Steps to connect to the DLT Infra Stack via the .NET DL Interface

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

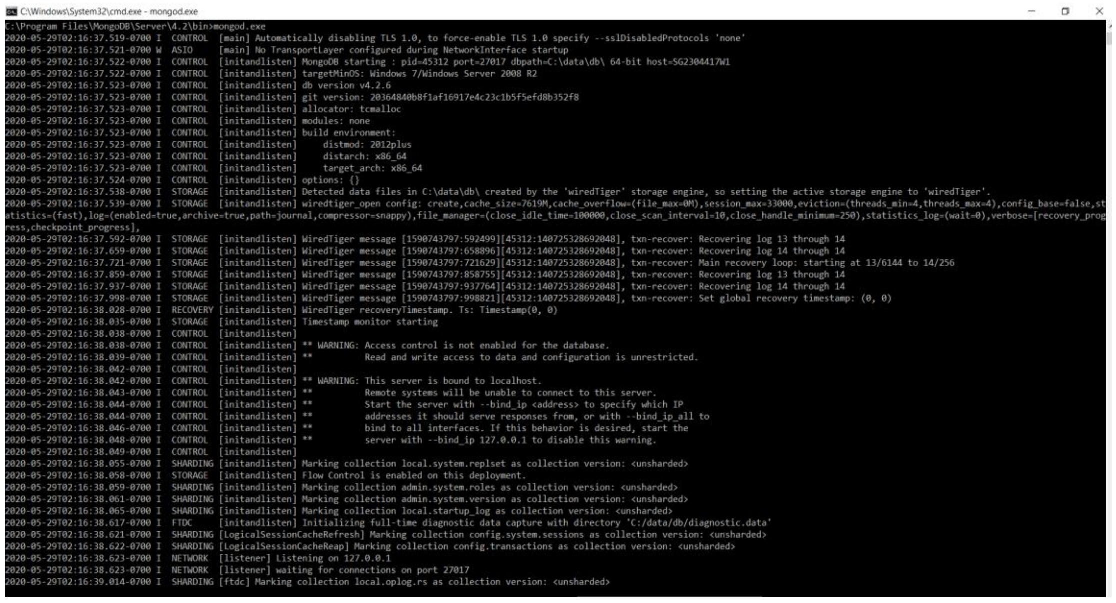
Pre-Requisites

Before using the DL Interface sample implementation code, partners are required to complete the below mentioned steps:

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. 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/
5. The steps below are illustrated using Visual Studio Code as the IDE. 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, navigate to the path where Mongo DB bin folder is present and run the `mongod.exe` command in order to run the Mongo Server, as shown below:

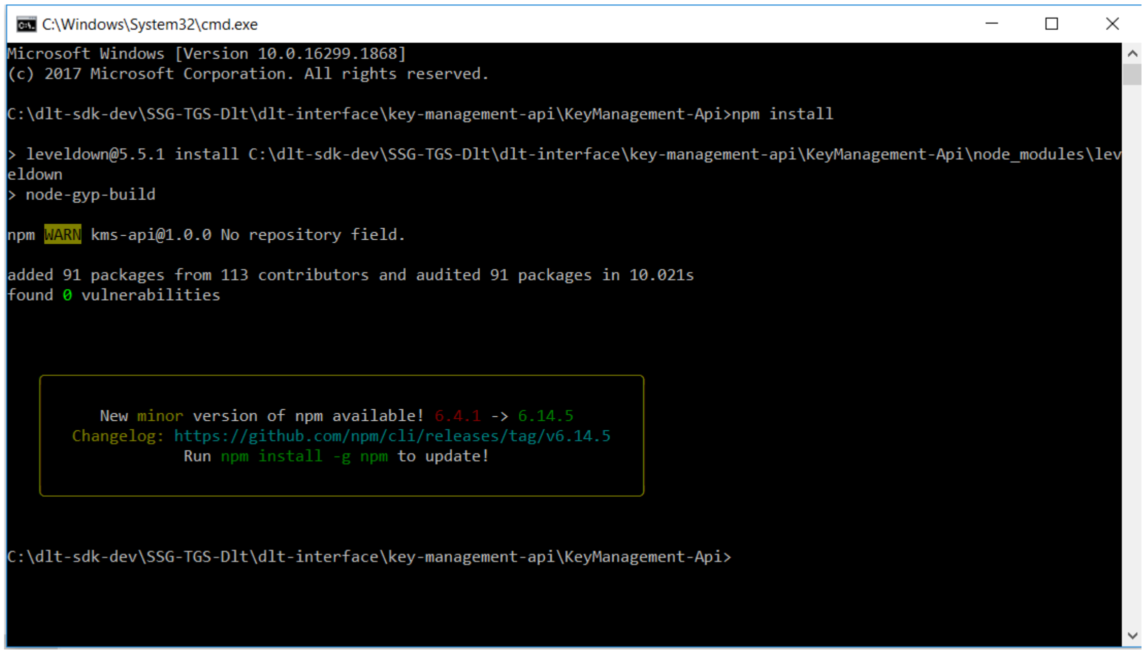


```
C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe
2020-05-29T02:16:37.519-0700 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-05-29T02:16:37.521-0700 W  ASIO     [main] No TransportLayer configured during NetworkInterface startup
2020-05-29T02:16:37.522-0700 I  CONTROL  [initandlisten] MongoDB starting : pid=45312 port=27017 dbpath=C:\data\db\ 64-bit host=SG2304417W1
2020-05-29T02:16:37.522-0700 I  CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten] db version v4.2.6
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten] git version: 20364840b8f1af16917e4c23c1b5f5efd8b352f8
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten] allocator: tcmalloc
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten] modules: none
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten] build environment:
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten]     distmod: 2012plus
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten]     distarch: x86_64
2020-05-29T02:16:37.523-0700 I  CONTROL  [initandlisten]     target_arch: x86_64
2020-05-29T02:16:37.524-0700 I  CONTROL  [initandlisten] options: {}
2020-05-29T02:16:37.538-0700 I  STORAGE  [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2020-05-29T02:16:37.539-0700 I  STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7619M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,st
atistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_prog
ress,checkpoint_progress],
2020-05-29T02:16:37.592-0700 I  STORAGE  [initandlisten] WiredTiger message [1590743797:592499][45312:140725328692048], txn-recover: Recovering log 13 through 14
2020-05-29T02:16:37.659-0700 I  STORAGE  [initandlisten] WiredTiger message [1590743797:658896][45312:140725328692048], txn-recover: Recovering log 14 through 14
2020-05-29T02:16:37.721-0700 I  STORAGE  [initandlisten] WiredTiger message [1590743797:721629][45312:140725328692048], txn-recover: Main recovery loop: starting at 13/6144 to 14/256
2020-05-29T02:16:37.859-0700 I  STORAGE  [initandlisten] WiredTiger message [1590743797:858755][45312:140725328692048], txn-recover: Recovering log 13 through 14
2020-05-29T02:16:37.937-0700 I  STORAGE  [initandlisten] WiredTiger message [1590743797:937764][45312:140725328692048], txn-recover: Recovering log 14 through 14
2020-05-29T02:16:37.998-0700 I  STORAGE  [initandlisten] WiredTiger message [1590743797:998821][45312:140725328692048], txn-recover: Set global recovery timestamp: (0, 0)
2020-05-29T02:16:38.028-0700 I  RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2020-05-29T02:16:38.035-0700 I  STORAGE  [initandlisten] Timestamp monitor starting
2020-05-29T02:16:38.038-0700 I  CONTROL  [initandlisten]
2020-05-29T02:16:38.038-0700 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-29T02:16:38.039-0700 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-05-29T02:16:38.042-0700 I  CONTROL  [initandlisten]
2020-05-29T02:16:38.042-0700 I  CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2020-05-29T02:16:38.043-0700 I  CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2020-05-29T02:16:38.044-0700 I  CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2020-05-29T02:16:38.044-0700 I  CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2020-05-29T02:16:38.046-0700 I  CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2020-05-29T02:16:38.048-0700 I  CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2020-05-29T02:16:38.049-0700 I  CONTROL  [initandlisten]
2020-05-29T02:16:38.055-0700 I  SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2020-05-29T02:16:38.058-0700 I  STORAGE  [initandlisten] Flow Control is enabled on this deployment.
2020-05-29T02:16:38.059-0700 I  SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2020-05-29T02:16:38.061-0700 I  SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-05-29T02:16:38.065-0700 I  SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-05-29T02:16:38.617-0700 I  FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2020-05-29T02:16:38.621-0700 I  SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
2020-05-29T02:16:38.622-0700 I  SHARDING [LogicalSessionCacheReap] Marking collection config.transactions as collection version: <unsharded>
2020-05-29T02:16:38.623-0700 I  NETWORK  [listener] Listening on 127.0.0.1
2020-05-29T02:16:38.623-0700 I  NETWORK  [listener] waiting for connections on port 27017
2020-05-29T02:16:39.014-0700 I  SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```

Figure 1: Start MongoDB

The mongo server is now up and running.

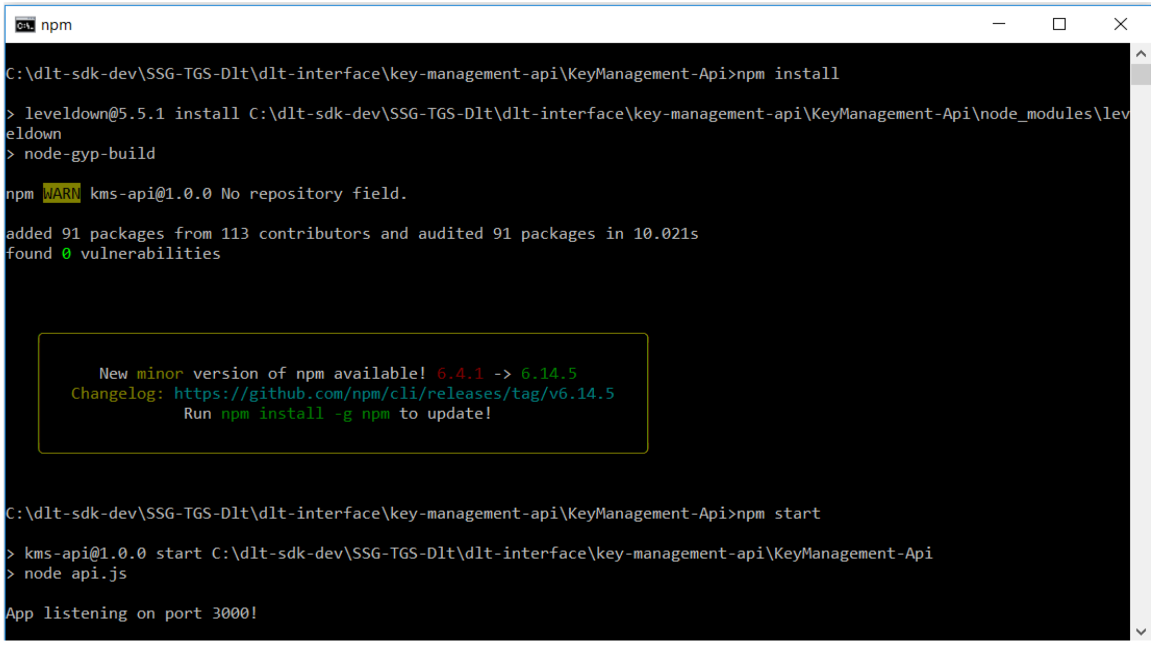
2. Unzip the "key-management-api-<version>.zip" folder on your local machine, navigate to the corresponding folder in the command line and run the `npm install` command to fetch all the node libraries, as shown below:



```
Microsoft Windows [Version 10.0.16299.1868]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api>npm install

> leveldown@5.5.1 install C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api\node_modules\lev
eldown
> node-gyp-build

npm WARN kms-api@1.0.0 No repository field.

added 91 packages from 113 contributors and audited 91 packages in 10.021s
found 0 vulnerabilities



       New minor version of npm available! 6.4.1 -> 6.14.5
   Changelog: https://github.com/npm/cli/releases/tag/v6.14.5
                 Run npm install -g npm to update!


C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api>
```

Figure 2: npm install

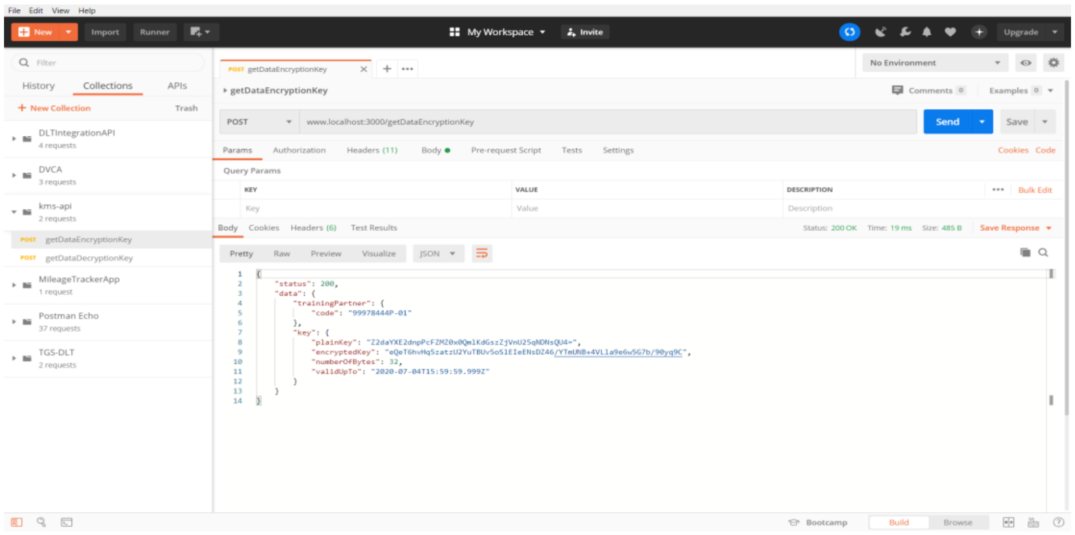
Once done, run the `npm start` command as shown below:



```
C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api>npm install

> leveldown@5.5.1 install C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api\node_modules\leveldown
> node-gyp-build

npm WARN kms-api@1.0.0 No repository field.

added 91 packages from 113 contributors and audited 91 packages in 10.021s
found 0 vulnerabilities



   ┌────────────────────────────────────────────────────────────┐
   │                                                            │
   │      New minor version of npm available! 6.4.1 -> 6.14.5   │
   │   Changelog: https://github.com/npm/cli/releases/tag/v6.14.5 │
   │               Run npm install -g npm to update!            │
   │                                                            │
   └────────────────────────────────────────────────────────────┘


C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api>npm start

> kms-api@1.0.0 start C:\dlt-sdk-dev\SSG-TGS-Dlt\dlt-interface\key-management-api\KeyManagement-Api
> node api.js

App listening on port 3000!
```

Figure 3: Running the Mock Key Management API

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:



POST www.localhost:3000/getDataEncryptionKey

Status: 200 OK Time: 19 ms Size: 485 B

```json
{
    "status": 200,
    "data": {
        "trainingPartner": {
            "code": "99978444P-01"
        },
        "key": {
            "plainKey": "Z2daYXE2dnpPcFZMZ0x0QmlKdGszZjVnU25qNDNsQU4=",
            "encryptedKey": "eQeT6hvHq5zatzU2YuTBUv5oS1EIeENsDZ46/YTmUNB+4VLla9e6w5G7b/90yq9C",
            "numberOfBytes": 32,
            "validUpTo": "2020-07-04T15:59:59.999Z"
        }
    }
}
```

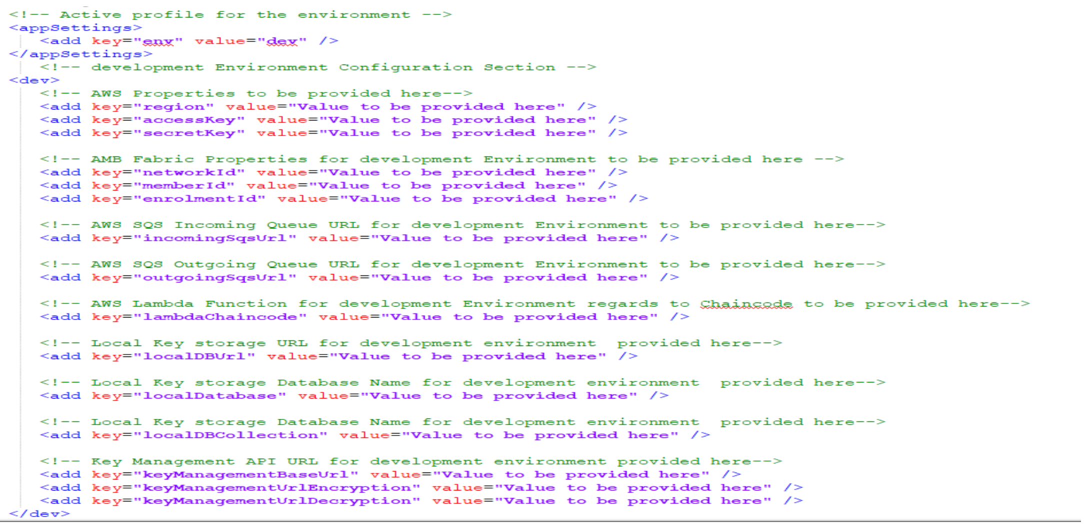
Figure 4: Successful response from Mock Key Management API

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 User.config file

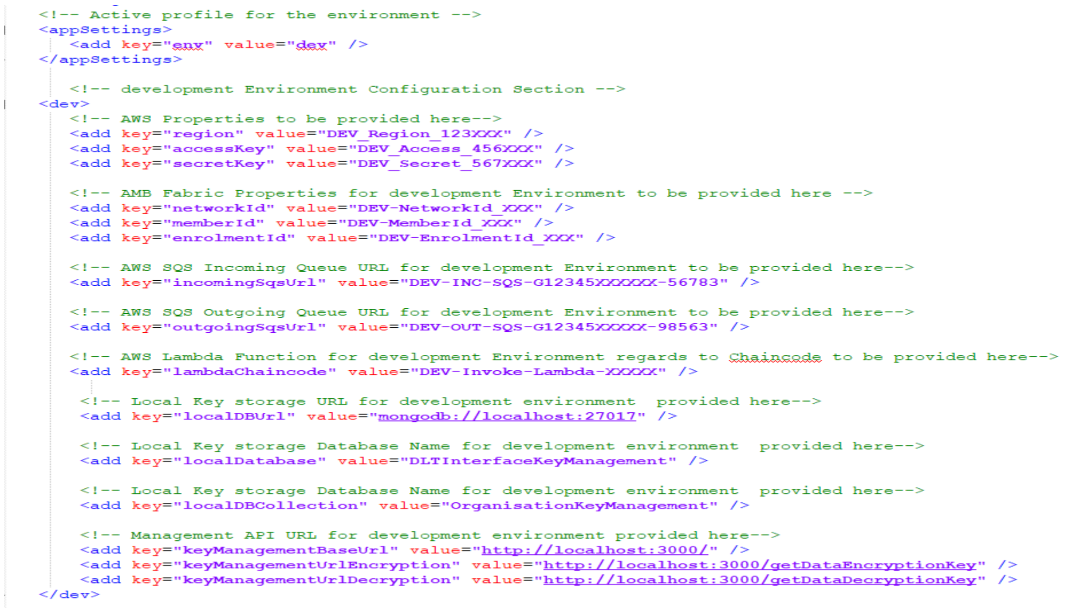
1. Unzip the dl-interface-<version>.zip archive. The contents of the archive are:
    - DL Interface sample implementation code (DLInterface)
    - User Configuration file (User.config)
2. Open the User.config file and edit the relevant properties for AWS credentials, Fabric properties, SQS endpoint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



```
<!-- Active profile for the environment -->
<appSettings>
   <add key="env" value="dev" />
</appSettings>
   <!-- development Environment Configuration Section -->
<dev>
   <!-- AWS Properties to be provided here-->
   <add key="region" value="Value to be provided here" />
   <add key="accessKey" value="Value to be provided here" />
   <add key="secretKey" value="Value to be provided here" />

   <!-- AMB Fabric Properties for development Environment to be provided here -->
   <add key="networkId" value="Value to be provided here" />
   <add key="memberId" value="Value to be provided here" />
   <add key="enrolmentId" value="Value to be provided here" />

   <!-- AWS SQS Incoming Queue URL for development Environment to be provided here-->
   <add key="incomingSqsUrl" value="Value to be provided here" />

   <!-- AWS SQS Outgoing Queue URL for development Environment to be provided here-->
   <add key="outgoingSqsUrl" value="Value to be provided here" />

   <!-- AWS Lambda Function for development Environment regards to Chaincode to be provided here-->
   <add key="lambdaChaincode" value="Value to be provided here" />

   <!-- Local Key storage URL for development environment  provided here-->
   <add key="localDBUrl" value="Value to be provided here" />

   <!-- Local Key storage Database Name for development environment  provided here-->
   <add key="localDatabase" value="Value to be provided here" />

   <!-- Local Key storage Database Name for development environment  provided here-->
   <add key="localDBCollection" value="Value to be provided here" />

   <!-- Key Management API URL for development environment provided here-->
   <add key="keyManagementBaseUrl" value="Value to be provided here" />
   <add key="keyManagementUrlEncryption" value="Value to be provided here" />
   <add key="keyManagementUrlDecryption" value="Value to be provided here" />
</dev>
```

Figure 5: User.config file before update

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



```
<!-- Active profile for the environment -->
<appSettings>
   <add key="env" value="dev" />
</appSettings>

   <!-- development Environment Configuration Section -->
<dev>
   <!-- AWS Properties to be provided here-->
   <add key="region" value="DEV_Region_123XXX" />
   <add key="accessKey" value="DEV_Access_456XXX" />
   <add key="secretKey" value="DEV_Secret_567XXX" />

   <!-- AMB Fabric Properties for development Environment to be provided here -->
   <add key="networkId" value="DEV-NetworkId_XXX" />
   <add key="memberId" value="DEV-MemberId_XXX" />
   <add key="enrolmentId" value="DEV-EnrolmentId_XXX" />

   <!-- AWS SQS Incoming Queue URL for development Environment to be provided here-->
   <add key="incomingSqsUrl" value="DEV-INC-SQS-G12345XXXXXX-56783" />

   <!-- AWS SQS Outgoing Queue URL for development Environment to be provided here-->
   <add key="outgoingSqsUrl" value="DEV-OUT-SQS-G12345XXXXX-98563" />

   <!-- AWS Lambda Function for development Environment regards to Chaincode to be provided here-->
   <add key="lambdaChaincode" value="DEV-Invoke-Lambda-XXXXX" />

    <!-- Local Key storage URL for development environment  provided here-->
    <add key="localDBUrl" value="mongodb://localhost:27017" />

    <!-- Local Key storage Database Name for development environment  provided here-->
    <add key="localDatabase" value="DLTInterfaceKeyManagement" />

    <!-- Local Key storage Database Name for development environment  provided here-->
    <add key="localDBCollection" value="OrganisationKeyManagement" />

    <!-- Management API URL for development environment provided here-->
    <add key="keyManagementBaseUrl" value="http://localhost:3000/" />
    <add key="keyManagementUrlEncryption" value="http://localhost:3000/getDataEncryptionKey" />
    <add key="keyManagementUrlDecryption" value="http://localhost:3000/getDataDecryptionKey" />
</dev>
```

Figure 6: User.config 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.

Importing the DL Interface code and connecting to the DLT Stack

1. Open Visual Studio 2019 and then click on the "*Open a project or solution*".

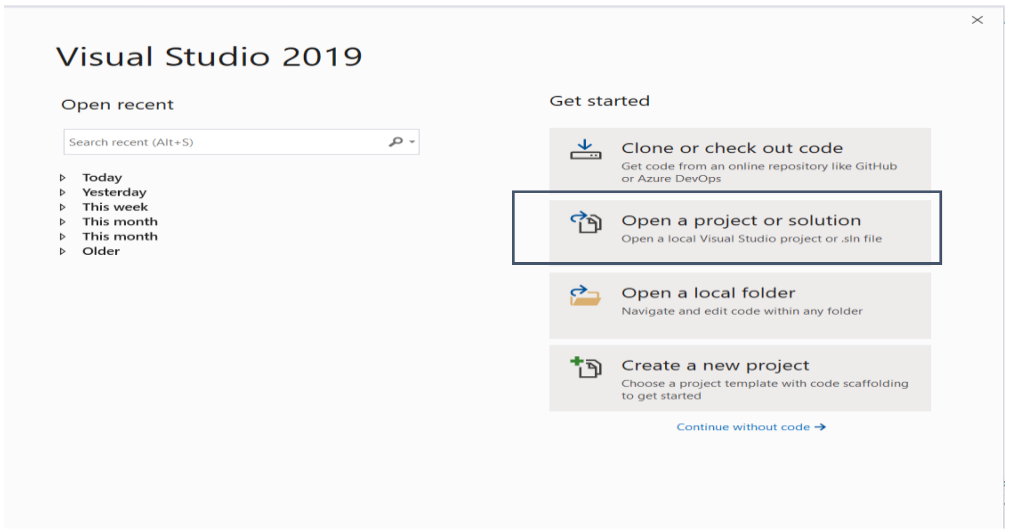

Visual Studio 2019
Open recent
Search recent (Alt+S)
Today
Yesterday
This week
This month
This month
Older
Get started
Clone or check out code
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

Figure 7: Open the .sln file

2. Locate the DLInterface.sln file by navigating to the dl-interface-<version> folder. The DLInterface sample implementation code will appear in the solution explorer.

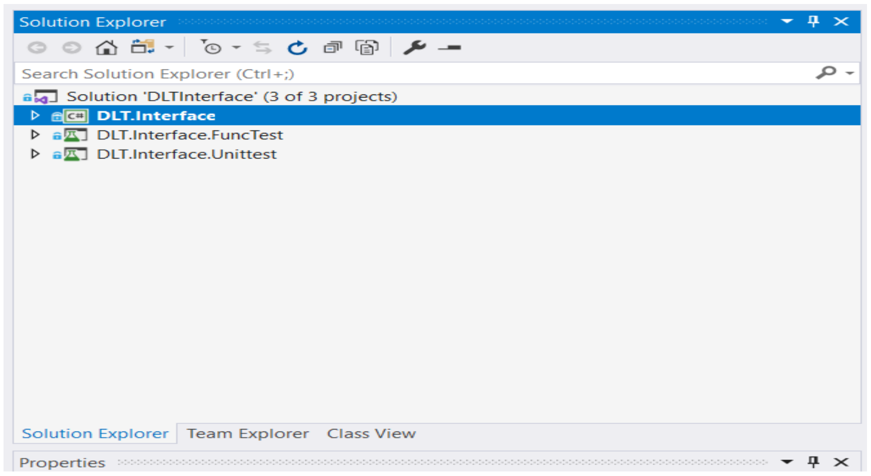
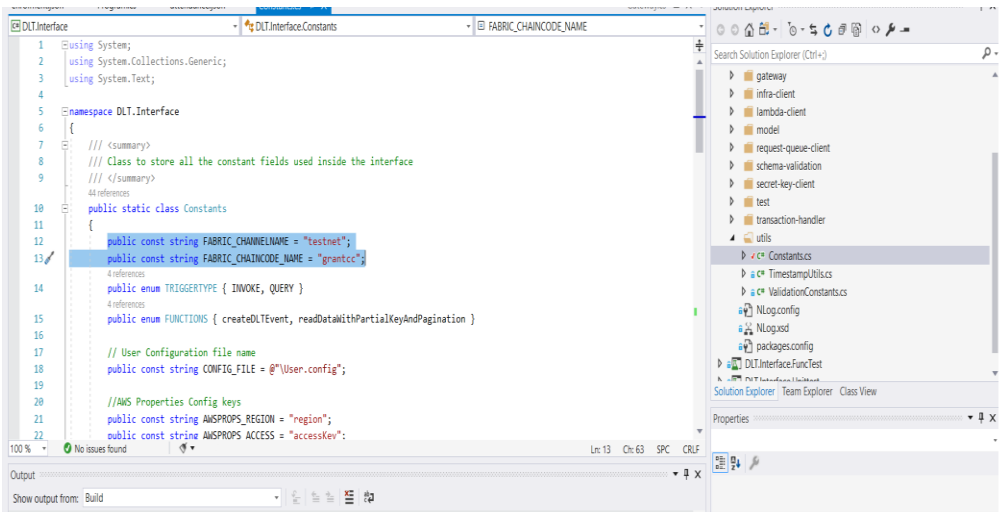


Figure 8: Open the solution file

3. Verify that the FABRIC_CHANNEL_NAME and FABRIC_CHAINCODE_NAME in Constants.cs is pointing to the correct channel name and chaincode name of the partners DLT Stack.

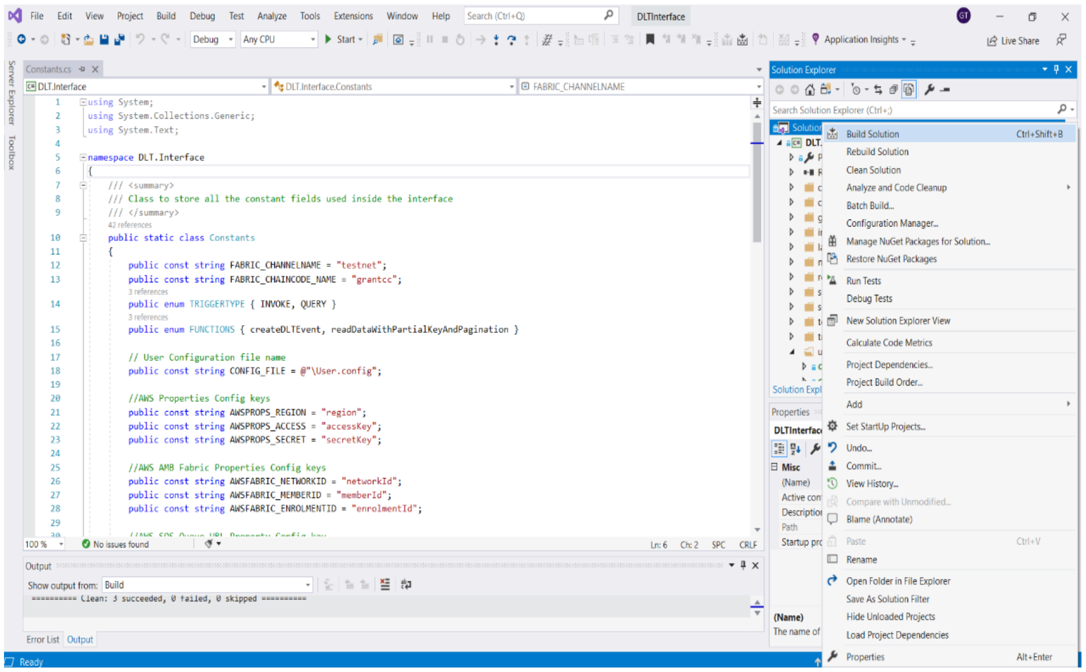


```
using System;
using System.Collections.Generic;
using System.Text;

namespace DLT.Interface
{
    /// <summary>
    /// Class to store all the constant fields used inside the interface
    /// </summary>
    44 references
    public static class Constants
    {
        public const string FABRIC_CHANNELNAME = "testnet";
        public const string FABRIC_CHAINCODE_NAME = "grantcc";
        4 references
        public enum TRIGGERTYPE { INVOKE, QUERY }
        4 references
        public enum FUNCTIONS { createDLTEvent, readDataWithPartialKeyAndPagination }

        // User Configuration file name
        public const string CONFIG_FILE = @"\User.config";

        //AWS Properties Config keys
        public const string AWSPROPS_REGION = "region";
        public const string AWSPROPS_ACCESS = "accessKey";
```

```
100 %    No issues found                                   Ln: 13   Ch: 63   SPC   CRLF
```

Figure 9: Channel and Chaincode names

Build the solution by choosing "*Build Solution*" on the Solution tab



```
using System;
using System.Collections.Generic;
using System.Text;

namespace DLT.Interface
{
    /// <summary>
    /// Class to store all the constant fields used inside the interface
    /// </summary>
    42 references
    public static class Constants
    {
        public const string FABRIC_CHANNELNAME = "testnet";
        public const string FABRIC_CHAINCODE_NAME = "grantcc";
        3 references
        public enum TRIGGERTYPE { INVOKE, QUERY }
        3 references
        public enum FUNCTIONS { createDLTEvent, readDataWithPartialKeyAndPagination }

        // User Configuration file name
        public const string CONFIG_FILE = @"\User.config";

        //AWS Properties Config keys
        public const string AWSPROPS_REGION = "region";
        public const string AWSPROPS_ACCESS = "accessKey";
        public const string AWSPROPS_SECRET = "secretKey";

        //AWS AMB Fabric Properties Config keys
        public const string AWSFABRIC_NETWORKID = "networkId";
        public const string AWSFABRIC_MEMBERID = "memberId";
        public const string AWSFABRIC_ENROLMENTID = "enrolmentId";
```

```
100 %    No issues found                                   Ln: 6   Ch: 2   SPC   CRLF
```

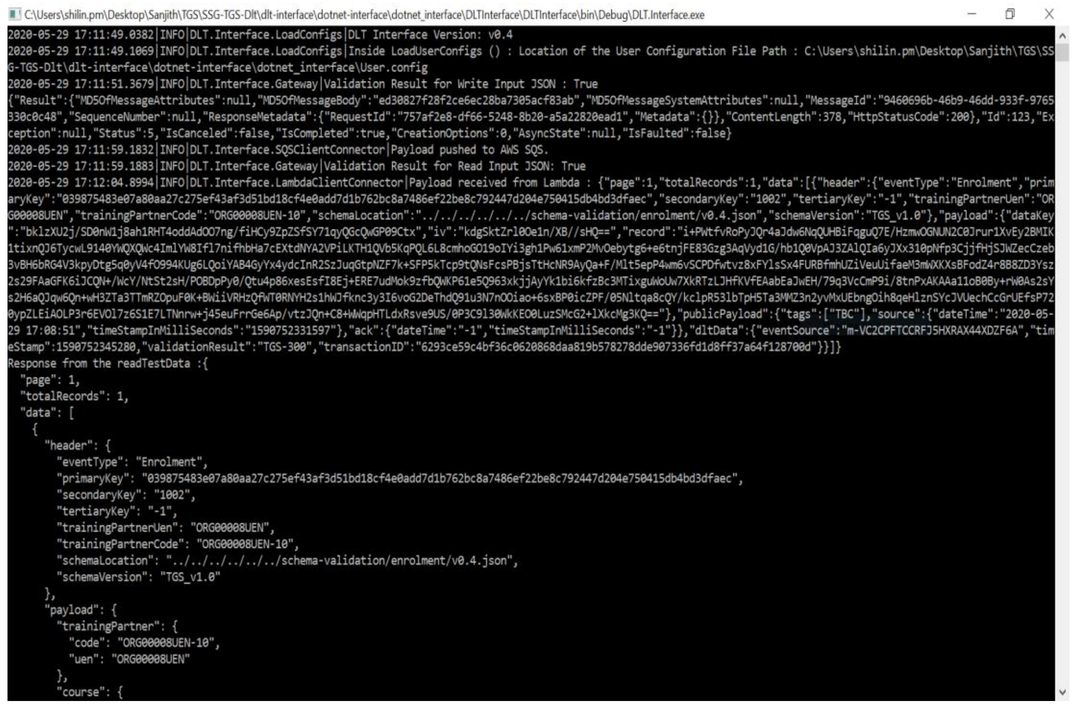
Output

Show output from: Build

```
========== Clean: 3 succeeded, 0 failed, 0 skipped ==========
```

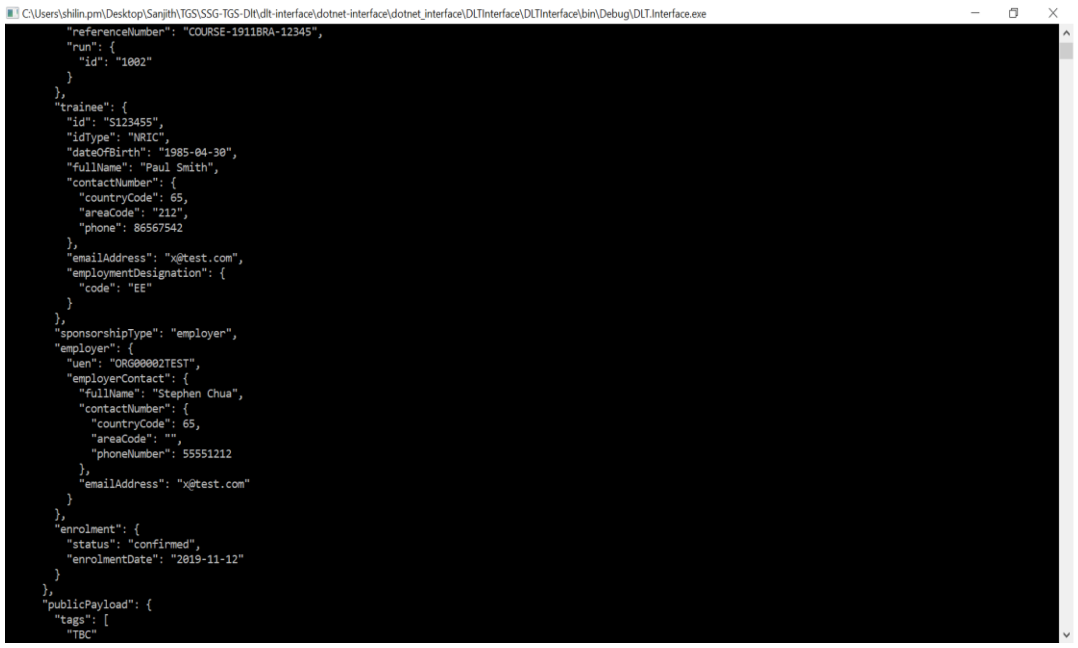
Figure 10: Build Solution

4. After the code build is successful , run `Ctrl+F5` to execute the test implementation to write to the AWS SQS Incoming queue and read from the lambda.



```
C:\Users\shilin.pm\Desktop\Sanjith\TGS\SSG-TGS-Dlt\dlt-interface\dotnet-interface\dotnet_interface\DLTInterface\DLTInterface\bin\Debug\DLT.Interface.exe
2020-05-29 17:11:49.0382|INFO|DLT.Interface.LoadConfigs|DLT Interface Version: v0.4
2020-05-29 17:11:49.1069|INFO|DLT.Interface.LoadConfigs|Inside LoadUserConfigs () : Location of the User Configuration File Path : C:\Users\shilin.pm\Desktop\Sanjith\TGS\SS
G-TGS-Dlt\dlt-interface\dotnet-interface\dotnet_interface\User.config
2020-05-29 17:11:51.3679|INFO|DLT.Interface.Gateway|Validation Result for Write Input JSON : True
{"Result":{"MD5OfMessageAttributes":null,"MD5OfMessageBody":"ed30827f28f2ce6ec28ba7305acf83ab","MD5OfMessageSystemAttributes":null,"MessageId":"9460696b-46b9-46dd-933f-9765
330c0c48","SequenceNumber":null,"ResponseMetadata":{"RequestId":"757af2e8-df66-5248-8b20-a5a22820ead1","Metadata":{}},"ContentLength":378,"HttpStatusCode":200},"Id":123,"Ex
ception":null,"Status":5,"IsCanceled":false,"IsCompleted":true,"CreationOptions":0,"AsyncState":null,"IsFaulted":false}
2020-05-29 17:11:59.1832|INFO|DLT.Interface.SQSClientConnector|Payload pushed to AWS SQS.
2020-05-29 17:11:59.1883|INFO|DLT.Interface.Gateway|Validation Result for Read Input JSON: True
2020-05-29 17:12:04.8994|INFO|DLT.Interface.LambdaClientConnector|Payload received from Lambda : {"page":1,"totalRecords":1,"data":[{"header":{"eventType":"Enrolment","prim
aryKey":"039875483e07a80aa27c275ef43af3d51bd18cf4e0add7d1b762bc8a7486ef22be8c792447d204e750415db4bd3dfaec","secondaryKey":"1002","tertiaryKey":"-1","trainingPartnerUen":"OR
G00008UEN","trainingPartnerCode":"ORG00008UEN-10","schemaLocation":"../../../../../../schema-validation/enrolment/v0.4.json","schemaVersion":"TGS_v1.0"},"payload":{"dataKey
":"bklzXU2j/SD0nW1j8ah1RHT4oddAdOO7ng/fiHCy9ZpZSfSY71qyQGcQwGP09Ctx","iv":"kdgSktZrl0Oe1n/XB//sHQ==","record":"i+PWtfvRoPyJQr4aJdw6NqQUHBiFqguQ7E/HzmwOGNUN2C0Jrur1XvEy2BMIK
1tixnQJ6TycwL9140YWQXQWc4ImlYW8Ifl7nifhbHa7cEXtdNYA2VPiLKTH1QVb5KqPQL6L8cmhoGO19oIYi3gh1Pw61xmP2MvOebytg6+e6tnjFE83Gzg3AqVyd1G/hb1Q0VpAJ3ZAlQIa6yJXx310pNfp3CjjfHjSJWZecCzeb
3vBH6bRG4V3kpyDtg5q0yV4fO994KUg6LQoiYAB4GyYx4ydcInR2SzJuqGtpNZF7k+SFP5kTcp9tQNsFcsPBjsTtHcNR9AyQa+F/Mlt5epP4wm6vSCPDfwtvz8xFYlsSx4FURBfmhUZiVeuUifaeM3mWXKXsBFodZ4r8B8ZD3Ysz
2s29FAaGFK6iJCQN+/WcY/NtSt2sH/POBDpPy0/Qtu4p86xesEsfI8Ej+ERE7udMok9zfbQWKP61e5Q963xkjjAyYk1bi6kfzBc3MTixguWoUw7XkRTzLJHfKVfEAabEaJwEH/79q3VcCmP9i/8tnPxAKAAa11oB0By+rW0As2sY
s2H6aQJqw6Qn+wH3ZTa3TTmRZOpuF0K+BWiiVRHzQfWT0RNYH2s1hWJfknc3y3I6voG2DeThdQ91u3N7nOOiao+6sxBP0icZPF/05Nltqa8cQY/kclpR53lbTpH5Ta3MMZ3n2yvMxUEbngOih8qeHlznSYcJVUechCcGrUEfsP72
0ypZLEiAOLP3r6EVOl7z6S1E7LTNnrw+j45euFrrGe6Ap/vtzJQn+C8+WWqpHTLdxRsve9US/0P3C9l30WkKEO0LuzSMcG2+lXkcMg3KQ=="},"publicPayload":{"tags":["TBC"],"source":{"dateTime":"2020-05-
29 17:08:51","timeStampInMilliSeconds":"1590752331597"},"ack":{"dateTime":"-1","timeStampInMilliSeconds":"-1"}},"dltData":{"eventSource":"m-VC2CPFTCCRFJ5HXRAX44XDZF6A","tim
eStamp":1590752345280,"validationResult":"TGS-300","transactionID":"6293ce59c4bf36c0620868daa819b578278dde907336fd1d8ff37a64f128700d"}}]}
Response from the readTestData :{
  "page": 1,
  "totalRecords": 1,
  "data": [
    {
      "header": {
        "eventType": "Enrolment",
        "primaryKey": "039875483e07a80aa27c275ef43af3d51bd18cf4e0add7d1b762bc8a7486ef22be8c792447d204e750415db4bd3dfaec",
        "secondaryKey": "1002",
        "tertiaryKey": "-1",
        "trainingPartnerUen": "ORG00008UEN",
        "trainingPartnerCode": "ORG00008UEN-10",
        "schemaLocation": "../../../../../../schema-validation/enrolment/v0.4.json",
        "schemaVersion": "TGS_v1.0"
      },
      "payload": {
        "trainingPartner": {
          "code": "ORG00008UEN-10",
          "uen": "ORG00008UEN"
        },
        "course": {
```

Figure 11: Run the .NET DL Interface



```
C:\Users\shilin.pm\Desktop\Sanjith\TGS\SSG-TGS-Dlt\dlt-interface\dotnet-interface\dotnet_interface\DLTInterface\DLTInterface\bin\Debug\DLT.Interface.exe
          "referenceNumber": "COURSE-1911BRA-12345",
          "run": {
            "id": "1002"
          }
        },
        "trainee": {
          "id": "S123455",
          "idType": "NRIC",
          "dateOfBirth": "1985-04-30",
          "fullName": "Paul Smith",
          "contactNumber": {
            "countryCode": 65,
            "areaCode": "212",
            "phone": 86567542
          },
          "emailAddress": "x@test.com",
          "employmentDesignation": {
            "code": "EE"
          }
        },
        "sponsorshipType": "employer",
        "employer": {
          "uen": "ORG00002TEST",
          "employerContact": {
            "fullName": "Stephen Chua",
            "contactNumber": {
              "countryCode": 65,
              "areaCode": "",
              "phoneNumber": 55551212
            },
            "emailAddress": "x@test.com"
          }
        },
        "enrolment": {
          "status": "confirmed",
          "enrolmentDate": "2019-11-12"
        }
      },
      "publicPayload": {
        "tags": [
          "TBC"
```

Figure 12: Run the .NET DL Interface (Contd)