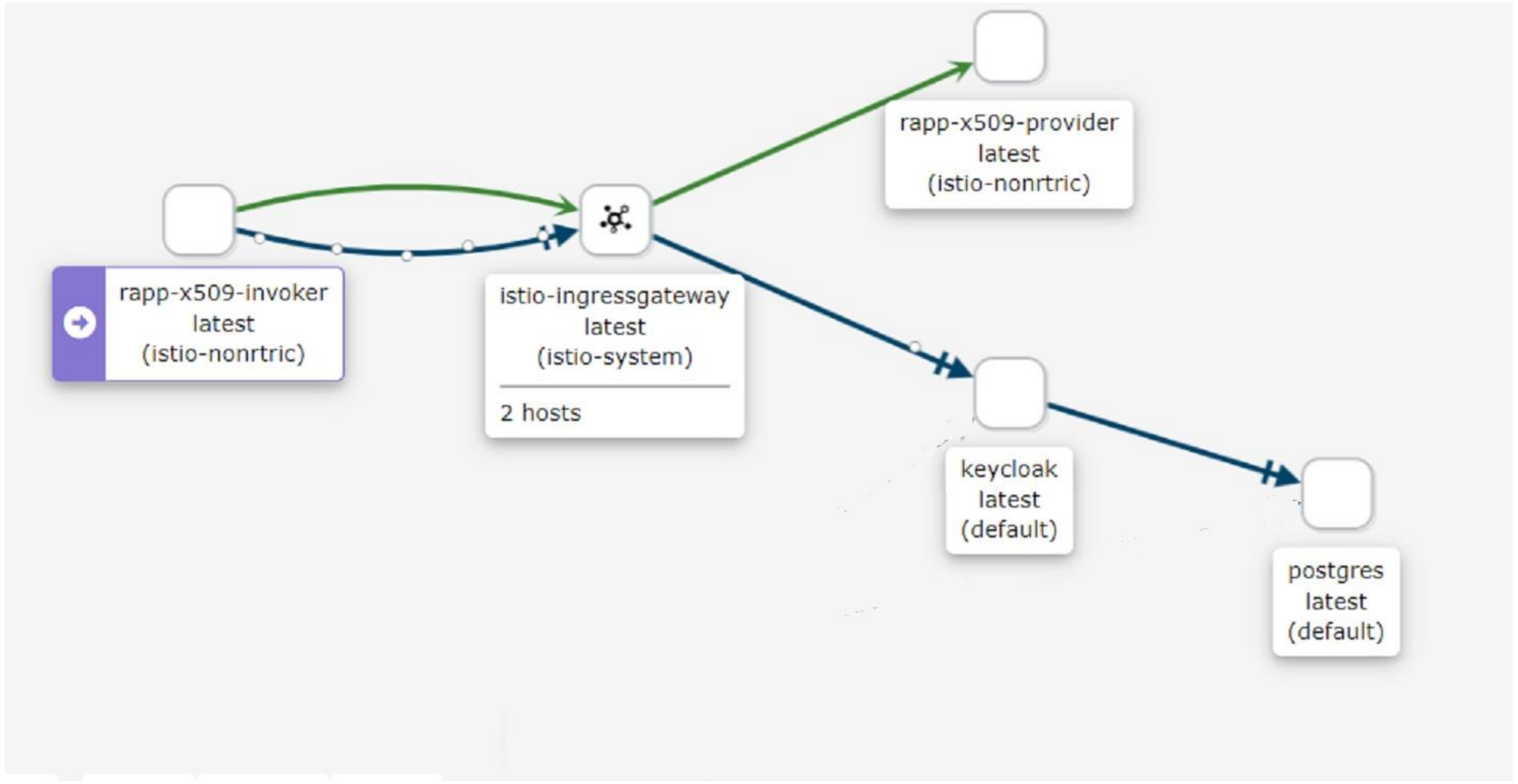
Sample istio AuthorizationPolicy

```
spec:
  rules:
  - from:
    - source:
        requestPrincipals:
        - http://192.168.49.2:31560/auth/realms/provider/   ←   ISS
  - to:
    - operation:
        methods:
        - GET   ←   Method call(s) allowed
        paths:
        - /rapp-provider   ←   Rapp prefix
    when:
    - key: request.auth.claims[clientRole]   ←   Field in token
      values:
      - provider-viewer   ←   Role Name
  selector:
    matchLabels:
      app.kubernetes.io/instance: rapp-provider
```

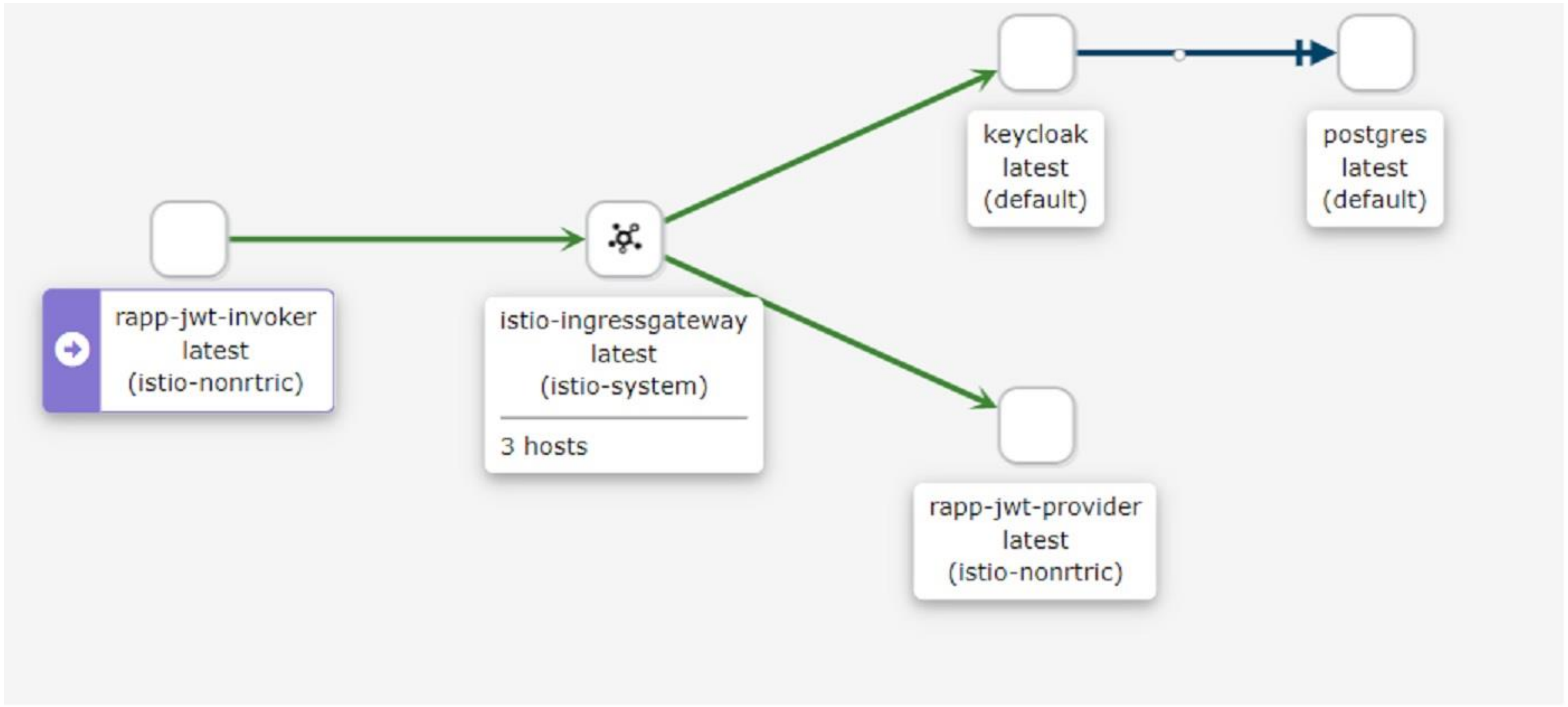
Rapp keycloak – "secret" flow



Rapp keycloak – "x509" flow



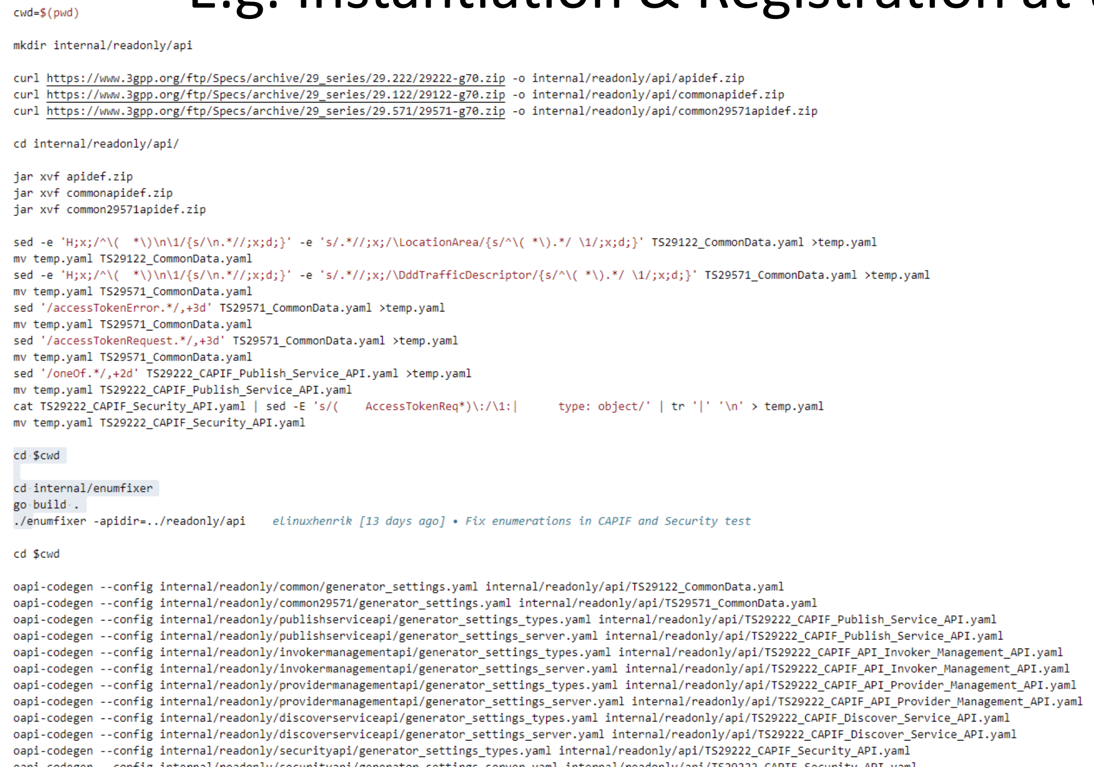
Rapp keycloak – “JWT” flow



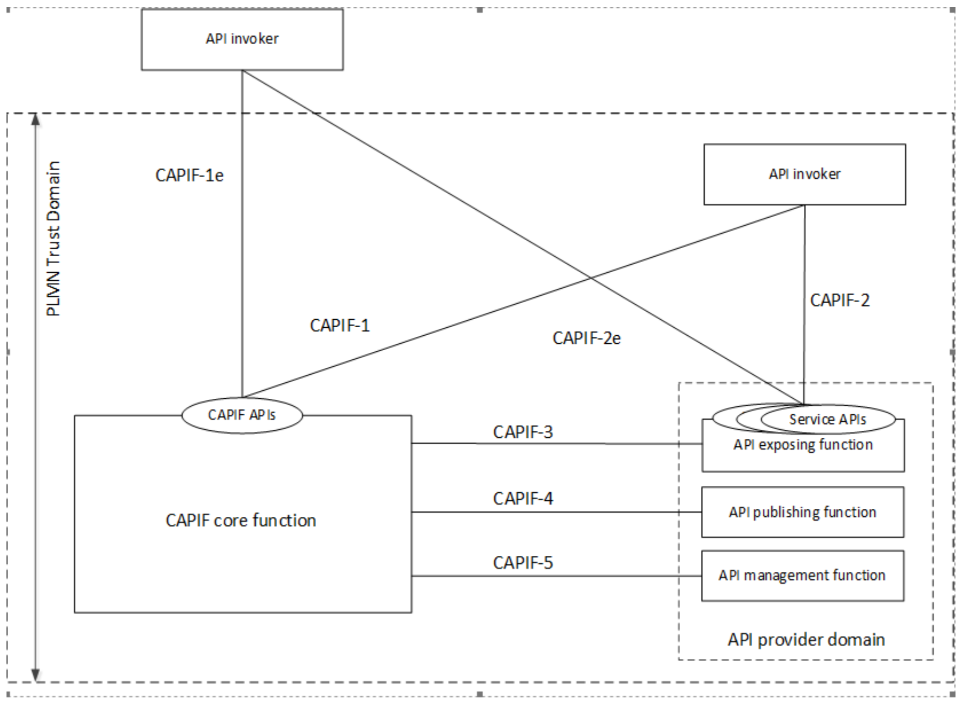
A little bit about using CAPIF functions (API) for Registration/Discovery

CAPIF APIs (3GPP TS 23.222)

- Majority of CAPIF API functions can be code-generated from 3GPP specs
- Work required to map from CAPIF registry operations to underlying platform capabilities to enforce Service exposure
  - E.g. Instantiation & Registration at the same time (Ref APP LCM in ONAP/EIAP)



```
cwd=$(pwd)

mkdir internal/readonly/api

curl https://www.3gpp.org/ftp/Specs/archive/29_series/29.222/29222-g70.zip -o internal/readonly/api/apidef.zip
curl https://www.3gpp.org/ftp/Specs/archive/29_series/29.122/29122-g70.zip -o internal/readonly/api/commonapidef.zip
curl https://www.3gpp.org/ftp/Specs/archive/29_series/29.571/29571-g70.zip -o internal/readonly/api/common29571apidef.zip

cd internal/readonly/api/

jar xvf apidef.zip
jar xvf commonapidef.zip
jar xvf common29571apidef.zip

sed -e 'H;x;/^\(  *\)\n\1/{s/\n.*//;x;d;}' -e 's/.*//;x;/\LocationArea/{s/^\( *\).*/ \1/;x;d;}' TS29122_CommonData.yaml >temp.yaml
mv temp.yaml TS29122_CommonData.yaml
sed -e 'H;x;/^\(  *\)\n\1/{s/\n.*//;x;d;}' -e 's/.*//;x;/\DddTrafficDescriptor/{s/^\( *\).*/ \1/;x;d;}' TS29571_CommonData.yaml >temp.yaml
mv temp.yaml TS29571_CommonData.yaml
sed '/accessTokenError.*/,+3d' TS29571_CommonData.yaml >temp.yaml
mv temp.yaml TS29571_CommonData.yaml
sed '/accessTokenRequest.*/,+3d' TS29571_CommonData.yaml >temp.yaml
mv temp.yaml TS29571_CommonData.yaml
sed '/oneOf.*/,+2d' TS29222_CAPIF_Publish_Service_API.yaml >temp.yaml
mv temp.yaml TS29222_CAPIF_Publish_Service_API.yaml
cat TS29222_CAPIF_Security_API.yaml | sed -E 's/(    AccessTokenReq*)\:/\1:|      type: object/' | tr '|' '\n' > temp.yaml
mv temp.yaml TS29222_CAPIF_Security_API.yaml

cd $cwd

cd internal/enumfixer
go build .
./enumfixer -apidir=../readonly/api    elinuxhenrik [13 days ago] • Fix enumerations in CAPIF and Security test

cd $cwd

oapi-codegen --config internal/readonly/common/generator_settings.yaml internal/readonly/api/TS29122_CommonData.yaml
oapi-codegen --config internal/readonly/common29571/generator_settings.yaml internal/readonly/api/TS29571_CommonData.yaml
oapi-codegen --config internal/readonly/publishserviceapi/generator_settings_types.yaml internal/readonly/api/TS29222_CAPIF_Publish_Service_API.yaml
oapi-codegen --config internal/readonly/publishserviceapi/generator_settings_server.yaml internal/readonly/api/TS29222_CAPIF_Publish_Service_API.yaml
oapi-codegen --config internal/readonly/invokermanagementapi/generator_settings_types.yaml internal/readonly/api/TS29222_CAPIF_API_Invoker_Management_API.yaml
oapi-codegen --config internal/readonly/invokermanagementapi/generator_settings_server.yaml internal/readonly/api/TS29222_CAPIF_API_Invoker_Management_API.yaml
oapi-codegen --config internal/readonly/providermanagementapi/generator_settings_types.yaml internal/readonly/api/TS29222_CAPIF_API_Provider_Management_API.yaml
oapi-codegen --config internal/readonly/providermanagementapi/generator_settings_server.yaml internal/readonly/api/TS29222_CAPIF_API_Provider_Management_API.yaml
oapi-codegen --config internal/readonly/discoverserviceapi/generator_settings_types.yaml internal/readonly/api/TS29222_CAPIF_Discover_Service_API.yaml
oapi-codegen --config internal/readonly/discoverserviceapi/generator_settings_server.yaml internal/readonly/api/TS29222_CAPIF_Discover_Service_API.yaml
oapi-codegen --config internal/readonly/securityapi/generator_settings_types.yaml internal/readonly/api/TS29222_CAPIF_Security_API.yaml
oapi-codegen --config internal/readonly/securityapi/generator_settings_server.yaml internal/readonly/api/TS29222_CAPIF_Security_API.yaml
```



… these were just brief glimpses. Follow-up discussions for those interested.