Oracle® Banking Electronic Data Exchange for Corporates

OBEDX-Trace Integration Guide Integration Guide

Patchset Release 14.7.4.0.0
G10292-01
June 2024

Oracle Banking Electronic Data Exchange for Corporates OBEDX-Trace Integration Guide Integration Guide, Patchset Release 14.7.4.0.0

G10292-01

Contents

Preface

Purpose

Purpose of this guide is to help you integrate Oracle Banking Electronic Data Exchange with Transformer (licensed product of Trace Financial Limited). Bank need to procure licenses for Transformer separately from Trace Financial Limited.

Audience

This guide is primarily intended for the following user/user roles:

Table 1 Audience

Role	Function
Implementation and IT Staff	Implementation and maintenance of the software

Acronyms and Abbreviations

The list of acronyms and abbreviations that are used in this guide are as follows:

Abbreviation	Description
OBEDX	Oracle Banking Electronic Data Exchange for Corporates

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.

Access to Oracle Support

Oracle customers that have purchased support 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.

Screenshot Disclaimer

Personal information used in the interface or documents are dummy and does not exist in the real world. It is only for reference purposes.

1

Integration Guide

1.1 Introduction

OBEDX supports processing of files uploaded in the canonical format, out of the box. Canonical formats supported out of the box are:

Transaction	Canonical format
Payments	pain001v6
Virtual Account Open	csv
Virtual Account Closure	csv

If banks want to support any other template or format, apart from the canonical formats mentioned above, such as MT, txt, csv, excel, etc., for payment processing or txt, XML, excel, or a csv file in a different template than what is supported for Virtual Account Management, then any third-party message transformation provider such as Transformer can be used to accomplish this.

This document briefs you about the specific steps and maintenances needed for Integration of these two products.

Following are the steps required integration with transformer:

1. Maintenance in transformer
2. Middleware service
3. Maintenance in Oracle Banking Routing Hub
4. Maintenance in Oracle Banking Electronic Data Exchange

1.2 Maintenance in Transformer

Transformer is a message formatting and translation toolkit. Using this toolkit, we can transform the files from corporate's preferred format to OBEDX format.

Transformer consist of two parts –

- Message & Mapping definition GUI toolkit used to define set of DictionAry Definitions (DAD)

- A run-time application programming interface (API) which uses the DAD to transform messages from one format to another.

Follow following steps to use transformer for message transformation

Prerequisites

1. Corporate preferred format/template – File format which a corporate upload
2. OBEDX format – This template is supplied by Oracle. The template can be found in the OBEDX Formats for Transformation.zip folder.

Steps to be followed

1. Create new project in Transformer
2. Create or import the message of corporate preferred format – The template which needs to be transformed.
3. Import the OBEDX format template as per below steps.
4. Tools > JSON schema import > Single file > Next > Schema file > select the schema file provided by Oracle > Next > Enter Group name > Next > Uncheck root schema check box (*but keep all the check boxes checked below root schema check box*) > Next > Move everything to selected window > Next > Finish
5. Do the mappings between preferred format and OBEDX format.
6. Add validations as per the requirement, refer table list of supported error codes
7. Create new exposed service and select the service builder as project jar builder
8. Select the java version 8
9. Add new exposed service operation select the operation type as OneToOneMapping
10. Select the appropriate message definition group and mapping definition
11. Test the mappings
12. Build the exposed service, this will generate the jar file
13. Note down the project key, service name & operation name generated on transformer's console.
14. Deploy this jar in middleware service.

List of supported status / error codes:

ERROR CODE	ERROR MSG
TRA-RLV-001	Record Level Validations Failed.
TRA-STX-001	Transaction syntax check failed
TRA-PAR-000	PARSING SUCESS
TRA-PAR-001	PARSING DONE with exceptions
TRA-RLV-002	Expected value for $1 is $2, actual value provided $3
TRA-STX-003	Invalid date $1
TRA-STX-002	Max length of the field with value $1 is breached. Expected max length is $2
TRA-STX-004	$1 is mandatory
TRA-STX-005	Invalid currency $1

> **Note:**
>
> Once should use above error codes while performing mapping in transformer's desktop application. $1 is placeholder and it should be replaced with appropriate value.

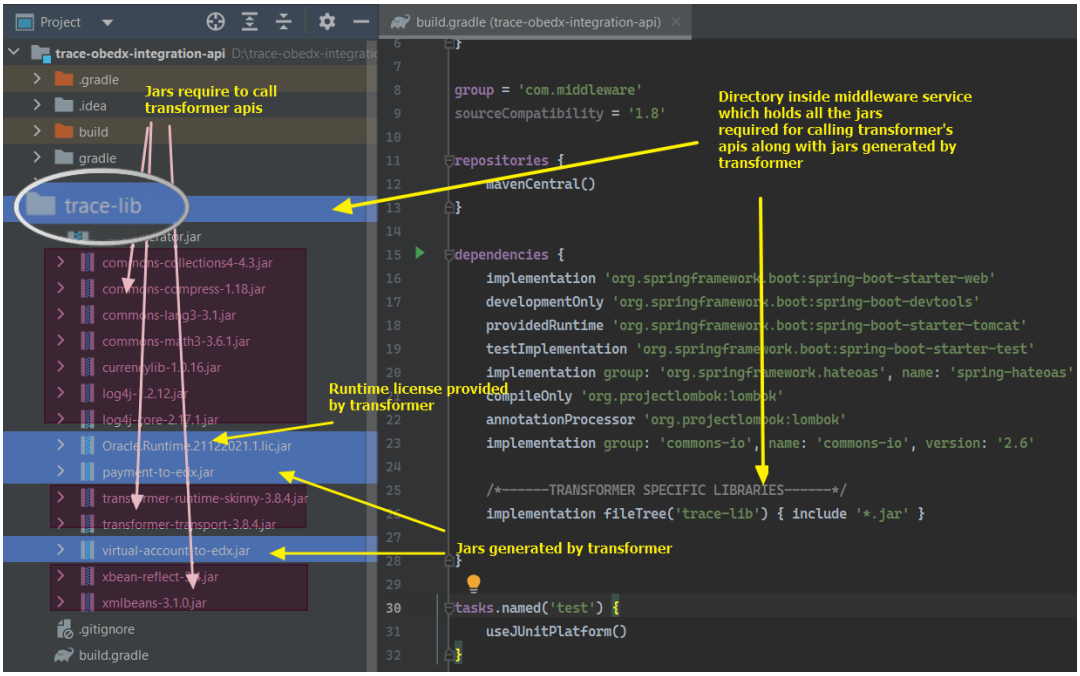
1.3 Middleware service

Middleware service acts as common access point for OBEDX to access Transformer's APIs. It is required for accessing the jars generated by transformer over the network.

Middleware service is wrapper around transformer's API. Since transformer may take some time for message transformation and enrichment, the middleware service must be designed in such a way that it should support asynchronous non-blocking communication.

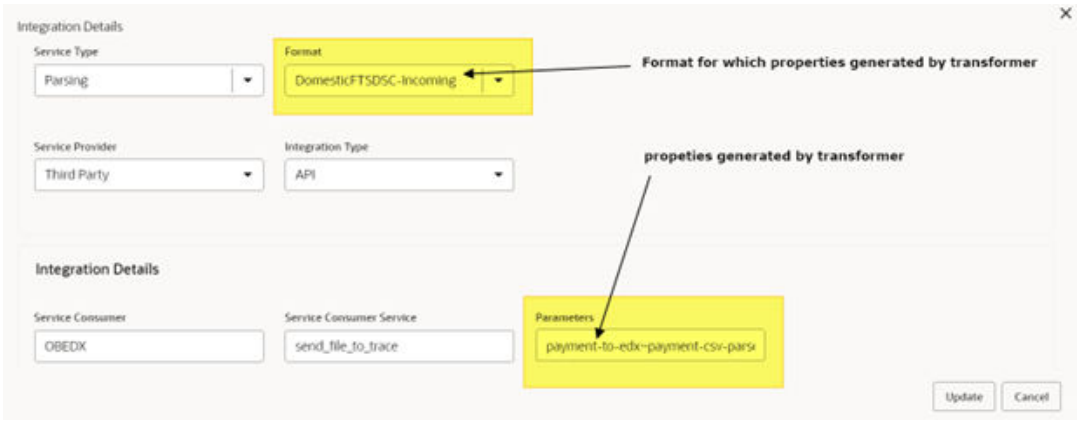
- Implementation team is required to build this service. Below are the expectations from the middleware service.
- Must be restful webservice
- Must support asynchronous and non-blocking API's
- Must have a rest endpoint to accept parsing requests from OBEDX with predefined request parameters
- Must be able to identify and decide which transformer api operation to call based on incoming format identfier
- Must be able to send acknolegement of parsing request
- Must be able to send the parsed response in OBEDX format along with all the validations / exceptions details if any to below end point.
- https://HOST_NAME:PORT_NUMBER/cmc-obrh-services/route/dispatch (OBRH_SERVICE_URL)
- Service must provide a directory to place all the jars required for transformer api's and jar generated by transformer.
- Refer below screenshot for more details

Figure 1-1 Middleware service



- After building the middleware service all the jars must be available in *WEB-INF/lib* folder for runtime access (refer line 26 from above screenshot).
- Transformer GUI Toolkit application generates three properties project key, service name & operation name on console, while building the exposed service or in the service info file present in jar generated by transformer at /META-INF directory
- These three properties must be configured in **Parameters** field present on integration preference screen UI as shown in below screenshot.
- One should follow below sequence while configuring these properties
- ***{project key}~{service name}~{operation name}*** (separator used is Tilde “~”)

Figure 1-2 Integration Details



- OBEDX will provide format identifier along with transaction name in below format

  format name-transaction name as a parsing request parameter named “format id” ;

- Along with properties configured in Parameters field on integration preferences UI as parsing request parameters named “parameters”.

Note that middleware service needs to be re-deployed every time new jar need to be added or modification done in existing mappings.

An example sudo code for middleware service written in java is provided below :

> **Note:**
>
> Code provided below is sample code for implementation partners to implement this service seamlessly. This code requires certain dependencies provided by transformer along with transformer’s runtime license jar.

Implementation team must retain the signature of submit() method, response headers, & response map keys as specified in the code below.

```
@RestControllerpublic class MiddlewareService{         @PostMapping("/
submit")    public ResponseDtoWrapper submit(@RequestBody MultipartFile file,
String fileRefId, String formatId,String
          parameters){             // perform technical validations such as
not null check... etc                // handleParsingRequest() must be
asynch (execute in separate
          thread).        handleParsingRequest(incomingFile, fileRefId,
originalFileName,
            formatId, parameters);                  //
sendAcknowledgement() will return the ResponseDtoWrapper without waiting for
            handleParsingRequest() to finish        return
sendAcknowledgement(originalFileName);
}         @Async    public void handleParsingRequest(MultipartFile file,
String fileRefId, String originalFileName,
            String formatId, String parameters) {                 String
responseFromTrace = callTransformerApi(formatId, file,
            parameters);
sendResponseToObrhService(responseFromTrace,
          fileRefId);    }         private String callTraceParserApi(String
formatId, File file, String parameters)
          {        Object output;        try {                   // Project
key, Service name, & Operation name will be generated by
          transformer         String [] parserParams =
          parameters.split(“~”);                   output =

ServicePerformerExecutor.performServiceWithSimpleExceptions(ClasspathProjectMa
nager.getInstance(),              parserParams[0],
parserParams[1],               parserParams[2],
Files.newInputStream(file.toPath()),              null,
OutputInstruction.asStringOutput());          String response = (String)
output;
return response;          }
catch (Exception exception) {             // log or handle exception here and
return String response           return
TECHNICAL_ERROR_IN_PARSING;         }     }          private void
sendResponseToObrhService(String responseFromTrace, String fileRefId)
```

```
{        restTemplate.postForObject(OBRH_SERVICE_URL,
          getObrhRequest(responseFromTrace,
                                              fileRefId),
                  ResponseDtoWrapper.class);} private HttpEntity<Map<String,
Map<String, String>>> getObrhRequest(String responseFromTrace,
            String fileRefId) {        Map<String, Map<String, String>>
responseMap =
            getResponseMap(responseFromTrace, fileRefId);
HttpHeaders httpHeaders = new HttpHeaders();        httpHeaders.add("appId",
"CMNCORE");        httpHeaders.add("userId", "EDXWORKFLOW");
httpHeaders.add("branchCode", "006");        httpHeaders.add("SERVICE-
CONSUMER", SERVICE-CONSUMER); // SERVICE-CONSUMER must be replaced with
name                                                                    of
            service consumer configured in OBRH
httpHeaders.add("entityId", "DEFAULTENTITY");        httpHeaders.add("SERVICE-
CONSUMER-SERVICE", SERVICE-CONSUMER-SERVICE); // SERVICE-CONSUMER-SERVICE
must                                                 be
            replaced with name of service consumer service configured in
OBRH
httpHeaders.setContentType(MediaType.APPLICATION_JSON);
return new HttpEntity<>(responseMap, headers);    }         public
Map<String, Map<String, String>> getResponseMap(String responseFromTrace,
String
            fileRefId) {        Map<String, String> paramMap = new
HashMap<>(2);        paramMap.put("input-file", fileRefId);
paramMap.put( "output-file", responseFromTrace);        Map<String,
Map<String, String>> responseMap =
              new HashMap<>(1);        responseMap.put("external-parser-
output", paramMap);                 // "input-file", "output-file" &
"external-parser-output" are keys of response
            map and are must NOT be changed in any case.
return responseMap;    }         private ResponseDtoWrapper
sendAcknowledgement(String originalFilename) {        ResponseDtoWrapper
responseDtoWrapper = new ResponseDtoWrapper();        ResponseDto responseDto
= new ResponseDto();        responseDto.addToResponseList(new
ResponseCode(ACCEPT_FOR_PARSE));
 responseDto.addToResponseList(new
ResponseCode(HttpStatus.ACCEPTED.toString()));
responseDto.setStatus("OK");
responseDto.setHttpStatusCode(HttpStatus.OK);        String requestId =
String.format(" %s-%s ", UUID.randomUUID(), originalFilename);
 responseDto.setRequestId(requestId);
responseDtoWrapper.setMessages(responseDto);        return
responseDtoWrapper;    }}
```

Middleware service's api must return ResponseDtoWrapper object, structure of which is as shown below.

```
import org.springframework.stereotype.Component;
@Componentpublic class ResponseDtoWrapper
{    private ResponseDto messages;    private ResponseResourceSupport
data;
 public ResponseDtoWrapper() {    }
public ResponseDto getMessages() {
return this.messages;    }     public void setMessages(ResponseDto messages)
```

```
{        this.messages = messages;    }     public ResponseResourceSupport
getData()
{        return this.data;    }     public void
setData(ResponseResourceSupport data)
{        this.data = data;    }}


import org.springframework.http.HttpStatus; import
java.io.Serializable;import java.math.BigDecimal;
import java.util.ArrayList;import java.util.List; public class ResponseDto
implements Serializable
 {    private static final long serialVersionUID = -8555557115768857726L;
 private String keyId;    private String status;    private
List<ResponseCode> codes = new ArrayList<>();
  private String requestId;    private HttpStatus httpStatusCode;    private
BigDecimal overrideAuthLevelsReqd;
public ResponseDto() {    }     public HttpStatus getHttpStatusCode()
{        return this.httpStatusCode;    }
   public void setHttpStatusCode(HttpStatus httpStatusCode)
{        this.httpStatusCode = httpStatusCode;    }     public String
getKeyId() {        return this.keyId;    }
   public void setKeyId(String keyId)
{        this.keyId = keyId;    }
  public void addToResponseList(ResponseCode respObj)
{        this.codes.add(respObj);    }
 public void removeFromResponseCodeList(ResponseCode respObj)
{        this.codes.remove(respObj);    }
 public void appendToResponseList(List<ResponseCode> respCodeListObj)
{        this.codes.addAll(respCodeListObj);    }
public String getStatus() {        return this.status;    }     public void
setStatus(String status)
{        this.status = status;    }
  public List<ResponseCode> getCodes()
{        return this.codes;    }     public void setCodes(List<ResponseCode>
codes) {
this.codes = codes;    }     public String getRequestId() {        return
this.requestId;    }
 public void setRequestId(String requestId) {        this.requestId =
requestId;    }
public BigDecimal getOverrideAuthLevelsReqd() {        return
this.overrideAuthLevelsReqd;    }
public void setOverrideAuthLevelsReqd(BigDecimal overrideAuthLevelsReqd)
{        this.overrideAuthLevelsReqd = overrideAuthLevelsReqd;}}


import java.io.Serializable;import java.math.BigDecimal;import
java.util.List;
public class ResponseCode implements Serializable {
 private String Code;    private String Desc;
private String Type;    private String Language;
 private List<Object> args;    private String arg;
  private boolean information;    private boolean override;
  private boolean error;    private BigDecimal overrideAuthLevelsReqd;
 public List<Object> getArgs() {        return this.args;    }
  public void setArgs(List<Object> args) {        this.args = args;    }
  public ResponseCode() {    }     public ResponseCode(String code)
```

```
{        this.Code = code;    }     public ResponseCode(String code, String
msg)
 {        this.Code = code;        this.arg = msg;    }     public String
getArg()
{        return this.arg;    }     public void setArg(String arg)
 {        this.arg = arg;    }     public ResponseCode(String code,
List<Object> msg)
 {        this.Code = code;        this.args = msg;    }     public String
getCode()
 {        return this.Code;    }     public void setCode(String code)
 {        this.Code = code;    }          public String getDesc()
{        return this.Desc;    }     public void setDesc(String desc)
{        this.Desc = desc;    }     public String getType()
{        return this.Type;    }     public void setType(String type)
{        this.Type = type;    }     public String getLanguage()
 {        return this.Language;    }     public void setLanguage(String
language)
 {        this.Language = language;    }     public boolean isError()
{        return null != this.Type &&
this.Type.equalsIgnoreCase("E");    }
public boolean isOverride() {        return null != this.Type &&
this.Type.equalsIgnoreCase("O");    }
  public boolean isInformation() {        return null != this.Type &&
this.Type.equalsIgnoreCase("I");    }
 public BigDecimal getOverrideAuthLevelsReqd() {        return
this.overrideAuthLevelsReqd;    }
public void setOverrideAuthLevelsReqd(BigDecimal overrideAuthLevelsReqd)
{        this.overrideAuthLevelsReqd = overrideAuthLevelsReqd;}}


import org.springframework.hateoas.RepresentationModel;
public class ResponseResourceSupport extends RepresentationModel
{    public ResponseResourceSupport() {    }}
```

1.4 Maintenance in OBRH

All the communication between middleware service and OBEDX will happen only via OBRH.

Steps to configure OBRH are as follows

1. Create if not exists, service consumer named “OBEDX”
2. Under the service consumer create service provider named “THIRD_PARTY”
   a. Provide the host and port of server where middleware service is deployed
   b. Edit the implementation and add headers and service path
3. Select the consumer service tab and add new consumer service
   a. Click on the service, by default transformation tab will be selected
   b. Click on add transformation, input required info and save
   c. Click on add route, input required info and save
4. Create if not exists, service consumer named “Transformer”
5. Under the service consumer create service provider named “OBEDX”
   a. Provide the host and port of server where obedx-workflow-service is deployed

b. Edit the implementation and add headers and service path

6. Select the consumer service tab and add new consumer service
   a. Click on the service, by default transformation tab will be selected
   b. Click on add transformation, input required info and save
   c. Click on add route, input required info and save

1.5 Maintenance in OBEDX

Maintenance in OBEDX requires two steps as follows

1. Create new format in OBEDX with a unique identifier
2. Add integration preference entry for new format along with properties generated by transformer to support parsing via Transformer as shown below.

Refer User manual for creating a format & integration preference.

2

Formats Supported (Out of the box)

2.1 Internal Fund Transfer Format 1

Format Supported - CSV

Field Descriptions

Field Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
1	mixedIdentifier	Y	This field can contain value A or B. Incase adhoc payment the value should be A	1	String
2	partyId	Y	Customer/Party id of customer who uploading the file. This value should be as per core banking data.	20	String
3	debitAccountId	Y	Source/Debit account number.	50	String
4	amount debit identifier	Y	Debit amount or Transfer amount Allowed Value - D/T	1	String
5	amount	Y	Transaction amount with format (###.##) for example 20.25, 20.00	-	BigDecimal
6	amountCurr	Y	3 digit transfer currency such as EUR, GBP	3	String

Field Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
7	valueDate	Y	Transfer date with format DD-MM-YYYY for example 26-03-2020	10	Date
8	creditAccountId	Y	Credit/ beneficiary account number	34	String
9	debitNarrative	Y	Narrative for debit account	35	String
10	creditNarrative	Y	Narrative for credit account	35	String
11	Deal ref number	N	Deal Reference Number	35	String
12	Email id	N	Email ID of the creditor	35	String
13	Charges account	N	Charges Account Details	35	String
14	Reference number	N	Reference Number	35	String

Field Mapping

Field Name	Canonical Field
mixedIdentifier	Not Required
partyId	initiatingPartyId
debitAccountId	drAccNo
amount debit identifier	Not Required
amount	amount
amountCurr	currency
valueDate	valueDate
creditAccountId	beneAccNo
debitNarrative	drAccTypePropietry
creditNarrative	remittanceInfo
Deal ref number	contractID
Email id	emailAddress
Charges account	chargesAccount
Reference number	instructionId

2.2 Internal Fund Transfer Format 2

Format Supported - CSV

Field Descriptions

Field Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
1	partyId	Y	Customer/Party id of customer who uploading the file. This value should be as per core banking data.	20	String
2	debitAccountId	Y	Source/Debit account number. Should be one account per file. It can either be VA or RA	50	String
3	amount debit identifier	Y	Debit amount or Transfer amount Allowed Value - D/T	1	String
4	totalamount	Y	Total transaction/ transfer amount with format (###.##) for example 100.50. Total amount should be equal to addition of credit/transfer amount from body section.	-	BigDecimal
5	amountCurrency	Y	3 digit transfer currency such as EUR, GBP	3	String
6	valueDate	Y	Transfer date with format DD-MM-YYYY for example 26-03-2020	10	Date
7	debitNarrative	Y	Narrative for debit account	35	String
8	Charges account	N	Charges Account Details	50	String
	Records Fields				
1	mixedIdentifier	Y	This field can contain value A or B. Incase adhoc payment the value should be A	1	String

Field Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
2	amount	Y	Transaction amount with format (###.##) for example 20.25, 20.00	-	BigDecimal
3	creditAccountId	Y	Credit/ beneficiary account number	34	String
4	creditNarrative	Y	Narrative for credit account	35	String
5	Deal ref number	N	Deal Reference Number	35	String
6	Email id	N	Email ID of creditor	35	String
7	Reference number	Y	Reference Number	35	String

Field Mapping

Field Sequence	Field Name	Canonical Field
1	partyId	initiatingPartyId
2	debitAccountId	drAccNo
3	amount debit identifier	Not Required
4	totalamount	totalAmount
5	amountCurr	currency
6	valueDate	valueDate
7	debitNarrative	drAccTypePropietry
8	Charges account	chargesAccount
	Records Fields	
1	mixedIdentifier	Not Required
2	amount	amount
3	creditAccountId	beneAccNo
4	creditNarrative	remittanceInfo
5	Deal ref no	contractID
6	Email id	emailAddress
7	Ref no	instructionId

2.3 Domestic Fund Transfer Format 1

Format Supported - CSV

Field Descriptions

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
1	mixedIdentifier	Y	This field can contain value A or B. Incase adhoc payment the value should be A	1	String
2	partyId	Y	Customer/Party id of customer who uploading the file. This value should be as per core banking data.	20	String
3	debitAccountId	Y	Source/Debit account number.	50	String
4	amount debit identifier	Y	Debit amount or Transfer amount Allowed Value - D/T	1	String
5	amount	Y	Transaction amount with format (###.##) for example 20.25, 20.00	-	BigDecimal
6	amountCurr	Y	3 digit transfer currency such as EUR, GBP	3	String
7	valueDate	Y	Transfer date with format DD-MM-YYYY for example 26-03-2020	10	Date
8	beneficiary name	Y	Name of Beneficiary/ Payee	35	String
9	creditAccountId	Y	Credit/ beneficiary account number	34	String
10	network	Y	Network type like ACH, NEFT, RTGS, SEPA for payment ACH=ACH NEFT=NEFT RTGS=RTGS SEPA=SEPA except this=SEPA	35	String
11	sortcode	Y	Beneficiary/ Payee bank/ clearing/ifsc code	11	String

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
12	creditNarrative1	N	Narrative for credit account	35	String
13	creditNarrative2	N	Narrative for credit account	35	String
14	creditNarrative3	N	Narrative for credit account	35	String
15	creditNarrative4	N	Narrative for credit account	35	String
16	Deal ref no	N	Deal Reference Number	35	String
17	Email id	N	Email ID of the creditor	35	String
18	Charges account	N	Charges Account Details	35	String
19	Ref no	N	Reference Number	35	String

Field Mapping

Sequence	Field Name	Canonical Field
1	mixedIdentifier	Not Required
2	partyId	initiatingPartyId
3	debitAccountId	drAccNo
4	amount debit identifier	Not Required
5	amount	amount
6	amountCurrency	currency
7	valueDate	valueDate
8	beneficiary name	beneName
9	creditAccountId	IBAN
10	network	extServiceLevelCd
11	sortcode	crBicFi
12	creditNarrative1	remittanceInfo
13	creditNarrative2	remittanceInfo
14	creditNarrative3	remittanceInfo
15	creditNarrative4	remittanceInfo
16	Deal reference number	contractID
17	Email id	emailAddress
18	Charges account	chargesAccount
19	Reference number	instructionId

2.4 Domestic Fund Transfer Format 2

Format Supported - CSV

Field Descriptions

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
	Header fields				
1	partyId	Y	Customer/Party id of customer who uploading the file. This value should be as per core banking data.	20	String
2	debitAccountId	Y	Source/Debit account number. Should be one account per file. It can either be VA or RA	50	String
3	amount debit identifier	Y	Debit amount or Transfer amount Allowed Value - D/T	1	String
4	totalamount	Y	Total transaction/ transfer amount with format (###.##) for example 100.50. Total amount should be equal to addition of credit/transfer amount from body section.	-	BigDecimal
5	amountCurr	Y	3 digit transfer currency such as EUR, GBP	3	String
6	network	Y	Network type like ACH, NEFT, RTGS for payment	35	String
7	valueDate	Y	Transfer date with format DD-MM-YYYY for example 26-03-2020	10	Date
8	Charges account	N	Charges Account Details	50	String
	Records Fields				
1	mixedIdentifier	Y	This field can contain value A or B. Incase adhoc payment the value should be A	1	String

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
2	amount	Y	Transaction amount with format (###.##) for example 20.25, 20.00	-	BigDecimal
3	beneficiary name	Y	Name of Beneficiary/ Payee	35	String
4	creditAccountId	Y	Credit/ beneficiary account number	34	String
5	sortcode	Y	Beneficiary/ Payee bank/ clearing/ifsc code	11	String
6	creditNarrative1	N	Narrative for credit account	35	String
7	creditNarrative2	N	Narrative for credit account	35	String
8	creditNarrative3	N	Narrative for credit account	35	String
9	creditNarrative4	N	Narrative for credit account	35	String
10	Deal reference no	N	Deal Reference Number	35	String
11	Email id	N	Email ID of Creditor	35	String
12	Reference Number	Y	Reference Number	35	String

Field Mapping

Sequence	Field Name	Canonical Field
	Header Fields	
1	partyId	initiatingPartyId
2	debitAccountId	drAccNo
3	amount debit identifier	
4	totalamount	
5	amountCurr	
6	network	extServiceLevelCd
7	valueDate	valueDate
8	Charges account	chargesAccount
	Records Fields	
1	mixedIdentifier	Not Required
2	amount	amount
3	beneficiary name	Creditor Name
4	creditAccountId	IBAN
5	sortcode	crBicFi
6	creditNarrative1	remittanceInfo

Sequence	Field Name	Canonical Field
7	creditNarrative2	remittanceInfo
8	creditNarrative3	remittanceInfo
9	creditNarrative4	remittanceInfo
10	Deal reference no	contractID
11	Email id	emailAddress
12	Reference Number	instructionId

2.5 International Fund Transfer Format 1

Format Supported - CSV

Field Descriptions

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
1	mixedIdentifier	Y	This field can contain value A.	1	String
2	partyId	Y	Customer/Party id of customer who uploading the file. This value should be as per core banking data.	20	String
3	debitAccountId	Y	Source/Debit account number. Should be one account per file. It can either be VA or RA	50	String
4	amount type	Y	Debit amt or Trfr amount	1	String
5	amount	Y	Transaction amount with format (###.##) for example 20.25, 20.00	-	BigDecimal
6	amountCurr	Y	3 digit transfer currency such as EUR, GBP	3	String
7	valueDate	Y	Transfer date with format DD-MM-YYYY for example 26-03-2020	10	Date
8	beneName	Y	Name of Beneficiary/ Payee	35	String
9	creditAccountId	Y	Credit/ beneficiary account number	34	String

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
10	beneAddrLine1	N	Beneficiary/ Payee address line 1		String
11	beneAddrLine2	N	Beneficiary/ Payee address line 2		String
12	beneCity	N	Beneficiary/ Payee city		String
13	beneCountry	N	Beneficiary/ Payee country		String
14	codeType	Y	Code/Payment Type of intermediary bank For swiftCode the value should be - SWI For National Clearing Code the value should be – NAC For Specific Bank details the value should be - SPE		String
15	bicCode	N	This will be swift code incase SWI This will be NCC code incase NCA This will be empty incase SPE		String
16	ultBankName	N	Beneficiary/ Payee Bank Name		String
17	ultAddrLine	N	Beneficiary/ Payee Bank address		String
18	ultCity	N	Beneficiary/ Payee Bank City		String
19	ultCountry	N	Beneficiary/ Payee Bank Country		String
20	paymentDetails 1	N	paymentDetails 1	35	String
21	paymentDetails 2	N	paymentDetails 2	35	String
22	paymentDetails 3	N	paymentDetails 3	35	String

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
23	paymentDetails 4	N	paymentDetails 4	35	String
24	charges	Y	Correspondence Charges expected value PAYEE, PAYER or SHARED mapping logic - SHARED--> SHAR PAYEE --> CRED PAYER --> DEBT		String
25	Deal ref no	N	Deal Reference Number	35	String
26	Email id	N	Email ID of the creditor	35	String
27	Charges account	N	Charges Account Details	35	String
28	OBDX ref no	N	OBDX Reference Number	35	String
	Structure for International Adhoc Payment via intermediary bank details				
1	mixedIdentifier	Y	Incase international adhoc payment the value should be AI	1	String
2	partyId	Y	Customer/Party id of customer who uploading the file. This value should be as per core banking data.	20	String
3	debitAccountId	Y	Source/Debit account number. Should be one account per file. It can either be VA or RA	50	String
4	amount type	Y	Debit amount or Transfer amount Allowed Value - D/T	1	String
5	amount	Y	Transaction amount with format (###.##) for example 20.25, 20.00	-	BigDecimal

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
6	amountCurr	Y	3 digit transfer currency such as EUR, GBP	3	String
7	valueDate	Y	Transfer date with format DD-MM-YYYY for example 26-03-2020	10	Date
8	beneficiary Name	Y	Name of Beneficiary/ Payee	35	String
9	creditAccountId	Y	Credit/ beneficiary account number	34	String
10	beneAddrLine1	N	Beneficiary/ Payee address line 1		String
11	beneAddrLine2	N	Beneficiary/ Payee address line 2		String
12	beneCity	N	Beneficiary/ Payee city		String
13	beneCountry	N	Beneficiary/ Payee country		String
14	intmdCodeType	N	Code/Payment Type of intermediary bank For swiftCode the value should be - SWI For National Clearing Code the value should be – NAC For Specific Bank details the value should be - SPE		String

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
15	intmdBicCode	N	This will be swift code incase SWI added for intermediary bank This will be NCC code incase NCA added for intermediary bank This will be empty incase SPE added for intermediary bank		String
16	intmdBankName	N	Intermediary bank name		String
17	intmdAddrLine	N	Intermediary bank address line		String
18	intmdCity	N	Intermediary bank city		String
19	intmdCountry	N	Intermediary bank country		String
20	codeType	Y	Code/Payment Type of intermediary bank For swiftCode the value should be - SWI For National Clearing Code the value should be – NAC For Specific Bank details the value should be - SPE		String
21	bicCode	N	This will be swift code incase SWI This will be NCC code incase NCA This will be empty incase SPE		String
22	ultBankName	N	Beneficiary/ Payee Bank Name		String

Sequence	Field Name	Mandatory	Remarks/Value	Max Length	Data Type
23	ultAddrLine	N	Beneficiary/ Payee Bank address		String
24	ultCity	N	Beneficiary/ Payee Bank City		String
25	ultCountry	N	Beneficiary/ Payee Bank Country		String
26	paymentDetails 1	N	paymentDetails 1	35	String
27	paymentDetails 2	N	paymentDetails 2	35	String
28	paymentDetails 3	N	paymentDetails 3	35	String
29	paymentDetails 4	N	paymentDetails 4	35	String
30	charge bearer	Y	Correspondence Charges expected value PAYEE, PAYER or SHARED mapping logic - SHARED--> SHAR PAYEE --> CRED PAYER --> DEBT		String
31	Deal ref no	N	Deal Reference Number	35	String
32	Email id	N	Email ID of the creditor	35	String
33	Charges account	N	Charges Account Details	35	String
34	Ref no	N	Reference Number	35	String

Field Mapping

Structure International Adhoc Payment

Sequence	Field Name	Canonical Field
1	mixedIdentifier	Not Required
2	partyId	initiatingPartyId
3	debitAccountId	drAccNo
4	amount type	NA
5	amount	amount
6	amountCurr	currency
7	valueDate	valueDate

Sequence	Field Name	Canonical Field
8	beneName	beneName
9	creditAccountId	beneAccNo
10	beneAddrLine1	crBuildingNo
11	beneAddrLine2	streetName
12	beneCity	townName
13	beneCountry	country
14	codeType	
15	bicCode	crBicFi
16	ultBankName	crAgentAccName
17	ultAddrLine	crAgentStreetName
18	ultCity	crAgentTownName
19	ultCountry	crAgentCountry
20	paymentDetails1	remittanceInfo
21	paymentDetails2	remittanceInfo
22	paymentDetails3	remittanceInfo
23	paymentDetails4	remittanceInfo
24	charges	chargeBearer
25	Deal ref no	contractID
26	Email id	emailAddress
27	Charges account	chargesAccount
28	OBDX ref no	instructionId
	Structure for International Adhoc Payment via intermediary bank details	
1	mixedIdentifier	Not Required
2	partyId	initiatingPartyId
3	debitAccountId	drAccNo
4	amount type	NA
5	amount	amount
6	amountCurr	currency
7	valueDate	valueDate
8	beneficiary Name	beneName
9	creditAccountId	beneAccNo
10	beneAddrLine1	crBuildingNo
11	beneAddrLine2	streetName
12	beneCity	townName
13	beneCountry	country
14	intmdCodeType	
15	intmdBicCode	intrmBicFi
16	intmdBankName	
17	intmdAddrLine	intrmAgentStreetName
18	intmdCity	intrmAgentTownName
19	intmdCountry	
20	codeType	
21	bicCode	crBicFi
22	ultBankName	crAgentAccName

Sequence	Field Name	Canonical Field
23	ultAddrLine	crAgentStreetName
24	ultCity	crAgentTownName
25	ultCountry	crAgentCountry
26	paymentDetails1	remittanceInfo
27	paymentDetails2	remittanceInfo
28	paymentDetails3	remittanceInfo
29	paymentDetails4	remittanceInfo
30	charge bearer	chargeBearer
31	Deal ref no	contractID
32	Email id	emailAddress
33	Charges account	chargesAccount
34	Reference number	instructionId

3
List of Topics

Chapters	Description
Chapter 1	Provides information on the intended audience. It also lists the various chapters covered in this manual.
Chapter 2	This chapter helps you to Integrate Oracle Banking Electronic Data Exchange for Corporates with Transformer.

Index