Steps to connect to the DLT Infra Stack via the Java DL Interface

This document is to support partners to connect their DLT Infra stack to the Java DL Interface sample implementation.

Pre-Requisites

1. Partners are firstly required to download the archives for the following:
    a. DL Interface: dl-interface-<version>.zip
    b. Key Management API: key-management-api-<version>.zip
    c. Lambda Applications: DLT-partners-archi-codes_<version>.zip
2. After successfully onboarding to the SSG Training Ecosystem DL Network, partners are to capture the AWS credentials, Fabric chaincode properties, Lambda properties and SQS queue endpoint information. These details are required to use the sample DL Interface.
3. Partners should then configure their AWS credentials. Refer to the following link for the steps to setup the same: https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-credentials.html
4. The Java DL Interface is compatible with Java version 8 (https://www.java.com/en/download/) and Maven version 10 (minimum) (https://maven.apache.org/download.cgi).
5. Partners are then required to install software for local configuration data storage and handling; specifically Mongo DB for local encrypted key storage and Node JS for running the mock Key Management API. Refer to the following links :
   Mongo DB : https://www.mongodb.com/download-center/community
   Node JS (Any version between 8 to 12) : https://nodejs.org/en/download/
6. The steps below are illustrated using Eclipse v4.6 as the IDE (https://wiki.eclipse.org/Eclipse/Installation). Partners are free to use any other IDE of their choice.

Configuring Mongo DB and the mock Key management API

1. In the command line (or terminal), start MongoDB as shown below:

```
[redacted]$ mongo
MongoDB shell version v4.2.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("10e4d4f2-0d04-45bd-be5f-a2eea9ab8a3b") }
MongoDB server version: 4.0.3
WARNING: shell and server versions do not match
Server has startup warnings:
2020-05-24T15:36:56.983+0800 I CONTROL  [initandlisten]
2020-05-24T15:36:56.983+0800 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-24T15:36:56.984+0800 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-05-24T15:36:56.984+0800 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

Figure 1: Run MongoDB

2. Start the Key Management API
   - Unzip the key-management-api-<version>.zip
   - On your terminal run:

```
cd key-management-api/KeyManagement-Api
npm install
```

```
npm start
```

To verify that the API is available on the localhost, import the “KeyManagement-Api.postman_collection.json” from the key-management-api-<version> folder, in the Postman client app and try calling the API:

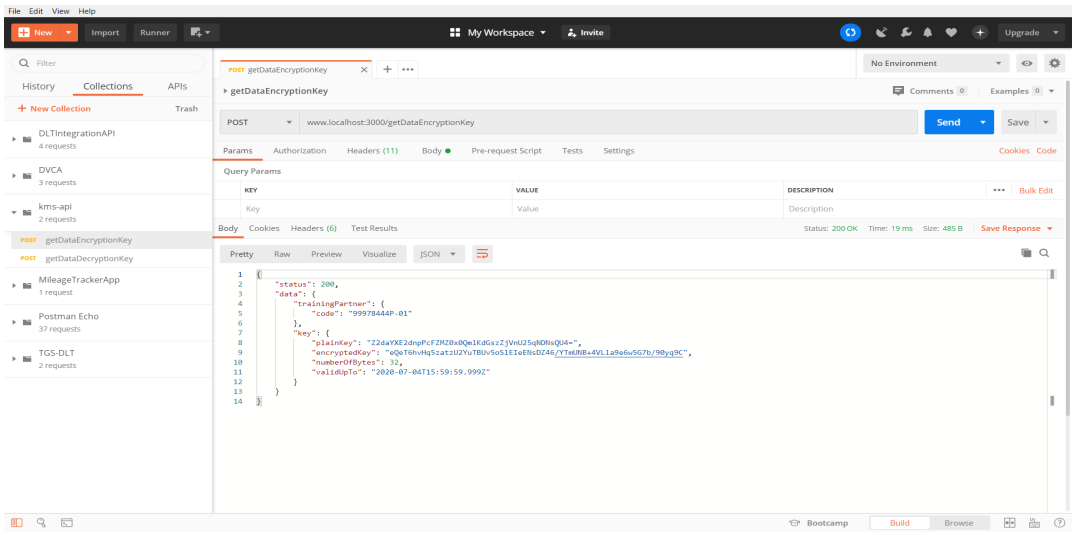

Figure 2: Successful response from Mock Key Management API

Return status of 200 indicates that the API is running on the localhost (port 3000).

Note: It is to be noted that the Key Management API is to be used only for dev and testing purpose until the real API endpoints are available on SSG’s API gateway.

Updating the DL Interface Configuration file

1. Unzip the dl-interface-<version>.zip archive. The contents of the archive are:
    - DL Interface sample implementation code (DLInterface)
    - DL Interface Configuration file (dl-interface.yaml)
2. Open the dl-interface.yaml file and edit the relevant properties for AWS credentials, Fabric properties, SQS endpoint URLs and Lambda properties (Refer to the section on Pre-requisites). Save the file after this is done.

dl-interface.yaml 〉 No Selection

```
dev:
  awsProperties:
    region: <region>
    accessKey: <accessKey>
    secretKey: <secretKey>

  dltProperties:
    networkId: <networkId>
    memberId: <memberId>
    enrolmentId: <enrolmentId>

  sqsUrl: <sqsUrl>
  lambdaChaincode: <lambdaChaincode>

  keyStorage:
    uri: mongodb://localhost:27017
    keyDB: dltKey
    keyCollection: orgKey

  keyMgmtAPI:
    url: http://localhost:3000
    getEncryptDataKeyEndpoint: /getDataEncryptionKey
    getDecryptDataKeyEndpoint: /getDataDecryptionKey

```

Figure 3: dl-interface.yaml file before update

A sample of the file after the required properties are updated is shown below:

dl-interface.yaml 〉 No Selection

```
dev:
  awsProperties:
    region: DEV_Region
    accessKey: DEV_AccessKey_XXXX
    secretKey: DEV_SecretKey_XXX/XXXX

  dltProperties:
    networkId: n-DEV_XXX
    memberId: m-DEV_XXX
    enrolmentId: DEV_admin

  sqsUrl: https://sqs.ap-southeast-1.amazonaws.com/XXX/XXX
  lambdaChaincode: Lambda-Ssg-Tgs-Prn-Apps-invoke-query-chaincode

  keyStorage:
    uri: mongodb://localhost:27017
    keyDB: dltKey
    keyCollection: orgKey

  keyMgmtAPI:
    url: http://localhost:3000
    getEncryptDataKeyEndpoint: /getDataEncryptionKey
    getDecryptDataKeyEndpoint: /getDataDecryptionKey

```

Figure 4: dl-interface.yaml file with sample values after update

Note: The credentials provided in the screenshots are just for reference purpose. These must be updated with the partners DLT stack properties.

Building the Java DL Interface

3. Build the DL Interface
   - Unzip the dl-interface-<version>.zip
   - On your terminal run:

     ```
     cd dl-interface
     ```

     If you want to skip running tests, run:

     ```
     mvn clean install -Dmaven.test.skip
     ```

     Otherwise, run:

     ```
     mvn clean install
     ```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  40.590 s
[INFO] Finished at: 2020-05-29T23:29:20+08:00
[INFO] ------------------------------------------------------------------------
```

Figure 5: DL Interface successful build

4. Install the jar to your maven repository
   - On your terminal run:
     `mvn install:install-file -Dfile=<FolderPath>/dl-interface/target/dl.interface-<version>-jar-with-dependencies.jar -DpomFile=<FolderPath>/dl-interface/pom.xml`

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------------< tgs:dl.interface >----------------------
[INFO] Building dl.interface 0.4.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-install-plugin:2.4:install-file (default-cli) @ dl.interfac
[INFO] Installing /Users[redacted]/dl-interface/target/dl.ir
[INFO] Installing /Users[redacted]/dl-interface/pom.xml to /
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.455 s
[INFO] Finished at: 2020-05-29T23:32:16+08:00
[INFO] ------------------------------------------------------------------------
```

Figure 6: Successful Installation of DL Interface jar to Maven repository

5. Add the dependency in your project's pom.xml:

```
<dependency>
      <groupId>tgs</groupId>
      <artifactId>dl.interface</artifactId>
      <version>0.X.0</version>
</dependency>
```

Using the DL Interface as dependency

After you have installed the DL Interface to your maven repository and added it to your project dependency, you can invoke the functions: `writeGrantsDataDLT` and `readGrantsDataDLT` from your project's working copy.
Below is the code snippet of a sample invocation of the DL Interface functions.

```
private Gateway gateway = new GatewayImpl();

public GatewayResponse writeGrantsData(RequestData request) {
    GatewayResponse response = new GatewayResponse();

    try {
        String requestString = new JsonMapper().serializeToJson(request);
        gateway.writeGrantsDataDLT(requestString);
        response.setStatus(HttpStatus.CREATED.value());
    } catch (JsonProcessingException e) {
        response.setStatus(HttpStatus.BAD_REQUEST.value());
        response.setResponse(e.getMessage());
    } catch (Exception e) {
        response.setStatus(HttpStatus.INTERNAL_SERVER_ERROR.value());
        response.setResponse(e.getMessage());
    }
    return response;
}

public GatewayResponse readGrantsData(RequestData request) {
    GatewayResponse response = new GatewayResponse();

    try {
        String requestString = new JsonMapper().serializeToJson(request);

        String result = gateway.readGrantsDataDLT(requestString);

        response.setStatus(HttpStatus.OK.value());
        response.setResponse(result);
    } catch (JsonProcessingException e) {
        response.setStatus(HttpStatus.BAD_REQUEST.value());
        response.setResponse(e.getMessage());
        e.printStackTrace();
    } catch (Exception e) {
        response.setStatus(HttpStatus.INTERNAL_SERVER_ERROR.value());
        response.setResponse(e.getMessage());
        e.printStackTrace();
    }
    return response;
}
```

Figure 7: Sample Invocation of DL Interface function

Running the DL Interface API

The DL Interface API also provides Rest Service API for the DL Interface sample implementation.

1. Build and run the API:
   - On your terminal run:
     ```
     cd dl-interface-api
     mvn clean install
     mvn spring-boot:run
     ```

```
calhost:27017] org.mongodb.driver.connection           : Opened connection [connectionId{localValue:1, serverValue:215}] to localh
          main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
calhost:27017] org.mongodb.driver.cluster              : Monitor thread successfully connected to server with description ServerDe
n=ServerVersion{versionList=[4, 0, 3]}, minWireVersion=0, maxWireVersion=7, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=
          main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
          main] s.d.s.w.s.ApiListingReferenceScanner     : Scanning for api listing references
          main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
          main] tgs.dl.infc.app.Application              : Started Application in 1.839 seconds (JVM running for 2.2)
```

Figure 8: DL Interface API Started

2. Access the swagger on your browser: http://localhost:8080/swagger-ui.html
3. Test the DL Interface API with the sample request on swagger ui.

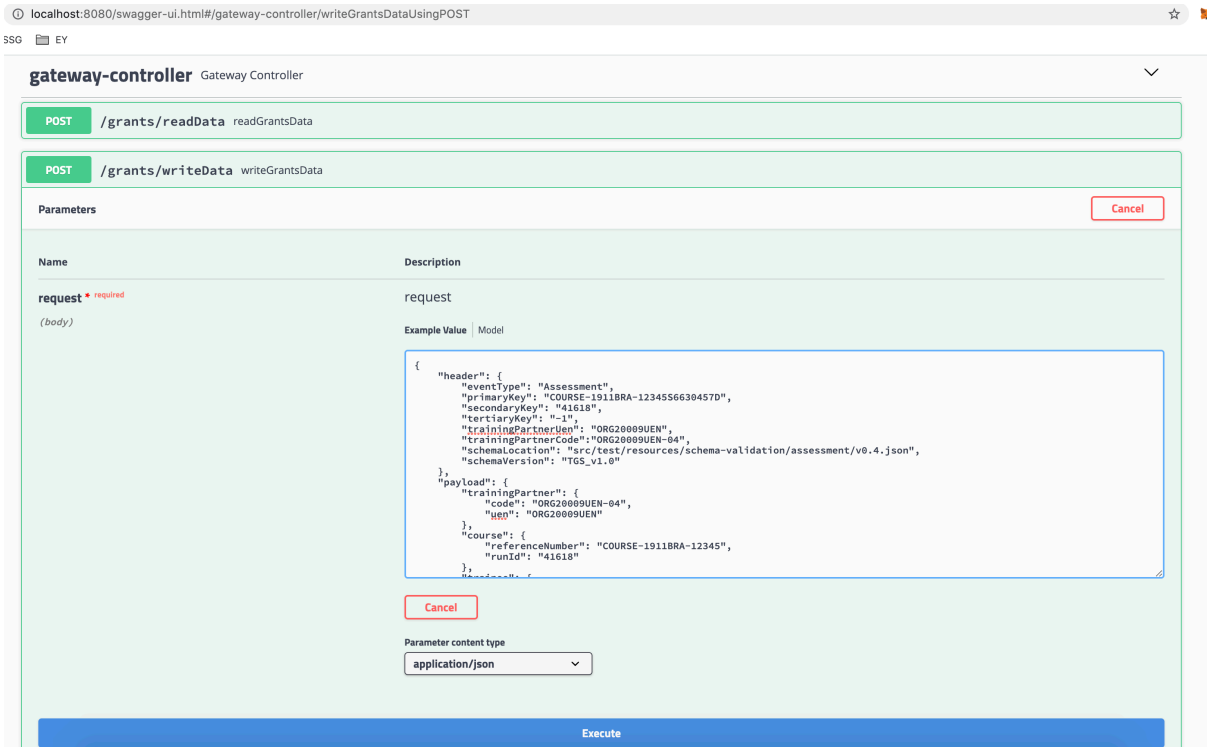

Figure 9: DL Interface Swagger UI