



**Support Handover Document**

**For**

**Fat Face - Integration HUB Phase-I and Phase-II**

13-Apr-2018

WHISHWORKS

www.whishworks.com

WW/PD/REC/SHD/1/HYD01/FFLL1

Status: Restricted

Reported By: Rupesh Sinha

*© WHISHWORKS 2018. 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 | Karthiraj Mani | [karthiraj.mani@whishworks.com](mailto:karthiraj.mani@whishworks.com) | Approver |
| 2 | Sougandh Golla | [Sougandh.golla@whishworks.com](mailto:Sougandh.golla@whishworks.com) | Reviewer |
| 3 | API Support Team | [fatface.supportteam@whishworks.com](mailto:fatface.supportteam@whishworks.com) | Reviewer |
| 4 | Mark Jones | [mark.jones@fatface.com](mailto:mark.jones@fatface.com) | Approver |
| 5 | Jim Autton | [jim.autton@fatface.com](mailto:jim.autton@fatface.com) | Approver |

**DOCUMENT CONTROL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared By** | **Reviewed By** | **Approved By** | **Approved Version** |
| Rupesh Sinha | Shruthi Thalari |  | 2.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**VERSION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 28-Nov-2017 | 1.0 | Initial version | Rupesh Sinha |
| 04-Dec-2017 | 1.1 | Updated all external links | Rupesh Sinha |
| 07-Feb-2018 | 1.2 | Updated Queries for retrieving CIMS request response After Promotional CR | Raviteja Alluri |
| 16-Mar-2018 | 1.3 | Updating Payment and refund failure queries | Raviteja Alluri |
| 13-Apr-2018 | 2.0 | Updated document with release-3 related details | Rupesh Sinha |

**REFERENCES**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Document/Artefact** | **Version** | **Date** |
| 1 | HLD - <https://fatface.atlassian.net/wiki/download/attachments/91790054/FatFace_DemandwareIntegr_FFLL1_HLD.docx?api=v2> | 2.3 | 24-Oct-2017 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**DEFINITIONS & ACRONYMS**

|  |  |  |
| --- | --- | --- |
| **#** | **Acronym** | **Expanded Form** |
| 1 | TBC | To be considered |
| 2 | SOA | Service Oriented Architecture |
| 3 | HLD | High Level Design |
| 4 | POS | Point of Sale |
| 5 | TBD | To be determined |
| 6 | ECOM | E-commerce |
| 7 | EPOS | Electronic Point of Sale |
| 8 | VPC | Virtual Private Cloud |
| 9 | VPN | Virtual Private Network |
| 10 | DW | Demandware |
| 11 | JMS | Java Messaging Service |
| 12 | SQS | Amazon Simple Queue Service |
| 13 | AWS | Amazon Web Service |

Table of Contents

[1. Introduction 6](#_Toc511387811)

[1.1 Purpose 6](#_Toc511387812)

[1.2 Business Overview 6](#_Toc511387813)

[1.3 References 7](#_Toc511387814)

[2. Pre-requisites 8](#_Toc511387816)

[2.1 Software 8](#_Toc511387817)

[2.2 Firewall Configuration 8](#_Toc511387818)

[2.3 Environment 9](#_Toc511387819)

[2.3.1 MuleSoft Server Configuration 9](#_Toc511387820)

[2.3.2 Database Server Configuration 9](#_Toc511387821)

[2.3.3 Dependent Application Configurations 10](#_Toc511387822)

[3. Dependencies 11](#_Toc511387823)

[3.1 Dependent Applications 11](#_Toc511387824)

[3.2 Dependent Services 11](#_Toc511387825)

[4. Scheduled Jobs 12](#_Toc511387826)

[5. Data Clean-up 13](#_Toc511387827)

[5.1 Database Clean-up 13](#_Toc511387828)

[5.2 File Repository Clean-up 13](#_Toc511387829)

[6. Deployment Process 14](#_Toc511387830)

[6.1 Source Control & Release Process 14](#_Toc511387831)

[6.2 Build Automation Process 14](#_Toc511387832)

[7. Testing 15](#_Toc511387833)

[7.1 Test Suites 15](#_Toc511387834)

[8. Maintenance 16](#_Toc511387835)

[8.1 Health Checks 16](#_Toc511387836)

[8.1.1 Functional Checks 16](#_Toc511387837)

[8.1.2 Sanity Checks 16](#_Toc511387838)

[8.1.3 Performance Checks 16](#_Toc511387839)

[8.1.4 Infrastructure Checks 16](#_Toc511387840)

[8.2 Monitoring Alerts 16](#_Toc511387841)

[8.3 Server Logs 18](#_Toc511387842)

[8.4 Troubleshooting 18](#_Toc511387843)

[9. Reporting 19](#_Toc511387844)

[10. Disaster Recovery Procedure 20](#_Toc511387845)

[11. Escalation Procedure 21](#_Toc511387846)

[12. Risks and Assumptions 22](#_Toc511387847)

[12.1 Known Issues/Risks 22](#_Toc511387848)

[12.2 Assumptions 23](#_Toc511387849)

[13. Appendix 24](#_Toc511387850)

# Introduction

## Purpose

Purpose of this Support Handover document, is to capture all technical details about the Fat Face Integration HUB Project for smoother transition from Development to Support phase of the project.

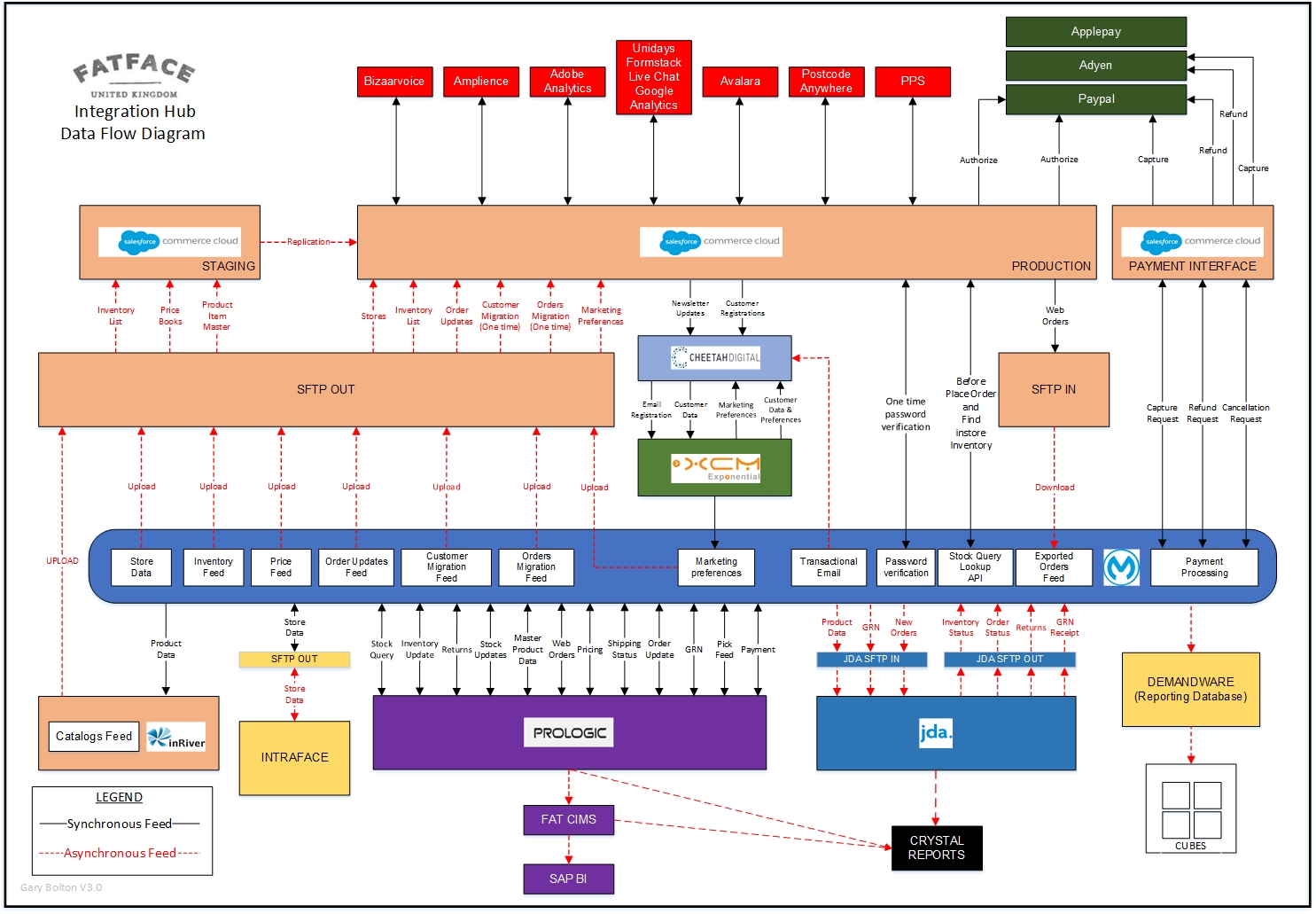
## Business Overview

Fat Face is a UK based lifestyle clothing brand, with a unique heritage, offering a wide range of high quality and affordable clothing, footwear and accessories.

They currently manage e-commerce transactions using a Venda platform, which does not offer multicurrency and multi geography capabilities, preventing the company from expanding globally and achieving its omnichannel retail objectives. In addition, point-to-point integration is used to connect legacy systems, impacting Fat Face’s ability to deliver on fast growing business demands.

Earlier in 2017, Fat Face decided to replace Venda with Salesforce Commerce Cloud (earlier known as Demandware) that would allow their e-commerce to operate in a multicurrency and multi geography environment.

There were several APIs built to power the ecommerce platform and replace their existing point-to-point integrations. The overall integration consists of a mixture of batch and synchronous APIs. Please see below diagram for the APIs developed for phase-I and phase-II.

****

## References

|  |  |
| --- | --- |
| Document Name | Document Location |
| Business/User Requirement Specification | <https://fatface.atlassian.net/secure/RapidBoard.jspa?rapidView=21&projectKey=IHUB&view=planning.nodetail&epics=visible> |
| Requirement Traceability Matrix | <https://drive.google.com/file/d/1-HBDjeDW0ild4c2gRSc29IYFjRYGXHpe/view> |
| SOA Service Registry | NA |
| Source Code | <https://github.com/FatFace/IntegrationHub> |
| Configuration files (Production  & Lower environments) | All config files are part of the code repository. The encryption key used for the production is as follows:   * Sandbox: jabdlgue34btiu2s * Production: jICm13MRF5IspiPe   The encryption algorithm used is AES. |
| DB Scripts | <https://github.com/FatFace/IntegrationHub/tree/master/database> |
| Service Design Documents (SDDs) or Low-Level Design Documents | <https://fatface.atlassian.net/wiki/spaces/IHUB/pages/93288997/2.4+SDD> |
| Data Mapping Sheets | <https://fatface.atlassian.net/wiki/spaces/IHUB/pages/93227839/2.3+Interface+Mapping+sheets> |
| MUnit Test Suites | MUnit test suites are available in GitHub under each application/project. |
| Functional Test Suites | There are no functional test suites. |
| SonarQube Code Review Report | <http://muledev.int.whishworks.com/view/FatFaceInt/> |
| Regression Test Suites | There are no regression test suites. |
| System Test Suites and Test Cases | <https://docs.google.com/spreadsheets/d/12_bDNcWfyiEmW4TDW7M5XTfwTwOiQvx-LIpRYytwkgI/edit#gid=233348737> |



# Pre-requisites

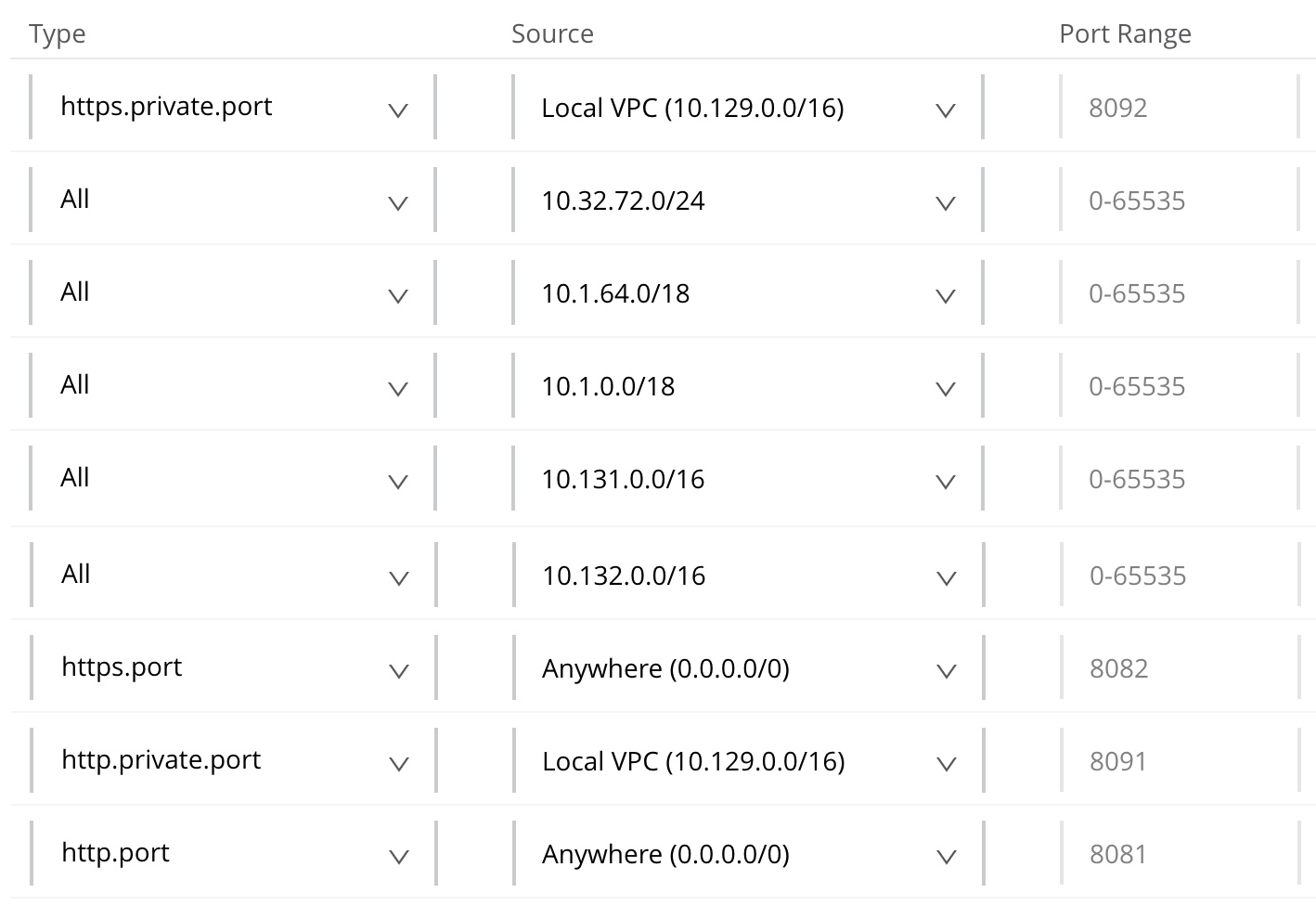
## Software

|  |  |
| --- | --- |
| Software Tools/Applications | Version |
| **CloudHub** | 3.8.4 Runtime |
| **GitHUB – Fat Face** | NA |
| **Jenkins – Fat Face** | 2.93 |
| **Confluence – Fat Face** | On cloud |
| **MS-SQL on-premise database** | 2016 |
| **AWS RDS Database (MS-SQL BYOL)** | On cloud (2016) |
| **AWS SQS** | On cloud |
| **Maytech SFTP** | On cloud |
| **SonarQube – Fat Face** | TBC |

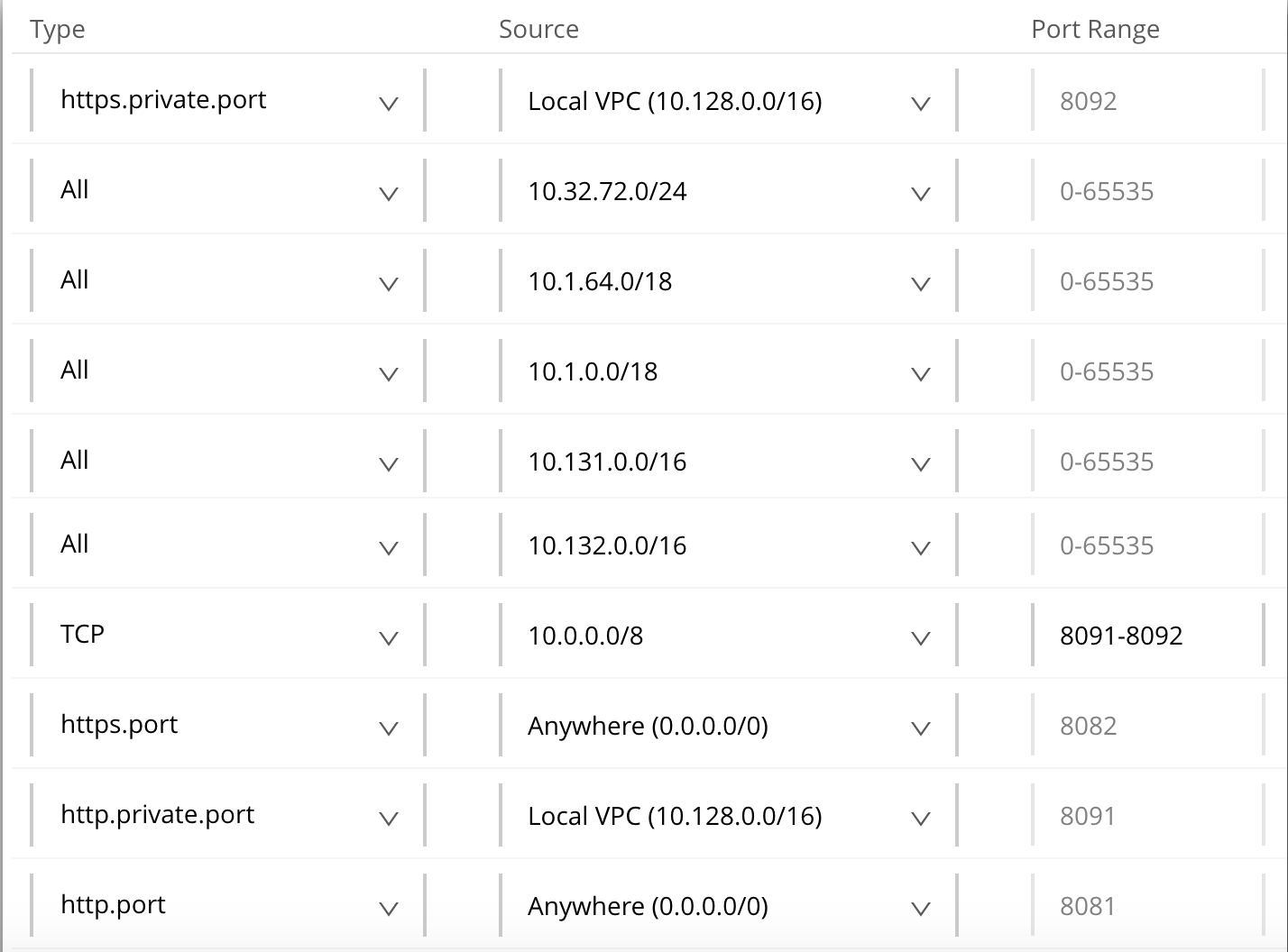
## Firewall Configuration

The firewall configuration is managed by Fat Face IT. Please contact Fat Face service delivery manager for any specific details around the Firewall configuration. The current firewall configuration in the CloudHub is given below for reference.

**Production Firewall configuration**



**Sandbox Firewall configuration**



## Environment

There are four environments created for Fat Face which are as shown below.

* Production
* UAT
* SIT
* Development

As most of the Fat Face backend systems have only two environments (test and production), all testing is done from SIT environment itself and UAT environment is currently unused. This may be used in future once a UAT environment for core systems such as CIMS and JDA are created.

There are 4vCores available for Production whereas Development, SIT and UAT collectively share 4vCores.

### MuleSoft Server Configuration

All MuleSoft applications are running on CloudHub. Details around other servers can be found in the section 4.1.1.42 of the Fat Face HLD. A quick link for the HLD us given below.

<https://fatface.atlassian.net/wiki/download/attachments/91790054/FatFace_DemandwareIntegr_FFLL1_HLD.docx?api=v2>

### Database Server Configuration

All on-premise database servers are managed by Fat Face IT and specific details are not available with us. Details on the AWS RDS database is available in section 4.1.1.42 of the Fat Face HLD.

### Dependent Application Configurations

NA

# Dependencies

## Dependent Applications

|  |  |
| --- | --- |
| Dependent Applications | Description |
| **MS SQL Server Demandware DB** | A reporting database hosted on premise to hold all order and payment related data. Please refer IHUB-166 and IHUB-166 and IHUB-168 for details on the reporting requirements. |
| **Maytech SFTP** | Used for all files-based integrations in Mule. Please refed HLD for details on SFTP directory structure and their usage. |
| **Amazon SQS** | Used for order create functionality. Please refer SDDs for IHUB-6 for details. |
| **Amazon RDS** | Used for data migration (one-time activity) and MuleSoft API audit (30 days of data is maintained in audit table). |

## Dependent Services

|  |  |
| --- | --- |
| Dependent Applications | Description |
| **CIMS SOAP web service** | Used for all transactions around CIMS. |
| **Demandware Payment API** | Used for payment capture, refund and cancellation for an order in Demandware payment gateway. |
| **CCMP Email API** | Used to send emails to customer for delivery, refund and cancellation scenarios. |
| **InRiver/PIM product API** | Used to send product feed to InRiver/PIM |

# Scheduled Jobs

All job schedules are captured in below spreadsheet.

<https://fatface.atlassian.net/wiki/download/attachments/104890369/FatFace-IHUB-Interface%20Schedule%20Catalog.xlsx?api=v2>

Please refer “Mule Job” sheet for details on Mule job frequencies.

# Data Clean-up

## Database Clean-up

A database clean-up job “Mule Audit DB Clean-up Job” is created in both sandbox and production that deletes audit record older than 30 days from “t\_audit” table. The job runs daily 1 minute past midnight. More details are available in [IHUB-1005](https://fatface.atlassian.net/browse/IHUB-1005).

## File Repository Clean-up

A job has been created on Maytech SFTP server by Peter Hunt to delete any files older than 30 days from all directories and subdirectories under “Fat Face” directory on both sandbox and production SFTP servers. Refer [IHUB-1006](https://fatface.atlassian.net/browse/IHUB-1006) for more details.

# Deployment Process

## Source Control & Release Process

Source code are maintained in Fat Face GitHub repository at below link.

<https://github.com/FatFace/IntegrationHub>

This is a private repository so one will need a Fat Face account to access the repository.

## Build Automation Process

Refer [Build and Deployment.docx](https://fatface.atlassian.net/wiki/download/attachments/118816769/Build%20and%20Deployment.docx?api=v2) for details on build and deployment process for all Fat Face MuleSoft applications.

# Testing

## Test Suites

No test suites were generated. Test cases and execution report is available at

<https://docs.google.com/spreadsheets/d/12_bDNcWfyiEmW4TDW7M5XTfwTwOiQvx-LIpRYytwkgI/edit#gid=1486494760>

# Maintenance

## Health Checks

### Functional Checks

Below listed functional checks can be performed daily

* Monitor CloudHub Alerts for Production deployed applications related Errors
* Monitor CloudHub schedule jobs for its status
* Monitor files are generated and copied to respective FTP or SFTP folders
* Monitor audit logs for any errors and liaise with business for reprocessing the failed records if required.

### Sanity Checks

Below listed sanity checks to be done daily

* Monitor CloudHub APIs are running
* Sanity check is performed when application is deployed in any environment like dev and sit. If changes deployed in production, will ask business team to process a single record to ensure that is the change is working fine.

### Performance Checks

Below listed performance checks to be done regularly.

* Monitor the CPU & Memory of CloudHub for its performance

### Infrastructure Checks

Below listed infrastructure checks to be performed manually

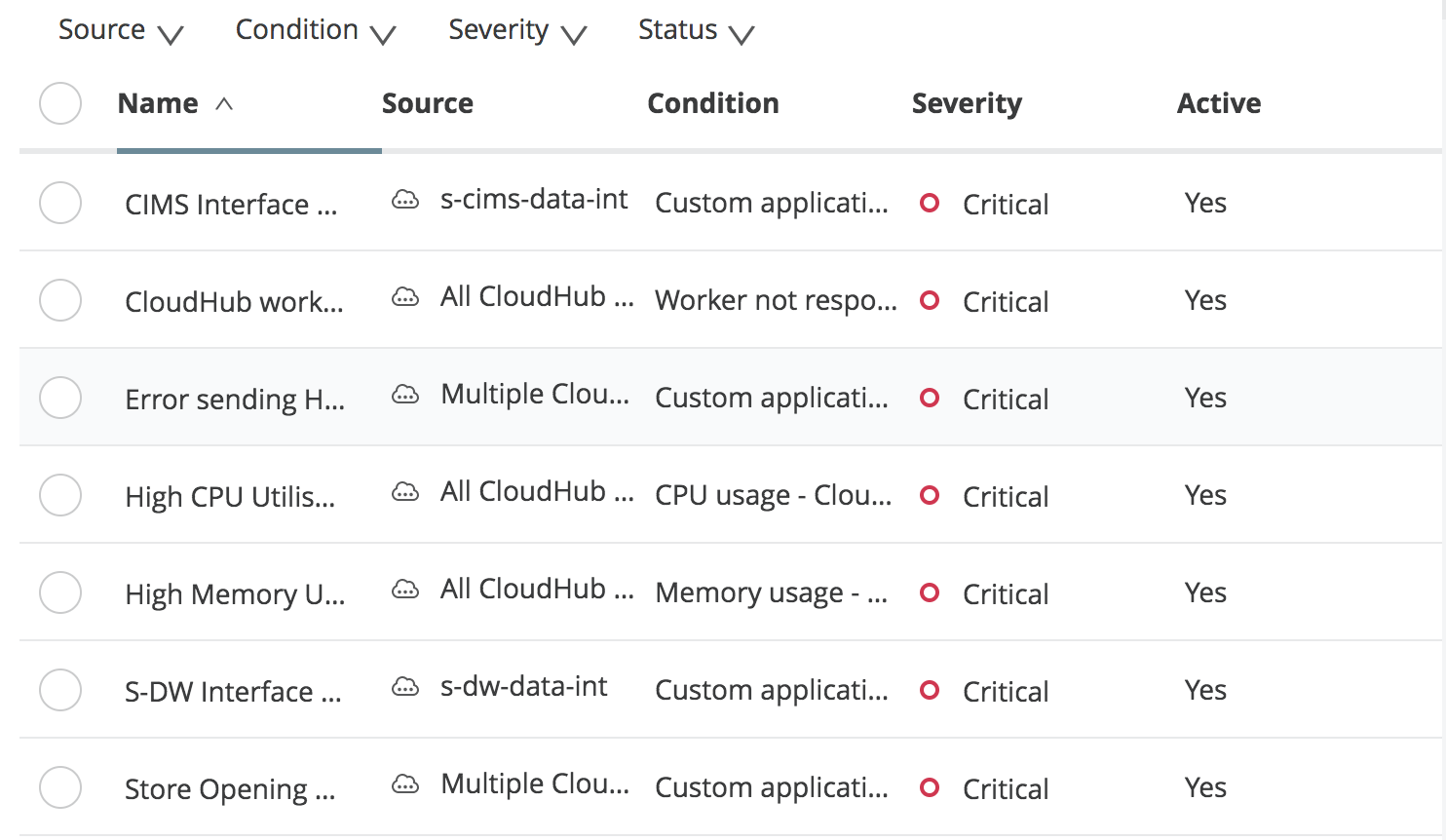
* Check if VPN connection or on-premise systems are working from both CloudHub and local machines.

## Monitoring Alerts

Following alerts are in scope for the current phase of the project.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Mule application name | Memory > 75 | CPU > 75 | Worker not responding | SFTP connection failure | HTTP connection failure | Database connection failure | AWS SQS connection failure |
| 1 | x-dw-web-api | X | X | X |  |  | X |  |
| 2 | x-payment-request-api | X | X | X |  |  | X |  |
| 3 | x-order-status-api | X | X | X |  |  | X |  |
| 4 | x-customer-api | X | X | X |  |  | X |  |
| 5 | p-cims-data-int | X | X | X |  | X | X | X |
| 6 | p-jda-data-int | X | X | X |  | X | X |  |
| 7 | p-dw-data-int | X | X | X |  | X | X |  |
| 8 | s-pim-data-int | X | X | X |  | X | X |  |
| 9 | s-jda-data-int | X | X | X | X | X | X |  |
| 10 | s-ccmp-data-int | X | X | X |  | X | X |  |
| 11 | s-dw-data-int | X | X | X | X | X | X |  |
| 12 | s-cims-data-int | X | X | X |  | X | X | X |
| 13 | s-intraface-data-int | X | X | X | X | X | X |  |

Following alerts have been created in CloudHub to address above alerting requirements.



Not all alerts have been tested. New alerts may need to be configured to meet above alerting requirements.

## Server Logs

Currently there are no automated log download in place. This is being investigated as part of [IHUB-1110](https://fatface.atlassian.net/browse/IHUB-1110). Until an automation is in place, logs should be periodically downloaded and checked for errors. Please maintain all logs on the SFTP server at “/FatFace/sftp/MuleSoftIntegration/<environment>/logs/” so that old logs files can be automatically deleted as part of the clean-up job.

## Troubleshooting

* Most of the issues are being notified via an email and the email contains the message\_id and request\_id of the transactions that has caused an alert. Troubleshooting of an issue should start with downloading the logs and tracing the error using the message\_id or request\_id. If sufficient details are not available in the logs, one can retrieve the payload from audit table and try and replicate the issue in lower environment with debug logs enabled. In case, message\_id or request\_id is not available in logs, the only way to trouble sheet the issue will be by downloading the logs and investigating the transaction that has failed.
* In case of connection issue between MuleSoft CloudHub environment and on-premise connection issue, support can deploy “net-tools-fatface-api” application (already available in applications) and open the web console of the application using “[**net-tools-fatface-api.eu.cloudhub.io**](http://net-tools-fatface-api.eu.cloudhub.io/)”. Further support can try to capture some stats such as ping, socket connection, traceroot etc. The same application is available in SIT environment with the name “sit-fatface-net-tools-api”. The credentials to login is part of the code. Please download the deployment ZIP and extract the login details from “mule-app.properties” file.
* Following are some of the key DEBUG logs that could be enabled in sandbox environment for troubleshooting any issues.
  + org.mule.transport.sftp – enable debug for SFTP transport
  + org.mule.transport.jdbc – enable debug for JDBC transport
  + com.ning.http – enable debug for HTTP calls
  + org.mule.module.http.internal.HttpMessageLogger – enable debug for HTTP calls
  + org.apache.cxf – enable debug for HTTP calls
  + org.mule – enable debug for all mule components

**Note** that DEBUG should not be enabled in Production as it may impact the performance of the applications. Any debugging should be carried out test environment and debug should be disabled once the investigation is done.

# Reporting

|  |  |  |  |
| --- | --- | --- | --- |
| Report Name | Report Format | Report Contents | ESB Service Support Contact |
| **Daily Health Check Report** | An excel sheet with checklist and verification details of all Mule ESB servers, SQL Server DB and Cloud Hub servers in both pre-production and production environments | Server Health Check List, ESB Application Statuses, Alerts, Batch Job Statuses, DB Alerts, Capacity Checks, Memory utilization and License Information etc. | TBC |
| **Weekly Status Report** | A weekly summary of support activities and incidents of the previous week | Key achievements of the past week, incident and service request summary, risks, issues and ticket ageing analysis etc. | TBC |
| **Monthly Report** | A detailed report with all incidents, problems and support activities carried out in the previous month | Incidents, Problems, changes, minutes of monthly review meeting etc. | TBC |

# Disaster Recovery Procedure

Yet to be identified.

# Escalation Procedure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Priority | Escalation Time  beyond SLA | Escalation Point 1 | Escalation Point 2 | Escalation Point 3 |
| **P1** | + 30 Min | TBC | TBC | TBC |
| **P2** | + 1 Hour | TBC | TBC | TBC |
| **P3** | + 1 Day | TBC | TBC | TBC |
| **P4** | + 3 Days | TBC | TBC | TBC |

Please refer the supplier documents in the below link for details on escalation procedure for individual suppliers.

<https://fatface.atlassian.net/wiki/spaces/IHUB/pages/115605505/Supplier+Documentation>

For FatFace escalation, you should contact the FatFace service delivery manager.

# Risks and Assumptions

## Known Issues/Risks

|  |  |
| --- | --- |
| Issue Description | Proposed/Intended Solution (If Any) |
| **Large file streaming/VPN connection issue/Planned downtime on CIMS/CIMS service is unavailable** | This will be planned with phase 2 release. Until this is fixed, support should work with business not queue orders in Demandware. In case of any downtime on CIMS side or any VPN issues, support should liaise with business to get the Demandware “Mule Order Export” directory location changed temporarily so that Demandware can continue to export the files on the SFTP server. Once the issues with VPN or CIMS is resolved, the files queued on the SFTP server should be processed manually by moving them to the correct export location. Max 5 MB files should be processed at once.  The details on how to change the Demandware “Mule Order Export” target location is captured in a separate document available in Fat Face confluence at below link.  [https://fatface.atlassian.net/wiki/download/attachments/115376129/Changing Order Export Job SFTP location in Demandware.docx?api=v2](https://fatface.atlassian.net/wiki/download/attachments/115376129/Changing%20Order%20Export%20Job%20SFTP%20location%20in%20Demandware.docx?api=v2) |
| **Audit database clean-up** | There is no automated audit DB clean job available in sandbox or production. Clean-up should be done manually until IHUB-1005 is fixed. |
| **SFTP files cleanup** | There is no automated SFTP files clean job available in sandbox or production. Clean-up should be done manually until IHUB-1006 is fixed. |
| **UTF-8 characters not support by CIMS** | There are few international orders which fail due to special characters in the address field. Not all special characters are currently supported by CIMS orderCreate service. Such issues will be reported via the alerts and support team should liaise with business to provide correct address and manually process those orders to CIMS.  The manual processing can be done using the internal worker URL for OrderCreate API or by calling CIMS web service directly using Soap-UI. |
| **Payment capture/refund/cancel failure** | Demandware payment API sometimes fail to process the payment and returns an error to MuleSoft. A retry is being implemented in MuleSoft API but a daily report will be required on the payment failure and CIMS may need to resend the payment request if payment processing fails from Demandware.  This report currently does not exists so manual query may be required. |
| **Object store key not found exception** | This issue occurs in “s-ccmp-data-int” application occasionally. This is a known bug on MuleSoft side and an enhancement request is logged for a fix. Please refer case number “00180765” on MuleSoft support for progress. |
| **Error '' occurred when trying to CDW** | This issue occurs in logs occasionally. This has no business impact. A MuleSoft ticket ([00180648](https://support.mulesoft.com/s/case/5003400000zPKRGAA4)) is logged for this issue and this is being investigated with MuleSoft and Maytech. |

## Assumptions

TBC

# Appendix

**Useful SQL queries around audit table.**

1. To get a report on all calls to price-book feed for a given date.

|  |
| --- |
| ;WITH XMLNAMESPACES ('http://xdi/FFService.wsdl' AS ns2,  'http://www.prologic.com/schema/cims/ff\_service/priceFeedResponse' AS priceFeed)  SELECT  ID, message\_id,  convert(xml,log\_message).value('(/ns2:PriceFeedResponse/priceFeed:priceFeedOUTPUT/@XDI\_ID)[1]', 'nvarchar(max)') as XDI\_ID,  convert(xml,log\_message).value('(/ns2:PriceFeedResponse/priceFeed:priceFeedOUTPUT/@MAX\_ID)[1]', 'nvarchar(max)') as MAX\_ID,  convert(xml,log\_message).value('(/ns2:PriceFeedResponse/priceFeed:priceFeedOUTPUT/@Type)[1]', 'nvarchar(max)') as FeedType,  convert(xml,log\_message).value('(/ns2:PriceFeedResponse/priceFeed:priceFeedOUTPUT/PRICE\_LISTS/PRICE\_LIST/@Name)[1]', 'nvarchar(max)') as PriceListName,  record\_create\_time as AuditCreatedTimestamp,  convert(xml,log\_message).value('(/ns2:PriceFeedResponse/priceFeed:priceFeedOUTPUT/@CalledAt)[1]', 'nvarchar(max)') as CIMSRequestTimestamp,  convert(xml,log\_message).value('(/ns2:PriceFeedResponse/priceFeed:priceFeedOUTPUT/@Timestamp)[1]', 'nvarchar(max)') as CIMSResponseTimestamp  FROM  t\_audit  where  log\_type='CALL EXIT'  and instance\_thread='get:/pricebook:s-cims-api-config'  and record\_create\_time>'YYYY-MM-DD'  order by Id desc; |

1. To get a report on all calls to stock feed for a given date.

|  |
| --- |
| ;WITH XMLNAMESPACES ('http://xdi/FFService.wsdl' AS ns2,  'http://www.prologic.com/schema/cims/ff\_service/stockFeedResponse' AS stockFeed)  SELECT  ID, message\_id,  convert(xml,log\_message).value('(/ns2:StockFeedResponse/stockFeed:stockFeedOUTPUT/@XDI\_ID)[1]', 'nvarchar(max)') as XDI\_ID,  convert(xml,log\_message).value('(/ns2:StockFeedResponse/stockFeed:stockFeedOUTPUT/@MAX\_ID)[1]', 'nvarchar(max)') as MAX\_ID,  convert(xml,log\_message).value('(/ns2:StockFeedResponse/stockFeed:stockFeedOUTPUT/@Type)[1]', 'nvarchar(max)') as FeedType,  record\_create\_time as AuditCreatedTimestamp,  convert(xml,log\_message).value('(/ns2:StockFeedResponse/stockFeed:stockFeedOUTPUT/@CalledAt)[1]', 'nvarchar(max)') as CIMSRequestTimestamp,  convert(xml,log\_message).value('(/ns2:StockFeedResponse/stockFeed:stockFeedOUTPUT/@Timestamp)[1]', 'nvarchar(max)') as CIMSResponseTimestamp  FROM  t\_audit  where  log\_type='CALL EXIT'  and instance\_thread = 'get:/inventory-list:s-cims-api-config'  and record\_create\_time>'YYYY-MM-DD'  order by Id desc; |

1. To get a report on all calls to product feed for a given date.

|  |
| --- |
| ;WITH XMLNAMESPACES ('http://xdi/FFService.wsdl' AS ns2,  'http://www.prologic.com/schema/cims/ff\_service/productFeedResponse' AS productFeed)  SELECT  ID, message\_id,  convert(xml,log\_message).value('(/ns2:ProductFeedResponse/productFeed:productFeedOUTPUT/@XDI\_ID)[1]', 'nvarchar(max)') as XDI\_ID,  convert(xml,log\_message).value('(/ns2:ProductFeedResponse/productFeed:productFeedOUTPUT/@MAX\_ID)[1]', 'nvarchar(max)') as MAX\_ID,  convert(xml,log\_message).value('(/ns2:ProductFeedResponse/productFeed:productFeedOUTPUT/@Type)[1]', 'nvarchar(max)') as FeedType,  record\_create\_time as AuditCreatedTimestamp,  convert(xml,log\_message).value('(/ns2:ProductFeedResponse/productFeed:productFeedOUTPUT/@CalledAt)[1]', 'nvarchar(max)') as CIMSRequestTimestamp,  convert(xml,log\_message).value('(/ns2:ProductFeedResponse/productFeed:productFeedOUTPUT/@Timestamp)[1]', 'nvarchar(max)') as CIMSResponseTimestamp  FROM  t\_audit  where  log\_type='CALL EXIT'  and instance\_thread='get:/products:s-cims-api-config'  and record\_create\_time>'YYYY-MM-DD'  order by Id; |

1. To retrieve an audit of all transactions made against an order

|  |
| --- |
| select \* from t\_audit where  record\_create\_time > '2017-11-04 00:07:00.00'  and message\_id in (select message\_id from t\_audit where record\_create\_time > '2017-11-04 00:07:00.00')  and log\_message like '%UKF1000096098%'  order by record\_create\_time asc |

1. To retrieve CIMS request and response for a given order and date.

|  |
| --- |
| select \* from t\_audit where  record\_create\_time > '2017-12-04 00:07:00.00'  and instance\_thread='s-cims-create-order-api-impl-sub-flow'  and request\_id in (  select request\_id from t\_audit where  record\_create\_time > '2017-12-04 00:07:00.00'  and instance\_thread='s-cims-create-order-api-impl-sub-flow'  )  and log\_type in ('ERROR', 'CALL ENTRY', 'SUCCESS')  and instance\_thread='s-cims-create-order-api-impl-sub-flow'  and log\_message like '%UKF1000210190%'  order by record\_create\_time asc |

1. To get the log message as XML where full request or response cannot be copied directly from management studio

|  |
| --- |
| select convert(xml,log\_message) as XMLData from t\_audit where request\_id='9e07c714-36ed-4e30-93bf-b6abb2a16e27'  OR  select convert(xml,log\_message) as XMLData from t\_audit where message\_id='9e07c714-36ed-4e30-93bf-b6abb2a16e27' |

1. To check for Demandware Payment API request and response for a given order and date.

|  |
| --- |
| select \* from t\_audit where  record\_create\_time > '2017-12-04 00:07:00.00'  and instance\_thread='post:/payment-request:application/json:s-dw-api-config'  and message\_id in (  select message\_id from t\_audit where  record\_create\_time > '2017-12-04 00:07:00.00'  and instance\_thread='post:/payment-request:application/json:s-dw-api-config'  and log\_message like '%UKF1000205850%'  )  and log\_type in ('CALL ENTRY', 'CALL EXIT')  and instance\_thread='post:/payment-request:application/json:s-dw-api-config'  order by record\_create\_time asc |

1. To check for CIMS Payment API request and response payload.

|  |
| --- |
| select \* from t\_audit where  record\_create\_time > '2017-12-04 00:07:00.00'  and instance\_thread='post:/payment-request:application/xml:x-payment-request-api-main-config'  and message\_id in (  select message\_id from t\_audit where  record\_create\_time > '2017-12-04 00:07:00.00'  and instance\_thread='post:/payment-request:application/xml:x-payment-request-api-main-config'  and log\_message like '%UKF1000205850%'  )  and log\_type in ('ENTRY', 'EXIT')  and instance\_thread='post:/payment-request:application/xml:x-payment-request-api-main-config'  order by record\_create\_time asc |

**Steps to resend Create Order request to CIMS using SoapUI**

|  |
| --- |
| --Use the following select to get CIMS Create Order XML Request. Modify record\_creation\_time as needed.  select distinct(log\_message) from t\_audit  where  instance\_thread = 's-order-api-main-sqs-create-order'  and record\_create\_time > '2017-11-14 22:00:00'  and log\_type ='CALL ENTRY'  order by record\_create\_time asc  Use SoapUI to send the XML request to  <http://cimslive.fatface.hq:8001/FatFaceLive/FFServicePort>  Use HTTP basic authorization (get username and password from the code)  Here is the sample request. Keep the yellow highlighted as it is and replace the blue highlighted lines with the XML from the audit table after removing “<?xml version='1.0' encoding='UTF-8'?>” from the request.  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ffs="http://xdi/FFService.wsdl">  <soapenv:Header/>  <soapenv:Body>  <ffs:OrderCreate xmlns:ffs="http://xdi/FFService.wsdl">  <orderCreateINPUT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.prologic.com/schema/cims/ff\_service/orderCreateRequest orderCreate.xsd" xmlns="http://www.prologic.com/schema/cims/ff\_service/orderCreateRequest">  -------------------  -------------------  </orderCreateINPUT>  </ffs:OrderCreate>  </soapenv:Body>  </soapenv:Envelope> |
|  |

**Useful queries around reporting database**

1. To check history of payment API calls for an order

|  |
| --- |
| select  PaymentRequest.TransactionId,  PaymentRequest.OrderNumber,  PaymentRequest.Operation,  PaymentRequest.Amount,  PaymentResponse.Status,  PaymentResponse.ErrorCode,  PaymentResponse.ErrorDescription,  PaymentRequest.CreatedOn,  PaymentResponseTransactions.Processor,  PaymentResponseTransactions.Amount  from  PaymentRequest left join  PaymentResponse on PaymentRequest.TransactionId=PaymentResponse.TransactionId  left join PaymentResponseTransactions on PaymentRequest.TransactionId=PaymentResponseTransactions.TransactionId  where  PaymentRequest.OrderNumber in ('UKF1000205850')  order by  PaymentResponse.CreatedOn asc |

To check order details in the Demandware order reporting tables

|  |
| --- |
| select \* from FF\_Order where order\_no='UKFT10125690'  select \* from FF\_Customer where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_ProductLineItem where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_OrderLineStatus where order\_no='UKFT10125690'  select \* from FF\_Payment where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_Address where customer\_id=(select ID from FF\_Customer where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690'))  select \* from FF\_OrderTotals where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_Totals where ID in (377650,377646,377652,377653,377654)  select \* from FF\_Shipment where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_ShipmentTotals where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_ShippingLineItem where Order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_Note where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from FF\_CustomAttribute where order\_id=(select ID from FF\_Order where order\_no='UKFT10125690')  select \* from [Status] where [order-no]='UKFT10125690' |

To check Payment capture failures.

select

PaymentRequest.OrderNumber,

FF\_Customer.customer\_email,

FF\_Payment.payment\_method,

PaymentRequest.Amount as CaptureAmount,

PaymentResponse.Status,

PaymentResponse.ErrorCode,

PaymentResponse.ErrorDescription,

PaymentResponse.CreatedOn PaymentCaptureRequestedOn

from

FF\_Order left join PaymentRequest

on FF\_Order.order\_no=PaymentRequest.OrderNumber

left join PaymentResponse

on PaymentRequest.TransactionId=PaymentResponse.TransactionId

left join FF\_Payment on FF\_Payment.order\_id=FF\_Order.ID

left join FF\_Customer on FF\_Customer.order\_id=FF\_Order.ID

where

(PaymentResponse.Status='error' )

and PaymentRequest.Operation='capture'

and PaymentRequest.OrderNumber not in

(select PaymentRequest.OrderNumbe from FF\_Order left join PaymentRequest

on FF\_Order.order\_no=PaymentRequest.OrderNumber left join PaymentResponse

on PaymentRequest.TransactionId=PaymentResponse.TransactionId

left join FF\_Payment on FF\_Payment.order\_id=FF\_Order.ID

where PaymentResponse.Status='success' and PaymentRequest.Operation='capture') and PaymentResponse.CreatedOn > '2018-03-15 00:00:00.0000000'

Order by

PaymentResponse.CreatedOn desc

To check Refund Failures

select

PaymentRequest.OrderNumber,

FF\_Customer.customer\_email,

FF\_Payment.payment\_method,

PaymentRequest.Amount as CaptureAmount,

PaymentResponse.Status,

PaymentResponse.ErrorCode,

PaymentResponse.ErrorDescription,

PaymentResponse.CreatedOn PaymentCaptureRequestedOn

from

FF\_Order left join PaymentRequest

on FF\_Order.order\_no=PaymentRequest.OrderNumber

left join PaymentResponse

on PaymentRequest.TransactionId=PaymentResponse.TransactionId

left join FF\_Payment on FF\_Payment.order\_id=FF\_Order.ID

left join FF\_Customer on FF\_Customer.order\_id=FF\_Order.ID

where

(PaymentResponse.Status='error' )

and PaymentRequest.Operation='refund'

and PaymentResponse.CreatedOn > '2018-03-15 00:00:00.0000000'

Order by

PaymentResponse.CreatedOn desc

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