Yugandhar Microservice Platform

API and Transaction Reference Guide

Yugandhar Microservice Platform Release - V1.0.0
Date – 23/05/2018

Contents

About Yugandhar Project

The Yugandhar Project is the umbrella project focused on building open source cloud ready solutions. The current offerings include Yugandhar Microservice Platform (MSP) and Yugandhar Open Master Data Management (MDM) Hub.

About Yugandhar Microservice Platform

Yugandhar Microservice Platform (also referred as Yugandhar MSP) provides framework for rapid development of your Microservice. This is an architecturally proven Springboot based application having all the basic components needed for a Microservice application to work which just needs to be extended as per your requirements.

Yugandhar Microservice platform codes with code generation templates for rapid development. Just design your data model and generate the code using Yugandhar templates, your base table services will be ready. To create the composite services, you may have to write custom code which would take minimal efforts. Your new Microservice would be ready in just few hours.

About this document

This document covers the Out of the box (OOTB) API and services provided by Microservice Platform which can be invoked from client applications. The services can be utilized to build a new composite service having custom business logic.

Invoking Transactions on Yugandhar Microservice Platform

- ## REST Webservice

Testing the Yugandhar Microservice Platform REST webservices is covered in Development Environment Setup guide section 'TEST With SOAPUI'.

- ## MQ interface

MQ messages can be invoked using MQ client. You may build your own java class to send message to request queue by taking reference from YugJMSMessageSender class, the response will be sent to default response queue. Also take help from activemq site http://activemq.apache.org/how-to-unit-test-jms-code.html.

We will be using SOAPUI based testing approach in this document.

The JSON Message layout

Common structure for invoking all transactions

All the transactions must follow a predefined format to invoke transaction on Microservice Platform. The request and response have Tranaction Transfer Object (TxnTransferObj) as the top level object for every transaction. The transfer object encapsulates Header and Payload objects. The response may additionally have error response object inside transfer object.

Request Message Layout

- **Header Object**

The header object provides the meta information about the transaction ,it has below attributes

- requesterLanguage: The language of the request. It is referred from CONFIG_LANGUAGE_CODE table.
- requesterLocale: The locale of the requester which can be used to write custom logic.
- userName: The username of the requester. This attribute is used for authorization
- userRole: The role of the requester. This attribute is used for authorization.
- txnCorrelationId: The correlation id of the message. This is sent back in the response as received in the request.
- txnMessageId: The message id of the transaction. The value of this attribute is used to update the UPDATED_TXN_REF_ID during persistent transaction. i.e. for create / update transaction all the tables getting updated will have UPDATED_TXN_REF_ID updated with this this id which can be used to trace the objects changed in one transaction.
- requestOriginSource: This attribute is used to define the request origin of the message. This attribute can be used to perform any additional logic while executing transaction.
- requesterSystem: This attribute is used to define the requesting system of the message. This attribute can be used to perform any additional logic while executing transaction.
- transactionResponseCode: This is the response code of the transaction which can either be SUCCESS or FAIL.
- requestTimeStamp: The timestamp when the request is received.
- responseTimeStamp: The timestamp when the response is sent back.
- totalExecutionTimeMillies: The total execution time of the transaction in milliseconds.
- transactionServiceName: The name of the transaction being invoked on Microservice Platform. E.g. createLegalEntity.

Sample header Object in the request -

```
"txnHeader": {
    "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "",
    "requestOriginSource": null,
    "requesterSystem": null,
     "transactionServiceName": "createAuthRolesRegistryBase"
   }
```

- **TxnPayload Object**

The txnPayload is the default object encapsulating the entire set of data objects.This object has the pagination related properties as well. Ensure to make an entry in this object for all the custom objects being developed manually or using Yugandhar Templates.

Refer Javadoc for the list of DOs and attributes of this object.

- **Primary Key Auto generation Vs providing in the request**

Snippet: createAuthRolesRegistryBase transaction with user provided idpk

```
      "transactionServiceName": " createAuthRolesRegistryBase "
   },
         "txnPayload": {
               "authRolesRegistryDO":      {
                        "primaryKeyDO": {
               "idPk": "77776663666AAAA211"
               },

               "roleName": "TESTROLE",
               "description": "TESTROLE"
 }
```

Snippet: createAuthRolesRegistryBase transaction to have auto generated idpk

```
      "transactionServiceName": " createAuthRolesRegistryBase "
   },
               "txnPayload": {
                "authRolesRegistryDO":      {
```

```
                    "idPk": null,
                    "roleName": "TESTROLE",
                    "description": "TESTROLE"
    }
```

Refer section 'Pluggable primary keys' section of 'Development and Customization guide' for more details.

Response Message Layout

- **Header Object**

Same as request object. Additionally 'totalExecutionTimeMillies' is sent in the response which denotes the time in milliseconds consumed for processing the request.

- **TxnPayload Object**

Same as request object. Additionally the response would have responseCode as 'SUCCESS' or 'FAIL' which denotes if the transaction is successful or failed in Microservice Platform.

- **Error Response**

Sample error response

```
{
  "responseCode": "FAIL",
  "txnHeader":   {
    "requesterLanguage": "1",
    "userName": "admin",
    "userRole": "admin",
    "txnMessageId": "1234567890123",
    "transactionServiceName": "createAuthRolesRegistryBase"
  },
  "txnPayload": {"errorResponseObj":   {
    "validationResult": false,
    "errorCode": "10073",
    "errorMessage": "authRolesRegistryDO.roleName should not be null",
    "additionalErrorMessage": "AuthRolesRegistryDO.roleName should not be null"
  }}
}
```

Sample Request and Response Message

Sample Request


```
{
        "txnHeader": {
         "requesterLanguage": "1",
     "requesterLocale": null,
     "userName": "admin",
     "userRole": "admin",
     "accessToken": null,
     "txnCorrelationId": null,
     "txnMessageId": "1234567890123",
     "requestOriginSource": null,
     "requesterSystem": null,
     "paginationStartIndex": null,
     "paginationEndIndex": null,
     "paginationTotalResultCount": null,
          "transactionServiceName":
"createAuthRolesRegistryBase"
        },
        "txnPayload": {
                         "authRolesRegistryDO":       {
                        "roleName": "TESTROLE",
                        "description": "TESTROLE"
     }

        }

}
```


Sample Response


```
{
        "responseCode": "SUCCESS",
        "txnHeader": {
                "requesterLanguage": "1",
                "userName": "admin",
                "userRole": "admin",
                "txnMessageId": "1234567890123",
                "transactionServiceName": "createAuthRolesRegistryBase",
                "totalExecutionTimeMillies": "116"
        },
        "txnPayload": {
                "authRolesRegistryDO": {
                        "idPk": "aba62bcd-7dc7-4937-89e6-4851362c4c76",
                        "version": 0,
                        "createdTs": "2018-05-22T10.20.03.999+0000",
                        "updatedTs": "2018-05-22T10.20.03.999+0000",
                        "updatedByUser": "admin",
                        "updatedByTxnId": "1234567890123",
                        "roleName": "TESTROLE",
                        "description": "TESTROLE"
                }
        }
}
```

List of Out of the Box Transactions

Base entity services

Base entity services perform CRUD (create, retrieve, update and delete) operations on single database entity e.g createAuthRolesRegistryBase or createConfigAppPropertiesBase. By default Yugandhar MSP perform only soft delete by setting the deleted_ts to mark the entity as deleted. To perform hard delete i.e. delete the row from database a new custom service needs to be written manually.

Composite Services

Composite entity services perform CRUD (create, retrieve, update and delete) operations on multiple database entities as single transaction. Also composite services can have composition of create/ retrieve/ update or delete in single services. e.g. the out of the box SearchAuthRolesService do the search multiple data tables and retrieve the results. You may create new composite service by taking reference of SearchAuthRolesService.

Application Configuration Management services

1. createConfigAppPropertiesBase

Description: create a record in CONFIG_APP_PROPERTIES table

Transaction Service Name: createConfigAppPropertiesBase

Request DO: ConfigAppPropertiesDO

Response DO: ConfigAppPropertiesDO

Mandatory Input: key, value

2. createConfigErrorcodeRegistryBase

Description:create a record in CONFIG_ERRORCODE_REGISTRY table

Transaction Service Name: createConfigErrorcodeRegistryBase

Request DO: ConfigErrorcodeRegistryDO

Response DO: ConfigErrorcodeRegistryDO

Mandatory Input: configLanguageCodeKey, errorCode, errorMessage

3. createConfigLanguageCodeBase

Description:create a record in CONFIG_LANGUAGE_CODE table.

Transaction Service Name: createConfigLanguageCodeBase

Request DO: ConfigLanguageCodeDO

Response DO: ConfigLanguageCodeDO

Mandatory Input: key, value

4. createConfigTxnRegistryBase

Description: create a record in CONFIG_TXN_REGISTRY table

Transaction Service Name: createConfigTxnRegistryBase

Request DO: ConfigTxnRegistryDO

Response DO: ConfigTxnRegistryDO

Mandatory Input: txnserviceName, txnserviceClass,txnserviceClassmethod

5. findAllConfigErrorcodeRegistryByLanguageCodeBase

Description:find all records in CONFIG_ERRORCODE_REGISTRY table for given language code

Transaction Service Name: findAllConfigErrorcodeRegistryByLanguageCodeBase

Request DO: ConfigErrorcodeRegistryDO

Response DO: ConfigErrorcodeRegistryDO

Mandatory Input: configLanguageCodeKey

inquiryFilter: *ACTIVE/INACTIVE/ALL*

6. findAllConfigLanguageCodesBase

Description:find all records in CONFIG_LANGUAGE_CODE table based on idpk

Transaction Service Name: findAllConfigLanguageCodesBase

Request DO: ConfigLanguageCodeDO

Response DO: ConfigLanguageCodeDO

Mandatory Input: N/A

inquiryFilter: *ACTIVE/INACTIVE/ALL*

7. findConfigAppPropertiesByBusinessKeyBase

Description: find a record in CONFIG_APP_PROPERTIES table based on key

Transaction Service Name: findConfigAppPropertiesByBusinessKeyBase

Request DO: ConfigAppPropertiesDO

Response DO: ConfigAppPropertiesDO

Mandatory Input: key

inquiryFilter: *ACTIVE/INACTIVE/ALL*

8. findConfigErrorcodeRegistryByBusinessKeyBase

Description:find a record in CONFIG_ERRORCODE_REGISTRY table based on business key

Transaction Service Name: findConfigErrorcodeRegistryByBusinessKeyBase

Request DO: ConfigErrorcodeRegistryDO

Response DO: ConfigErrorcodeRegistryDO

Mandatory Input: configLanguageCodeKey, errorCode

9. findConfigLanguageCodeByBusinessKeyBase

Description:find a record in CONFIG_LANGUAGE_CODE table based on business key

Transaction Service Name: findConfigLanguageCodeByBusinessKeyBase

Request DO: ConfigLanguageCodeDO

Response DO: ConfigLanguageCodeDO

Mandatory Input: key

10. findConfigTxnRegistryByBusinessKeyBase

Description: find a record in CONFIG_TXN_REGISTRY table based on business key

Transaction Service Name: findConfigTxnRegistryByBusinessKeyBase

Request DO: ConfigTxnRegistryDO

Response DO: ConfigTxnRegistryDO

Mandatory Input: txnserviceName

11. retrieveConfigAppPropertiesBase

Description: retrieve a record in CONFIG_APP_PROPERTIES table based on idpk

Transaction Service Name: retrieveConfigAppPropertiesBase

Request DO: ConfigAppPropertiesDO

Response DO: ConfigAppPropertiesDO

Mandatory Input: idPk

12. retrieveConfigErrorcodeRegistryBase

Description:retrieve a record in CONFIG_ERRORCODE_REGISTRY table based on idpk

Transaction Service Name: retrieveConfigErrorcodeRegistryBase

Request DO: ConfigErrorcodeRegistryDO

Response DO: ConfigErrorcodeRegistryDO

Mandatory Input: idPk

13. retrieveConfigLanguageCodeBase

Description: retrieve a record in CONFIG_LANGUAGE_CODE table based on idpk

Transaction Service Name: retrieveConfigLanguageCodeBase

Request DO: ConfigLanguageCodeDO

Response DO: ConfigLanguageCodeDO

Mandatory Input: idPk

14. retrieveConfigTxnRegistryBase

Description: retrieve a record in CONFIG_TXN_REGISTRY table based on idpk

Transaction Service Name: retrieveConfigTxnRegistryBase

Request DO: ConfigTxnRegistryDO

Response DO: ConfigTxnRegistryDO

Mandatory Input: idPk

15. updateConfigAppPropertiesBase

Description*:* update a record in CONFIG_APP_PROPERTIES table based on idpk

Transaction Service Name: updateConfigAppPropertiesBase

Request DO: ConfigAppPropertiesDO

Response DO: ConfigAppPropertiesDO

Mandatory Input: idPk and version

16. updateConfigErrorcodeRegistryBase

Description: update a record in CONFIG_ERRORCODE_REGISTRY table based on idpk

Transaction Service Name: updateConfigErrorcodeRegistryBase

Request DO: ConfigErrorcodeRegistryDO

Response DO: ConfigErrorcodeRegistryDO

Mandatory Input: idPk and version

17. updateConfigLanguageCodeBase

Description: update a record in CONFIG_LANGUAGE_CODE table based on idpk

Transaction Service Name: updateConfigLanguageCodeBase

Request DO: ConfigLanguageCodeDO

Response DO: ConfigLanguageCodeDO

Mandatory Input: idPk, version

18. updateConfigTxnRegistryBase

Description: update a record in CONFIG_TXN_REGISTRY table based on idpk

Transaction Service Name: updateConfigTxnRegistryBase

Request DO: ConfigTxnRegistryDO

Response DO: ConfigTxnRegistryDO

Mandatory Input: idPk, version

Authorization Framework services

1. createAuthRolesRegistryBase

Description: create a record in AUTH_ROLES_REGISTRY table

Transaction Service Name: createAuthRolesRegistryBase

Request DO: AuthRolesRegistryDO

Response DO: AuthRolesRegistryDO

Mandatory Input: roleName

2. updateAuthRolesRegistryBase

Description: update a record in AUTH_ROLES_REGISTRY based on idPk

Transaction Service Name: updateAuthRolesRegistryBase

Request DO: AuthRolesRegistryDO

Response DO: AuthRolesRegistryDO

Mandatory Input: idPk, version, roleName

3. retrieveAuthRolesRegistryBase

Description: retrieve a record fromAUTH_ROLES_REGISTRY based on idPk

Transaction Service Name: retrieveAuthRolesRegistryBase

Request DO: AuthRolesRegistryDO

Response DO: AuthRolesRegistryDO

Mandatory Input: idPk

4. **findAuthRolesRegistryByBusinessKeyBase**

Description: find a record from AUTH_ROLES_REGISTRY based on business key

Transaction Service Name: findAuthRolesRegistryByBusinessKeyBase

Request DO: authRolesRegistryDO

Response DO: authRolesRegistryDO

Mandatory Input: roleName

5. **createAuthUserRegistryBase**

Description: create a record in AUTH_USER_REGISTRY table

Transaction Service Name: createAuthUserRegistryBase

Request DO: authUserRegistryDO

Response DO: authUserRegistryDO

Mandatory Input: userName

6. **updateAuthUserRegistryBase**

Description: update a record in AUTH_USER_REGISTRY based on idPk

Transaction Service Name: updateAuthUserRegistryBase

Request DO: authUserRegistryDO

Response DO: authUserRegistryDO

Mandatory Input: idPk, version, userName

7. **retrieveAuthUserRegistryBase**

Description: retrieve a record from AUTH_USER_REGISTRY based on idPk

Transaction Service Name: retrieveAuthUserRegistryBase

Request DO: authUserRegistryDO

Response DO: authUserRegistryDO

Mandatory Input: idPk

8. findAuthUserRegistryByBusinessKeyBase

Description: *f*ind a record from AUTH_USER_REGISTRY based business key

Transaction Service Name: findAuthUserRegistryByBusinessKeyBase

Request DO: authUserRegistryDO

Response DO: authUserRegistryDO

Mandatory Input: userName

9. createAuthUserroleAccesscontrolBase

Description: create a record in AUTH_USERROLE_ACCESSCONTROL table

Transaction Service Name: createAuthUserroleAccesscontrolBase

Request DO: authUserroleAccesscontrolDO

Response DO: authUserroleAccesscontrolDO

Mandatory Input: profileType, authUserRoleRegistryIdpk, configTxnRegistryIdpk

10. updateAuthUserroleAccesscontrolBase

Description: update a record in AUTH_USERROLE_ACCESSCONTROL table based on idPk

Transaction Service Name: updateAuthUserroleAccesscontrolBase

Request DO: authUserroleAccesscontrolDO

Response DO: authUserroleAccesscontrolDO

Mandatory Input: idPk, version, profileType, authUserRoleRegistryIdpk, configTxnRegistryIdpk

11. retrieveAuthUserroleAccesscontrolBase

Description: retrieve a record from AUTH_USERROLE_ACCESSCONTROL based on idPk

Transaction Service Name: retrieveAuthUserroleAccesscontrolBase

Request DO: authUserroleAccesscontrolDO

Response DO: authUserroleAccesscontrolDO

Mandatory Input: idPk

12. findAuthUserroleAccesscontrolByBusinessKeyBase

Description: find a record from AUTH_USERROLE_ACCESSCONTROL table based on business key

Transaction Service Name: findAuthUserroleAccesscontrolByBusinessKeyBase

Request DO: authUserroleAccesscontrolDO

Response DO: authUserroleAccesscontrolDO

Mandatory Input: profileType, authUserRoleRegistryIdpk, configTxnRegistryIdpk

13. findAuthUserroleAccesscontrolByRegistryIdpkBase

Description: find a record from AUTH_USERROLE_ACCESSCONTROL based on business key

Transaction Service Name: findAuthUserroleAccesscontrolByRegistryIdpkBase

Request DO: authUserroleAccesscontrolDO

Response DO: *authUserroleAccesscontrolDO*

Mandatory Input: profileType, authUserRoleRegistryIdpk

14. createAuthUserRoleAssocBase

Description: create a record in AUTH_USER_ROLE_ASSOC table

Transaction Service Name: createAuthUserRoleAssocBase

Request DO: authUserRoleAssocDO

Response DO: authUserRoleAssocDO

Mandatory Input: authRolesRegistryIdpk, authUserRegistryIdpk

15. updateAuthUserRoleAssocBase

Description: update a record in AUTH_USER_ROLE_ASSOC based on idPk

Transaction Service Name*:* *updateAuthUserRoleAssocBase*

Request DO*:* *authUserRoleAssocDO*

Response DO: *authUserRoleAssocDO*

Mandatory Input: *idPk, version , authRolesRegistryIdpk, authUserRegistryIdpk*

16. retrieveAuthUserRoleAssocBase

Description: retrieve a record from AUTH_USER_ROLE_ASSOC based on idPk

Transaction Service Name: retrieveAuthUserRoleAssocBase

Request DO: authUserRoleAssocDO

Response DO: authUserRoleAssocDO

Mandatory Input: idPk

17. findAuthUserRoleAssocByBusinessKeyBase

Description: find a record from AUTH_USER_ROLE_ASSOC based on business key

Transaction Service Name: findAuthUserRoleAssocByBusinessKeyBase

Request DO: authUserRoleAssocDO

Response DO: authUserRoleAssocDO

Mandatory Input: authRolesRegistryIdpk and authUserRegistryIdpk

18. findAllRecordsByAuthUserRegistryIdpkBase

Description: find AUTH_USER_ROLE_ASSOC based on authUserRegistryIdpk

Transaction Service Name: findAllRecordsByAuthUserRegistryIdpkBase

Request DO: authUserRoleAssocDO

Response DO: authUserRoleAssocDO

Mandatory Input: authUserRegistryIdpk

19. findAllRecordsByAuthRolesRegistryIdpkBase

Description: find all records from AUTH_USER_ROLE_ASSOC based on authRolesRegistryIdpk

Transaction Service Name: findAllRecordsByAuthRolesRegistryIdpkBase

Request DO: authUserRoleAssocDO

Response DO: authUserRoleAssocDO

Mandatory Input: authRolesRegistryIdpk

Composite Services

1. searchAuthRoles

Description: This service returns the roles based on username and/or role name. This transaction search the database as per below condition

1. If username and userRole are present in the request then this transaction will return the result only if below conditions are satisfied
   - ✓ The given user must be present in AUTH_USER_REGISTRY table.
   - ✓ The given role must be present in AUTH_ROLES_REGISTRY table.
   - ✓ The user must be associated with the given role in AUTH_USER_ROLE_ASSOC table.
   - ✓ The result will have the rows from AUTH_ROLES_REGISTRY table for given user name and role name combination.
2. If userRole is present and username is null in the request then this transaction will return the result only if below conditions are satisfied
   - ✓ The given role must be present in AUTH_ROLES_REGISTRY table.
   - ✓ The result will have the rows from AUTH_ROLES_REGISTRY table for given role.
3. If username is present and userRole is null in the request then this transaction will return the result only if below conditions are satisfied
   - ✓ The given user must be present in AUTH_USER_REGISTRY table.
   - ✓ The result will have the rows from AUTH_ROLES_REGISTRY table to which the user is associated in AUTH_USER_ROLE_ASSOC table. i.e. all the roles to which a user is associated with will be returned.

Transaction Service Name: searchAuthRoles

Request DO: searchAuthAccessControlDO

Response DO: authRolesRegistryDOList (empty list if no records found)

Mandatory Input: rolename or username

Pagination Supported: Yes

Filter: ACTIVE / INACTIVE / ALL

Wildcard Support: Yes. % can be used for wildcard searches

Use Case –

1. To check if given userName and roleName combination is valid
2. To check if roleName is valid
3. To get all the roles associated with a valid user

Special Notes – This service is used to in authorization framework.

2. searchAuthUsers

Description: This service returns the roles based on username and/or role name. This transaction search the database as per below condition

4. If username and userRole are present in the request then this transaction will return the result only if below conditions are satisfied
    - ✓ The given user must be present in AUTH_USER_REGISTRY table.
    - ✓ The given role must be present in AUTH_ROLES_REGISTRY table.
    - ✓ The user must be associated with the given role in AUTH_USER_ROLE_ASSOC table.
    - ✓ The result will have the rows from AUTH_USER_REGISTRY table for given user name and role name combination.
5. If userRole is present and username is null in the request then this transaction will return the result only if below conditions are satisfied
    - ✓ The given role must be present in AUTH_ROLES_REGISTRY table.
    - ✓ The result will have the rows from AUTH_USER_REGISTRY table for given role.
6. If username is present and userRole is null in the request then this transaction will return the result only if below conditions are satisfied
    - ✓ The given user must be present in AUTH_USER_REGISTRY table.
    - ✓ The result will have the rows from AUTH_USER_REGISTRY table to which the user is associated in AUTH_USER_ROLE_ASSOC table. i.e. all the users to which a role is associated with will be returned.

Transaction Service Name: searchAuthRoles

Request DO: searchAuthAccessControlDO

Response DO: authUserRegistryDOList (empty list if no records found)

Mandatory Input: rolename or username

Pagination Supported: Yes

Filter: ACTIVE / INACTIVE / ALL

Wildcard Support: Yes. % can be used for wildcard searches

Use Case –

1. To check if given userName and roleName combination is valid
2. To check if userName is valid
3. To get all the users associated with a valid role

Special Notes – N/A

3. searchAuthAccessControl

Description: This service is used to get the transaction service name based on userName or roleName

The response is sent back based on below rule

1. If username and userRole are present in the request then authorization if all of the below conditions are satisfied
   - ✓ The given user must be present in AUTH_USER_REGISTRY table.
   - ✓ The given role must be present in AUTH_ROLES_REGISTRY table.
   - ✓ The user must be associated with the given role in AUTH_USER_ROLE_ASSOC table
   - ✓ The response will be sent back having all the rows if txnserviceName have entry in Access control table for given role i.e. AUTH_USERROLE_ACCESSCONTROL have entry for given role and transaction service name.
2. If userRole is present and username is null in the request then role is authorized to perform the given transaction if all the below conditions are satisfied
   - ✓ The given role must be present in AUTH_ROLES_REGISTRY table.
   - ✓ The response will be sent back having all the rows if txnserviceName have entry in Access control table for given role i.e. AUTH_USERROLE_ACCESSCONTROL have entry for given role and transaction service name.
3. If username is present and userRole is null in the request then user is authorized to perform the given transaction if all the below conditions are satisfied
   - ✓ The given user must be present in AUTH_USER_REGISTRY table.
   - ✓ The response will be sent back having all the rows if txnserviceName have entry in Access control table for given user i.e. AUTH_USERROLE_ACCESSCONTROL have entry for given user and transaction service name.

Transaction Service Name: searchAuthAccessControl

Request DO: searchAuthAccessControlDO

Response DO: authUserRegistryDOList (empty list if no records found)

Mandatory Input: txnserviceName and either rolename or userName

Pagination Supported: Yes

Filter: ACTIVE / INACTIVE / ALL

Wildcard Support: NO

Use Case – To check if given user or role is authorized to invoke a given transaction.

Special Notes – This is the primary service used by Microservice Platform to perform authorication of the request based on userName/roleName and txnserviceName

Reference data Table Management services

All the reference data services follow same structure as follows. The transactionServiceName and txnPayload.xxxxxxxDO will change as per the reference list name.

Create Request - sample JSON message

```
{
       "txnHeader": {
        "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
    "requestOriginSource": null,
    "requesterSystem": null,
       "transactionServiceName": "create<referenceTableName>Base"
       },
       "txnPayload": {
                       "<referenceTableName>DO":       {
                       "idPk": null,
                       "version": null,
                       "createdTs": null,
                       "deletedTs": null,
                       "updatedTs": null,
                       "updatedByUser": null,
                       "updatedByTxnId": null,
                       "configLanguageCodeKey": null,
                       "key": null,
                       "value": null,
                       "description": null
    }}

}
```

update request - sample JSON message

```
{
       "txnHeader": {
              "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
```

```
    "requestOriginSource": null,
    "requesterSystem": null,
       "transactionServiceName": "update<referenceTableName>Base"
       },
       "txnPayload": {
                       "<referenceTableName>DO":      {
                       "idPk": null,
                       "version": null,
                       "createdTs": null,
                       "deletedTs": null,
                       "updatedTs": null,
                       "updatedByUser": null,
                       "updatedByTxnId": null,
                       "configLanguageCodeKey": null,
                       "key": null,
                       "value": null,
                       "description": null
    }

       }

}
```

Retrieve Request - sample JSON message

```
{
       "txnHeader": {
              "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
    "requestOriginSource": null,
    "requesterSystem": null,
        "transactionServiceName": "retrieve<referenceTableName>Base"
       },
       "txnPayload": {
                       "<referenceTableName>DO":      {
      "idPk": "1"

    }

       }

}
```

findByBusinessKey request - sample JSON message.

```
{
       "txnHeader": {
              "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
    "requestOriginSource": null,
    "requesterSystem": null,
              "transactionServiceName": "find<referenceTableName>ByBusinessKeyBase"
       },
       "txnPayload": {
                     "<referenceTableName>DO":      {
              "configLanguageCodeKey": "1",
      "key": "ACTIVE"
    }

       }

}
```

findAllRecordsByLanguageCode request - sample JSON message.

The response of this message is the list object of request DO. E.g. if the request do is refCountryIsoDO then the response object (in the txnPayload) would be refCountryIsoDOList.

```
{
       "txnHeader": {
              "requesterLanguage": "1",
              "userName": "Rakesh9",
              "userRole": "admin9",
              "txnMessageId": "12312311115999",
              "transactionServiceName":
"findAll<referenceTableName>ByLanguageCodeBase"
       },
       "txnPayload": {
              "paginationIndexOfCurrentSlice": 0,
              "paginationPageSize": 25,
              "<referenceTableName>DO":      {
              "configLanguageCodeKey": "1"
    }
```

```
        }

}
```

All the below mentioned services would have same structure as mentioned above so those are not covered in detail independently.

List of Reference data Management services following common request structure

1. ***createRefCountryIsoBase***
2. ***createRefCurrencyBase***
3. ***createRefLanguageCodeBase***
4. ***findAllRefCountryIsoByLanguageCodeBase***
5. ***findAllRefCurrencyByLanguageCodeBase***
6. ***findAllRefLanguageCodeByLanguageCodeBase***
7. ***findRefCountryIsoByBusinessKeyBase***
8. ***findRefCurrencyByBusinessKeyBase***
9. ***findRefLanguageCodeByBusinessKeyBase***
10. ***retrieveRefCountryIsoBase***
11. ***retrieveRefCurrencyBase***
12. ***retrieveRefLanguageCodeBase***
13. ***updateRefCountryIsoBase***
14. ***updateRefCurrencyBase***
15. ***updateRefLanguageCodeBase***