Consumer

Table of Contents

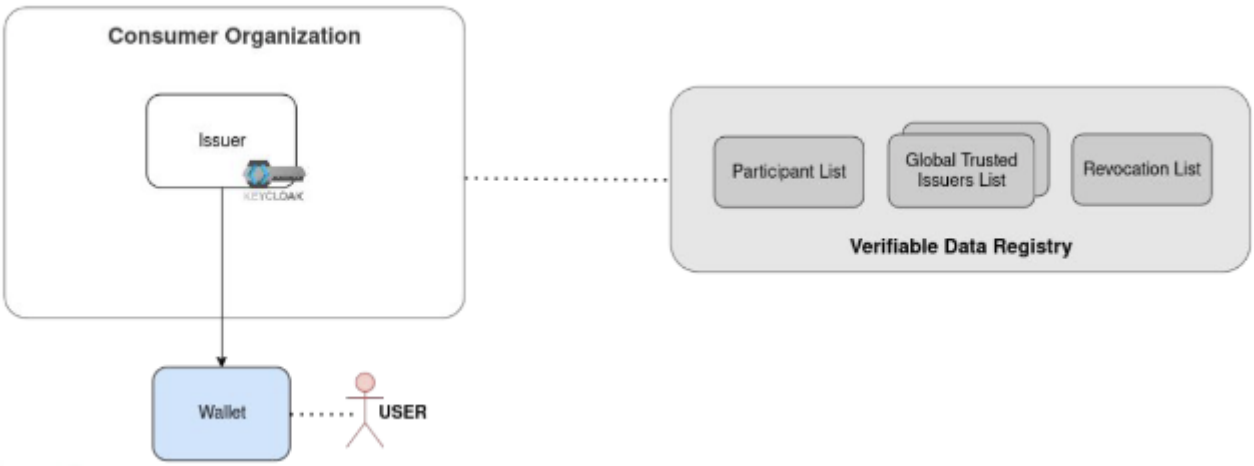
Introduction

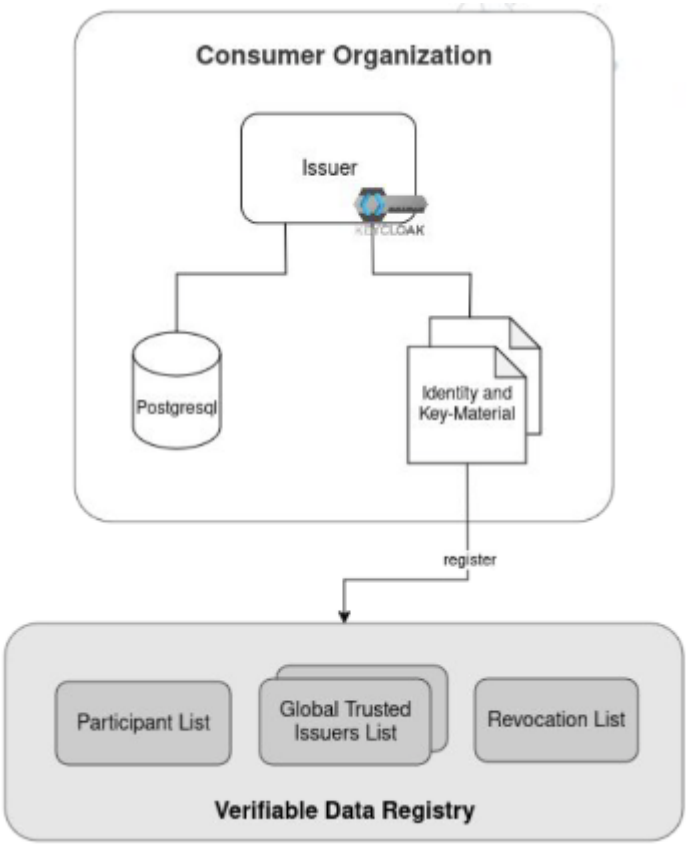
A consumer is **registered at the Verifiable Data Registry** and **issues credentials** to its users and services.

Main components of the consumer include:

- The credential issuer **Keycloak**
- **Postgresql** database for Keycloak

The organization has to be **registered** at the Verifiable Data Registry, with the corresponding **identity** and **key material** that have been created and provided.





Configuration values.yaml

Postgresql

```
# -- database for keycloak as the issuer needs to be registered
postgresql:
  primary:
    persistence:
      enabled: true
      # use one of the classes provided by your cluster
      storageClass: local-path
```

Keycloak

```
# -- keycloak as issuer of verifiable credentials is required
keycloak:
  ingress:
    enabled: true
    hostname: keycloak-consumer.127.0.0.1.nip.io

  externalDatabase:
    host: postgresql
    database: keycloak
    user: postgres
    existingSecret: database-secret
```

```
    existingSecretPasswordKey: postgres-admin-password

  extraEnvVars:
    # import the configured realm
    - name: KEYCLOAK_EXTRA_ARGS
      value: "--import-realm"
    # enable the issuance feature
    - name: KC_FEATURES
      value: "oid4vc-vci"
    # indicates ssl is terminated at the edge
    - name: KC_PROXY
      value: "edge"
    # password for reading the key store connected to the did
    - name: STORE_PASS
      value: test
    # keycloak admin password
    - name: KC_ADMIN_PASSWORD
      valueFrom:
        secretKeyRef:
          name: issuance-secret
          key: keycloak-admin
    # log level for keycloak
    - name: KC_LOG_LEVEL
      value: INFO
    # hostname of keycloak to be set as part of the realm config
    - name: KC_HOSTNAME
      value: keycloak-consumer.127.0.0.1.nip.io
    # did of the consumer
    - name: DID
      value: "did:key:<CONSUMER-KEY>"

  extraVolumeMounts:
    - name: did-material
      mountPath: /did-material/cert.pfx
      subPath: cert.pfx
    - name: realms
      mountPath: /opt/bitnami/keycloak/data/import

  extraVolumes:
    - name: did-material
      secret:
        secretName: consumer-identity
    - name: realms
      configMap:
        name: test-realm-realm

  realm:
    frontendUrl: http://keycloak-consumer.127.0.0.1.nip.io:8080
    import: true
    name: test-realm
    clientRoles: <CONSUMER-CLIENT-ROLES>
```

```
    users: <CONSUMER-USERS>
    clients: <CONSUMER-CLIENTS>
```

Keycloak Realm configuration

1. CONSUMER-CLIENT-ROLES

```
"${DID}": [
    {
        "name": "READER",
        "description": "Is allowed to see offers etc.",
        "clientRole": true
    },
    {
        "name": "OPERATOR",
        "description": "Is allowed to operate clusters.",
        "clientRole": true
    }
]
```

2. CONSUMER-USERS

```
{
    "username": "test-user",
    "enabled": true,
    "email": "test@user.org",
    "firstName": "Test",
    "lastName": "Reader",
    "credentials": [
        {
            "type": "password",
            "value": "test"
        }
    ],
    "clientRoles": {
        "${DID}": [
            "OPERATOR"
        ],
        "account": [
            "view-profile",
            "manage-account"
        ]
    },
    "groups": []
}
```

3. CONSUMER-CLIENTS

```
{
    "clientId": "${DID}",
    "enabled": true,
    "description": "Client to connect test.org",
    "surrogateAuthRequired": false,
    "alwaysDisplayInConsole": false,
    "clientAuthenticatorType": "client-secret",
    "defaultRoles": [],
    "redirectUris": [],
    "webOrigins": [],
    "notBefore": 0,
    "bearerOnly": false,
    "consentRequired": false,
    "standardFlowEnabled": true,
    "implicitFlowEnabled": false,
    "directAccessGrantsEnabled": false,
    "serviceAccountsEnabled": false,
    "publicClient": false,
    "frontchannelLogout": false,
    "protocol": "oid4vc",
    "attributes": {
        "client.secret.creation.time": "1675260539",
        "vc.user-credential.format": "jwt_vc",
        "vc.user-credential.scope": "UserCredential",
        "vc.verifiable-credential.format": "jwt_vc",
        "vc.verifiable-credential.scope": "VerifiableCredential",
        "vc.operator-credential.format": "jwt_vc",
        "vc.operator-credential.scope": "OperatorCredential"
    },
    "protocolMappers": [
        {
            "name": "target-role-mapper",
            "protocol": "oid4vc",
            "protocolMapper": "oid4vc-target-role-mapper",
            "config": {
                "subjectProperty": "roles",
                "clientId": "${DID}",
                "supportedCredentialTypes": "OperatorCredential"
            }
        },
        {
            "name": "context-mapper",
            "protocol": "oid4vc",
            "protocolMapper": "oid4vc-context-mapper",
            "config": {
                "context":
"https://www.w3.org/2018/credentials/v1",
                "supportedCredentialTypes":
"VerifiableCredential,UserCredential,OperatorCredential"
            }
        },
```

```
        {
            "name": "email-mapper",
            "protocol": "oid4vc",
            "protocolMapper": "oid4vc-user-attribute-mapper",
            "config": {
                "subjectProperty": "email",
                "userAttribute": "email",
                "supportedCredentialTypes":
"UserCredential,OperatorCredential"
            }
        },
        {
            "name": "firstName-mapper",
            "protocol": "oid4vc",
            "protocolMapper": "oid4vc-user-attribute-mapper",
            "config": {
                "subjectProperty": "firstName",
                "userAttribute": "firstName",
                "supportedCredentialTypes":
"UserCredential,OperatorCredential"
            }
        },
        {
            "name": "lastName-mapper",
            "protocol": "oid4vc",
            "protocolMapper": "oid4vc-user-attribute-mapper",
            "config": {
                "subjectProperty": "lastName",
                "userAttribute": "lastName",
                "supportedCredentialTypes":
"UserCredential,OperatorCredential"
            }
        }
    ],
    "authenticationFlowBindingOverrides": {},
    "fullScopeAllowed": true,
    "nodeReRegistrationTimeout": -1,
    "defaultClientScopes": [],
    "optionalClientScopes": []
}
```

Deployment of the Consumer

1. Create an identity for the consumer

1.1 Create a folder for the consumer identity material

```
mkdir consumer-identity
```

1.2 Generate the **private key** - do not get confused about the curve: openssl uses the name `prime256v1` for `secp256r1`(as defined by P-256)

```
openssl ecparam -name prime256v1 -genkey -noout -out consumer-identity/private-key.pem
```

1.3 Generate corresponding **public key**

```
openssl ec -in consumer-identity/private-key.pem -pubout -out consumer-identity/public-key.pem
```

1.4 Create a **(self-signed) certificate**

```
openssl req -new -x509 -key consumer-identity/private-key.pem -out consumer-identity/cert.pem -days 360
```

1.5 Export the **keystore**

```
openssl pkcs12 -export -inkey consumer-identity/private-key.pem -in consumer-identity/cert.pem -out consumer-identity/cert.pfx -name didPrivateKey
```

1.6 Check the contents

```
keytool -v -keystore consumer-identity/cert.pfx -list -alias didPrivateKey
```

1.7 Generate **DID** from the keystore

```
wget https://github.com/wistefan/did-helper/releases/download/0.1.1/did-helper

chmod +x did-helper

./did-helper -keystorePath ./consumer-identity/cert.pfx -keystorePassword=test
```

2. Create 'consumer' namespace

```
kubectl create namespace consumer
```

3. Deploy the key into the cluster

```
kubectl create secret generic consumer-identity --from-file=consumer-
identity/cert.pfx -n consumer
```

4. Install the consumer

```
helm install consumer-dsc data-space-connector/data-space-connector --
version 7.17.0 -f consumer/values.yaml --namespace=consumer

watch kubectl get pods -n consumer
```

The issuer can be accessed at: http://keycloak-consumer.127.0.0.1.nip.io:8080/realms/test-realm/account/oid4vci

5. Register the consumer at the Trust Anchor

```
curl -X POST http://til.127.0.0.1.nip.io:8080/issuer \
--header 'Content-Type: application/json' \
--data '{
    "did": "did:key:<CONSUMER-KEY>",
    "credentials": []
}'
```

6. Verify that the consumer has been registered correctly by querying the Trust Anchor's TIL API to fetch the list of the issuers

```
curl -X GET http://tir.127.0.0.1.nip.io:8080/v4/issuers
```

The credential can be decoded at https://jwt.io/.

Verify that the consumer is working correctly

Use the script `get_credential_for_consumer.sh` to get a credential from the deployed and registered consumer.

```
export USER_CREDENTIAL=$(./scripts/get_credential_for_consumer.sh
http://keycloak-consumer.127.0.0.1.nip.io:8080 operator-credential);
```

```
echo ${USER_CREDENTIAL}
```

Uninstall

```
helm uninstall consumer-dsc -n consumer
```

Return to index