**Yugandhar Microservice Platform**

**API and Transaction Reference Guide**

Yugandhar Microservice Platform Release - V1.0.0

Date – 23/05/2018

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Copyright [2017] [Yugandhar Microservice Platform]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Contents

[About Yugandhar Project 5](#_Toc515536803)

[About this document 5](#_Toc515536804)

[Invoking Transactions on Yugandhar Microservice Platform 6](#_Toc515536805)

[ REST Webservice 6](#_Toc515536806)

[ MQ interface 6](#_Toc515536807)

[The JSON Message layout 7](#_Toc515536808)

[Common structure for invoking all transactions 7](#_Toc515536809)

[Request Message Layout 7](#_Toc515536810)

[ Header Object 7](#_Toc515536811)

[ TxnPayload Object 8](#_Toc515536812)

[ Primary Key Auto generation Vs providing in the request 8](#_Toc515536813)

[Response Message Layout 9](#_Toc515536814)

[ Header Object 9](#_Toc515536815)

[ TxnPayload Object 9](#_Toc515536816)

[ Error Response 9](#_Toc515536817)

[Sample Request and Response Message 10](#_Toc515536818)

[Out of the Box Transactions 11](#_Toc515536819)

[Base entity services 11](#_Toc515536820)

[Composite Services 11](#_Toc515536821)

[Application Configuration Management services 12](#_Toc515536822)

[1. createConfigAppPropertiesBase 12](#_Toc515536823)

[2. createConfigErrorcodeRegistryBase 12](#_Toc515536824)

[3. createConfigLanguageCodeBase 12](#_Toc515536825)

[4. createConfigTxnRegistryBase 13](#_Toc515536826)

[5. findAllConfigErrorcodeRegistryByLanguageCodeBase 13](#_Toc515536827)

[6. findAllConfigLanguageCodesBase 13](#_Toc515536828)

[7. findConfigAppPropertiesByBusinessKeyBase 14](#_Toc515536829)

[8. findConfigErrorcodeRegistryByBusinessKeyBase 14](#_Toc515536830)

[9. findConfigLanguageCodeByBusinessKeyBase 14](#_Toc515536831)

[10. findConfigTxnRegistryByBusinessKeyBase 15](#_Toc515536832)

[11. retrieveConfigAppPropertiesBase 15](#_Toc515536833)

[12. retrieveConfigErrorcodeRegistryBase 15](#_Toc515536834)

[13. retrieveConfigLanguageCodeBase 15](#_Toc515536835)

[14. retrieveConfigTxnRegistryBase 16](#_Toc515536836)

[15. updateConfigAppPropertiesBase 16](#_Toc515536837)

[16. updateConfigErrorcodeRegistryBase 16](#_Toc515536838)

[17. updateConfigLanguageCodeBase 16](#_Toc515536839)

[18. updateConfigTxnRegistryBase 17](#_Toc515536840)

[Authorization Framework services 18](#_Toc515536841)

[1. createAuthRolesRegistryBase 18](#_Toc515536842)

[2. updateAuthRolesRegistryBase 18](#_Toc515536843)

[3. retrieveAuthRolesRegistryBase 18](#_Toc515536844)

[4. findAuthRolesRegistryByBusinessKeyBase 19](#_Toc515536845)

[5. createAuthUserRegistryBase 19](#_Toc515536846)

[6. updateAuthUserRegistryBase 19](#_Toc515536847)

[7. retrieveAuthUserRegistryBase 19](#_Toc515536848)

[8. findAuthUserRegistryByBusinessKeyBase 20](#_Toc515536849)

[9. createAuthUserroleAccesscontrolBase 20](#_Toc515536850)

[10. updateAuthUserroleAccesscontrolBase 20](#_Toc515536851)

[11. retrieveAuthUserroleAccesscontrolBase 20](#_Toc515536852)

[12. findAuthUserroleAccesscontrolByBusinessKeyBase 21](#_Toc515536853)

[13. findAuthUserroleAccesscontrolByRegistryIdpkBase 21](#_Toc515536854)

[14. createAuthUserRoleAssocBase 21](#_Toc515536855)

[15. updateAuthUserRoleAssocBase 21](#_Toc515536856)

[16. retrieveAuthUserRoleAssocBase 22](#_Toc515536857)

[17. findAuthUserRoleAssocByBusinessKeyBase 22](#_Toc515536858)

[18. findAllRecordsByAuthUserRegistryIdpkBase 22](#_Toc515536859)

[19. findAllRecordsByAuthRolesRegistryIdpkBase 23](#_Toc515536860)

[Composite Services 24](#_Toc515536861)

[1. searchAuthRoles 24](#_Toc515536862)

[2. searchAuthUsers 25](#_Toc515536863)

[3. searchAuthAccessControl 25](#_Toc515536864)

[Reference data Table Management services 27](#_Toc515536865)

[Create Request - sample JSON message 27](#_Toc515536866)

[update request - sample JSON message 27](#_Toc515536867)

[Retrieve Request - sample JSON message 28](#_Toc515536868)

[findByBusinessKey request - sample JSON message. 29](#_Toc515536869)

[findAllRecordsByLanguageCode request - sample JSON message. 29](#_Toc515536870)

[List of Reference data Management services following common request structure 30](#_Toc515536871)

# 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** | **Sample Response** |
| {  "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"*  }    }    } | {  "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

### createConfigAppPropertiesBase

#### Description: create a record in CONFIG\_APP\_PROPERTIES table

#### Transaction Service Name: createConfigAppPropertiesBase

#### Request DO: ConfigAppPropertiesDO

#### Response DO: ConfigAppPropertiesDO

#### Mandatory Input: key, value

### 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

### createConfigLanguageCodeBase

#### Description:create a record in CONFIG\_LANGUAGE\_CODE table.

#### Transaction Service Name: createConfigLanguageCodeBase

#### Request DO: ConfigLanguageCodeDO

#### Response DO: ConfigLanguageCodeDO

#### Mandatory Input: key, value

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### createAuthRolesRegistryBase

#### Description: create a record in AUTH\_ROLES\_REGISTRY table

#### Transaction Service Name: createAuthRolesRegistryBase

#### Request DO: AuthRolesRegistryDO

#### Response DO: AuthRolesRegistryDO

#### Mandatory Input: roleName

### 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

### retrieveAuthRolesRegistryBase

#### Description: retrieve a record fromAUTH\_ROLES\_REGISTRY based on idPk

#### Transaction Service Name: retrieveAuthRolesRegistryBase

#### Request DO: AuthRolesRegistryDO

#### Response DO: AuthRolesRegistryDO

#### Mandatory Input: idPk

### 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

### createAuthUserRegistryBase

#### Description: create a record in AUTH\_USER\_REGISTRY table

#### Transaction Service Name: createAuthUserRegistryBase

#### Request DO: authUserRegistryDO

#### Response DO: authUserRegistryDO

#### Mandatory Input: userName

### 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

### 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

### findAuthUserRegistryByBusinessKeyBase

#### Description: find a record from AUTH\_USER\_REGISTRY based business key

#### Transaction Service Name: findAuthUserRegistryByBusinessKeyBase

#### Request DO: authUserRegistryDO

#### Response DO: authUserRegistryDO

#### Mandatory Input: userName

### 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

### 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

### 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

### 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

### 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

### 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

### 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*

### 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

### 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

### findAllRecordsByAuthUserRegistryIdpkBase

**Description:** find AUTH\_USER\_ROLE\_ASSOC based on authUserRegistryIdpk

**Transaction Service Name:** findAllRecordsByAuthUserRegistryIdpkBase

**Request DO:** authUserRoleAssocDO

**Response DO:** authUserRoleAssocDO

**Mandatory Input:** authUserRegistryIdpk

### 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

### 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.

### searchAuthUsers

**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\_USER\_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\_USER\_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\_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

### 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***