Oracle Banking APIs

UK Open Banking Taxonomy Configuration Guide

Release 18.3.0.0.0

Part No. F12055-01

December 2018

UK Open Banking Taxonomy Configuration Guide
December 2018
Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Table of Contents

1. Preface

1.1 Intended Audience

This document is intended for the following audience:

- Customers
- Partners

1.2 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

1.3 Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.4 Structure

This manual is organized into the following categories:

Preface gives information on the intended audience. It also describes the overall structure of the User Manual.

Introduction provides brief information on the overall functionality covered in the User Manual.

The subsequent chapters provide information on transactions covered in the User Manual.

Each transaction is explained in the following manner:

- Introduction to the transaction
- Screenshots of the transaction
- The images of screens used in this user manual are for illustrative purpose only, to provide improved understanding of the functionality; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.
- Procedure containing steps to complete the transaction- The mandatory and conditional fields of the transaction are explained in the procedure.

If a transaction contains multiple procedures, each procedure is explained. If some functionality is present in many transactions, this functionality is explained separately.

1.5 Related Information Sources

For more information on Oracle Banking APIs Release 18.3.0.0.0, refer to the following documents:

- Oracle Banking APIs Licensing Guide
- Oracle Banking APIs Installation Manuals

2. Introduction

Taxonomy validation is used to validate the input fields of the request object for each UK Open Banking service in OBAPI application. The validation of fields can be configured using database table properties.

Note: This has been added as part of **18.3.1.3.0** patch release, and is not available in **18.3.0.0.0** release.

3. Taxonomy Validation Table Structure

DIGX_OB_VALIDATION_CONFIG

This table is used to define taxonomy for request object fields of the open banking services. The entries should be available as day zero and any updates to this table requires server restart. Following table denotes the columns of this table and their description. These column values determine the validation definition of the

Column name	Type	Description
QUALIFIEDFIELD	VARCHAR2(1000) NOT NULL	Unique identifier of the field to be validated. This should be concatenation of service name and field name.
MINLENGTH	NUMBER	Minimum length required for the field
MAXLENGTH	NUMBER	Maximum length allowed for the field
MANDATORY	VARCHAR2(1)	Is the field value mandatorily required? (Y/N) Default-N
PATTERN	VARCHAR2(255)	Regex pattern required to be validated (if any). e.g. [a-zA-Z0-9]*
ERR_COD_PAT	VARCHAR2(255)	Error code to be thrown if the pattern validation fails
ERR_COD_LENG	VARCHAR2(255)	Error code to be thrown if length criteria fails
ERR_COD_MAND	VARCHAR2(255)	Error code to be thrown if mandatory criteria fails
VALIDATION_CLASS	VARCHAR2(255)	Fully qualified name of the class that needs to be invoked if any specific validation is required.

Note:
1) In QUALIFIEDFIELD column, use the fully qualified name of the service class along with method name. The fields in the request DTO of the method should be added directly to the method name. The method name and field name should be separated by '.'. If there are DTO objects in the request DTO, then the corresponding entries should be done by maintaining its hierarchical name.
Any fields in the parent DTO of the request DTO can be added in this column in similar way.

A sample value for this column can be:
com.ofss.digx.app.openbanking.service.consent.uk.DomesticPaymentConsent.create.initiation.creditorAccount.name.
2) MINLENGTH and MAXLENGTH will be used to check the length of the string input. In case of number input type, these columns will be used to specify the allowed range of the number input.
3) Proper error codes should be maintained for the fields based on their validation definition.
4) If any specific validation is required for a field, the validation logic can be defined in a custom validation class. This can be mapped using column 'VALIDATION_CLASS'. In this column the fully qualified name of the custom class must be given. Once this mapping is done, only mandatory check from this table will be performed.

Rules to create custom validator class:
a) The class should implement the interface 'ITaxonomyValidator'
b) Use getInstance method to return object of the class. Preferably use singleton pattern for validator
c) Override the validate method to provide the validation logic for respective input object. Use 'validationErrors' to add errors in case validation failures.
d) A reference custom validation class is provided in templates section

4. UK Open Banking Specific Validation Error Code Mapping

Please refer to document **Oracle Banking APIs UK Open Banking Configuration Guide (section 9)** to map UK Open Banking specified error codes for validation.

5. Templates

Following is a Sample template to create custom validator class.

In this example, OBAPI custom data type ‘CurrencyAmount’ is used as an input object.

```
package com.ofss.digx.app.openbanking.validation.taxonomy.uk;


import java.util.List;
import java.util.logging.Logger;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


import com.ofss.digx.datatype.CurrencyAmount;
import com.ofss.fc.infra.log.impl.MultiEntityLogger;
import com.ofss.fc.infra.validation.error.ValidationError;

public class CurrencyAmountValidator implements ITaxonomyValidator {


private CurrencyAmountValidator() {


}
private static class CurrencyAmountValidatorHolder {
private static final CurrencyAmountValidator INSTANCE = new CurrencyAmountValidator();
}
    public static CurrencyAmountValidator getInstance() {
    return CurrencyAmountValidatorHolder.INSTANCE;
}

@Override
public void validate(Object val, String serviceName, String fieldKey, List<ValidationError> validationErrors) {


CurrencyAmount currAmount = (CurrencyAmount) val;


    if (currAmount.getAmount() == null) {
        validationErrors.add(new ValidationError(serviceName, fieldKey + ".amount", null, "DIGX_OB_ERR_001_MAN", null));
        } else {

            Pattern pattern = Pattern.compile("^\\d{1,13}\\.\\d{1,5}$");

Matcher matcher = pattern.matcher(currAmount.getAmount().toPlainString());
            if (!matcher.matches()) {
                validationErrors
                        .add(new ValidationError(serviceName, fieldKey
+ ".amount", null, "DIGX_OB_ERR_001_PAT", null));

            }

        }
        if (currAmount.getCurrency() == null) {
            validationErrors
                    .add(new ValidationError(serviceName, fieldKey +
".currency", null, "DIGX_OB_ERR_001_MAN", null));
        } else {
            Pattern pattern = Pattern.compile("^[A-Z]{3,3}$");

            Matcher matcher = pattern.matcher(currAmount.getCurrency());
            if (!matcher.matches()) {
                validationErrors.add(
                        new ValidationError(serviceName, fieldKey +
".currency", null, "DIGX_OB_ERR_001_PAT", null));
            }

        }
    }

}
```

6. Blacklisted Input Set

Blacklisted input set is used to validate the String input fields of the request object for all UK Open Banking service's in OBAPI application. This validation will restrict entering any input as defined in blacklisted input set.

System will throw error in-case any blacklist input is found in request object.

Blacklist input set to be configured in table **DIGX_FW_CONFIG_ALL_B.** multiple values can be given in this property separated by comma (,).

Default **PROP_VALUE** property is set to '<html>,<script>,<' .

To modify existing blacklist set please refer below sample query.

update DIGX_FW_CONFIG_ALL_B set PROP_VALUE='<input to be blacklisted>' where PROP_ID='UK_BLACKLIST_CHAR ';