Domestic Payments Implementation Guide
Oracle Banking Digital Experience Cloud Service
Release 22.2.1.0.0

Part No. F76129-01

May 2023

ORACLE

Domestic Payments Implementation Guide

May 2023

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/

Table of Contents

1. Preface 1–1

1.1 Intended Audience 1–1

1.2 Documentation Accessibility 1–1

1.3 Access to Oracle Support 1–1

1.4 Structure 1–1

2. Introduction to this guide 2–2

3. Adding a local Network 3–1

3.1 Single One Time Payments 3–1

3.2 Standing Instructions 3–4

4. Bank Identification Code Lookup 4–1

5. Implementing the Local Payment Service 5–1

6. Business Policy for Local Payment Service 6–1

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.

The subsequent chapters describes following details:

- Introduction
- Preferences & Database
- Configuration / Installation.

2. Introduction to this guide

Domestic Payments, also referred to as Local Payments differ from region to region. The rules and regulations that govern local payments vary from country to country.

Out of the box the base product supports the following local payments:

1. SEPA Credit Transfer when the payment processor is Oracle Banking Payments.
2. NEFT, RTGS, IMPS networks (Indian Payment Networks) when the payment processor is Oracle FLEXCUBE Retail.

For other local payments the base product has a framework in place which will help an implementation team implement local payments for any country/region by using a combination of configurations and custom code.

The purpose of this document is to guide an implementation team through the steps required to implement a local payment.

Home

3. Adding a local Network

3.1 Single One Time Payments

1. Two task codes can potentially be defined per local payment network. One for domestic payments done via the Transfer Money transaction and one for domestic payments done via the Adhoc Payments transaction.
2. For example BACS, CHAPS are local payment networks in the UK. You may want to define either 1 or 2 task codes for each of these networks – if applicable to your implementation.

   Having 2 different task codes is not mandatory but it's possible. It serves the purpose when the requirement is to define one set of limits for payments done using a network via Transfer Money, and a different set of limits for payments done using the same network via Adhoc Payments.

   Note: For the purpose of illustration only, SEPA Credit Transfer configurations have been taken as a sample in screenshots in all the steps that will follow.
   The screenshots serve only as examples and this guide in no way suggests that you use the same content.

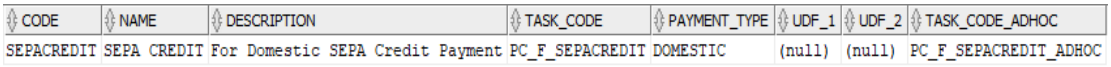
3. Insert entries for the task codes (1 or 2 depending on the requirement as stated in point #1) in the table DIGX_CM_TASK.

ID	NAME	PARENT_ID	EXECUTABLE	TASK_TYPE	MODULE_TYPE
PC_F_SEPACREDIT	Domestic Payment SEPA CREDIT	PC_F_NPDOM	Y	FINANCIAL_TRANSACTION	PC
PC_F_SEPACREDIT_ADHOC	Adhoc Domestic Payment SEPA CREDIT	PC_F_NPDOM	Y	FINANCIAL_TRANSACTION	PC

4. Insert corresponding entries in the table DIGX_CM_TASK_ASPECTS

TASK_ID	ASPECT	ENABLED
PC_F_SEPACREDIT	blackout	Y
PC_F_SEPACREDIT	limit	Y
PC_F_SEPACREDIT	working-window	Y
PC_F_SEPACREDIT_ADHOC	limit	Y

5. Insert an entry for the new task code defined for a network, in the table DIGX_PY_NETWORK



CODE	NAME	DESCRIPTION	TASK_CODE	PAYMENT_TYPE	UDF_1	UDF_2	TASK_CODE_ADHOC
SEPACREDIT	SEPA CREDIT	For Domestic SEPA Credit Payment	PC_F_SEPACREDIT	DOMESTIC	(null)	(null)	PC_F_SEPACREDIT_ADHOC

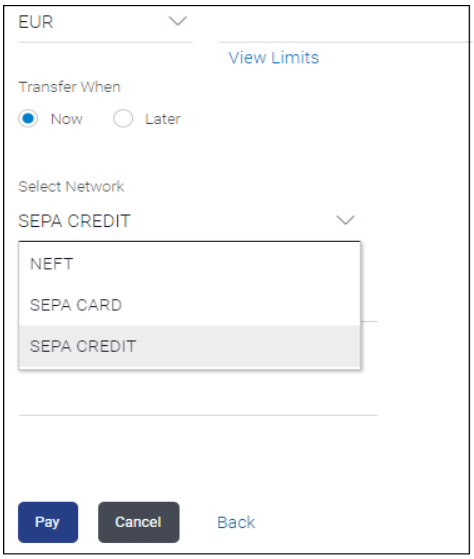
Column Name	Significance
CODE	Unique identifier for the network being implemented. Implementers can choose any value of their choice. We recommend keeping this value restricted to alphabets only. For UI Localization – displaying network name in local language, make changes in the NLS entries using the UI Toolkit, generate the components and then run the build. The key against which the local language entry needs to be made would be the CODE.
TASK_CODE	Task Code, for which entries have been made in DIGX_CM_TASK
PAYMENT_TYPE	Needs to be DOMESTIC always. Do not choose a different value.
TASK_CODE_ADHOC	If you have defined a different task code for Adhoc Payments for this network, then that task code needs to be mentioned here, else keep this value the same as TASK_CODE.
NAME	User friendly name given to identify the local payment network. No significance from the base framework perspective.
DESCRIPTION	Description of the local payment network. No significance from the base framework perspective.

6. Insert an entry for the new task code defined for a network, in the table DIGX_PY_NETWORK_HOST.

NETWORK_CODE	DETERMINANT_VALUE	HOST_NETWORK_CODE	ENABLED	VALUE_DATE_DEF	CURRENCIES
SEPACREDIT	OBDX_BU	STEP2	Y	com.ofss.digx.app.payment.service.genericDate.GenericPaymentDate	(null)

Column Name	Significance
NETWORK_CODE	Same value as the one defined in column CODE of DIGX_PY_NETWORK
DETERMINANT_VALUE	Entity Identifier. This is not a payment specific term and is used throughout the OBDX Application.
HOST_NETWORK_CODE	If the Payments Processor is Oracle Banking Payments: This value is utilized as one of the inputs to the Oracle Banking Payments Credit Value Date Service, when the Suggestive Credit Value Date for a payment is being fetched from the Payments Processor.
ENABLED	The network will be visible in the dropdown on the Payee and Payments screen only if the value is Y
VALUE_DATE_DEF	Use the value shown for SEPACREDIT, unless the requirement is to fetch the Suggestive Credit Value Date differently for each network, in which case you need to write your custom Service and register it here.
CURRENCIES	Comma Separated list of currencies to be shown on the Transfer Money screen on selection of a local payment network. If no currency is configured here then the default currency configured for the Region of the entity is used on the screen.

7. After following the above steps, the configured network will start showing in the Network Dropdown. The following is a snapshot of the relevant portion of the Transfer Money screen where you can expect the network to show up.



The network will also start showing up in Adhoc Payments screen, Add Payee screen as well as Multiple Transfer screen.

3.2 Standing Instructions

The base product screens will send fields that are common across different types of domestic SI's. For custom local networks, the implementer will have to pass the network specific fields to the REST API via the dictionary fields.

To configure local networks for Domestic Standing Instructions, please follow the steps below:

1. Create a Java class (Custom Domain Class) specific to the custom network. The class must extend the Domain class `com.ofss.digx.domain.payment.entity.instructions.PaymentInstruction`.

   This class should contain all the fields which are being sent from the UI via the Dictionary array of the payload (Network specific fields). Also create getters and setters for the fields.
2. The Custom Domain Class must have a field defined to store the custom network. This field should be private and getter – setter methods for the same must be present.
3. Create a custom ORM file for the Custom Domain Class. This ORM must contain all the fields of Domestic SI required for the custom network. (Fields defined in Custom Domain Class and inherited fields from `PaymentInstruction` class)
4. Create a custom Database Table where all the Network specific standing instruction data will be stored. ORM created in step 3 must map fields to this table.

5. Create a class (*Will be referred to as the Converter class hereafter in the document*) which will be responsible for mapping of Request DTO received from UI along with Dictionary array into Custom Domain Class.

   The Converter class must extend abstract class `com.ofss.digx.app.payment.assembler.service.instruction.payout.GenericDomesticPayoutInstructionConvertor` and implement the methods `toDomainObjectPaymentInstruction()` and `fromDomainObjectPaymentInstruction()`

6. Make an entry into Database table `DIGX_PY_INSTRUCTION_CONFIG`.

The following is a sample entry – for reference only:

	NETWORK	ASSEMBLER_NAME
1	SEPACREDIT	com.ofss.digx.app.payment.assembler.service.instruction.payout.SepaPayoutInstructionConvertor

Column Name	Significance
NETWORK	Same value as the one defined in column CODE of DIGX_PY_NETWORK for your network. This value must be same as NETWORK column of table DIGX_PY_DOMESTIC_PAYEE for payee to which SI is getting initiated.
ASSEMBLER_NAME	This column specifies fully qualified class name of the Converter class.

7. Create a Java class (*Will be referred to as the Local Repository Adapter hereafter in the document*) which must extend `com.ofss.digx.framework.domain.repository.adapter.AbstractLocalRepositoryAdapter<PaymentInstruction>` and implement `com.ofss.digx.domain.payment.entity.instructions.payout.repository.adapter.IGenericDomesticPayoutRepositoryAdapter`.

   You need to implement `create()`, `read()`, `update()` and `process()` methods.

8. The following code snippets can be used as a reference. Replace occurrences of `SepaPayoutInstruction` with your Custom Domain Class name.

```
@Override
public void create(PaymentInstruction object) throws Exception {
    SepaPayoutInstruction obj = (SepaPayoutInstruction) object;
    if (obj.getKey() != null) {
        obj.getKey().setDeterminantValue(
                DeterminantResolver.getInstance().fetchDeterminantValue(PaymentInstruction.class.getName()));
    }
    super.insert(obj);
}
```

```
@Override
public void update(PaymentInstruction object) throws Exception {
    SepaPayoutInstruction obj = (SepaPayoutInstruction) object;
    if (obj.getKey() != null) {
        obj.getKey().setDeterminantValue(
                DeterminantResolver.getInstance().fetchDeterminantValue(PaymentInstruction.class.getName()));
    }
    super.update(obj);
}
```

```
@Override
public PaymentInstruction read(InstructionKey key) throws Exception {
    SepaPayoutInstruction domainObj = new SepaPayoutInstruction();
    key.setDeterminantValue(
            DeterminantResolver.getInstance().fetchDeterminantValue(PaymentInstruction.class.getName()));
    domainObj = (SepaPayoutInstruction) super.load(SepaPayoutInstruction.class, key);
    return domainObj;
}
```

Just return null in the `process()` method.

9. Insert a record into the table `DIGX_FW_CONFIG_ALL_B` to register the Local Repository Adapter that we just created.

PROP_ID	`PAYOUT_INSTRUCTION_REPOSITORY_ADAPTER_{NETWORK}`
PROP_VALUE	Fully qualified class name of the local repository adapter
CATEGORY_ID	`repositoryadapterconfig`

The NETWORK string appended to `PAYOUT_INSTRUCTION_REPOSITORY_ADAPTER_` must be same as the NETWORK value entered in table `DIGX_PY_INSTRUCTION_CONFIG` in step 6.

Sample entry from `DIGX_FW_CONFIG_ALL_B` for SEPACREDIT network.

	PROP_ID	CATEGORY_ID	PROP_VALUE
1	PAYOUT_INSTRUCTION_REPOSITORY_ADAPTER_SEPACREDIT	repositoryadapterconfig	com.ofss.digx.domain.payment.entity.instructions.payout.repository.adapter.SepaNetworkInstructionAdapter

10. Create another java class (*Will be referred to as the Remote Repository Adapter hereafter in the document*) which must extend
    `com.ofss.digx.framework.domain.repository.adapter.AbstractRemoteRepositoryAdapter<PaymentInstruction>`
    and implement
    `com.ofss.digx.domain.payment.entity.instructions.payout.repository.adapter.IGenericDomesticPayoutRepositoryAdapter`.
11. You can just return null in the `create()`, `update()` and `read()` methods.
12. Override the `process()` method to call the Host/Backend Payment Processor for initiation of the SI.

    At this point in the code, an implementer has all the data that he needs in order to call the backend payment processor and initiate the SI.
13. Replace `SepaPayoutInstruction` with the Custom Domain Class in the sample snippet shown below, to achieve the same result for your custom network SI.

```
@Override
public PaymentInstruction process(PaymentInstruction paymentInstruction) throws Exception {
    SepaPayoutInstruction sepaInstructionPayout = (SepaPayoutInstruction) paymentInstruction;

    SessionContext sessionContext = (SessionContext) ThreadAttribute.get(ThreadAttribute.SESSION_CONTEXT);
    if (sessionContext.getServiceCallContextType() != null
            && sessionContext.getServiceCallContextType() == ServiceCallContextType.VALIDATE) {
        return sepaInstructionPayout;
    }
    INetworkPaymentInstructionAdapter adapter = null;
    try {
        adapter = ExtxfaceAdapterFactory.getInstance().getAdapter(INetworkPaymentInstructionAdapter.class,
                "processNetworkPaymentInstruction_" + sepaInstructionPayout.getNetwork(),
                DeterminantResolver.getInstance().getDeterminantTypeForObject(PaymentInstruction.class.getName()));
    } catch (NullPointerException | RunTimeException e1) {
        Exception e = new Exception();
        e.setErrorCode(PaymentErrorConstants.PY_NETWORK_ADAPTER_NOT_FOUND);
        e.setLocalizedMessage(
                ErrorManager.buildErrorMessage(PaymentErrorConstants.PY_NETWORK_ADAPTER_NOT_FOUND, null));
        throw e;
    }
    GenericDomesticInstructionReadResponse domainDto = null;
    SepaPayoutInstructionConvertor assembler = new SepaPayoutInstructionConvertor();
    domainDto = assembler.fromDomainObjectPaymentInstruction(sepaInstructionPayout);
    NetworkPaymentInstructionResponseDomainDTO response = adapter.processNetworkPaymentInstruction(domainDto);
    sepaInstructionPayout.getReference().setExternalReferenceId(response.getHostReference());
    return sepaInstructionPayout;

}
```

14. Insert a record into the table `DIGX_FW_CONFIG_ALL_B` to register the Remote Repository Adapter just created.

PROP_ID	`PAYOUT_INSTRUCTION_REPOSITORY_ADAPTER_REMOTE_{NETWORK}`
PROP_VALUE	Fully qualified class name of the remote repository adapter
CATEGORY_ID	`repositoryadapterconfig`

The NETWORK string appended to `PAYOUT_INSTRUCTION_REPOSITORY_ADAPTER_REMOTE_` must be same as the NETWORK value entered in table `DIGX_PY_INSTRUCTION_CONFIG` in step 6.

Sample entry from `DIGX_FW_CONFIG_ALL_B` for SEPACREDIT network.

PROP_ID	CATEGORY_ID	PROP_VALUE
PAYOUT_INSTRUCTION_REPOSITORY_ADAPTER_REMOTE_SEPACREDIT	repositoryadapterconfig	com.ofss.digx.domain.payment.entity.instructions.payout.repository.adapter.SepaNetworkInstructionRemoteAdapter

Note: On Clicking CONFIRM on the Review Screen, there is a REST call (Method = PATCH). All the fields which are sent in the Dictionary array in the Initiate to Review page transition – POST call must be sent again in PATCH call.
Otherwise these fields will not be available on approver screen.
Also, Currencies configured in Single one time payments (Section 3.1, Point number 6) will be applicable for SI.

Home

4. Bank Identification Code Lookup

Once the local network starts showing up on the Payments page, a lookup service would be required to lookup BIC Codes using the selected network. This would be needed when adding a payee.

The following query returns the host adapter configured in the Database for the lookup service

```
select * from DIGX_FW_CONFIG_ALL_O where PROP_ID like
'%com.ofss.digx.extxface.payment.adapter.payee.IFinancialInstitutionAda
pter.listGenericClearingCodes%'
```

The PROP_ID seen in the query above is a fully qualified class name of an interface. To override the default lookup service (which caters to only SEPA), a host adapter which implements the interface and the method seen above needs to be written and the fully qualified class name of the custom host adapter needs to be updated in the PROP_VALUE column against this PROP_ID.

A server restart would be required for the changes to be reflected.

Home

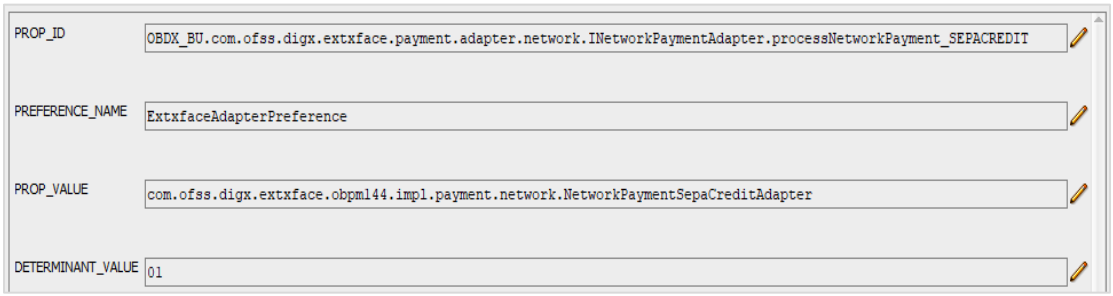
5. Implementing the Local Payment Service

Some amount of custom code writing will be required to achieve the last leg of implementing a local payment, which is posting an initiated payment to the payments processor.

This would entail writing a custom host adapter.

An entry needs to be inserted in DIGX_FW_CONFIG_ALL_O to register the custom host adapter.

The following image shows the out of the box entry in the same table for SEPA Credit Transfer:



The PROP_ID is of the format:

<Entity_ID>.<Fully Qualified Class Name of INetworkPaymentAdapter Interface>.<methodName>_<CODE>

Where

Entity_ID = Entity ID selected during installation. For example OBDX_BU.

methodName = processNetworkPayment (always)

CODE = Value of the column CODE in DIGX_PY_NETWORK

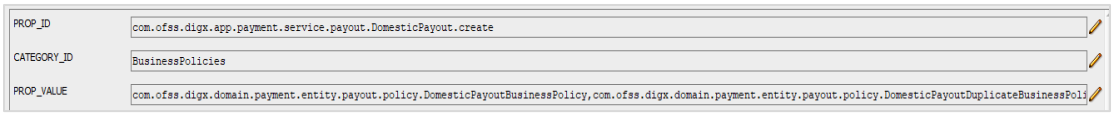
The PROP_VALUE column will contain the fully qualified class name of the host adapter, which must implement the interface and method mentioned in the PROP_ID.

Home

6. Business Policy for Local Payment Service

Copy the following entry from DIGX_FW_CONFIG_ALL_B and insert into DIGX_FW_CONFIG_ALL_O

This is for the purposes of overriding the base product business policies for the local payment service implemented.



In the PROP_VALUE column, add the custom business policy to the existing comma separated list.

Override `isPolicyToBeValidated()` method in the new business policy.

For example: The following implementation of `isPolicyToBeValidated()` will ensure that this business policy will get picked if a SEPA payment service is being invoked.

```
@Override
protected boolean isPolicyToBeValidated() {
    return paymentBusinessPolicyData.getNetwork().equals("SEPACREDIT");
}
```

Note: The above method needs to be implemented only for the Business Policy to get picked. For the actual business policy to be executed you will need to implement the `validatePolicy()` as usual.

Home