



Taking a pro-active fraud detection/prevention approach: A smart way to reduce potential high costs of fraudulent transactions

Abstract

Consider a scenario where a person fills out a credit card authorization form with all the information needed to place a valid transaction. A cyber thief manages to steal the form without leaving any clue to the credit card owner. In the next hour, a fraudulent transaction has been placed on your site. As the credit card information is correct the transaction is approved and the products are being shipped to the specified shipping address. The owner of the credit card finds out about the fraud and tries to cancel the transaction but the goods are already been delivered.

Every year major retail online stores loses thousands of dollars in customer trust and legal battles to settle credit card fraud disputes. One way to avoid this is to have a fraud check before the payment has been approved and deposited. There are some third party companies that use risk management and analysis tools to predict and prevent payment card fraud. In this case study we will talk about how to integrate IBM WebSphere Commerce with Retail Decisions Fraud check service to protect the transaction on your ecommerce site.

Retail Decisions (ReD), with an experience of over 20 years, is recognized as a leader in fraud prevention and payment processing. ReD is a specialist supplier to the payments industry worldwide and it has clients from banking and the broader e-commerce , global telecommunications, retail, travel and petroleum, sectors.

With powerful out-of the-box capabilities and easy-to-use business user tools, IBM WebSphere Commerce delivers a proven, flexible solution that scales to meet your business requirements no matter what your industry, size of company, or selling model is. IBM WebSphere Commerce can be integrated with a JAVA API provided by "Retail Decisions" to incorporate payment fraud prevention. ReD is able to predict and prevent payment card fraud, process high volumes of credit card and electronic check transactions and has a platform for a full range of transaction based value-added services.

This document covers the technical and theoretical aspects of the integration between IBM WebSphere Commerce and Retail Decisions Fraud Detection API.





Problem Description

As an eCommerce business you can't completely rely on eCommerce payment processing operators to protect you, the merchant, from the use of unauthorized credit cards, fraud, chargebacks and the subsequent theft of your product and services. These issues are severe and can take several months to materialize and can involve potentially hundreds of dollars of resolution fees, lost products, shipping charges, and materials.

So the problem focus is to avoid the frauds in payment processing by implementing a pre proactive fraud detection and prevention system. On the other hand, creating your own business logic or database to check for fraud can be difficult and time consuming also.

Solution

We suggest taking a pro-active fraud detection/prevention approach to screening transactions before shipping of goods to avoid high costs due to fraudulent transactions. In addition, we also suggest that unless your organizations have prior experience implementing a credit card fraud system it is economical and more reliable to use automated processes from companies that provide Fraud Check. Most of our IBM WebSphere Commerce customers prefer Retail Decisions Fraud prevention system due to its easy compatibility and high reliability.

An automated process to detect the fraudulent transactions is provided by Retail Decisions (ReD). We can integrate a client side API provided by the publisher ReD, to send the order, payment and customer details to the LiveProcessor server. LiveProcessor is the Retail Decisions electronic payment processing solution. LiveProcessor combines a significant number of payment and billing capabilities with the most comprehensive set of security and fraud prevention technologies to provide a reliable, convenient and secure gateway for e-commerce, catalog and call center / IVR payments.

The Java Enterprise API is a LiveProcessor programming interface that enables Java programmers to quickly incorporate payment processing and fraud prevention functionality into Java application, applets, or servlets. The API communicates with LiveProcessor to perform all functions. Designed to run within the retailer's data center, LiveProcessor eliminates gateway transaction fees and provides a secure payment processing application that supports Visa CVV2, MasterCard CVC2 and Amex CID security codes, procurement level 2 and 3 cards, electronic check processing, foreign currency settlement support and multiple bank interfaces in a single product.





The API provides several advanced features including:

- **Secure Connection** – The Java Enterprise API establishes a secure SSLv3 connection to support SSL encrypted transmission of data between the client and LiveProcessor.
- **Encryption** - A separate method is provided to optionally encrypt the account number prior to submission to the API. The encryption method uses RSA public key encryption. If the account number is not encrypted prior to execution, the API will automatically encrypt the number before transmitting the request to LiveProcessor.
- **Load balancing** – the API will distribute processing across multiple LiveProcessor at multiple sites.
- **Fail-Over** – the API will fail-over to an alternate LiveProcessor server (if available) due to network errors or ENETPP errors. In addition, the API will fail-over to backup sites (if available) when necessary.

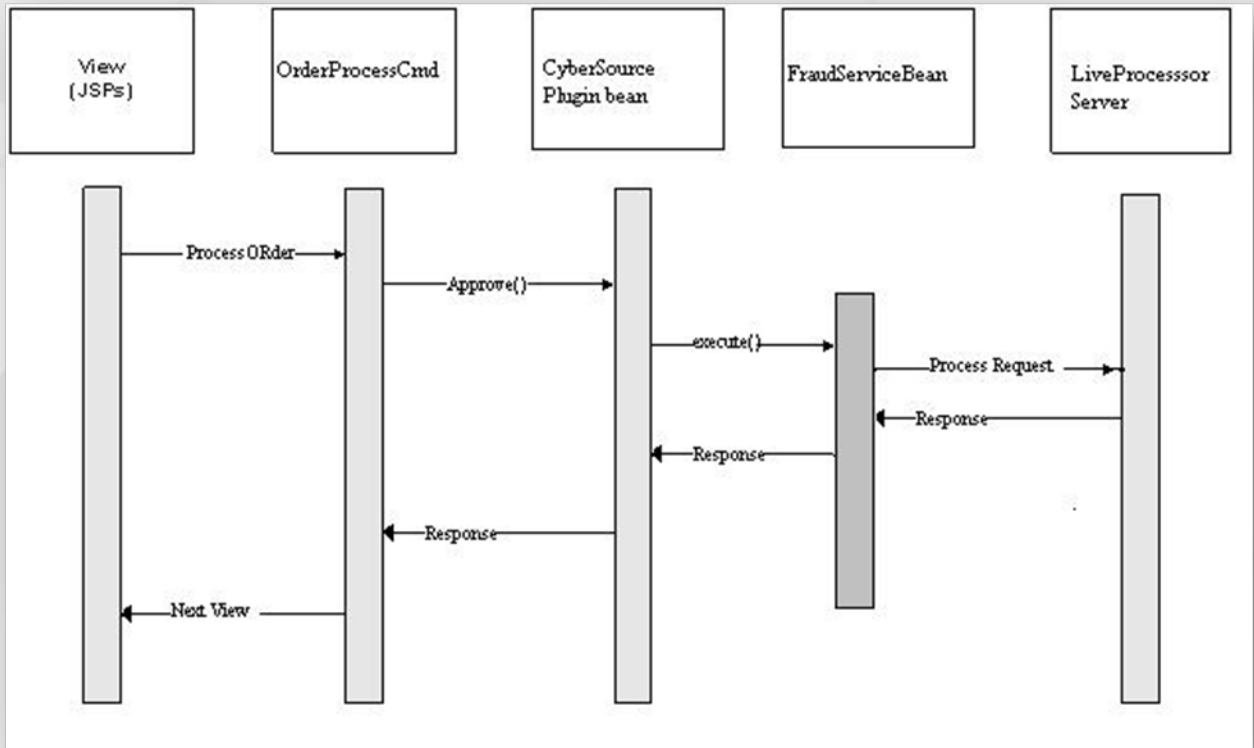
The solution i.e. integration of IBM WebSphere Commerce with Retail Decisions LiveProcessor consists of the following steps:

- API Installation
- API Configuration (editing the .ini file provided)
- Logging Configuration (Log4J Configuration)
- WCS Configuration

WCS Configuration

The flow of calls to different beans in the WebSphere Commerce Environment is shown in below figure.





In WCS we need to call the LiveProcessor API from a suitable position in the logic flow. The steps to be accomplished are as follows;

- Create a service bean for communication between LiveProcessor and WCS.
- In the service bean define all fields that are needed by the LiveProcessor server in the request. These fields should have getter and setter methods. All of these fields will be mapped with some fields that are available in WCS environment and with some others that are provided by the LiveProcessor team and are specific to client.
- We need to call this service bean from a point in the order process flow where the customer enters the payment details and the system tries to use some payment processing system to get the payment. In our case we used CyberSource Payment Service provider (PSPP). So when the OrderProcessCommand calls CyberSource's Approve() method the CyberSource plugin approves if the details provided by the customer are valid or not i.e. it validates the amount, the account number etc.
- We will call a method that will get data from the order, payment and customer details and sets it in the FraudService bean, after the CyberSource plugin gets the response.





- This method will call the execute() method of FraudServiceBean after setting the data in fields.
- execute() method in FraudServiceBean will need to instantiate a LPClient object like

```
LPClient client = new LPClient(false);
```

The parameter passed to this is a flag for using secure mode or not. LPClient can be created in either secure or clear mode. When running in secure mode, the SSL connection will be established using the certificates specified in the initialization file's [Server] and [Client] sections. When running in clear mode, only important data fields such as the account number are encrypted using the data encryption keys specified in the initialization file [Keys] section.

The LPClient can process requests on multiple LiveProcessor servers in multiple sites. Requests are automatically distributed among the servers in the first site, but site fail-over will be initiated automatically if all servers at a given site are unavailable (unreachable or returning ENETPP errors). During site fail-over, the LPClient fails over to the next site as specified in the init file.

- We need to encrypt the Account Number that is to be sent to LiveProcessor Server. This can be done by calling the LPEncrypt class. This class needs the public key defined in the ini file for encryption. The encryption function will encrypt clear text and encode the result into Base64 format. These encryption functions are designed to work with LiveProcessor server. The encryption function uses RSA public key to encryption the data, then encodes the result in Base64 format with a proprietary header. This can be done in the following way inside execute() method of FraudServiceBean.

```
try {  
String sAcct = LPEncrypt.encrypt(client, null,  
getAccountNumber());  
request.setField("ACCT_NUM", sAcct);  
} catch (Exception e) {  
e.printStackTrace();  
}
```

- Instantiate an LPTransaction object to represent the request, or call the clear() method to reset an LPTransaction for another request. The LPTransaction class represents a LiveProcessor credit card and check order request





- Assign client (*LPTransaction*) request properties, including (at a minimum) request and payment method information, using the *setField* method.

Example:

```
request.setField("ORD_ID",getOrderId());  
request.setField("SHIP_FNAME",getShippingFirstName());  
request.setField("SHIP_LNAME",getShippingLastName());
```

- Call the *process()* method on *LPTransaction* object to perform a request using the assigned request properties. The API will connect to a server, send the request, receive a response, and disconnect. The server will be selected automatically by the API from the configuration file. Sends the request to LiveProcessor server for processing and gets the response. Upon successful transmission of the request, set the response properties and return to the caller.
- Read the response properties using the *getField* method of *LPTransaction* to determine the status of the request. Three possible outcomes from Retail Decisions system are accept, challenge, or deny.
 - Deny will present the user with a hard stop to the customer.
 - Challenge will go through the system, and will be flagged with an attribute as challenged.
 - Accept will go through the system as usual.
- On the status returned from the retail decisions, take the appropriate action specified above.

Summary

In this article we came to know how we can integrate the Fraud Detection and prevention solution API provided by Retail Decisions within the IBM WebSphere commerce environment. First we need to configure and setup the configuration file and then write our code to fetch data for the LiveProcess request from our environment. This solution will stop order processing if there is a fraudulent transaction detected.

"Retail Decisions' LiveProcessor provides us with the best of both worlds -- a long-term, fast and dependable payment processing solution for our expanding transaction volumes, and a solution that provides us with the ability to safeguard the confidentiality of our customers' payment information."

For more information, please dial us on 1-630-355-6292 (Monday - Friday: 8AM to 6PM) , alternatively you can email us at info@royalcyber.com or for any inquiry [Contact us](#) and our team of consultants will assist you further. Contact us with your requirements by clicking the link below: [WebSphere Commerce Inquiry Form](#)

