# Fanatics

# *Design Document*

# *Payment Processing - Settlement*

**Authors: Lightwell Inc.**

**Date Created:** April 14, 2016

**Last Updated:** 5/11/2016

**Version:** 0.3

**File Name: Fanatics Payment Settlement.docx**

Document Revision History

| Revision | **Date** | **Description (Changes Made)** | **Author** |
| --- | --- | --- | --- |
| 0.1 | April 14, 2016 | Initial Draft | Sourav Bairagi |
| 0.2 | May 3, 2016 | Updated with Internal review changes | Sai Tatineni |
| 0.3 | May 10, 2016 | Updated with Brian’s review Comments | Sai Tatineni |

Document Approval History

| **Name** | **Signature** | **Date** | **Version Approved / Comments** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Related or Reference Documents

| **Document Name** | **Description** | **Owner** | **Location** |
| --- | --- | --- | --- |
| Fanatics\_Triple Crown\_BRD\_Financial\_v2.0.docx | Requirements document |  | [Fanatics\_Triple Crown\_BRD\_Financial\_v2.0.docx](https://fanatics.atlassian.net/wiki/download/attachments/38469851/Fanatics_Triple%20Crown_BRD_Financial_v2.0.docx?version=1&modificationDate=1456257406322&api=v2) |
| Fanatics OMS Solution Document - version 0.4 | Solution design document |  | [Fanatics OMS Solution Document - version 0.4](https://fanatics.atlassian.net/wiki/download/attachments/39289178/Fanatics_OMS_SolutionDefinition.doc?version=6&modificationDate=1457642161657&api=v2) |
| OMS Payment Settlement | Mapping Document |  | [OMS Payment Settlement](https://fanatics.atlassian.net/wiki/download/attachments/36045148/OMS%20Payment%20Settlement%20Service%20Mapping%20Template.xlsx?version=2&modificationDate=1461007691471&api=v2) |

Useful URL’s

| **URL** | **Description** |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Table of Contents

[1. Introduction 5](#_Toc450752150)

[1.1 Purpose 5](#_Toc450752151)

[2. Payment Settlement Overview 5](#_Toc450752152)

[2.1 Overview 5](#_Toc450752153)

[2.2 Assumptions 6](#_Toc450752154)

[2.3 Process Flow Diagram 6](#_Toc450752155)

[3. Impacted Components 7](#_Toc450752156)

[3.1 ‘Payment\_Collection’ Transaction 7](#_Toc450752157)

[3.2 ‘Payment\_Execution’ Transaction 7](#_Toc450752158)

[3.3 ‘YFSCollectionCreditCardUE’ user exit 7](#_Toc450752159)

[3.4 ‘YFSCollectionStoredValueCardUE’ user exit 7](#_Toc450752160)

[3.5 ‘YFSCollectionOthersUE’ user exit 7](#_Toc450752161)

[3.6 FanaticsSettlementInvocationService’ 8](#_Toc450752162)

[3.7 ‘COLLECTION\_FAILED’ event on ‘PAYMENT\_EXECUTION’ transaction 8](#_Toc450752163)

[3.8 ‘CHECK\_PAYMENT\_ALERTS.0001.ex’ time triggered transaction (agent server) 8](#_Toc450752164)

[4. Configuration Details 9](#_Toc450752165)

[4.1 Transaction: PAYMENT\_COLLECTION 9](#_Toc450752166)

[4.1 Transaction: PAYMENT\_EXECUTION 11](#_Toc450752167)

[4.2 User Exit: ‘YFSCollectionCreditCardUE’ 12](#_Toc450752168)

[4.3 User Exit: ‘YFSCollectionStoredValueCardUE’ 13](#_Toc450752169)

[4.4 User Exit: ‘YFSCollectionOthersUE’ 14](#_Toc450752170)

[4.5 Service: ‘YCD\_ExecuteCollectionCreditCard\_Proxy\_1.0’ 15](#_Toc450752171)

[4.6 Service: ‘YCD\_ExecuteCollectionSVC\_Proxy\_1.0 16](#_Toc450752172)

[4.7 Service: ‘YCD\_ExecuteCollectionOthers\_Proxy\_1.0’ 17](#_Toc450752173)

[4.7.1 API Component 17](#_Toc450752174)

[4.8 Service: ‘FanaticsSettlementInvocationService’ 18](#_Toc450752175)

[4.8.1 REST call Utility API Component 19](#_Toc450752176)

[4.9 Payment Alert Queue Configuration 20](#_Toc450752177)

[4.10 Event: ‘COLLECTION\_FAILED’ 20](#_Toc450752178)

[4.11 Hold: PAYMENT\_CAPTURE\_HOLD 23](#_Toc450752179)

[4.12 Service: FanaticsTasksOnPaymentHardDecline 24](#_Toc450752180)

[4.13 Service: FanaticsTasksOnPaymentServiceUnavailable 25](#_Toc450752181)

[4.13 Transaction: CHECK\_PAYMENT\_ALERTS.0001.ex 26](#_Toc450752182)

[4.14 Payment Method configuration 27](#_Toc450752183)

[4.14.1 Payment Types: 27](#_Toc450752184)

[4.14.2 Payment Card Types 37](#_Toc450752185)

[4.15 Post Conditions 40](#_Toc450752186)

[4.16 Test Cases 40](#_Toc450752187)

# Introduction

## Purpose

The purpose of this document is to provide an overview of the implementation and design of the payment settlement interface between OMS and Payment Gateway. This document also gives a technical overview of how settlement processing is setup in OMS. This document only covers the settlement process for payment processing.

IBM OMS will use a common enterprise payment service to process all payment transactions (authorization, capture, refunds, release authorization holds) across all payment methods.

# Payment Settlement Overview

## Overview

Once OMS receives the ship confirmation message, OMS will generate an invoice for the shipped items and a request will be generated to process a settlement/capture for the value of the shipped goods. Existing authorization will be used to make settlement requests. In case of credit/debit card payment method, same authorization ID will be used for multiple settlement requests on a multi shipment order.

OMS may initiate multiple settlement requests if the order gets shipped at different points in time.

OMS will initiate one settlement call per payment type if an order has multiple payment methods and the settlement for an invoice is to be charged on multiple payment methods.

Settlement on an order that has multiple payment methods will follow a charge sequence as defined in the below table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Payment Type** | **Payment Type Group** | **Charge Seq** | **Max charge limit** |
| Gift Card | Store value card | 2 | Up to available balance. |
| Gift Certificate | Store value card | 2 | Up to available balance. |
| Credit/Debit Cards (Visa, Master, Amex, Discover) | Credit card | 3 | Max charge limit is not set. |
| PayPal | Others | 4 | Max charge limit set at time of order capture |
| Account Balance | Store value card | 1 | Up to available balance. |

If OMS is not able to process a capture/settlement request due to a hard decline from the payment processor it will do the following

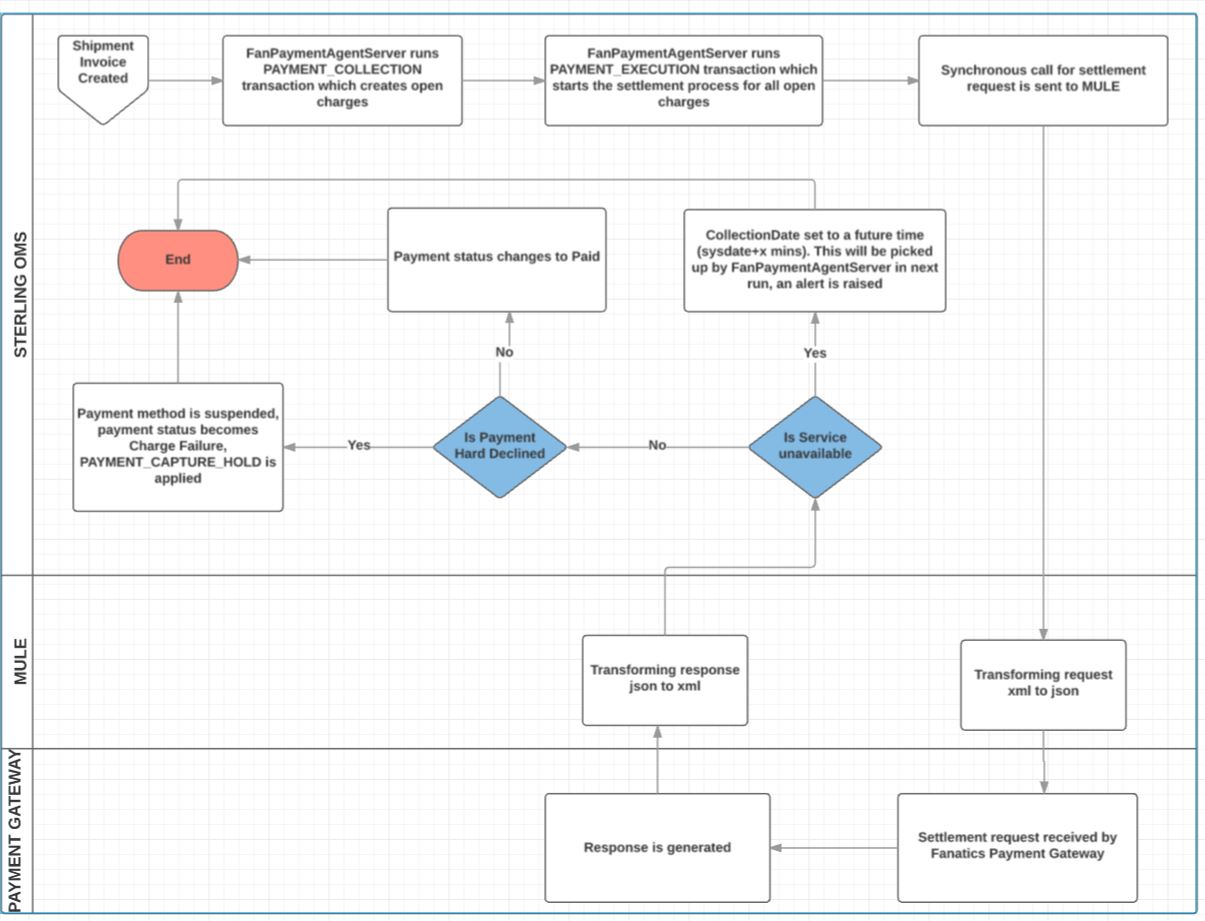
* The payment status will move to ‘await\_pay\_info’ and the payment method will be suspended for both charge and authorization in OMS
* An alert will be raised and assigned to the respective queue.

CSR will need to review the order and reach out to customer to acquire a new payment method through the call center application.

## Assumptions

|  |  |
| --- | --- |
| **S.No** | * Assumptions |
|  | Sensitive payment information will be tokenized prior to sending the order to IBM OMS. |
|  | OMS will send the tokenized payment information to the payment service during settlement call. |

## Process Flow Diagram



# Impacted Components

## ‘Payment\_Collection’ Transaction

* This transaction will be set to run on ‘**FanPaymentAgentServer’**. This transaction will create charge requests for all the charges on a particular order.

## ‘Payment\_Execution’ Transaction

* This transaction will be set to run on ‘**FanPaymentAgentServer**’. This transaction will execute and settle all the open charges of a particular order.
* ‘**YFSCollectionCreditCardUE’** on this transaction will be implemented to settle the charges on credit/debit card payment methods.
* ‘**YFSCollectionStoredValueCardUE’** on this transaction will be implemented to settle the charges on Gift card and Gift certificate payment methods.
* ‘**YFSCollectionOthersUE’** on this transaction will be implemented to settle the charges on PayPal and Account balance payment methods.
* ‘**COLLECTION\_FAILED**’ event on this transaction is configured to raise alerts whenever a settlement attempt fails.

## ‘YFSCollectionCreditCardUE’ user exit

* This user exit will be implemented as a java class **com.fanatics.sterling.ue.** **FANYFSCollectionCreditCardUEImpl** which is an exact replica of ‘**com.yantra.pca.ycd.ue.YCDCollectionCreditCardUEImpl**’. This will be used for credit/ debit card payment methods.
* This is the OOB code that calls the proxy service ‘**YCD\_ExecuteCollectionCreditCard\_Proxy\_1.0**’
* This service will in turn invoke ‘**FanaticsSettlementInvocationService**’ that contains the REST call to external payment service.

## ‘YFSCollectionStoredValueCardUE’ user exit

* This user exit will be implemented as a java class **com.fanatics.sterling.ue.** **FANYFSCollectionSVCUEImpl** which is an exact replica of ‘**com.yantra.pca.ycd.ue.YCDCollectionSVCImpl**’. This will be used for gift cards/gift certificate payment methods.
* This is the OOB code that calls the proxy service ‘**YCD\_ExecuteCollectionSVC\_Proxy\_1.0’**
* This service will in turn invoke ‘**FanaticsSettlementInvocationService**’ that contains the REST call to external payment service.

## ‘YFSCollectionOthersUE’ user exit

* OOB class ‘**com.yantra.pca.ycd.ue.YCDCollectionOtherUEImpl**’ is extended and a new java class ‘**com.fanatics.sterling.ue.FANCollectionOthersImpl**’ is created. This class will call the proxy service ‘**YCD\_ExecuteCollectionOTHERS\_Proxy\_1.0’** for PayPal and Account Balance Payment Types.
* This user exit will be implemented as the newly created java class.
* This service will in turn invoke ‘**FanaticsSettlementInvocationService**’ that contains the REST call to external payment service.

## FanaticsSettlementInvocationService’

* ‘FanaticsSettlementInvocationService’ is a new reusable service that can be invoked for all the payment methods and used to invoke the REST call to the external payment service.
* This service can also modify the input to be sent to the REST call and also modify the output from REST call (to convert the output to OMS system format if needed).

## ‘COLLECTION\_FAILED’ event on ‘PAYMENT\_EXECUTION’ transaction

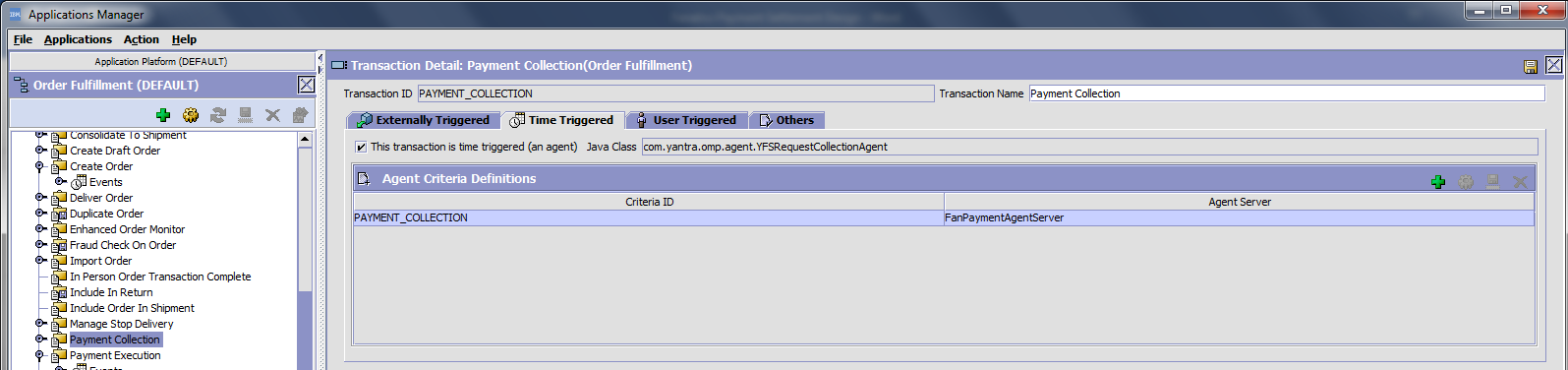
* Event handler will be configured to raise an alert when a payment collection fails for service unavailable. A service ‘**FanaticsTasksOnPaymentServiceUnavailable’** will be called which will call OOB API changeOrder to stamp the next CollectionDate. This Alert will be created in ‘Payment Failures’ queue which will be assigned to relevant users.
* If there is a hard decline on payment ‘**FanaticsTasksOnPaymentHardDecline’** payment will be called which will put a **PAYMENT\_CAPTURE\_HOLD** on the order.

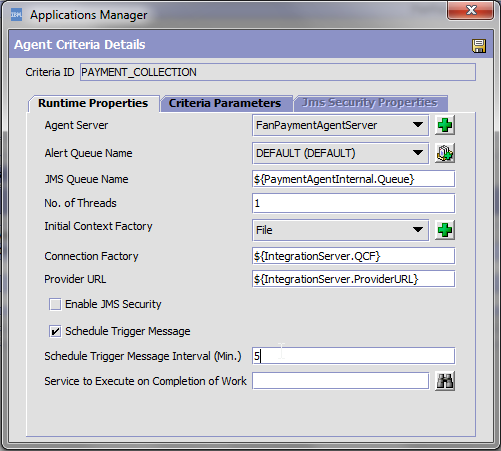
## ‘CHECK\_PAYMENT\_ALERTS.0001.ex’ time triggered transaction (agent server)

* This will call a java class com.fanatics.sterling.agent.FANCheckForPaymentAlerts which will extend YCPBaseAgent().
* In the getJobs() section OOB API **getExceptionList** is called with ExceptionType=**’PAYMENT\_SRVC\_UNAVAIL’**. Unique orders from that list are taken and checked if payment is settled for them. All exceptions for settled orders are resolved.

# Configuration Details

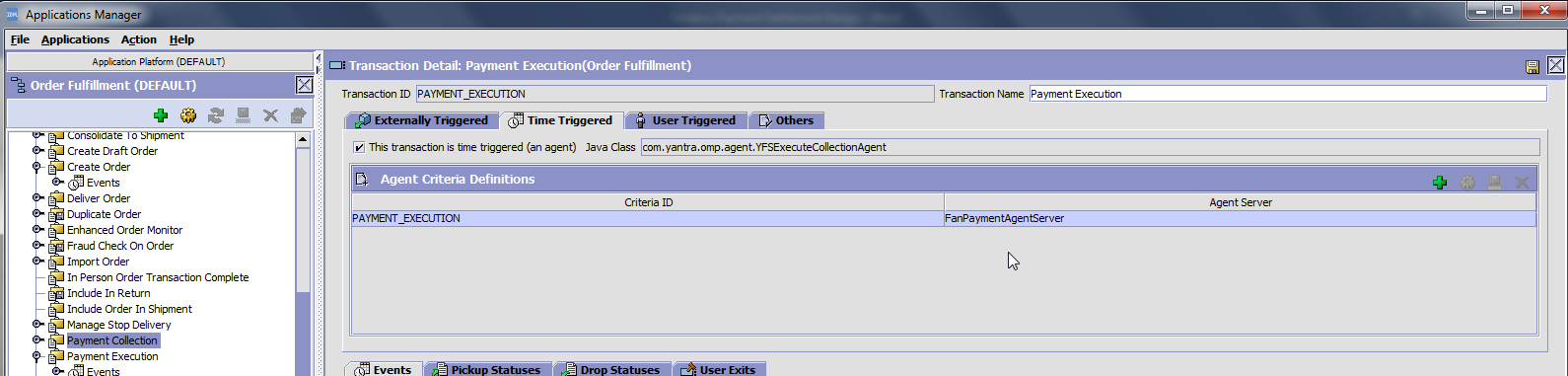
## Transaction: PAYMENT\_COLLECTION

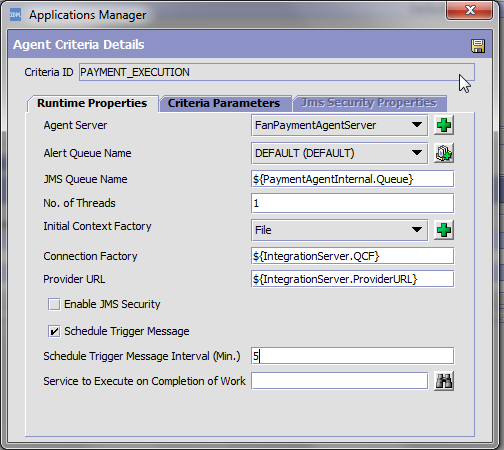




|  |  |
| --- | --- |
| **Name** | **Value** |
| Transaction ID | PAYMENT\_COLLECTION |
| AgentServer Name | FanPaymentAgentServer |
| Schedule Trigger Interval | 5 |

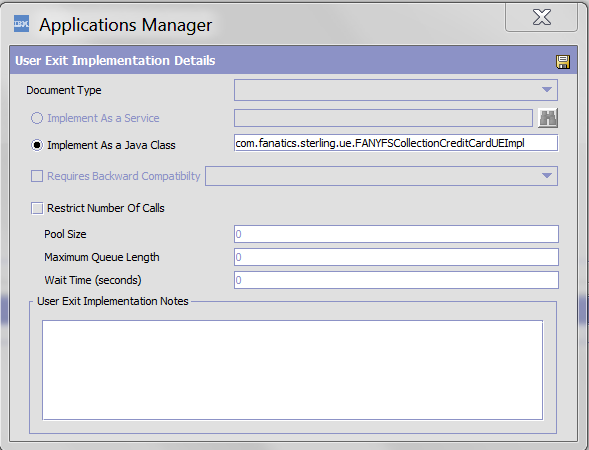
## Transaction: PAYMENT\_EXECUTION





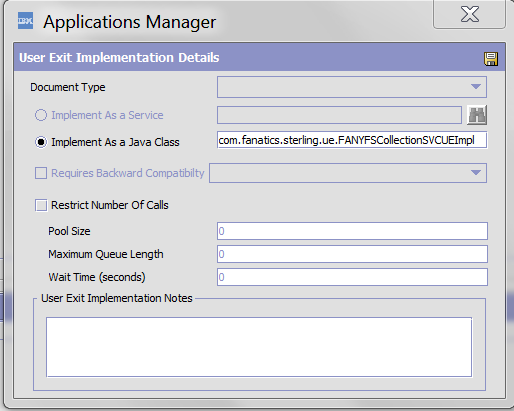
|  |  |
| --- | --- |
| **Name** | **Value** |
| Transaction ID | PAYMENT\_EXECUTION |
| AgentServer Name | FanPaymentAgentServer |
| Schedule Trigger Interval | 5 |

## User Exit: ‘YFSCollectionCreditCardUE’



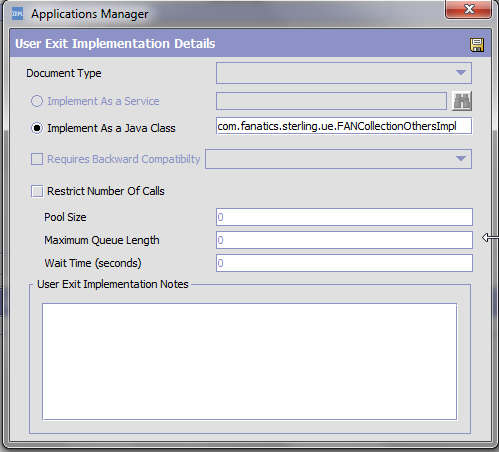
|  |  |
| --- | --- |
| **Name** | **Value** |
| Implement As a Java Class | com.fanatics.sterling.ue.FANYFSCollectionCreditCardUEImpl |

## User Exit: ‘YFSCollectionStoredValueCardUE’



|  |  |
| --- | --- |
| **Name** | **Value** |
| Implement As a Java Class | com.fanatics.sterling.ue.FANYFSCollectionSVCImpl |

## User Exit: ‘YFSCollectionOthersUE’

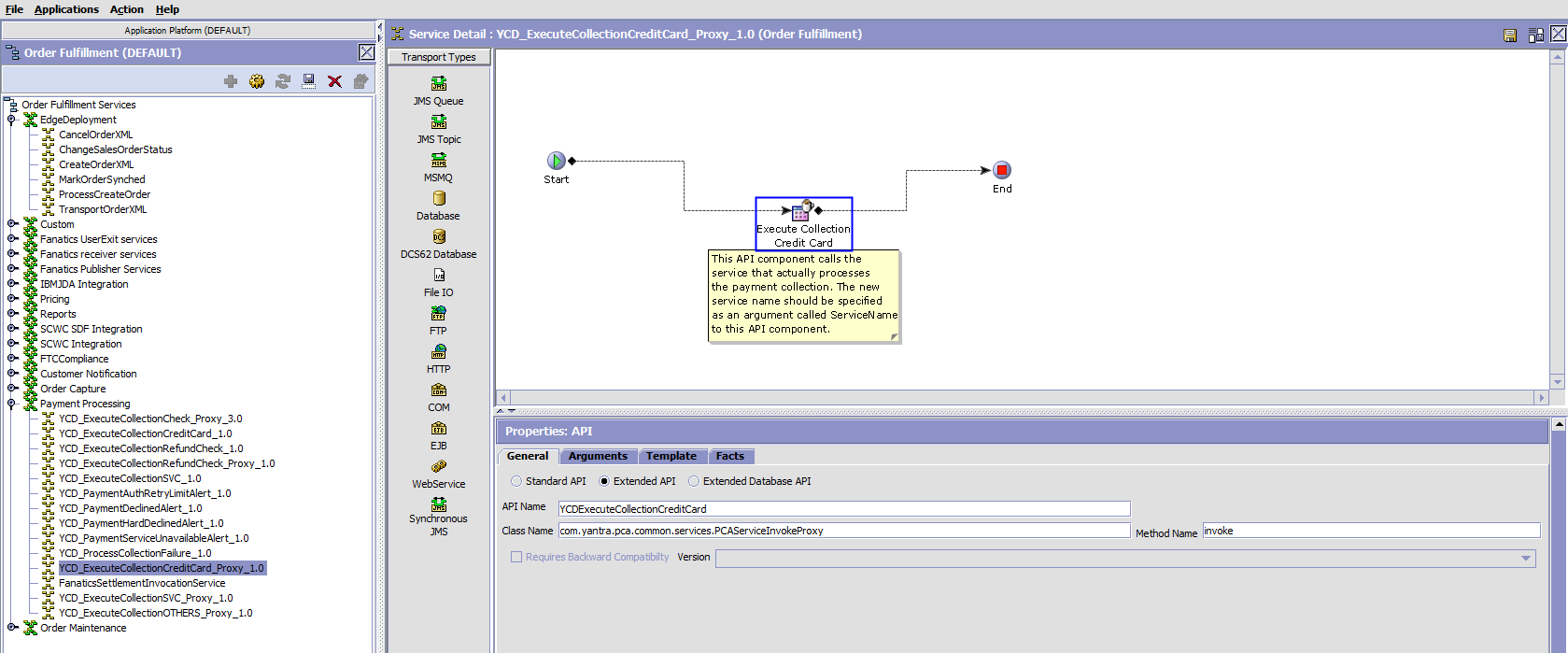


|  |  |
| --- | --- |
| **Name** | **Value** |
| Implement As a Java Class | com.fanatics.sterling.ue.FANCollectionOthersImpl |

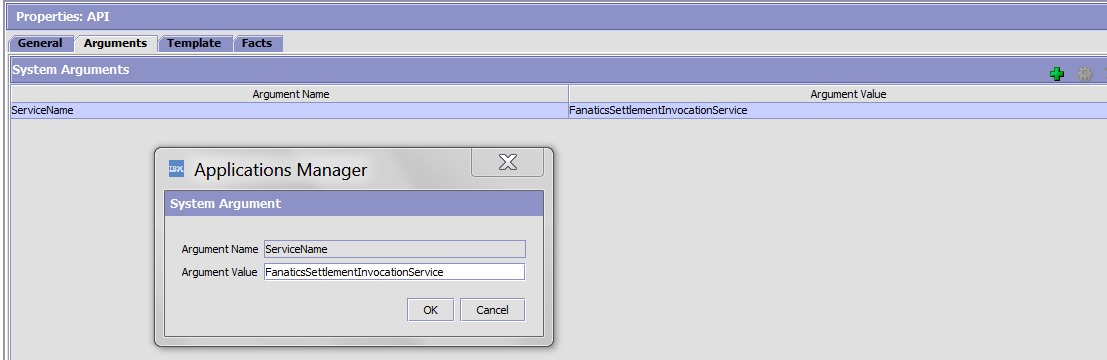
## Service: ‘YCD\_ExecuteCollectionCreditCard\_Proxy\_1.0’

Navigate to Application Platform 🡪 Process Modelling 🡪 Order Fulfillment 🡪 Services 🡪 Payment Processing.

Modify the existing service.



In the parameters tab, configure the service name that invokes the REST webservice call.

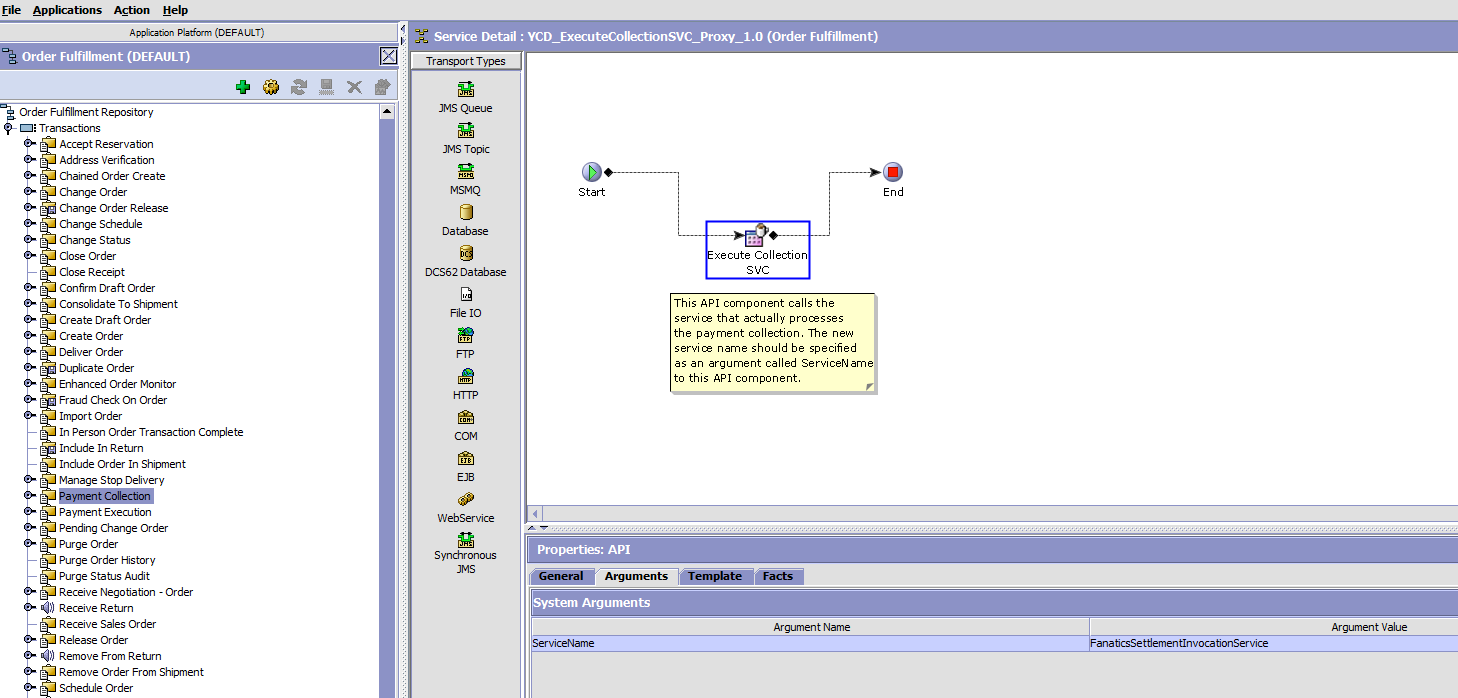


|  |  |
| --- | --- |
| **Name** | **Value** |
| Service Name | FanaticsSettlementInvocationService |

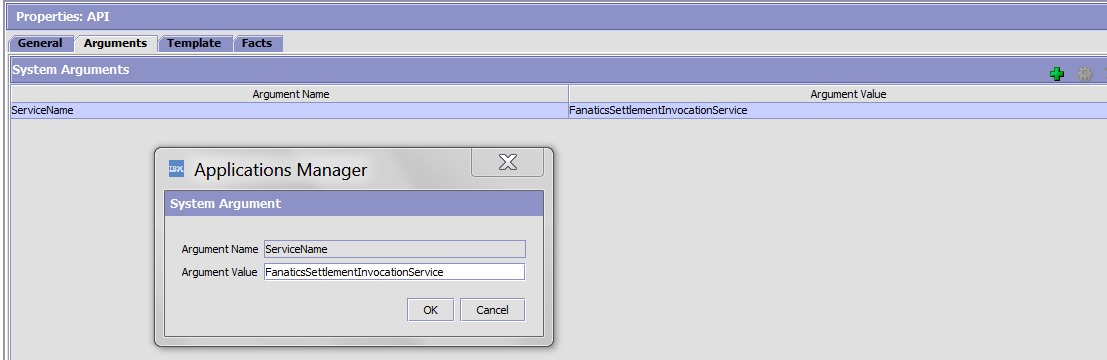
## Service: ‘YCD\_ExecuteCollectionSVC\_Proxy\_1.0

Navigate to Application Platform 🡪 Process Modelling 🡪 Order Fulfillment 🡪 Services 🡪 Payment Processing.

Modify the existing service.



In the parameters tab, configure the service name that invokes the REST webservice call.



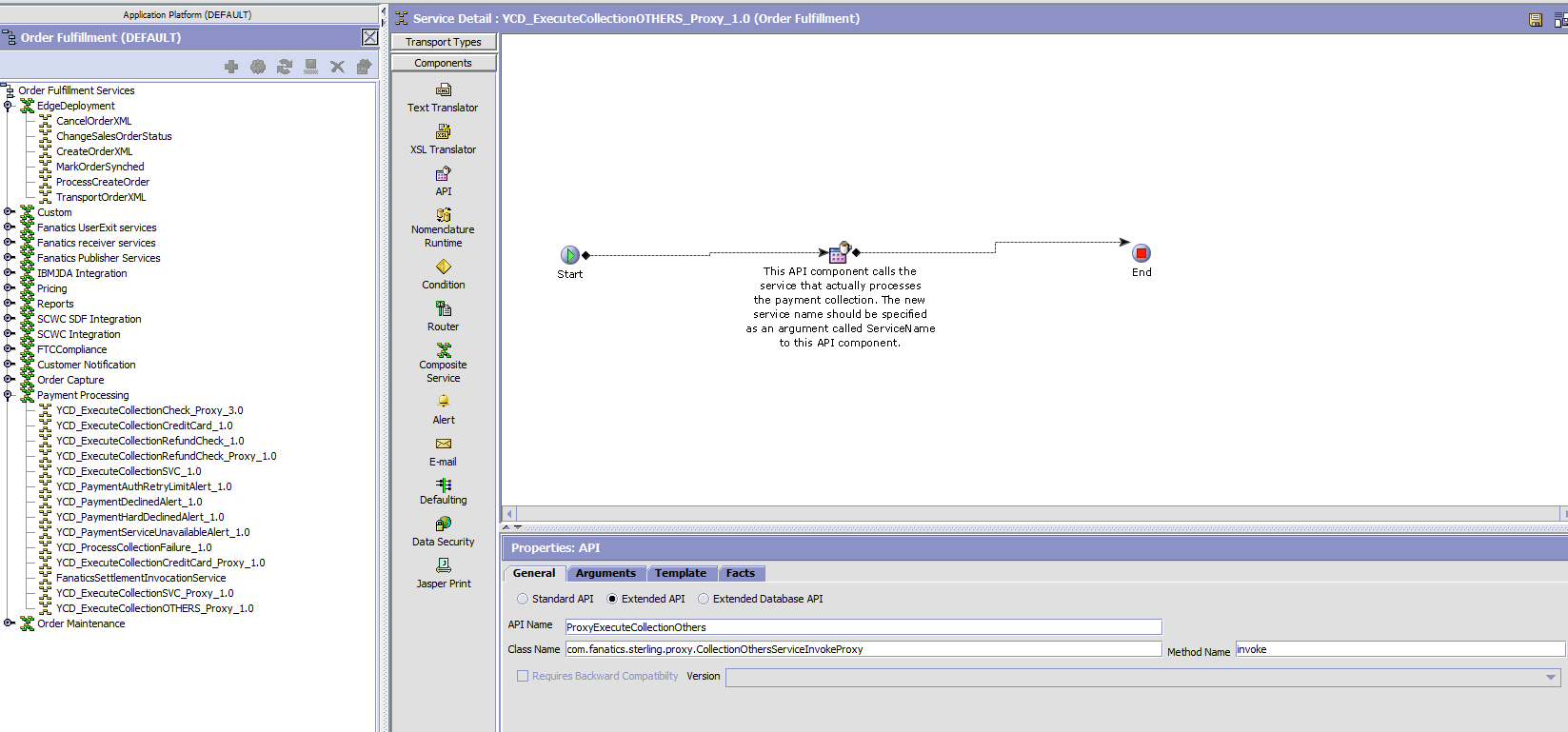
|  |  |
| --- | --- |
| **Name** | **Value** |
| Service Name | FanaticsSettlementInvocationService |

## Service: ‘YCD\_ExecuteCollectionOthers\_Proxy\_1.0’

Navigate to Application Platform 🡪 Process Modelling 🡪 Order Fulfillment 🡪 Services 🡪 Payment Processing.

Create a new service:

|  |  |
| --- | --- |
| **Name** | **Value** |
| Process Type | Order Fulfillment |
| Service Group | Payment Processing |
| Service Name | YCD\_ExecuteCollectionOthers\_Proxy\_1.0 |
| Service Type | Synchronous |



### API Component

#### General tab

|  |  |
| --- | --- |
| **Name** | **Value** |
| Extended API | Y |
| API Name | ProxyExecuteCollectionOthers |
| Class Name | com.fanatics.sterling.ue.FANCollectionOthersImpl |
| Method Name | invoke |

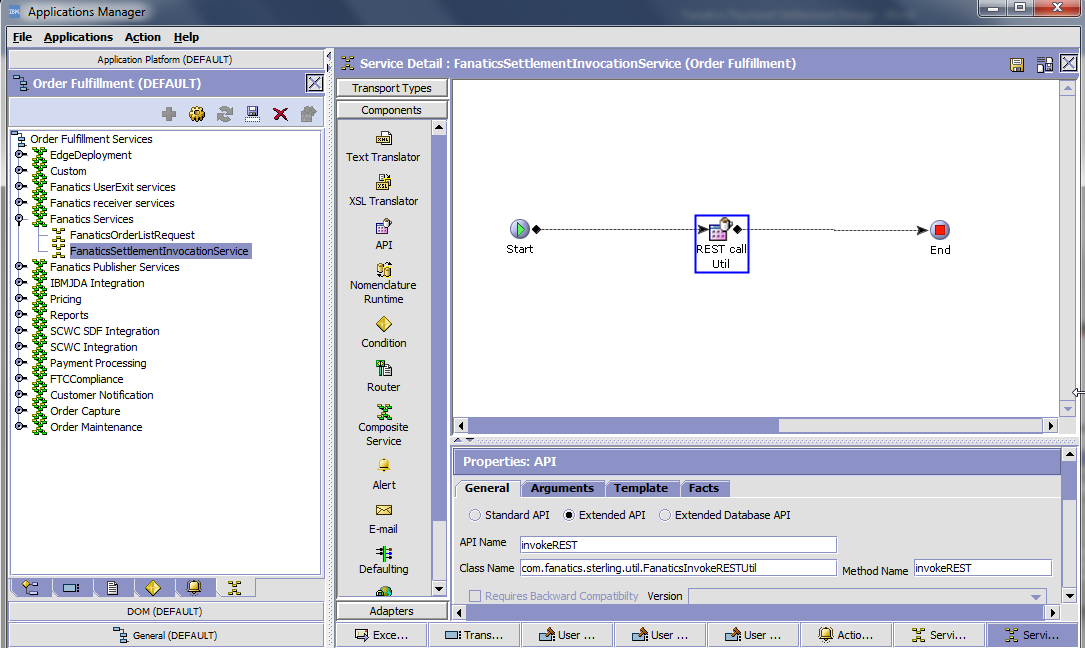
#### Arguments tab

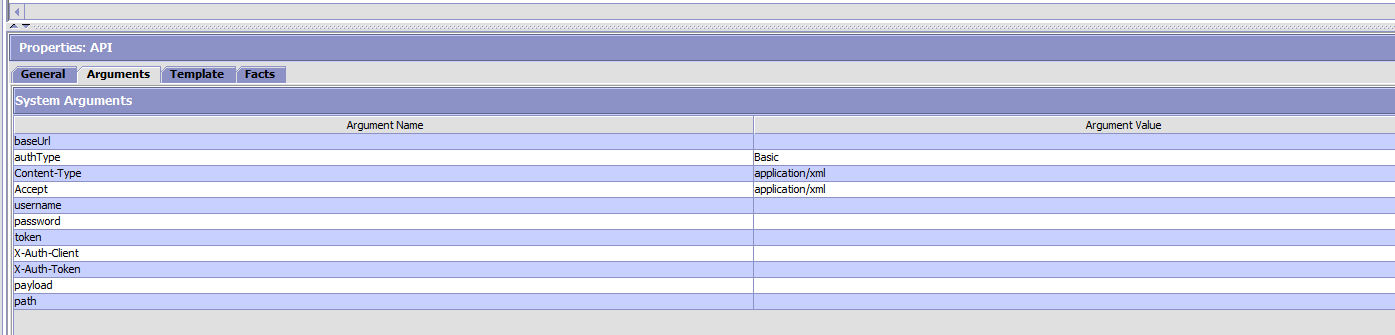
|  |  |
| --- | --- |
| **Name** | **Value** |
| Service Name | FanaticsSettlementInvocationService |

## Service: ‘FanaticsSettlementInvocationService’

The table below provides information on where to look for the Sterling service configuration for this service:

|  |  |
| --- | --- |
| **Name** | **Value** |
| Process Type | Order Fulfillment |
| Service Group | Fanatics Services |
| Service Name | FanaticsSettlementInvocationService |
| Service Type | Synchronous |





### REST call Utility API Component

The first component in the FanaticsSettlementInvocationService is the custom API utility that invokes the REST call.

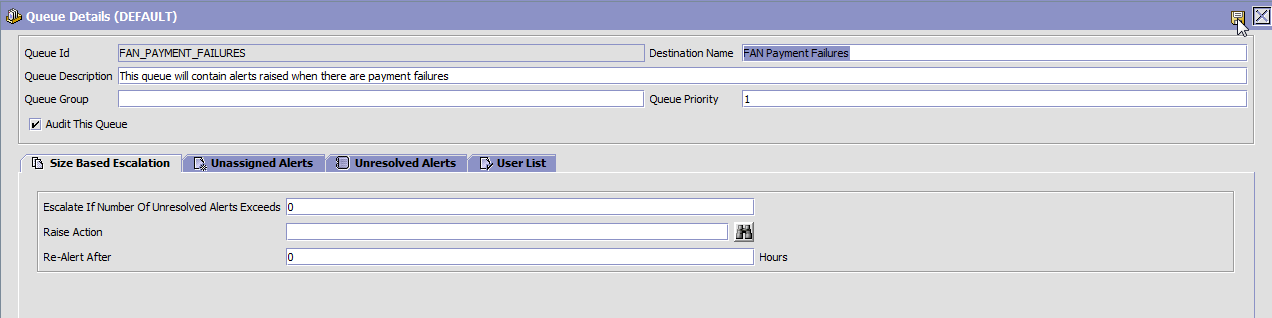
#### General tab

|  |  |
| --- | --- |
| **Name** | **Value** |
| Extended API | Y |
| API Name | invokeREST |
| Class Name | com.fanatics.sterling.util.FanaticsInvokeRESTUtil |
| Method Name | invokeREST |

#### Arguments tab

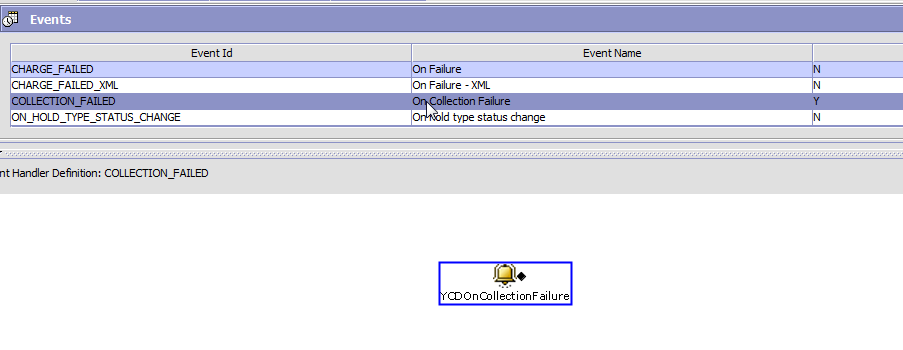
|  |  |
| --- | --- |
| **Name** | **Value** |
| baseUrl | TBD |
| path | TBD |
| authType | Basic |
| Content-Type | application/xml |
| Accept | application/xml |
| username | TBD |
| password | TBD |
| token | TBD |
| X-Auth-Client | TBD |
| X-Auth-Token | TBD |
| payload | TBD |

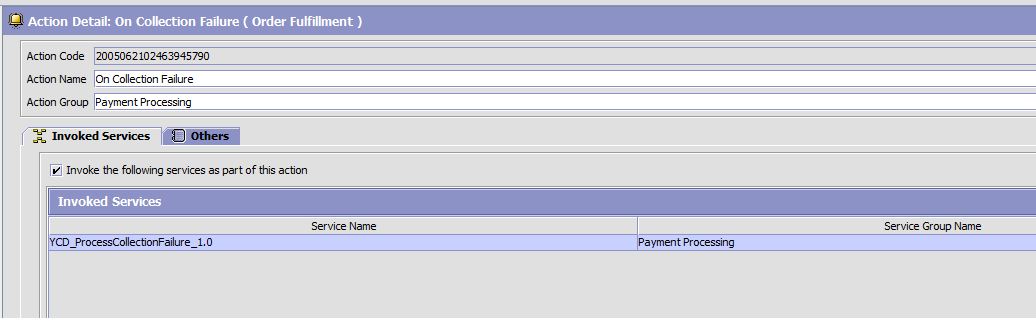
## Payment Alert Queue Configuration

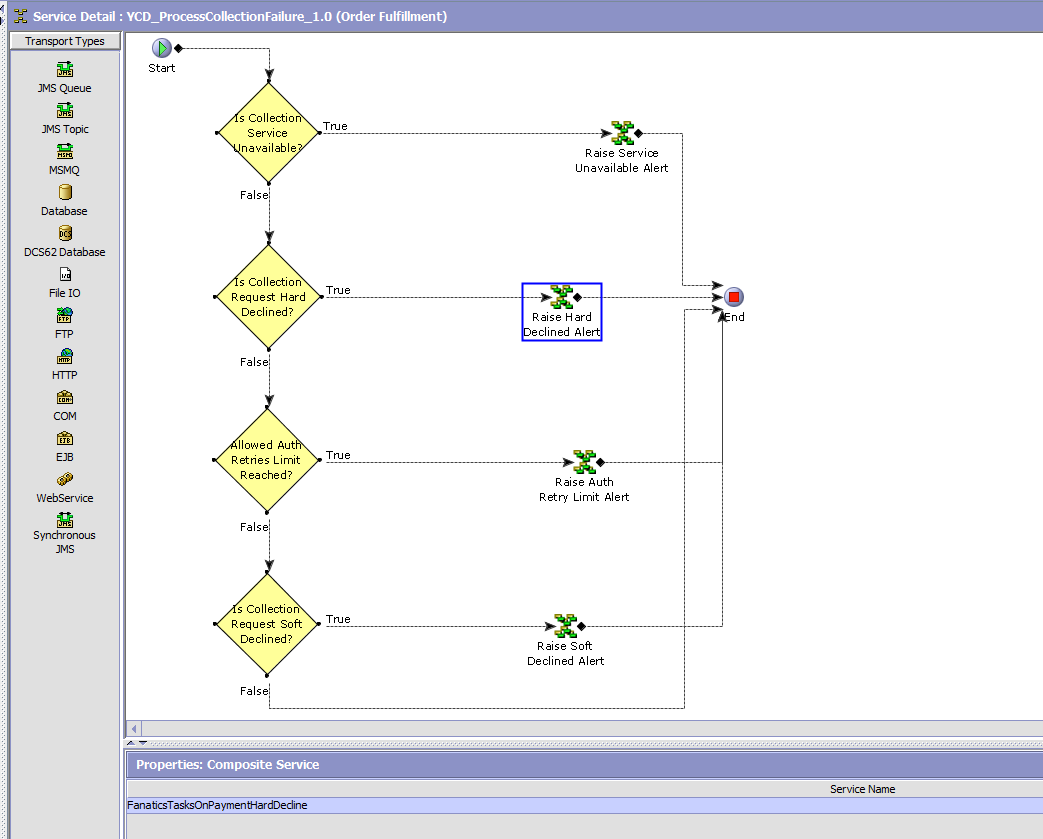
Create a new queue under Application Platform 🡪 Queue Management  


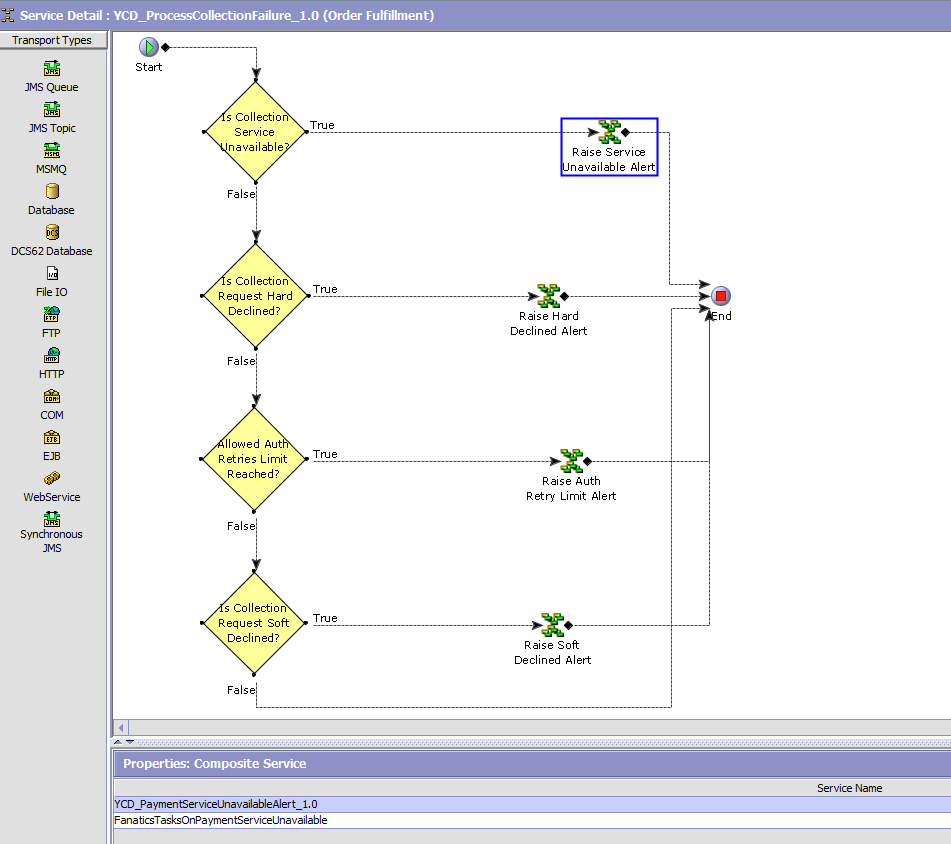
|  |  |
| --- | --- |
| **Name** | **Value** |
| Queue Id | FAN\_PAYMENT\_FAILURES |
| Queue Description | This queue will contain alerts raised when there are payment failures |
| Destination Name | FAN Payment Failures |

## Event: ‘COLLECTION\_FAILED’





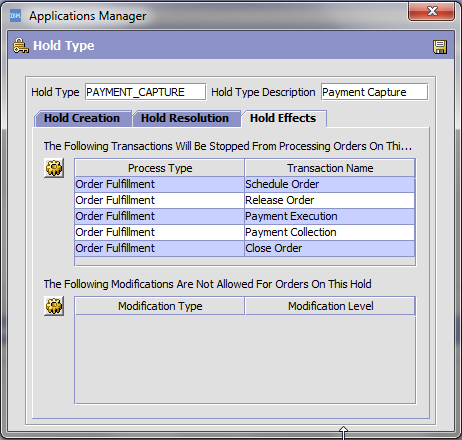
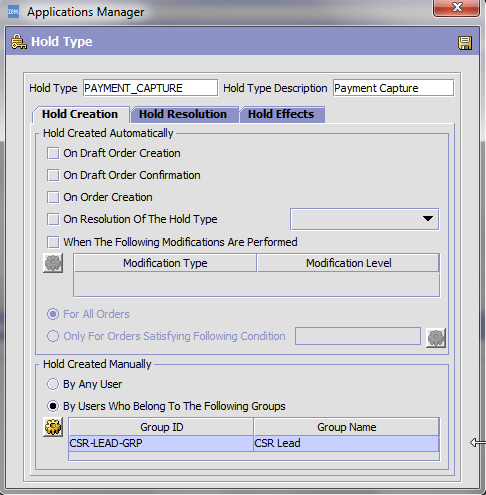




## Hold: PAYMENT\_CAPTURE\_HOLD

Navigate to Configurator 🡪 Applications 🡪 Distributed Order Management 🡪 Document Specific 🡪 Sales Order 🡪 Fulfillment 🡪 Hold Types, create a new Order Hold Type

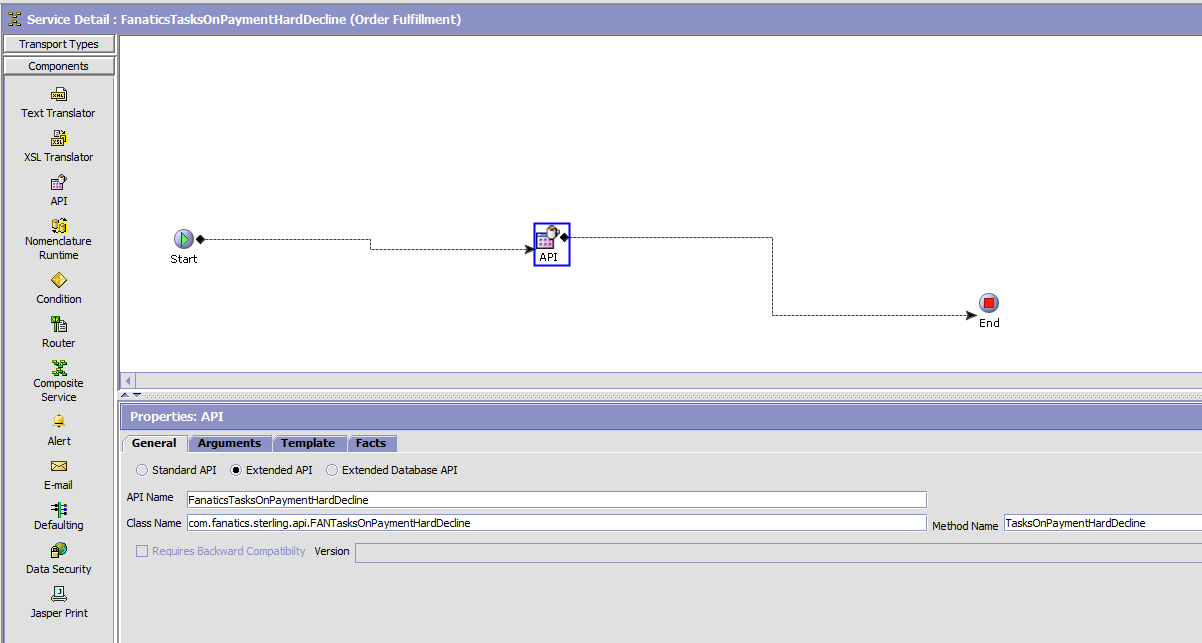
|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Hold Type | PAYMENT\_CAPTURE\_HOLD |
| Hold Type Description | Payment Capture Hold |
| By Users Who Belong to the following user groups | CSR\_LEAD |
| Hold Effect | Schedule Order, Release Order, Payment Execution, Payment Collection, Close Order |



## Service: FanaticsTasksOnPaymentHardDecline

Create a new service:

|  |  |
| --- | --- |
| **Name** | **Value** |
| Process Type | Order Fulfillment |
| Service Group | Payment Processing |
| Service Name | FanaticsTasksOnPaymentHardDecline |
| Service Type | Synchronous |



#### General Tab

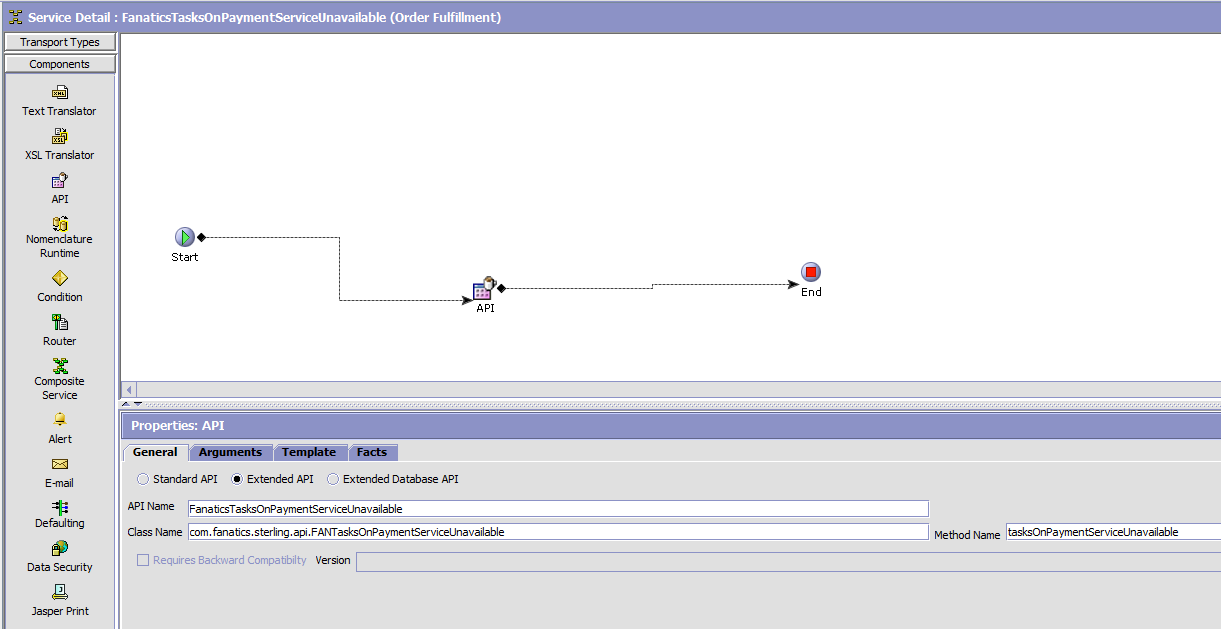
|  |  |
| --- | --- |
| **Name** | **Value** |
| API Name | FanaticsTasksOnPaymentHardDecline |
| Class Name | com.fanatics.staerling.api.FanaticsTasksOnPaymentHardDecline |
| Method Name | TasksOnPaymentHardDecline |

This service will create a CAPTURE\_HOLD on the order by calling changeOrder OOB API. In future, this class can be utilized to perform more tasks if needed.

## Service: FanaticsTasksOnPaymentServiceUnavailable

Create a new service:

|  |  |
| --- | --- |
| **Name** | **Value** |
| Process Type | Order Fulfillment |
| Service Group | Payment Processing |
| Service Name | FanaticsTasksOnPaymentServiceUnavialable |
| Service Type | Synchronous |



#### General Tab

|  |  |
| --- | --- |
| **Name** | **Value** |
| API Name | FanaticsTasksOnPaymentServiceUnavailable |
| Class Name | com.fanatics.sterling.api.FanaticsTasksOnPaymentServiceUnavailable |
| Method Name | TasksOnPaymentServiceUnavailable |



#### Arguments Tab

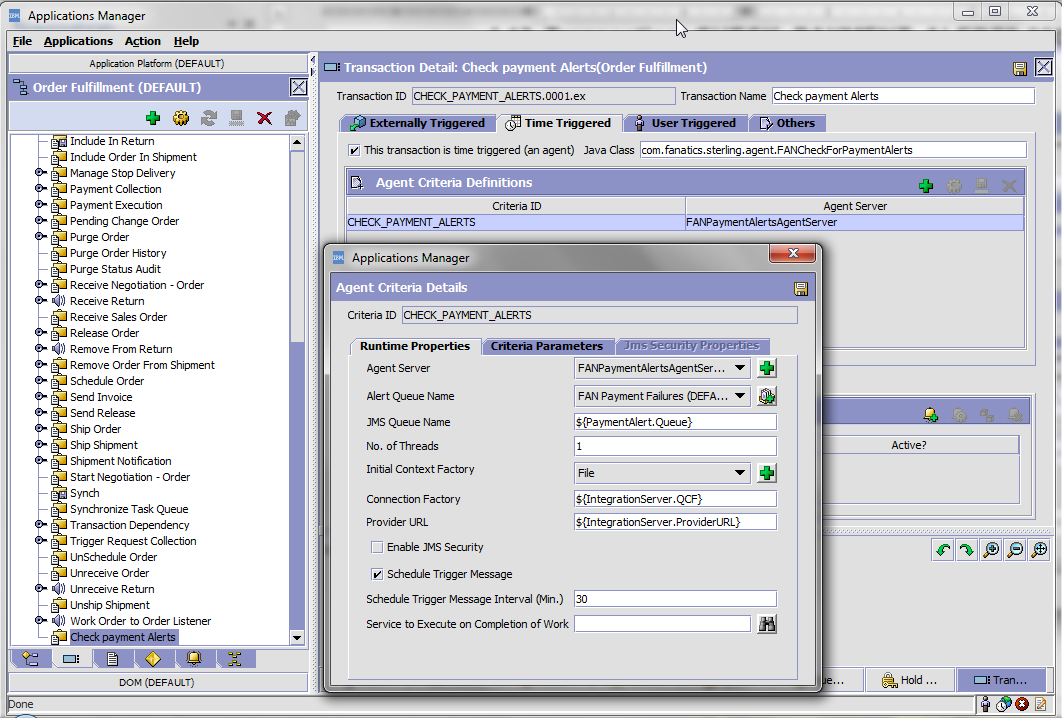
|  |  |
| --- | --- |
| **Name** | **Value** |
| RetryIntervalMinutes | 30 |

This service will call OOB API changeOrder to stamp the next CollectionDate at the level Order/PaymentMethods/PaymentMethod/PaymentDetails/@CollectionDate.

* New CollectionDate = sysdate + RetryIntervalMinutes.

## Transaction: CHECK\_PAYMENT\_ALERTS.0001.ex

Create a new transaction in ‘OrderFulfillment 🡪 Transactions’

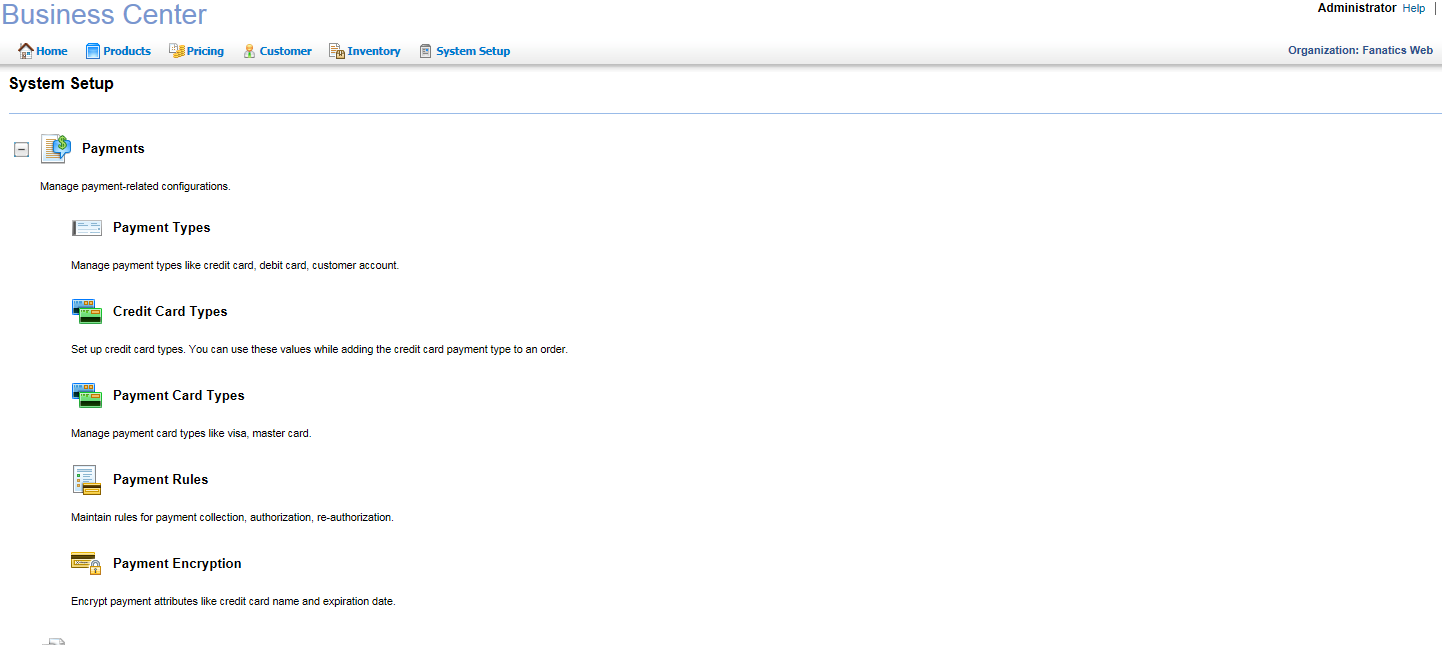


|  |  |
| --- | --- |
| **Name** | **Value** |
| Transaction ID | CHECK\_PAYMENT\_ALERTS |
| Transaction Name | Check Payment Alerts |
| Time Triggered | Checked |
| Java Class | com.fanatics.sterling.agent.FANCheckForPaymentAlerts |
| Criteria ID | CHECK\_PAYMENT\_ALERTS |
| AgentServer Name | FANPaymentAlertsAgentServer |
| Alert Queue Name | FAN Payment Failures |
| JMS Queue Name | ${PaymentAlert.Queue} |
| No. of Threads | 1 |
| Initial Context Factory | File |
| Provider URL | ${AgentServer.ProviderURL} |
| Connection Factory | ${AgentServer.QCF} |
| Schedule Trigger Message Interval | 30 |

## Payment Method configuration

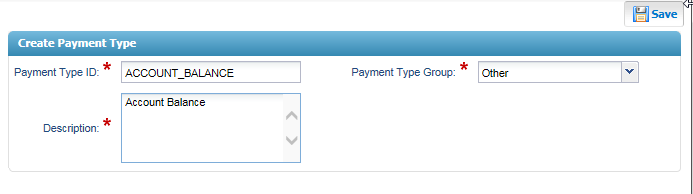
Navigate to Sterling Business Center -> Load ‘FANATICS\_WEB’ 🡪 System Setup Tab

-> Payments 🡪 Payment Types.



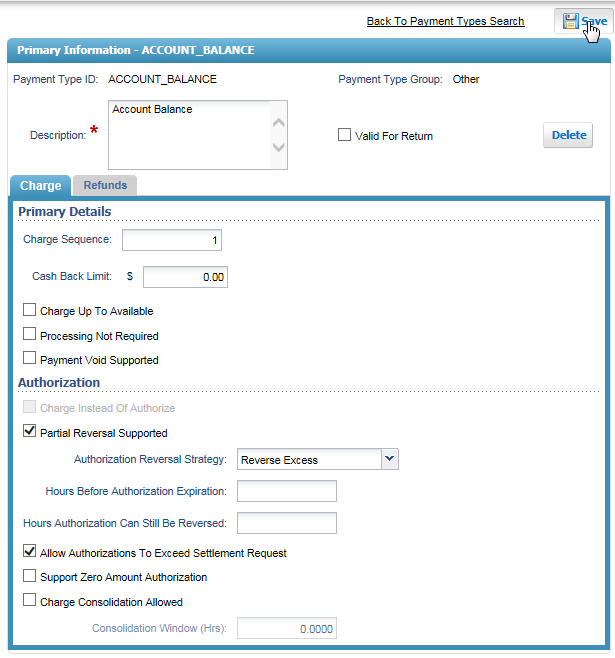
### Payment Types:

* **ACCOUNT BALANCE**
  + Navigate to Payment Type -> Create Payment Type



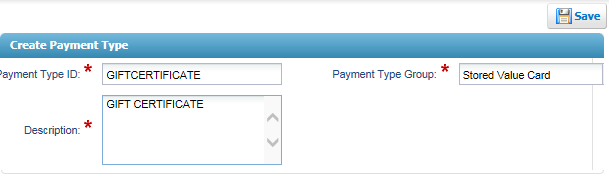
|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment Type ID | ACCOUNT\_BALANCE |
| Payment Type Group | Other |
| Description | Account Balance |

* + Save
  + Configure the payment type parameters as follows:



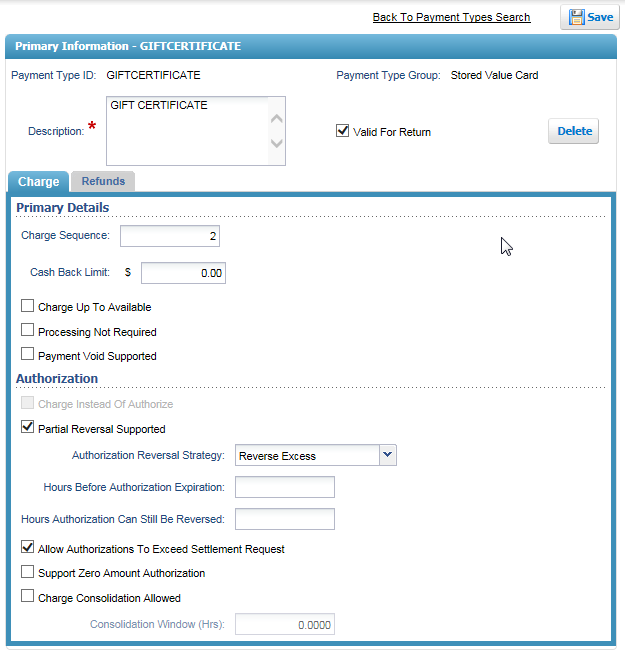
|  |  |
| --- | --- |
| **Name** | **Value** |
| Valid For return | Unchecked |
| Charge Sequence | 1 |
| Partial Reversal Supported | Checked |
| Authorization Reversal Strategy | Reverse Excess |
| Allow Authorization To Exceed Settlement Request | Checked. |

* + Save
* **GIFT CERTIFICATE**
  + Navigate to Payment Type -> Create Payment Type



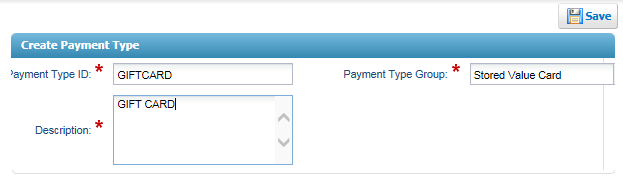
|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment Type ID | GIFTCERTIFICATE |
| Payment Type Group | Stored value Card |
| Description | GIFT CERTIFICATE |

* + Save
  + Configure the payment type parameters as follows:



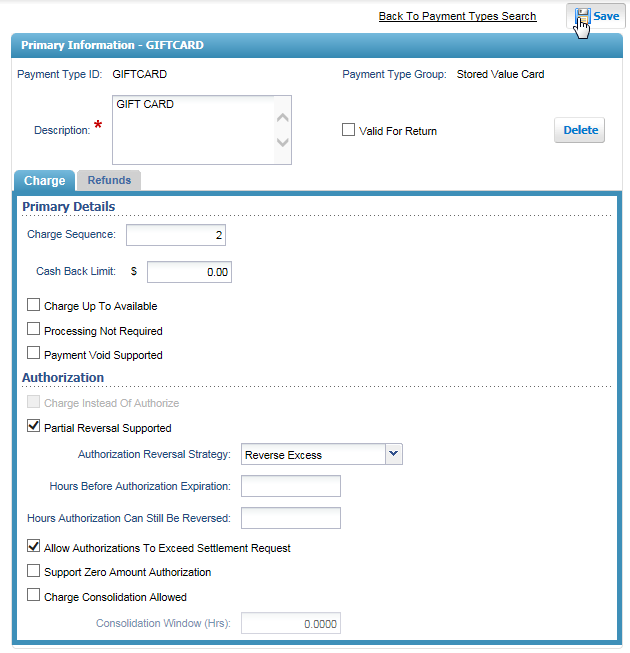
|  |  |
| --- | --- |
| **Name** | **Value** |
| Valid For return | Checked |
| Charge Sequence | 2 |
| Partial Reversal Supported | Checked |
| Authorization Reversal Strategy | Reverse Excess |
| Allow Authorization To Exceed Settlement Request | Checked |

* + Save
* **GIFT\_CARD**
  + Navigate to Payment Type -> Create Payment Type



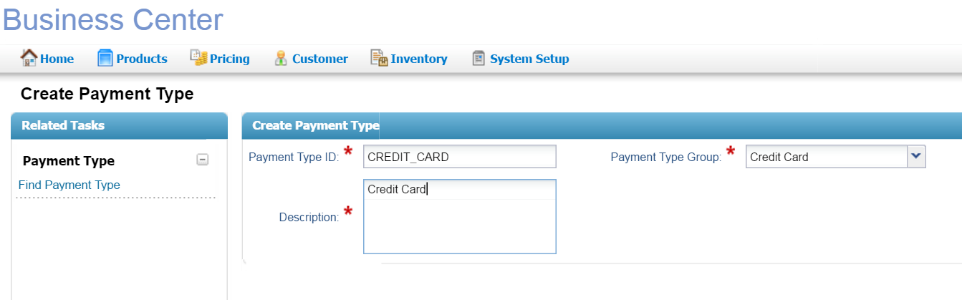
|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment Type ID | GIFTCARD |
| Payment Type Group | Stored value Card |
| Description | GIFT CARD |

* + Save
  + Configure the payment type parameters as follows:



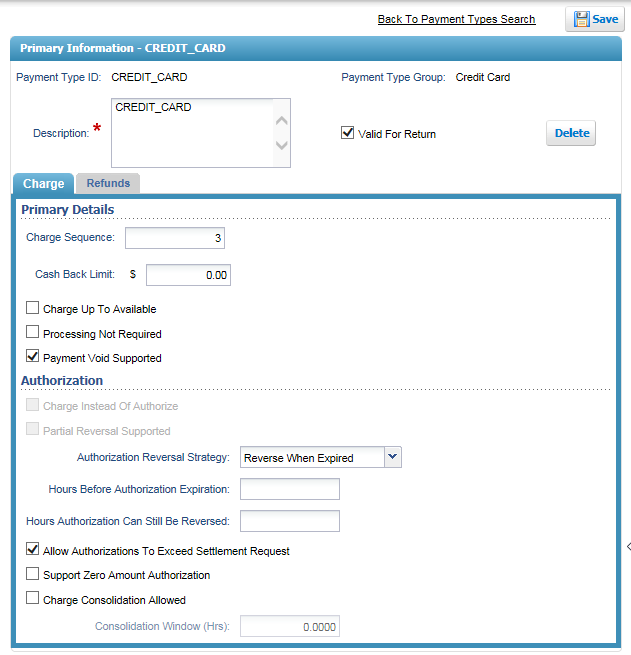
|  |  |
| --- | --- |
| **Name** | **Value** |
| Charge Sequence | 2 |
| Partial Reversal Supported | Checked |
| Authorization Reversal Strategy | Reverse Excess |
| Allow Authorization To Exceed Settlement Request | Checked |

* + Save
* **CREDIT\_CARD**
  + Navigate to Payment Type -> Create Payment Type



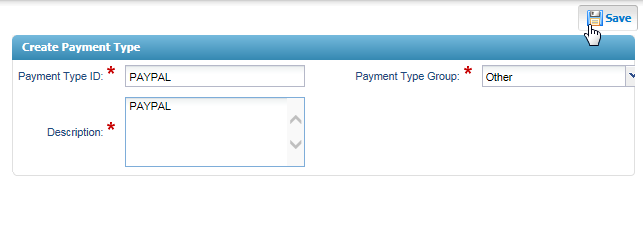
|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment Type ID | CREDIT\_CARD |
| Payment Type Group | Credit Card |
| Description | Credit Card |

* + Save
  + Configure the payment type parameters as follows:



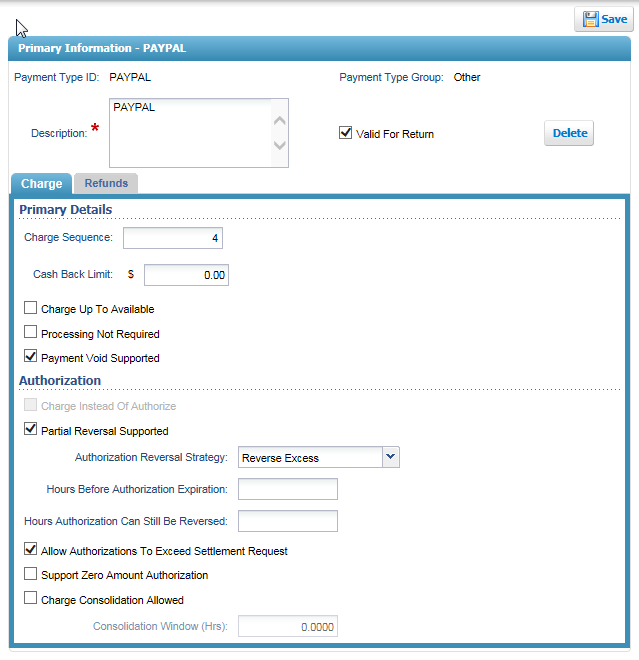
|  |  |
| --- | --- |
| **Name** | **Value** |
| Valid For return | Checked |
| Charge Sequence | 3 |
| Payment Void Supported | Checked |
| Authorization Reversal Strategy | Reverse When Expired |
| Allow Authorization To Exceed Settlement Request | Checked |

* + Save
* **PAYPAL**
  + Navigate to Payment Type -> Create Payment Type



|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment Type ID | PAYPAL |
| Payment Type Group | Other |
| Description | PAYPAL |

* + Save
  + Configure the payment type parameters as follows:



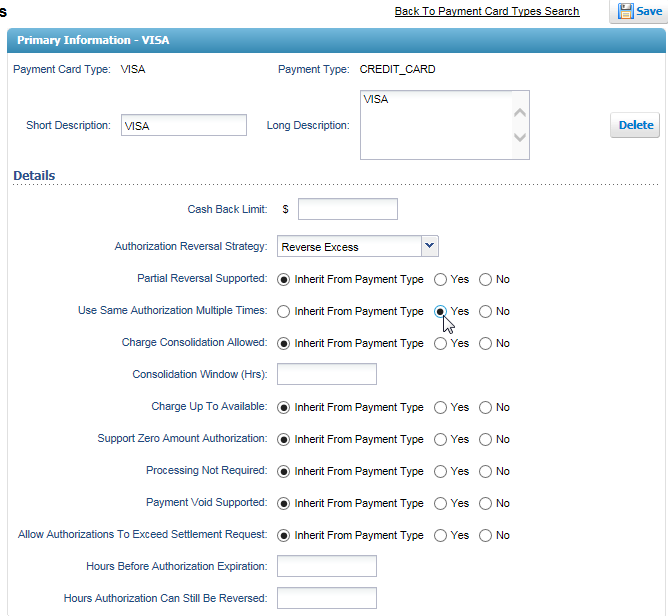
|  |  |
| --- | --- |
| **Name** | **Value** |
| Valid For return | Checked |
| Charge Sequence | 4 |
| Payment Void Supported | Checked |
| Partial Reversal Supported | Checked |
| Authorization Reversal Strategy | Reverse Excess |
| Allow Authorization To Exceed Settlement Request | Checked |

* + Save

### Payment Card Types

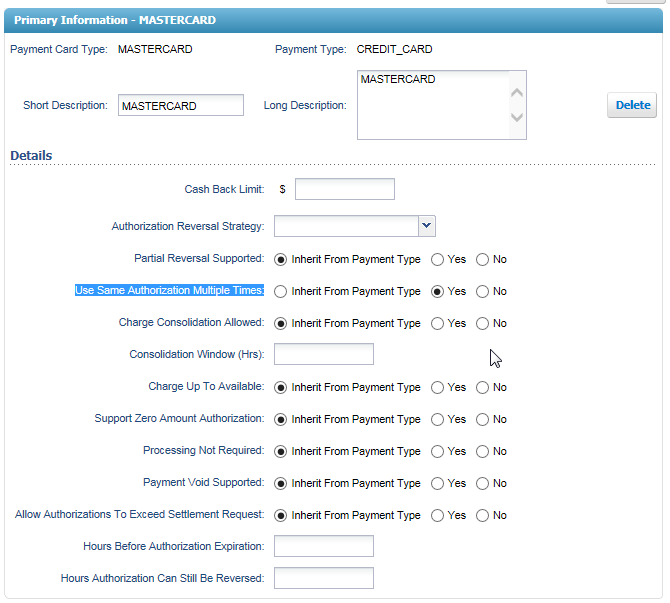
Navigate to Payments -> Create Payment Card Type.

Create the Payment Card Type and save. Configure the parameters as follows:

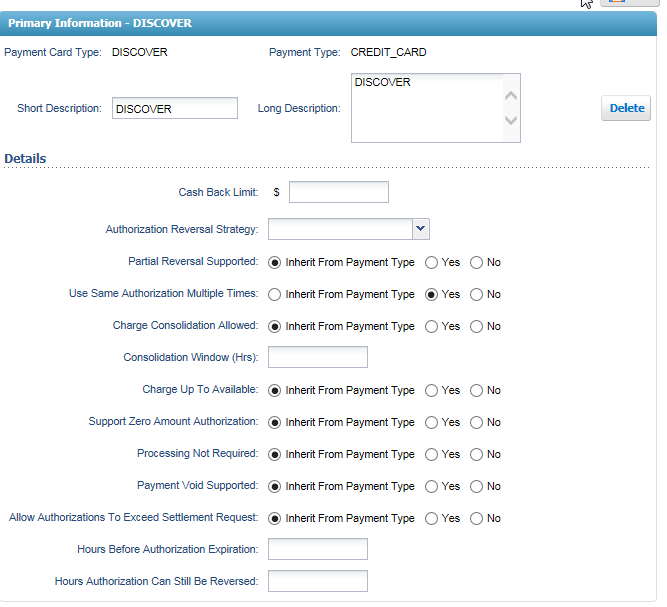


|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment card type | VISA |
| Payment Type | CREDIT\_CARD |
| Short Description | VISA |
| Long Description | VISA |
| Use Same Authorization Multiple Times | Yes |

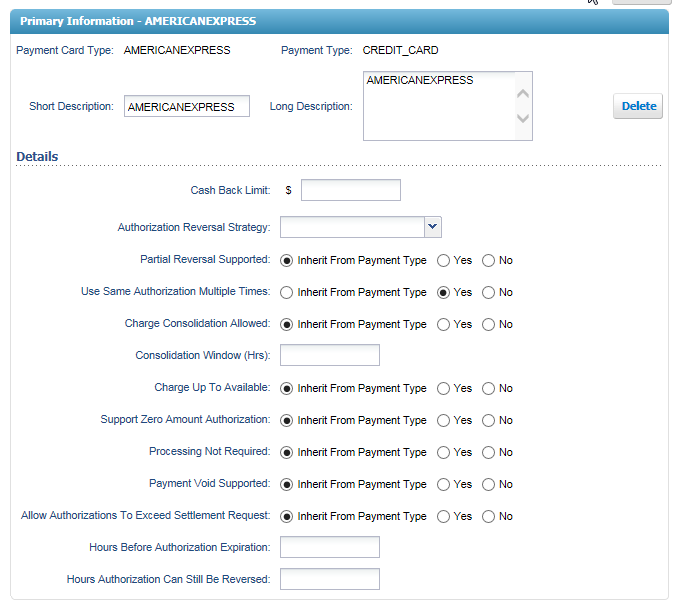
Create other payment card types as follows:



|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment card type | MASTERCARD |
| Payment Type | CREDIT\_CARD |
| Short Description | MASTERCARD |
| Long Description | MASTERCARD |
| Use Same Authorization Multiple Times | Yes |



|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment card type | DISCOVER |
| Payment Type | CREDIT\_CARD |
| Short Description | DISCOVER |
| Long Description | DISCOVER |
| Use Same Authorization Multiple Times | Yes |



|  |  |
| --- | --- |
| **Name** | **Value** |
| Payment card type | AMERICANEXPRESS |
| Payment Type | CREDIT\_CARD |
| Short Description | AMERICANEXPRESS |
| Long Description | AMERICANEXPRESS |
| Use Same Authorization Multiple Times | Yes |

## Post Conditions

Upon successful execution, following situation will be true:

* The Invoices on the order will be Collected.

## Test Cases

