Contents

[1. Introduction 4](#_Toc463376682)

[1.1 Document Purpose 4](#_Toc463376683)

[1.2 Project Information 4](#_Toc463376684)

[1.2.1 Refined Estimate Details 5](#_Toc463376685)

[2. Technical Design Overview 7](#_Toc463376686)

[2.1 System Architecture 7](#_Toc463376687)

[2.1.1 On-line Systems Flow 7](#_Toc463376688)

[2.1.2 Batch System Flow 7](#_Toc463376689)

[2.1.3 Database Structure 7](#_Toc463376690)

[2.1.4 File Layout 7](#_Toc463376691)

[2.1.5 Configuration 7](#_Toc463376692)

[2.2 Related Systems and Interfaces 7](#_Toc463376693)

[2.3 Performance Concerns 7](#_Toc463376694)

[2.4 Security Considerations 7](#_Toc463376695)

[2.4.1 Compliance with FIS Security Standards 7](#_Toc463376696)

[2.4.2 Data Confidentiality requirements (encryption/masking/tokenization) 7](#_Toc463376697)

[2.4.3 Access Control 8](#_Toc463376698)

[3. Technical Design Details 8](#_Toc463376699)

[3.1.1 Component Summary 8](#_Toc463376700)

[3.1.2 Component Details 8](#_Toc463376701)

[3.1.3 Technical Education Prerequisites/Dependencies (Optional) 9](#_Toc463376702)

[3.1.4 Risk 9](#_Toc463376703)

[3.1.5 Technical Alternatives Considered 9](#_Toc463376704)

[4. Use Case/Test Case Scenarios: 9](#_Toc463376705)

[5. Assumptions 9](#_Toc463376706)

[6. Installation Prerequisites 9](#_Toc463376707)

[7. Production Installation Considerations 9](#_Toc463376708)

[8. Other Technical Requirements/Notes 9](#_Toc463376709)

[9. Appendices 10](#_Toc463376710)

[9.1 Distribution List 10](#_Toc463376711)

[9.2 Version History 10](#_Toc463376712)

[10. Estimate and Approval 10](#_Toc463376713)

1. Introduction

If you haven't already supplied a project name, go to the first page and enter a project name in the appropriate place.

* 1. Document Purpose

Describe the purpose of this document and specify the intended audience. Modify the following generic text as needed to include any project specific purposes or requirements.

The Detailed Design and Estimate is provided for customer review and approval to verify the identified solution represents customer’s specifications and requirements as documented.

This document is a working document that fluctuates as scope change requests are approved for the project. All changes to the scope and functional designs will require updates to the design and estimate if applicable.

* 1. Project Information

Visa has added a unique identifier to the header of each week’s routing table, and acquirers must provide the unique identifier in full financial ATM transactions and their reversals to confirm a current routing table is used.

* + 1. Component Details

Create a new dll EEDLL with eftposexception as namespace.

CXOSEEnn

New class EFTPosImportRecord in EEDLL derived from APIImportRecord (BC54.cpp) model like VCRImportRecord.

In parse() we can add the ResponseData tag and check for “ResponseData” in the handler endElement() to call the caseManifest->import()

string strBuffer("<ResponseData>");

strBuffer.append(strDATA\_DATA\_BUFFER);

strBuffer.append("</ResponseData>");

CXOSEEnn

New class EFTPosImportHandler in EEDLL derived from DNHandlerBase(VE08.cpp) model like VCRHandler.

Based upon the type of response we must set the class here, that should be present in DZ03.cpp to import and to write that transition in the DM5 file.

(Use the current state in the incoming queue response to set the class, for eg <currentState>disputeIntitiated</currentState> should be mapped to a new ChargebackFinancial class under EEDLL

**issuerReview1 should be mapped to PreArb** class under EEDLL

|  |  |  |
| --- | --- | --- |
|  |  |  |

(we can use REQ\_TYPE that we create in API\_QUEUE\_CONTROL to set the class for Accept,Reject and Extend API Calls)

CXOSDZ03

Add the new classes in the constructor and switch case statements.

EB00.cpp

intitalize() if API\_TYPE is EFTPOS, we can initialize the above new classes.

onReset(), onResume() should be changed to support EFTPOS.

BC53.cpp

Handle EFTPOS API\_TYPE

GE02.cpp

We can map network id 062 to EHB (EFTPos)

Map the below xml element in the constructor

caseReferenceNumber to EFTPOS\_CASE\_NO in CaseEFTPosSegment

institutionShortName to ACQ\_ID in CaseEFTPosSegment

processingType to PROCESSING\_TYPE in CaseEFTPosSegment

expiryDate to EXPIRY\_DATE in CasePhaseEFTPosSegment

currentstate to CURRENT\_STATE in CasePhaseEFTPosSegment

phaseExpiryDate to PHASE\_EXPIRY\_DATE in CasePhaseEFTPosSegment

CXOSESnn

New Class CaseEFTPosSegment in ESDLL

CXOSESnn

New Class CasePhaseEFTPosSegment in ESDLL

Following modules needs to changed to support the new segments in case create/Case Update and case search

EM00.cpp

EQ00.cpp

EX01.cpp

CXOXCREX,

CXOXRDM5 template,

Add the above xml elements to the DM5 template

Create new sections for the below dispute types.

* ChargebackExtend
* PreArbExtend

CXOSEX01

When the chargeback extend or pre arb extend is selected we should have a constraint in the rules “CHECK\_EXT”, in the server code for that constraint we need to check if there is any record in EMS\_TRANSITION table for that case\_id with STATUS\_NEXT as EXTD, if yes then we should send an error message saying “Dispute cannot be extended more than once”, the same should be applicable in the DM5 import. We have to report this error in the DM3 report.

DZ02.CPP

Handle the action EXP\_EHB and initialize EFTPosExportAction

CXOSEEnn

New EEDLL class EFTPosExportAction

EX70.cpp add all the transitions where we need to make API calls in here. Please go through the below test cases to come up with the list.

Eg. EHBREP1FWRD

For Accept and Reject api set the xml\_request to disputeEvidence, and req\_type should be Accept or Reject.

For Extend there is no xml\_request as we don’t need to send anything here.

CXOSEEnn

Like VCRExport, create EHBExport in EEDLL

In the insert() we have to prefix the req\_type with /ACQ\_ID/EFTPOS\_CASE\_NO/+(strREQ\_TYPE from EX70.cpp)

DZ06.cpp : Initialize the EFTPosExport class

Database changes

New Table

EMS\_CASE\_EHB

Case\_id

EFTPOS\_CASE\_NO

ACQ\_ID

PROCESSING\_TYPE

EMS\_PHASE\_EHB

CASE\_ID

TSTSAMP\_CREATED

EXPIRY\_DATE CHAR(8)

PHASE\_EXPIRY\_DATE CHAR(8)

CURRENT\_STATE VARCHAR(99)

CXOXEHB.txt

The template should be like

<?xml version="1.0" encoding="UTF-8" ?>

&disputeEvidence

<disputeEvidence>

  <comments>~O.COMMENTS.</comments>

  <documents>

    <content>~D.DOC\_PATH.[</content](file:///\\MKEDDH01\DNDocs\OUT\VNT\202109\ABCDEFG1234.PDF%3c/content)>

    <name>~D.DOC\_PATH.</name>

  </documents>

</disputeEvidence>

Below is xml sample that needs to be written out for Accept and Reject scenario

<?xml version="1.0" encoding="UTF-8" ?>

&disputeEvidence

<disputeEvidence>

  <comments>a comment on why this a disputed transaction</comments>

  <documents>

    <content[\\MKEDDH01\DNDocs\OUT\VNT\202109\ABCDEFG1234.PDF</content](file:///\\MKEDDH01\DNDocs\OUT\VNT\202109\ABCDEFG1234.PDF%3c\content)>

    <name>[\\MKEDDH01\DNDocs\OUT\VNT\202109\ABCDEFG1234.PDF</name](file:///\\MKEDDH01\DNDocs\OUT\VNT\202109\ABCDEFG1234.PDF%3c\name)>

  </documents>

</disputeEvidence>

Test Cases for PayPal

**Accept Chargeback flow**

1. Code to import the dm5 and make the below transition,

CHB1 SDRC to ACPF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to ACPS/ACPR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

CHB1 ACPR to ACPF = EXPAPI action, code to send the API call out

Code to process the response and transition the case to ACPS/ACPR

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

CHB1 SDRC to EXTD = EXPAPI action, code to send the API call out

Code to process the response and transition the case to EXTS/EXTR

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

CHB1 EXTR to EXTD = EXPAPI action, code to send the API call out

Code to process the response and transition the case to EXTS/EXTR

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

CHB1 EXTS to ACPF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to ACPS/ACPR,

Code to send this in the DM5 file.

**Decline Chargeback flow**

1. Code to import the dm5 and make the below transition,

CHB1 SDRC to REP1 FWRD = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to SDRC/REJR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

REP1 REJR to FWRD = EXPAPI action, code to send the API call out

Code to process the response and transition the case to SDRC/REJR

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

CHB1 EXTS to REP1 FWRD = EXPAPI action, code to send the API call out

Code to process the response and transition the case to SDRC/REJR

Code to send this in the DM5 file.

**Accept Representment Flow**

1. Code to process the queue response and make the below transition,

REP1 SDRC to ACRD

Code to send this in the DM5 file.

**Decline Representment Flow (Incoming Pre-arb)**

1. Code to process the queue response and make the below transition,

REP1 SDRC to PARB RECD

Code to send this in the DM5 file.

**Accept/Extend Pre-Arb Flow**

1. Code to import the dm5 and make the below transition,

PARB RECD to ACPF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to ACPS/ACPR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

PARB ACPR to ACPF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to ACPS/ACPR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

PARB RECD to EXTD = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to EXTS/EXTR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

PARB EXTR to EXTD = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to EXTS/EXTR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

PARB EXTS to ACPF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to ACPS/ACPR,

Code to send this in the DM5 file.

**Decline Pre-Arb Flow**

1. Code to import the dm5 and make the below transition,

PARB RECD to DNYF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to DNYS/DNYR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

PARB DNYR to DNYF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to DNYS/DNYR,

Code to send this in the DM5 file.

1. Code to import the dm5 and make the below transition,

PARB EXTS to DNYF = EXPAPI action, code to send the API call out,

Code to process the response and transition the case to DNYS/DNYR,

Code to send this in the DM5 file.

**Accept Pre-Arb Denial**

1. Code to process the queue response and make the below transition,

PARB DNYS to PARB DNAD

Code to send this in the DM5 file.

Supporting Documentation.

Business flow Diagrams from EFTPOS



Rules Flow Diagram



EftPos 3rd Party Spreadsheet



EFTPOS Schema

