



**S-DW-API**

**Service Design Document**

**For**

**FATFACE - Demandware Integration**

28-JUN-2017

WHISHWORKS

www.whishworks.com

WW/PD/REC/SDD/HYD01/FFLL1

Status: Restricted

Prepared By: Rupesh Sinha

*© WHISHWORKS 2017. All rights reserved.*

*No part of this document may be reproduced or transmitted in any form or by any means electronic or mechanical including photocopying and recording or by any information storage or retrieval system except as may be expressly permitted by WHISHWORKS*

**DISTRIBUTION LIST**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Contact Details** | **Purpose** |
| 1 | Mark Jones | mark.jones@fatface.com | Reviewer |
| 2 | Samin Sarkar | samin.sarkar@fatface.com | Reviewer |
| 3 | Gary Bolton | gary.bolton@fatface.com | Reviewer |
| 4 | John Ray | john.ray@fatface.com | Reviewer |
| 5 | Graeme Harkins | graeme.harkins@fatface.com | Reviewer |
| 6 | Shruthi Thalari | shruthi.thalari@whishworks.com | Informed |

**DOCUMENT CONTROL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared By** | **Reviewed By** | **Approved By** | **Approved Version** |
| [rupesh.sinha@whishworks.com](mailto:rupesh.sinha@whishworks.com) | [shruthi.thalari@whishworks.com](mailto:shruthi.thalari@whishworks.com) |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**VERSION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11-May-2017 | 0.1 | Initial Draft of the SDD | [rupesh.sinha@whishworks.com](mailto:rupesh.sinha@whishworks.com) |
| 26-Jun-2017 | 1.0 | Migrated to new SDD template and updated details to adopt the new Fatface deployment model based out on 4 vCores. | [rupesh.sinha@whishworks.com](mailto:rupesh.sinha@whishworks.com) |
| 19-Jul-2017 | 1.1 | Updated post payment API to cover details around invoking UK or US payment APIs based on order number | [rupesh.sinha@whishworks.com](mailto:rupesh.sinha@whishworks.com) |
|  |  |  |  |

**REFERENCES**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Document/Artefact** | **Version** | **Date** |
| 1 | [Interface Mapping Sheet](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Payment%20Request%20-%20CIMS-to-DW.xlsx?api=v2) | 0.2 | 5-Jun-2017 |
| 2 | [Astound ISD](https://drive.google.com/open?id=0B-gZPhTdfLuXMXRwZGJMZXVJOG8) | v.6 | 8-May-2017 |
| 3 | [Prologic Business Spec](https://fatface.atlassian.net/wiki/download/attachments/92938901/CID-2479%20-%20Extract%20of%20Web%20Order%20status%20for%20website%203rd%20party%20interface%20-%20s2.pdf?api=v2) | s2 | 5-Jun-2017 |
| 4 | [Payment Request System API Design Document](https://drive.google.com/open?id=0B5ptsQthYS0lTjBPbDdsR1FqYTg) | 0.2 | 5-Jun-2017 |
| 5 | [HTTP Status Codes](https://drive.google.com/open?id=0BwCs8xj46n0QR1ZhZHYxanZvdUU) | 0.1 | 13-Dec-2017 |
| 6 | [Fat Face High Level Design Document](https://fatface.atlassian.net/wiki/download/attachments/91790054/FatFace_DemandwareIntegr_FFLL1_HLD.docx?api=v2) | 1.5 | 20-Apr-2017 |

**DEFINITIONS & ACRONYMS**

|  |  |  |
| --- | --- | --- |
| **#** | **Acronym** | **Expanded Form** |
| 1 | SCV | Single Customer View |
| 2 | SOA | Service Oriented Architecture |
| 3 | SLA | Service Level Agreement |
| 4 | QOS | Quality of Service |
| 5 | SDD | Service Design Document |
| 6 | WSDL | Web Services Description Language |
| 7 | SOAP | Simple Object Access Protocol |
| 8 | HTTP | Hypertext Transfer Protocol |
| 9 | S-API | System API |
| 10 | PSV | Pipe separated value |
| 11 | RAML | Restful Modelling Language |
| 12 | X-API | Experience API |
| 13 | API | Application Programming Interface |
| 14 | P-API | Process API |
| 15 | IHUB | Integration Hub |
| 16 | REST | Representational State Transfer |
| 17 | DW | Demandware |

Table of Contents

[1. Introduction 8](#_Toc488270102)

[1.1 Purpose 8](#_Toc488270103)

[1.2 Scope 8](#_Toc488270104)

[1.3 Out of Scope 8](#_Toc488270105)

[1.4 Document Ownership 8](#_Toc488270106)

[2 API/Service Definition 9](#_Toc488270107)

[2.1 Risks 9](#_Toc488270110)

[2.2 Assumptions 9](#_Toc488270111)

[2.3 Issues 10](#_Toc488270112)

[2.4 Dependencies 10](#_Toc488270113)

[2.5 Constraints 10](#_Toc488270114)

[2.6 API/Service Design 10](#_Toc488270115)

[3 API/Service Method Definition 11](#_Toc488270116)

[3.1 PUT/PRICEBOOK 11](#_Toc488270119)

[3.1.1 Implementation 11](#_Toc488270120)

[3.1.2 Method Design 11](#_Toc488270125)

[3.1.3 Summary 12](#_Toc488270126)

[3.1.4 Request Schema 12](#_Toc488270127)

[3.1.5 Request Mapping 12](#_Toc488270128)

[3.1.6 Response Schema 12](#_Toc488270129)

[3.1.7 Response Mapping 12](#_Toc488270130)

[3.1.8 Error Handling 12](#_Toc488270131)

[3.1.9 Auditing 13](#_Toc488270132)

[3.1.10 Logging 13](#_Toc488270133)

[3.2 PUT/INVENTORY 14](#_Toc488270134)

[3.2.1 Implementation 14](#_Toc488270135)

[3.2.2 Method Design 14](#_Toc488270140)

[3.2.3 Summary 15](#_Toc488270141)

[3.2.4 Request Schema 15](#_Toc488270142)

[3.2.5 Request Mapping 15](#_Toc488270143)

[3.2.6 Response Schema 15](#_Toc488270144)

[3.2.7 Response Mapping 15](#_Toc488270145)

[3.2.8 Error Handling 15](#_Toc488270146)

[3.2.9 Auditing 15](#_Toc488270147)

[3.2.10 Logging 16](#_Toc488270148)

[3.3 PUT/ORDERS-STATUS 17](#_Toc488270149)

[3.3.1 Implementation 17](#_Toc488270150)

[3.3.2 Method Design 17](#_Toc488270155)

[3.3.3 Summary 18](#_Toc488270156)

[3.3.4 Request Schema 18](#_Toc488270157)

[3.3.5 Request Mapping 18](#_Toc488270158)

[3.3.6 Response Schema 18](#_Toc488270159)

[3.3.7 Response Mapping 18](#_Toc488270160)

[3.3.8 Error Handling 18](#_Toc488270161)

[3.3.9 Auditing 18](#_Toc488270162)

[3.3.10 Logging 19](#_Toc488270163)

[3.4 POST/PAYMENT-REQUEST 20](#_Toc488270164)

[3.4.1 Implementation 20](#_Toc488270165)

[3.4.2 Method Design 23](#_Toc488270170)

[3.4.3 Summary 23](#_Toc488270171)

[3.4.4 Request Schema 23](#_Toc488270172)

[3.4.5 Request Mapping 24](#_Toc488270173)

[3.4.6 Response Schema 24](#_Toc488270174)

[3.4.7 Response Mapping 25](#_Toc488270175)

[3.4.8 Error Handling 25](#_Toc488270176)

[3.4.9 Auditing 25](#_Toc488270177)

[3.4.10 Logging 25](#_Toc488270178)

[4 API/Service NFR 26](#_Toc488270179)

[4.1 Security 26](#_Toc488270183)

[4.2 Auditing 26](#_Toc488270184)

[4.3 Performance 26](#_Toc488270185)

[4.4 Exception and Error Handling and Alerting 26](#_Toc488270186)

[4.5 Application Monitoring 26](#_Toc488270187)

[4.6 Batch Processing and Scheduling 26](#_Toc488270188)

[4.7 Build and Deployment Solution 27](#_Toc488270189)

[4.7.1 Code Versioning 27](#_Toc488270190)

[4.7.2 Continuous Integration 27](#_Toc488270192)

[4.8 Tools and Technologies 27](#_Toc488270193)

[4.8.1 Development Tools 27](#_Toc488270195)

[4.8.2 Technologies 27](#_Toc488270196)

[4.8.3 Environment 27](#_Toc488270197)

[5 Testing 28](#_Toc488270198)

[5.1 Testing Approach 28](#_Toc488270202)

[5.1.1 Functional Testing 28](#_Toc488270203)

[5.1.2 Non-Functional testing 28](#_Toc488270204)

[5.1.3 Testing Tools 28](#_Toc488270205)

[5.1.4 Mocking frameworks 28](#_Toc488270206)

[6 Appendix 29](#_Toc488270207)

Table of Figures

[Figure 1 Put Pricebook API Design 11](#_Toc488270211)

[Figure 2 Put Inventory API Design 14](#_Toc488270212)

[Figure 3 Put Order Status API Design 17](#_Toc488270213)

[Figure 4 Post Payment Request API Method Design 20](#_Toc488270214)

[Figure 5 Payment Request Reporting Database Design 21](#_Toc488270215)

# Introduction

## Purpose

The purpose of this document is to provide the detailed design for the Demandware system APIs along with the interaction methods/operations, protocols and Quality of Service (QoS). It also provides request and response structure and the entity relationships/mappings. It also details the non-functional requirements design like debug logging, audit logging, error/exception handling, SLAs, Security.

## Scope

Following user stories are considered in scope for this service

* IHUB-23: Update regular list price: CIMS to Demandware
* IHUB-24: Publish future date markdown price book from CIMS to Demandware
* IHUB-12: Inventory updates from CIMS to Demandware
* IHUB-17: Order update sent from CIMS to Demandware
* IHUB-19: Take payment notification - CIMS to Demandware
* IHUB-20: Refund request notification: CIMS to Demandware
* IHUB-28: Payment update notification: Demandware – CIMS
* IHUB-29: Refund update notification: Demandware – CIMS
* IHUB-21: Order cancellation: - CIMS to Demandware
* IHUB-169: Order Refund /Cancellation request CIMS to Demandware directed to ECommerceDW
* IHUB-170: Order Payment update details CIMS to Demandware directed to ECommerceDW

## Out of Scope

Following are considered to be out of scope for this design document

* Any queuing or batching of target API calls is out of scope
* Any business alerting for summary of success and failure of capture, refund or cancellation transaction is out of scope
* Multiple attempts will be made to connect to Demandware SFTP server. If all attempts fail, Mule API will log an exception and raise an alert notification. Handling of failed requests will have to be addressed manually.

## Document Ownership

This document and the contract of the Service Operation it defines is owned by:

* The Delivery Team during delivery phases and
* The Support Team following the successful go live of the API

# API/Service Definition



|  |  |  |
| --- | --- | --- |
| API/Service Title | DEMANDWARE SYSTEM API | |
| API/Service Name | S-DW-API | |
| API/Service Version | V1 | |
| Brief Description | This system API provides several operations under different resources. Except payment request, all other integration with Demandware is via a file on the SFTP server. The system API uses JSON as the data model for request and response whereas the downstream system Demandware uses XML for all file based integration. Payment API uses JSON on both DW and Mule layer. All Mule system APIs are restful. | |
| Summary of Methods | PUT/pricebook – to update product price changes in DW  PUT/inventory – to update inventory in DW  PUT/order-status – to update order status in DW  POST/payment-request – to send payment request to DW | |
| Criticality | High | |
| Layer/Category | System | |
| Usage | Real Time | |
| Base URI | CloudHub:  https://<env>-s-dw-api-<version>.eu.cloudhub.io/api/ | |
| Transport Protocal(s) | HTTPS | |
| Service Level Access (SLA) | NA | |
| Quality of Service (QoS) | Avg. Response Time (Secs) | Refer [Fat Face NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) for detail |
| Avg. Payload Size (KB/MB) | Refer [Fat Face NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) for detail |
| Max Payload Size (KB/MB) | Refer [Fat Face NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) for detail |
| Avg. Volumes (per day) | Refer [Fat Face NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) for detail |
| Peak Volumes | Refer [Fat Face NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) for detail |
| Peak Volumes Timespan | Refer [Fat Face NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) for detail |
| Availability | High Availability |

## Risks

1. Delay in getting DW payment api web service stub may lead to a risk for SIT.

## Assumptions

1. Demandware will be able to import multiple files from the SFTP server.
2. Any alerting or notifications for failure in processing a file exported via Mule on Demandware SFTP server will be managed by Demandware
3. All UK orders will start with “UKF” and all US orders will start with “USF”. These patterns will be used to invoke Demandware UK or US payment APIs.

## Issues

NA

## Dependencies

1. Astound to share SFTP directory structure for all Demandware file inbound and outbound interfaces.

## Constraints

NA

## API/Service Design

* This API will expose several operations under different resources.
* Each of these resources and their operations provides an abstraction layer on top of the SFTP based or API based integration with Demandware.
* The data model used by this API will be JSON both for request and response.
* Demandware will use its own data model using XML for all file based inbound and outbound interfaces whereas the payment API will use JSON.

# API/Service Method Definition



## PUT/PRICEBOOK

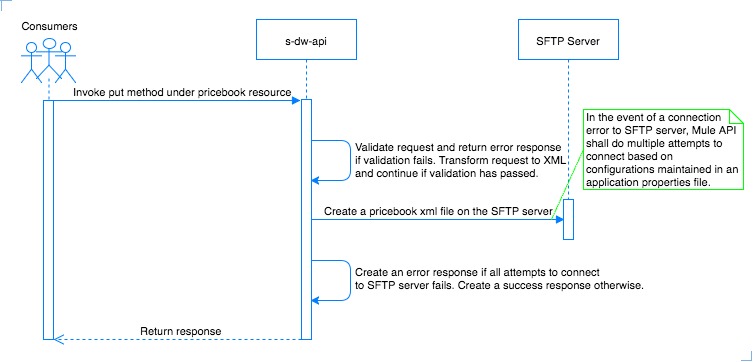


Figure 1 Put Pricebook API Design

### Implementation

* This API method accepts JSON request via HTTPS endpoint.
* This API method will first validate the request received and will return an error response back to the consumer if validation fails. Validation is mainly done to ensure all required inputs are present and request adheres to the schema.
* If validation fails, the API will return HTTP status code 400 with a message body describing that the error was caused due to bad request.
* If request is validated successfully, the API creates an XML file on the Demandware SFTP server.
* If target Demandware SFTP server is not reachable, the API shall do multiple attempts to connect to the SFTP server based on no. of attempts and reconnection delay configured in a property file.
* If all reconnection attempts fail, the system API shall return an error response back to the caller.
* If file is written successfully, system API shall return a success response to the consumer.
* The file creation shall be done in an async manner (one-way SFTP).



### Method Design

Following is the list of recommended Mule ESB components to be used for this method.

* HTTP Listener – to configure HTTP inbound endpoint for this API operation
* SFTP connector & SFTP outbound endpoint – to create a file on the SFTP server.
* DataWeave – to transform incoming XML request to outgoing JSON request and transform JSON response to XML response.
* Sub-flow/private flow – to modularize various steps involved in the method implementation. An example could be having a separate flow that handles request entry in the reporting database and calling this flow from Async scope using flow reference component.
* Flow Reference – to invoke a sub-flow or private flow
* VM outbound and inbound endpoint – to implement reliable accusation pattern for this API.
* Reference Exception Strategy – to invoke a global shared exception strategy defined in the application.

### Summary

This API method is used to create a pricebook XML file on Demandware SFTP server. This API will use the following API components for executing the project

* JSON schema – to describe response structure for this API.
* RAML – to define API contracts. This will use the JSON schema internally.
* APIkit Console – to simulate and test the API contract
* APIkit Router – to route API call to the correct method

### Request Schema

Refer the [Mapping sheet for IHUB-23 & IHUB-24](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Price%20Data%20Feed%20-%20CIMS-to-DW.xlsx?api=v2) for response schema design. The mapping sheet currently lists the data elements received from DW in XML format. This needs to be transformed to a JSON structure for the system API.

### Request Mapping

Refer the [Mapping sheet for IHUB-23 & IHUB-24](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Price%20Data%20Feed%20-%20CIMS-to-DW.xlsx?api=v2) for field mapping details.

### Response Schema

Following tables shows the response schema structure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Cardinality | Length | Valid Data | Description |
|  | message-id | String | 1..1 | 36 | UUID | Unique message Id for the transaction |
|  | status | String | 1..1 | 100 | Success/Failure | Success or failure based on API processing result |
|  | error-code | String | 0..1 | 10 | 400 | API internal error code if any |
|  | error-description | String | 0..1 | 100 | Bad request | API error description |

Sample Response:

HTTP status code with corresponding message body shall be returned in the response.

### Response Mapping

Refer the [Mapping sheet for IHUB-23 & IHUB-24](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Price%20Data%20Feed%20-%20CIMS-to-DW.xlsx?api=v2) for field mapping details.

### Error Handling

As described in section [3.1.1](#_Implementation_1).

### Auditing

Service will audit any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

### Logging

Service will log any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

## PUT/INVENTORY

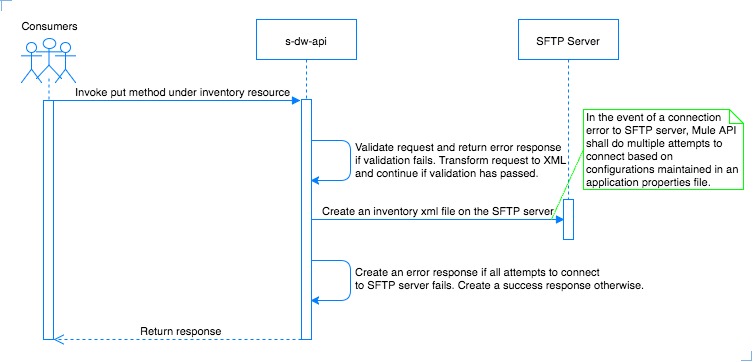


Figure 2 Put Inventory API Design

### Implementation

* This API method accepts JSON request via HTTPS endpoint.
* This API method will first validate the request received and will return an error response back to the consumer if validation fails. Validation is mainly done to ensure all required inputs are present and request adheres to the schema.
* If validation fails, the API will return HTTP status code 400 with a message body describing that the error was caused due to bad request.
* If request is validated successfully, the API creates an XML file on the Demandware SFTP server.
* If target Demandware SFTP server is not reachable, the API shall do multiple attempts to connect to the SFTP server based on no. of attempts and reconnection delay configured in a property file.
* If all reconnection attempts fail, the system API shall return an error response back to the caller.
* If file is written successfully, system API shall return a success response to the consumer.
* The file creation shall be done in an async manner (one-way SFTP).



### Method Design

Following is the list of recommended Mule ESB components to be used for this method.

* HTTP Listener – to configure HTTP inbound endpoint for this API operation
* SFTP connector & SFTP outbound endpoint – to create a file on the SFTP server.
* DataWeave – to transform incoming XML request to outgoing JSON request and transform JSON response to XML response.
* Sub-flow/private flow – to modularize various steps involved in the method implementation. An example could be having a separate flow that handles request entry in the reporting database and calling this flow from Async scope using flow reference component.
* Flow Reference – to invoke a sub-flow or private flow
* VM outbound and inbound endpoint – to implement reliable accusation pattern for this API.
* Reference Exception Strategy – to invoke a global shared exception strategy defined in the application.

### Summary

This API method is used to create an inventory XML file on Demandware SFTP server. This API will use the following API components for executing the project

* JSON schema – to describe response structure for this API.
* RAML – to define API contracts. This will use the JSON schema internally.
* APIkit Console – to simulate and test the API contract
* APIkit Router – to route API call to the correct method

### Request Schema

Refer the [Mapping sheet for IHUB-12](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Inventory%20Update%20CIMS-to-DW.xlsx?api=v2) for response schema design. The mapping sheet currently lists the data elements received from DW in XML format. This needs to be transformed to a JSON structure for the system API.

### Request Mapping

Refer the [Mapping sheet for IHUB-12](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Inventory%20Update%20CIMS-to-DW.xlsx?api=v2) for field mapping details.

### Response Schema

Following tables shows the response schema structure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Cardinality | Length | Valid Data | Description |
|  | message-id | String | 1..1 | 36 | UUID | Unique message Id for the transaction |
|  | status | String | 1..1 | 100 | Success/Failure | Success or failure based on API processing result |
|  | error-code | String | 0..1 | 10 | 400 | API internal error code if any |
|  | error-description | String | 0..1 | 100 | Bad request | API error description |

Sample Response:

HTTP status code with corresponding message body shall be returned in the response.

### Response Mapping

Refer the [Mapping sheet for IHUB-12](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Inventory%20Update%20CIMS-to-DW.xlsx?api=v2) for field mapping details.

### Error Handling

As described in section [3.2.1](#_Implementation_2).

### Auditing

Service will audit any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

### Logging

Service will log any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

## PUT/ORDERS-STATUS

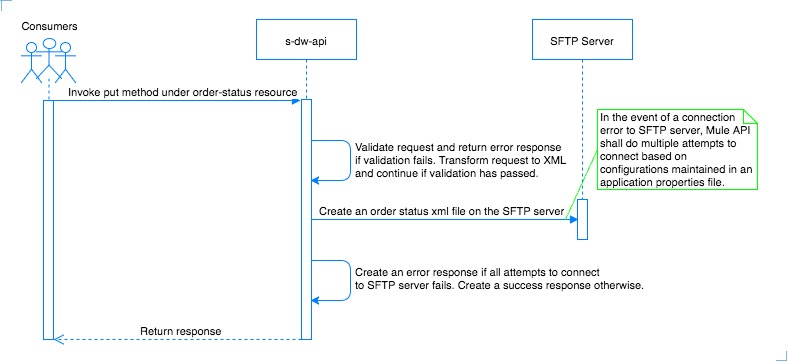


Figure 3 Put Order Status API Design

### Implementation

* This API method accepts JSON request via HTTPS endpoint.
* This API method will first validate the request received and will return an error response back to the consumer if validation fails. Validation is mainly done to ensure all required inputs are present and request adheres to the schema.
* If validation fails, the API will return HTTP status code 400 with a message body describing that the error was caused due to bad request.
* If request is validated successfully, the API creates an XML file on the Demandware SFTP server.
* If target Demandware SFTP server is not reachable, the API shall do multiple attempts to connect to the SFTP server based on no. of attempts and reconnection delay configured in a property file.
* If all reconnection attempts fail, the system API shall return an error response back to the caller.
* If file is written successfully, system API shall return a success response to the consumer.
* The file creation shall be done in an async manner (one-way SFTP).



### Method Design

Following is the list of recommended Mule ESB components to be used for this method.

* HTTP Listener – to configure HTTP inbound endpoint for this API operation
* SFTP connector & SFTP outbound endpoint – to create a file on the SFTP server.
* DataWeave – to transform incoming XML request to outgoing JSON request and transform JSON response to XML response.
* Sub-flow/private flow – to modularize various steps involved in the method implementation. An example could be having a separate flow that handles request entry in the reporting database and calling this flow from Async scope using flow reference component.
* Flow Reference – to invoke a sub-flow or private flow
* VM outbound and inbound endpoint – to implement reliable accusation pattern for this API.
* Reference Exception Strategy – to invoke a global shared exception strategy defined in the application.

### Summary

This API method is used to create an order status update XML file on Demandware SFTP server. This API will use the following API components for executing the project

* JSON schema – to describe response structure for this API.
* RAML – to define API contracts. This will use the JSON schema internally.
* APIkit Console – to simulate and test the API contract
* APIkit Router – to route API call to the correct method

### Request Schema

Refer the [Mapping sheet for IHUB-17](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Order%20Status%20Update%20CIMS-to-DW.xlsx?api=v2) for response schema design. The mapping sheet currently lists the data elements received from DW in XML format. This needs to be transformed to a JSON structure for the system API.

### Request Mapping

Refer the [Mapping sheet for IHUB-17](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Order%20Status%20Update%20CIMS-to-DW.xlsx?api=v2) for field mapping details.

### Response Schema

Following tables shows the response schema structure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Cardinality | Length | Valid Data | Description |
|  | message-id | String | 1..1 | 36 | UUID | Unique message Id for the transaction |
|  | status | String | 1..1 | 100 | Success/Failure | Success or failure based on API processing result |
|  | error-code | String | 0..1 | 10 | 400 | API internal error code if any |
|  | error-description | String | 0..1 | 100 | Bad request | API error description |

Sample Response:

HTTP status code with corresponding message body shall be returned in the response.

### Response Mapping

Refer the [Mapping sheet for IHUB-17](https://fatface.atlassian.net/wiki/download/attachments/93227839/WW-PD-REC-IMS-Interface%20Mapping%20-%20Order%20Status%20Update%20CIMS-to-DW.xlsx?api=v2) for field mapping details.

### Error Handling

As described in section [3.3.1](#_Implementation_4).

### Auditing

Service will audit any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

### Logging

Service will log any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

## POST/PAYMENT-REQUEST

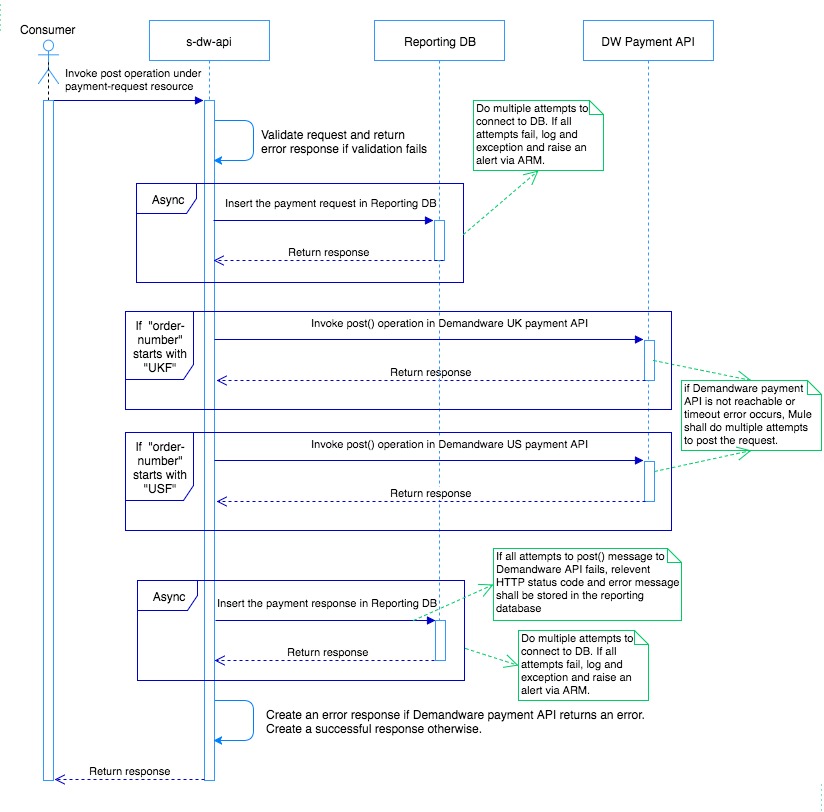


Figure 4 Post Payment Request API Method Design

### Implementation

* Make a request entry in audit log [asynchronous call].
* Make a request entry in the reporting database [asynchronous call].
* Check if the mandatory/required fields null or empty, if null or empty then exit/terminate and return an appropriate error code ad error message. HTTP status code 400 shall be returned as part of the response.
* Based on the data in request, prepare a JSON request to invoke post operation on the target Demandware API.
* If order-number in request starts with “UKF”, invoke post operation on Demandware UK payment API.
* If order-number in request starts with “USF”, invoke post operation on Demandware US payment API.
* If target API is not reachable or not responding, do multiple retry at a defined interval. Both reconnection attempts and retry interval shall be configured in Mule application property file.
* Make a response entry in audit log [asynchronous call].
* Make a response entry in the reporting database [asynchronous call]. This entry shall be done no matter response is success or error.
* Transform the target Demandware API response to system API response and return the response to the consumer.

**Reporting Database Design:**

Following diagram shows the DB structure to be used to store data into the reporting database:

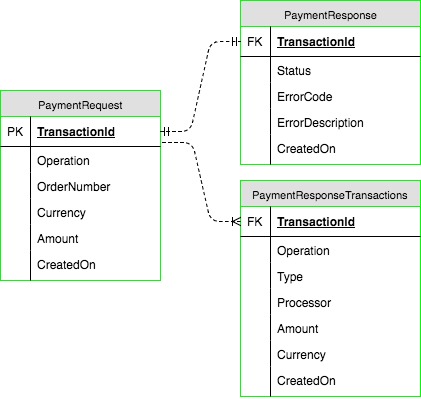
****

Figure 5 Payment Request Reporting Database Design

Table definitions:

PaymentRequest

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Length | Valid Data | Description | Constraints |
|  | TransactionId | string | 36 | "2d159786-245b-11e7-93ae-92361f002671" | Unique identifier of transaction to avoid duplicated requests | Primary Key |
|  | Operation | string | 10 | "capture" "refund" "cancel" | The API operation |  |
|  | OrderNumber | string | 30 | "123456789" | Web order number |  |
|  | Currency | string | 3 | "GBP" | Order currency |  |
|  | Amount | decimal | 13,4 | "101.50" | Amount to be captured or refunded |  |
|  | CreatedOn | datetime | NA | Default date time format | Date time when record was inserted into the DB | This should be by default inserted in the database. |

PaymentResponse

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Length | Valid Data | Description | Constraints |
|  | TransactionId | string | 36 | "2d159786-245b-11e7-93ae-92361f002671" | Unique identifier of transaction | Foreign Key |
|  | Status | string | 10 | "success" "error" | Status of the performed operation |  |
|  | ErrorCode | string | 4 | "01”  “400” | Error code |  |
|  | ErrorDescription | string | 256 | "Payment Gateway Service Unavailable"  “Bad request” | Error message |  |
|  | CreatedOn | datetime | NA | Default date time format | Date time when record was inserted into the DB | This should be by default inserted in the database. |

PaymentResponseTransactions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Length | Valid Data | Description | Constraints |
|  | TransactionId | string | 36 | "2d159786-245b-11e7-93ae-92361f002671" | Unique identifier of transaction | Foreign Key |
|  | Operation | string | 10 | "capture" "refund" "cancel" | Type of transaction |  |
|  | Type | string | 10 | "CASH" "GIFT" | Operation object |  |
|  | Processor | string | 256 | "ADYEN\_CREDIT" "PAYPAL\_EXPRESS" "GIFTCARD" | Payment processor for operation |  |
|  | Amount | decimal | 13,4 | "101.50" | Amount of operation |  |
|  | Currency | string | 3 | "GBP" | 3-letter currency code |  |
|  | CreatedOn | datetime | NA | Default date time format | Date time when record was inserted into the DB | This should be by default inserted in the database. |

**Reporting Implementation:**

1. The service request shall be stored in “PaymentRequest” table
2. The service response shall be stored in “PaymentResponse” and “PaymentResponseTransactions” tables.
3. In the event of an error received from target Demandware payment API, “PaymentResponse” table shall have an entry for each failure. The status field in this table shall be set as “error” and HTTP error code and error message received from the target API shall be stored into “ErrorCode” and “ErrorDescription” fields respectively.
4. The datetime filed on each of these tables shall be automatically populated. This can be handled by setting a trigger for the tables.

### Method Design

Following is the list of recommended Mule ESB components to be used for this method.

* HTTP Listener – to configure HTTP inbound endpoint for this API operation
* DataWeave – to transform incoming request to outgoing request
* HTTP request – to invoke target system API.
* Generic Database Connector – to configure database connection to the reporting database
* Async Scope – to asynchronously audit incoming request and outgoing response. This will also be used to make a request and response entry in the reporting database.
* Sub-flow/private flow – to modularize various steps involved in the method implementation. An example could be having a separate flow that handles request entry in the reporting database and calling this flow from Async scope using flow reference component.
* Flow Reference – to invoke a sub-flow or private flow
* VM outbound and inbound endpoint – to implement reliable accusation pattern for this API.
* Until Successful – to implement reconnection attempt to the target API.
* Reference Exception Strategy – to invoke a global shared exception strategy defined this API.

Note – above list is not necessarily the entire list of component that will be used but shows the important components that can be used for this API operation implementation.

### Summary

This API will use the following API components for executing the project

* JSON schema – to describe response structure for this API.
* RAML – to define API contracts. This will use the JSON schema internally.
* APIkit Console – to simulate and test the API contract
* APIkit Router – to route API call to the correct method

### Request Schema

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Cardinality | Length | Valid Data | Description |
|  | message-id | string | 0..1 | 36 | "2d159786-245b-11e7-93ae-92361f002671" | Unique identifier of transaction to avoid duplicated requests |
|  | operation | string | 1..1 | 10 | "capture" "refund" "cancel" | The API operation |
|  | order-number | string | 1..1 | 30 | "123456789" | Web order number |
|  | currency | string | 0..1 | 3 | "GBP" | Order currency |
|  | amount | decimal | 0..1 | 13,4 | "101.50" | Amount to be captured or refunded |

Sample Request:

|  |
| --- |
| {  “**message-id**”: “2d159786-245b-11e7-93ae-92361f002671”  “**operation**”: “capture”  “**order-number**”: “123456789”  “**currency**”: “GBP”  “**amount**”: “101.50”  } |

### Request Mapping

Refer the interface mapping sheet given in [references](#REFERENCES) section.

### Response Schema

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Attribute | Type | Cardinality | Length | Valid Data | Description |
|  | message-id | string | 1..1 | 36 | "2d159786-245b-11e7-93ae-92361f002671" | Unique identifier of transaction |
|  | status | string | 1..1 | 10 | "success" "error" | Status of the performed operation |
|  | error-code | string | 0..1 | 2 | "01" | Error code |
|  | error-description | string | 0..1 | 256 | "Payment Gateway Service Unavailable" | Error message |
|  | operation | string | 1..1 | 10 | "capture" "refund" "cancel" | Type of transaction |
|  | type | string | 0..1 | 10 | "CASH" "GIFT" | Operation object |
|  | processor | string | 0..1 | 256 | "ADYEN\_CREDIT" "PAYPAL\_EXPRESS" "GIFTCARD" | Payment processor for operation |
|  | amount | decimal | 0..1 | 13,4 | "101.50" | Amount of operation |
|  | currency | string | 0..1 | 3 | "GBP" | 3-letter currency code |

Sample Response:

|  |
| --- |
| {  “**message-id**”: “2d159786-245b-11e7-93ae-92361f002671”  “**status**”: “success”  “**transactions**”:  [  {  “**operation**”: “capture”  “**type**”: “CASH”  “**processor**”: “ADYEN\_CREDIT”  “**amount**”: “50.50”  “**currency**”: “GBP”  },  {  “**operation**”: “capture”  “**type**”: “GIFT”  “**processor**”: “GIFTCARD”  “**amount**”: “50.50”  “**currency**”: “GBP”  }  ]  } |

### Response Mapping

Refer the interface mapping sheet given in [references](#REFERENCES) section.

### Error Handling

As described in section [3.4.1](#_Implementation_3).

### Auditing

Service will audit any inbound and outbound messages using custom audit and logging framework. Details around the audit can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

### Logging

Service will log messages using custom audit and logging framework. Details around the logging can be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

# API/Service NFR



## Security

Following security shall be implemented/configured in this API:

1. The API shall be private and should use HTTPS i.e., expose internally using port 8092.
2. Client-id security shall be enforced to secure the API call.

## Auditing

Custom audit framework will be used for auditing all incoming and outgoing messages. Details around the audit framework be found in the HLD section 3.1.1.6. Link to HLD document can be found in [references](#REFERENCES) section of this document.

## Performance

Refer [FatFace NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) sheet for performance requirements.

## Exception and Error Handling and Alerting

There is no specific error handling scenario that has been identified or discussed. Standard error handling will be required to ensure the following:

* Any exception received from target CIMS web service is caught, logged and relevant error code and message shall be returned to the consumer.
* Any component level error such as data weave transformation error shall be handled and appropriate error code and message shall be returned to the consumer.
* If connection to any target system API fails, implementation shall do multiple reconnection attempts. If all attempts exhausts, Mule shall catch the exception and appropriate error code and error message should be returned to the consumer.

## Application Monitoring

* MuleSoft Anypoint Runtime Manager will be used to monitor the application
* MuleSoft API analytics dashboard will be used to monitor API consumption

## Batch Processing and Scheduling

NA

## Build and Deployment Solution

### Code Versioning



GitHub is used for software configuration management. Access to GitHub is provided by FatFace.

### Continuous Integration

Jenkins installed in Amazon Cloud will be used to pull source files from GitHub for building and deploying applications in CloudHub. FatFace is yet to provide access to Jenkins server in Cloud.

TBC – Additional details will be added after setting up Jenkins jobs in Cloud.

## Tools and Technologies



### Development Tools

* Anypoint Studio – for API implementation
* API Manager – for API design & policy configuration
* Anypoint Runtime Manager – for application deployment and alert configuration
* Maven – for defining the application dependencies

### Technologies

* Java
* Mule Expression Language
* XML
* JSON
* Groovy

### Environment

Refer HLD section 4.1.1.45 for details related to environments to be used. Link to HLD document can be found in [references](#REFERENCES) section of this document.

# Testing



## Testing Approach

The MuleSoft Payemt Demandware System API service will be invoked by the Payment experience API. The service will accept the JSON request and post to the DW service which receives success or error response in return. In case of any connection issues service to perform retry attempts and return success or error response. The testing to be carried out by verifying the log files, response generated by the application.

### Functional Testing

* Request format validation.
* Data mapping and Business rules validation.
* Response format validation.

### Non-Functional testing

Any non-functional requirements mentioned in [FatFace NFR](https://fatface.atlassian.net/wiki/download/attachments/92322210/Fat%20Face%20-%20NFR.xlsx?api=v2) document will be covered.

### Testing Tools

* Soap-UI
* Ready API
* HeidiSQL

### Mocking frameworks

NA

# Appendix



NA

\*\*\*End of Document\*\*\*