Integration BluePrint

OLFM – SADAD integration

Author: Marwan Zakhia

Creation Date: April 14, 2015

Last Updated: April 14, 2015

Version: 0.1 - Draft

1. **Title, Subject, Last Updated Date, Reference Number**, **and** **Version** are marked by a Word Bookmark so that they can be easily reproduced in the header and footer of documents. When you change any of these values, be careful not to accidentally delete the bookmark. **You can make bookmarks visible by selecting Tools->Options…View and checking the Bookmarks option in the Show region.**

**Approvals:**

|  |  |
| --- | --- |
| Mohammad Abumalloh |  |
| Osama AlDameri |  |

1. To add additional approval lines, press [Tab] from the last cell in the table above.
2. You can delete any elements of this cover page that you do not need for your document.

# Document Control

## Change Record

9

| Date | Author | Version | Change Reference |
| --- | --- | --- | --- |
| 14-April-2015 | Marwan Zakhia | 0.1 | Initial or Draft Version |
|  |  |  |  |
|  |  |  |  |

## Reviewers

| Name | Position |
| --- | --- |
| Mohammad Abumalloh | SBM Project Manager |
| Osama Al Dameri | EMAD Project Manager |
|  |  |
|  |  |

Contents

Document Control 2

Change Record 2

Reviewers 2

Introduction 4

Purpose 4

Intended Audience 4

Document Summary 4

PROJECT OVERVIEW 5

Integration Overview 6

Overview 6

Integration Summary 7

Integration BluePrint 8

OLFM-BillUpload 8

OLFM Bill Load 1

SADAD-Bill Confirmation 1

OLFM-AccountUpload 1

SADAD-Account Confirmation 1

SADAD-Payment Notification 1

OLFM Payment Load 1

OLFM Payment UpLoad 1

Appendix 2

1. To update the table of contents, put the cursor anywhere in the table and press [F9]. To change the number of levels displayed, select the menu option Insert‑>Index and Tables, make sure the Table of Contents tab is active, and change the Number of Levels to a new value.

# Introduction

The purpose of this task is to assemble all the information that is required to describe the design of a software component into a complete Design Specification. This task is not a substitute for executing the individual design tasks. This specification work product can serve as a structure for completing the design for each component by providing pointers back into the Design Tasks:   
- DS.040 Develop Design Architecture Description  
- DS.080 Design Software Components  
- DS.090 Design Data  
- DS.100 Design Behavior  
- DS.130 Design User Interface

## Purpose

The purpose of this document is to describe, at a high level, the complete technical solution to be implemented using the Middleware Platform at ALMajdouie. This document will describe the interaction of the various parts of the solution and call out any technical considerations that need to be applied in the authoring of the Technical Specifications.

## Intended Audience

The intended audience for this document are:

• Middleware Integration Analysts to co-ordinate the completion of this document

• Middleware Solution Architect in production of integration solution architecture

• Testers as this will assist the writing of test cases

• Infrastructure Architects for the capacity planning, network and security non-functional requirement sections.

• Stakeholders or Authorised Personnel who need to sign off the detailed requirements contained in this document as it forms the basis for the scope which estimates will be based on

## Document Summary

This document covers the following areas:

• Project Overview

• Individual integration blueprint

• Service identification

• Service Descriptions

# PROJECT OVERVIEW



Figure : Integration Overview

The scope of this project is to expose as a reusable web services called *“proxy services”*, the below SADAD interfaces on the Oracle Service Bus:

1. Account Upload
2. Account Confirmation
3. Bill Upload
4. Bill Confirmation
5. Bill Load
6. Payment Notification
7. Payment Load
8. Payment Upload
9. Payment Confirmation

OSB will be responsible of the following actions:

1. Virtualization of the services
2. Reporting and monitoring
3. Routings
4. Messages transformation and validation
5. Messages enrichment
6. Error Handling
7. Apply security and adding policies.

For that, these exposed services will be reusable and interoperable and will be accessible to any third consumer party.

A service contract document for these services will be provided in a separate document, providing the integration details for third consumer parties.

The integration of the OLFM (Oracle Lease and Finance Management) with the OSB exposed services is within the scope of this project.

OLFM is an Oracle application, and then a SCA composite SOA layer will be exposed between the OSB and OLFM.

The SCA composite will be composed from the “*oracle application”* adapter and a BPEL component that consume the OSB proxy services. The JCA oracle application adapter will be used for integrating the OLFM application with the SOA platform.

# Integration Overview

## Overview

This blueprint documents the details of each integration required by this project. An integration summary is provided followed by a detailed section for integration.

An Integration Identifier is provided as a means to facilitate easy identification and recognition of each integration, both in documentation and general conversation.

The details for each integration will include identifying the services required for each integration, depicting the service interaction, and detailing any limitations and/or specific requirements of the interaction.

## Integration Summary

| **Integration ID** | **Interface Name** | **Functionality** | **Consumer** | **Provider** |
| --- | --- | --- | --- | --- |
| **OLFM-BillUpload** | BillUpload | OLFM Lease invoicing process creates multiple bills at night when SADAD systems are expected to have lower usage. OLFM then pushes the prepared Bill(s) via OSB to SADAD. | OLFM | SADA |
| **OLFM-BillLoad** | BillLoad | OLFM Lease invoicing process creates bill on completion of the Invoicing activity. OLFM then pushes the prepared Bill via OSB to SADAD. | OLFM | SADAD |
| **SADAD-BillConfirmation** | BillConfirmation | Bill confirmation message is sent to OLFM after the load process for a specific file has been completed within SADAD system. This confirmation message provides a Bill Load Summary report and a list of rejected records. | SADAD | OLFM |
| **OLFM-AccountUpload** | AccountUpload | OLFM pushes the prepared Account to SADAD. | OLFM | SADAD |
| **OLFM-AccountConfirmation** | AccountConfirmation | The Account confirmation message is sent to OLFM after the load process for a specific file has been completed within SADAD system. This confirmation message provides an Account Load Summary report and a list of rejected records. | SADAD | OLFM |
| **SADAD-Payment Notification** | PaymentNotification | A real-time alert from SADAD to the Biller when a new payment has been confirmed. | SADAD | OLFM |
| **OLFM-Payment Load** | PaymentLoad | Biller informs SADAD about payment done through NON-SADAD channel, by loading the payment record using this interface. | OLFM | SADAD |
| **OLFM-Payment Upload** | PaymentUpload | Biller informs SADAD about payment done through NON-SADAD channel, by loading a batch of payment record using this interface. | OLFM | SADAD |
| **OLFM-PaymentConfirmation** | PaymentConfirmation | The Payment confirmation message is sent to Almajdouie after the load process for a specific file has been completed. This confirmation message provides a Payment Upload summary report and a list of rejected records | SADAD | OLFM |

# Integration BluePrint

## OLFM-BillUpload

### Description

This interface provides the functionality to push the Bill(s) created at the OLFM level to SADAD.

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge (Even, it is an asynchronous transport )

### Integration Interaction



**cs.pay.abcs.Bills Oracle Application Adapter:** This exposed service is the oracle application adapter that will be used to connect to OLFM, and then to extract the following elements as xml elements:

1. Local\_Sender
2. Rq\_Timestamp
3. Due\_Amount
4. Due\_Date
5. Bill\_Cycle
6. Billing\_Account
7. Expiry\_Date

**cs.pay.abcs.OLFM (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to a specific BPEL component and then to a specific external OSB exposed service. For this operation, the mediator will route the message to the cs.pay.func.uploadBill BPEL component.

**cs.pay.func.uploadBill (BPEL):** it will be used to transform the XML message (ABM) generated by the Oracle Application adapter to the ebm message (the message of the exposed OSB webservice) and vice versa. This includes the transformation of the error message generated by the OSB proxy service to the OLFM error format message.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then error compensation etc…

**cs.pay.ebs.Bills (proxy service):** This is the external business service. Usually third parties consumers should call this webservice. If their request message doesn’t fit with the ebm of this proxy service, a layer should be added between third party consumer and OSB proxy service like it is done for OLFM, note that this additional layer may be another osb proxy service and not only a SCA soa.

This ebs is responsible of the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

### Error Handling

The OLFM application will need to handle a variety of errors, most of which will be return to OLFM which will map these to end-user-consumable error messages.

**Business Errors**

The following business exceptions have been documented based on the SADAD interfaces (with some rationalization)

* Invalid XML. Schema validation failed.
* Invalid or missing HTTP Header sender-id or rquid
* Missing data stream or valid content-type application/octet-stream
* Invalid message code. Expected BUPLQ
* Invalid message type. Expected BillUploadRq
* Internal error has occurred

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the EBS should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## OLFM Bill Load

### Description

This interface provides the functionality to push the Bill(s) created at the OLFM level to SADAD.

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**cs.pay.abcs.Bills Oracle Application Adapter:** This exposed service is the oracle application adapter that will be used to connect to OLFM, and then to extract the following elements as xml elements:

1. Local\_Sender
2. Rq\_Timestamp
3. Due\_Amount
4. Due\_Date
5. Bill\_Cycle
6. Billing\_Account
7. Expiry\_Date

Note: most probably, this will be the same adapter used for the UploadBill operation.

**cs.pay.abcs.OLFM (oracle mediator) :** This will be used as a message router (it is the same mediator used in the UploadBill operation). It will route all the messages based on a defined criteria to a specific BPEL component and then to a specific external OSB exposed service. For this operation, the mediator will route the message to the cs.pay.func. loadBill BPEL component.

**cs.pay.func. loadBill (BPEL):** it will be used to transform the XML message (ABM) generated by the Oracle Application adapter to the ebm message (the message of the exposed OSB webservice) and vice versa. This includes the transformation of the error message generated by the OSB proxy service to the OLFM error format message.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then error compensation etc…

**cs.pay.ebs.Bills (proxy service):** This is the external service. Usually third parties consumers should call this webservice. If their request message doesn’t fit with the ebm of this proxy service, a layer should be added between third party consumer and OSB proxy service like it is done for OLFM, note that this additional layer may be another osb proxy service and not only a SCA soa.

This ebs is responsible of the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

It is the same proxy used in the BillUpload operation.

### Error Handling

The OLFM application will need to handle a variety of errors, most of which will be return to OLFM which will map these to end-user-consumable error messages.

**Business Errors**

The following business exceptions have been documented based on the SADAD interfaces (with some rationalization)

* Due date is not in acceptable format
* Open date is not in acceptable format
* Close date is not in acceptable format
* Payment Status Effective Date for Payment Id [pmtId] is Invalid
* Payment Status Current Amount is Invalid for Payment Id
* Invalid Service Type for Account number{account number}
* Invalid Category Code for the Account number{account number}
* The Account is Deactivated
* Payment Upload Error(s)
* Active Bill already exists for the Bill Cycle
* Invalid Bill Status Code
* Bill Cycle does not exist
* Expiry date is not in acceptable format
* No active normal bill exists for this high toll bill
* Payment completion failed
* Internal Validation Error (e.g. duplicate statement, etc.)
* Duplicate Statement
* Deactivated Service Type for Account Number
* Deactivated Category for the Account Number
* Bill Identification Rule does not exist for the category
* Statement with same number already exists for the merchant
* New one-off bill already exists for the category
* Bill number does not exist
* Recurring category already exists for the category Number XXXXX
* Minimum partial payment amount is not a valid currency value
* Minimum advance payment amount is not a valid currency value
* Maximum advance payment amount is not a valid currency value
* Duplicate RqUID, file not processed
* A newer version of the corresponding bill already exists in the system
* One or more payments associated with the statement couldn’t be completed due to invalid state
* One or more payments cancelled after generating this statement
* One or more payments reversed after generating this statement
* Invalid XML. Schema validation failed.
* Invalid or missing HTTP Header sender-id or rquid
* Missing data stream or valid content-type application/octet-stream
* Invalid message code. Expected BUPLQ
* Invalid message type. Expected BillUploadRq
* Internal error has occurred

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the EBS should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## SADAD-Bill Confirmation

### Description

This interface provides the functionality to push the Bill(s) created at the OLFM level to SADAD.

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**chn.ext.SADAD.abcs.Confirmation\_1.0 (proxy service):** SADAD will call this exposed service. This proxy service will be build based on the schema(s) and wsdl provided from SADAD.

For that, no naming standard is used for this proxy. This proxy will transform the SADAD message to the ebs message.

**cs.pay.ebs.Bills\_1.0 proxy service:** This is the external that should be called by other third parties. It does the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

**cs.pay.abcs.Confirmation (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to the specific BPEL component. For this operation, the mediator will route the message to the cs.pay.func. loadBill BPEL component.

**cs.pay.func. confirmBill (BPEL):** it will be used to transform the XML message received from the OSB to the abm message of the oracle application adapter and vice versa.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then providing an error compensation etc…

**Oracle application adapter:** This will be used to connect to OLFM and then to push to it the Bill Load Summary.

The Following elements will be sent to OLFM:

* Rq\_Timestamp
* Msg\_Code
* Status\_Code
* RqUID
* AsyncRqUID
* Upload\_Date
* Processing\_Date
* Success\_Record\_Count
* Error\_Record\_Count
* Pending\_Record\_Count
* Error\_Code
* Error\_Msg
* Billing\_Account
* Bill\_Cycle

OLFM should reply with the following:

* Local\_Sender
* Rs\_Timestampe
* Msg\_Code
* Status\_Code
* RqUID (recommended)
* Async\_Rq\_UID
* Upload\_Date
* Processing\_Date
* Success\_Record\_Count
* Error\_Record\_Count

### Error Handling

Error that will be generated from OLFM during the processing of the request message received from SADAD:

* Invalid XML. Schema validation failed.
* Failed to get Sender ID, Reciever ID or Message Code from XML message.
* Failed to uncompress input data
* Invalid value for HTTP Header Content-Encoding.Expected 'gzip'
* Invalid AsyncRqUID
* Generic Error, see embedded Error Description
* SuccessRecordCount, ErrorRecordCount, and PendingRecordCount do not add up to the number of records in the corresponding BillUploadRq or BillDownloadRs
* The ErrorRecordCount value does not add up to the number of records in the BillConfirmationRq message
* The BillCategory value is either unrecognized or not the same as provided in corresponding upload file.
* The ServiceType value is either unrecognized or not the same as provided in corresponding upload file.

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the ebs should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## OLFM-AccountUpload

### Description

This interface provides the functionality to push the Account(s) created at the OLFM level to SADAD.

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge (Even, it is an asynchronous transport )

### Integration Interaction



**cs.pay.abcs.Accounts (Oracle Application Adapter):** This exposed service is the oracle application adapter that will be used to connect to OLFM, and then to extract the following elements as xml elements:

1. Local\_Sender
2. Rq\_Timestamp
3. Biller\_Id
4. Billing\_Acct
5. Service\_Type
6. Account\_Status\_Code
7. Beneficiar\_Id
8. Beneficiar\_Type

**cs.pay.abcs.OLFM (oracle mediator) :** This will be used as a message router . It will route all the messages based on a defined criteria to a specific BPEL component and then to a specific external OSB exposed service. For this operation, the mediator will route the message to the cs.pay.func.uploadAccounts BPEL component.

**cs.pay.func.uploadAccounts (BPEL):** it will be used to transform the XML message (ABM) generated by the Oracle Application adapter to the ebm message (the message of the exposed OSB webservice) and vice versa. This includes the transformation of the error message generated by the OSB proxy service to the OLFM error format message.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then error compensation etc…

**cs.pay.ebs.Accounts (proxy service):** This is the external business service. Usually third parties consumers should call this webservice. If their request message doesn’t fit with the ebm of this proxy service, a layer should be added between third party consumer and OSB proxy service like it is done for OLFM, note that this additional layer may be another osb proxy service and not only a SCA soa.

This ebs is responsible of the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

### Error Handling

The OLFM application will need to handle a variety of errors, most of which will be return to OLFM which will map these to end-user-consumable error messages.

**Business Errors**

* Failed to get Sender ID, Receiver ID or Message Code from XML message.
* Failed to read SADADProfile.cnf file.
* Failed to write the request message to Staging File System.
* Failed to uncompressed input data
* Invalid value for HTTP Header Content-Encoding.Expected 'gzip'
* Failed to generate SFS file name

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the ebs should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## SADAD-Account Confirmation

### Description

This interface provides the functionality to send the Account confirmation message to OLFM after the load process for a specific file has been completed within SADAD. This confirmation message provides an Account Load Summary report and a list of rejected records.

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**chn.ext.SADAD.abcs.Confirmation\_1.0 (proxy service):** SADAD will call this exposed service. This proxy service will be build based on the schema(s) and wsdl provided from SADAD.

For that, no naming standard is used for this proxy. It will transform the request send from SADAD to the ebs message.

**cs.pay.ebs.Accounts\_1.0 proxy service:** This is the external that should be called by other third parties. It does the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

**cs.pay.abcs.Confirmation (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to the specific BPEL component. For this operation, the mediator will route the message to the cs.pay.func. ConfirmAccount BPEL component.

**cs.pay.func. confirmAccount (BPEL):** it will be used to transform the XML message received from the OSB to the abm message of the oracle application adapter and vice versa.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then providing an error compensation etc…

**Oracle application adapter:** This will be used to connect to OLFM and then to push to it the Bill Load Summary.

The Following elements will be sent to OLFM:

* Rq\_Timestamp
* Msg\_Code
* Status\_Code
* RqUID
* AsyncRqUID
* Upload\_Date
* Processing\_Date
* Success\_Record\_Count
* Error\_Record\_Count
* Error\_Code
* Error\_Msg
* Billing\_Account
* Official\_Id

Official\_Id\_TypeOLFM should reply with the following:

* Local\_Sender
* Rs\_Timestampe
* Msg\_Code
* Status\_Code
* RqUID (recommended)
* Async\_Rq\_UID
* Upload\_Date
* Processing\_Date
* Success\_Record\_Count
* Error\_Record\_Count

### Error Handling

Error that will be generated from OLFM during the processing of the request message received from SADAD:

* Invalid XML. Schema validation failed.
* Failed to get Sender ID, Reciever ID or Message Code from XML message.
* Failed to uncompress input data
* Invalid value for HTTP Header Content-Encoding.Expected 'gzip'
* Invalid AsyncRqUID
* Generic Error, see embedded Error Description
* SuccessRecordCount and ErrorRecordCount do not add up to the number of records in the corresponding AccountUploadRq message
* The ErrorRecordCount value does not add up to the number of records in the AccountConfirmationRq message

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the ebs should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## SADAD-Payment Notification

### Description

This interface provides the functionality to alert to the Biller. The message is generated for new payment that has been confirmed, or reversed.

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**chn.ext.SADAD.abcs.Notification\_1.0 (proxy service):** SADAD will call this exposed service. This proxy service will be build based on the schema(s) and wsdl provided from SADAD.

For that, no naming standard is used for this proxy. It will transform the request send from SADAD to the ebs message.

**cs.pay.ebs.Payments\_1.0 proxy service:** This is the external that should be called by other third parties. It does the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

**cs.pay.abcs.Notification (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to the specific BPEL component. For this operation, the mediator will route the message to the cs.pay.func.notifyPayment BPEL component.

**cs.pay.func.notifyPayment (BPEL):** it will be used to transform the XML message received from the OSB to the abm message of the oracle application adapter and vice versa.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then providing an error compensation etc…

**Oracle application adapter:** This will be used to connect to OLFM and then to notify it with the payment notification.

The Following elements will be sent to OLFM:

* Rq\_Timestamp
* Msg\_Code
* Status\_Code
* RqUID
* Billing\_Account
* Payment\_Id
* OfficialId
* Error\_Code
* Error\_Msg

OLFM should reply with the following:

* Local\_Sender
* Rs\_Timestampe
* Msg\_Code
* Status\_Code
* RqUID (recommended)

### Error Handling

Error that will be generated from OLFM during the processing of the request message received from SADAD:

* Failed to uncompress input data
* Invalid value for HTTP Header Content-Encoding.Expected 'gzip'
* One or more payments were rejected

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the ebs should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## OLFM Payment Load

### Description

This interface provides the functionality to inform SADAD of any payment that is done through the biller channel (NON-SADAD channel).

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**cs.pay.abcs.Payments Oracle Application Adapter:** This exposed service is the oracle application adapter that will be used to connect to OLFM, and then to extract the following elements as xml elements:

* Local\_Sender
* Rq\_Timestampe
* Msg\_Code
* RqUID (recommended by OLFM)
* Due\_Amount
* Due\_Date
* Billing\_Account
* Payload\_Timestamp

**cs.pay.abcs.OLFM (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to a specific BPEL component and then to a specific external OSB exposed service. For this operation, the mediator will route the message to the cs.pay.func. loadPayment BPEL component.

**cs.pay.func. loadBill (BPEL):** it will be used to transform the XML message (ABM) generated by the Oracle Application adapter to the ebm message (the message of the exposed OSB webservice) and vice versa. This includes the transformation of the error message generated by the OSB proxy service to the OLFM error format message.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then error compensation etc…

**cs.pay.ebs.Payments (proxy service):** This is the external service. Usually third parties consumers should call this webservice. If their request message doesn’t fit with the ebm of this proxy service, a layer should be added between third party consumer and OSB proxy service like it is done for OLFM, note that this additional layer may be another osb proxy service and not only a SCA soa.

This ebs is responsible of the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

### Error Handling

The OLFM application will need to handle a variety of errors, most of which will be return to OLFM which will map these to end-user-consumable error messages.

**Business Errors**

The following business exceptions have been documented based on the SADAD interfaces (with some rationalization)

* Invalid Merchant Id
* Merchant is not registered with SADAD
* Invalid Customer Account Number
* The Account is not Active
* Payment Upload File date is in wrong format
* Payment Status Effective date cannot be a future date.
* Payment Status Effective date is in wrong format
* Payment Process date cannot be a future date
* Payment Process date is in wrong format
* The Payment Amount is invalid
* The due date of the Payment is invalid
* Duplicate payment data
* No associated Statement(s) exist for the Payment
* Invalid XML. Schema validation failed.
* Failed to get Sender ID, Reciever ID or Message Code from XML message.
* Failed to get Signature value.
* Failed to get Certificate details from partner profile.
* Failed to get message parameters from config file
* Failed to get partner profile from config file.
* Failed to read MessageDetails.cnf file
* Failed to read PartnerDetails.cnf
* Failed to validate the message
* Verification failed for the certificate used for signing the data.
* Signature verification failed.
* Failed to process Real-Time Payment.
* Failed to sign or persist response message..
* Failed to attach signature to the message
* Failed to sign the data

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the EBS should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## OLFM Payment UpLoad

### Description

This interface provides the functionality to inform SADAD of any payments that are done through the biller channel (NON-SADAD chanel).

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**cs.pay.abcs.Payments Oracle Application Adapter:** This exposed service is the oracle application adapter that will be used to connect to OLFM, and then to extract the following elements as xml elements:

* Local\_Sender
* Rq\_Timestampe
* Msg\_Code
* RqUID (recommended by OLFM)
* Due\_Amount
* Due\_Date
* Billing\_Account
* Payload\_Timestamp

**cs.pay.abcs.OLFM (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to a specific BPEL component and then to a specific external OSB exposed service. For this operation, the mediator will route the message to the cs.pay.func. uploadPayment BPEL component.

**cs.pay.func.uploadBill (BPEL):** it will be used to transform the XML message (ABM) generated by the Oracle Application adapter to the ebm message (the message of the exposed OSB webservice) and vice versa. This includes the transformation of the error message generated by the OSB proxy service to the OLFM error format message.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then error compensation etc…

**cs.pay.ebs.Payments (proxy service):** This is the external service. Usually third parties consumers should call this webservice. If their request message doesn’t fit with the ebm of this proxy service, a layer should be added between third party consumer and OSB proxy service like it is done for OLFM, note that this additional layer may be another osb proxy service and not only a SCA soa.

This ebs is responsible of the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

### Error Handling

The OLFM application will need to handle a variety of errors, most of which will be return to OLFM which will map these to end-user-consumable error messages.

**Business Errors**

The following business exceptions have been documented based on the SADAD interfaces (with some rationalization)

* Failed to get Sender ID, Reciever ID or Message Code from XML message.
* Failed to read SADADProfile.cnf file.
* Failed to write the request message to Staging File System.
* Failed to uncompress input data
* Invalid value for HTTP Header Content-Encoding.Expected 'gzip'
* Failed to generate SFS file name

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the EBS should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

## SADAD-Payment Confirmation

### Description

This interface provides the functionality to send the Payment confirmation message to Almajdouie after the load process for a specific file has been completed. This confirmation message provides a Payment Upload summary report and a list of rejected records

### Assumptions

The following assumptions apply to this section of this blueprint

* SOA will hold no state
* SOA will hold no money  
  This means SOA will not hold onto ‘money’ and therefore need to be accounted for. SADAD will always be the source of truth and the source of all money accounting and recording.
* SOA will persist a Message footprint into a dedicated DB for tracking and arbitration purpose.

### Patterns

This service will be implemented using the following patterns (documented in the *Solution Architecture Patterns* document):

* Synchronous Request Acknowledge

### Integration Interaction



**PaymentConfirmation (proxy service):** SADAD will call this exposed service. This proxy service will be build based on the schema(s) and wsdl provided from SADAD.

For that, no naming standard is used for this proxy. It will transform the request send from SADAD to the ebs message.

**cs.pay.ebs.Accounts\_1.0 proxy service:** This is the external that should be called by other third parties. It does the following actions:

1. Message transformation
2. Message enrichment.
3. Message validation
4. Reporting and tracing
5. Routing and orchestration (between the database where a footprint is inserted and the SADAD webservice)

**cs.pay.abcs.Confirmation (oracle mediator) :** This will be used as a message router. It will route all the messages based on a defined criteria to the specific BPEL component. For this operation, the mediator will route the message to the cs.pay.func.confirmPayment BPEL component.

**cs.pay.func. confirmPayment(BPEL):** it will be used to transform the XML message received from the OSB to the abm message of the oracle application adapter and vice versa.

It could be used to integrate any business rules, domain value mapping or XREF tables that are changeable at runtime. It could contain hydration and dehydration point and then providing an error compensation etc…

**Oracle application adapter:** This will be used to connect to OLFM and then to push the Payment record Summary.

The Following elements will be sent to OLFM:

* Rq\_Timestamp
* Msg\_Code
* Status\_Code
* RqUID
* AsyncRqUID
* Upload\_Date
* Processing\_Date
* Success\_Record\_Count
* Error\_Record\_Count
* Error\_Code
* Error\_Msg
* Billing\_Account
* Bill\_Cycle

OLFM should reply with the following:

* Local\_Sender
* Rs\_Timestampe
* Msg\_Code
* Status\_Code
* RqUID (recommended)
* Async\_Rq\_UID
* Upload\_Date
* Processing\_Date
* Success\_Record\_Count
* Error\_Record\_Count

### Error Handling

Error that will be generated from OLFM during the processing of the request message received from SADAD:

* Invalid schema
* Unable to parse corresponding request message
* Failed to get Sender ID, Reciever ID or Message Code from XML message.
* Failed to uncompress input data
* Invalid value for HTTP Header Content-Encoding.Expected 'gzip'
* Invalid AsyncRqUID
* Generic Error, see embedded Error Description

**System Errors**

These are errors that are a result of a technical issue. The following are potential Technical Errors

* Timeout due to network error
* Exception in code due to unexpected result
* SADAD unavailable.

Typical error handling outcomes for these types of errors a little more complicated. In most of these cases standard error handle would apply and the ebs should attempt to retry the request (given the assumption above the at-least once messaging applies here). Failing this, the error will be communicated to OLFM via the standard fault EBO.

1. Add open issues that you identify while writing or reviewing this document to the open issues section. As you resolve issues, move them to the closed issues section and keep the issue ID the same. Include an explanation of the resolution.  
     
   When this work product is complete, any open issues should be transferred to the project- or process-level Issue Log (Manage focus area) and managed using a project level Issue Form (Manage focus area). In addition, the open items should remain in the open issues section of this work product, but flagged in the resolution column as being transferred.

# Appendix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Doc. S.No | Component | Document Description | Reference | Comments |
|  |  |  |  |  |