Software Documentation : Java Generic Amazon Flexible Payment System Component

This page last changed on Nov 18, 2011 by delemach.

1. Scope

1.1 Overview

Create a component to abstract the complexity of using Amazon FPS, and to allow for it's easy integration into web applications. We will be using this component to add support for clients to pay directly in TopCoder Direct.

1.1.1 Version

1.0.0

1.2 Logic Requirements

The component needs to support several client charging scenarios, which are detailed in the functional requirements below. Notably, the charge scenarios will be variations where some payment is charged immediately, and then subsequent payments are deferred. The subsequent charges/payments must not require re-authorization from the client.

We expect that you will be implementing Multi-User Payment Tokens, the Reserve Method and the Settle Method to support these scenarios. (This is just a starting point - you will need to do your own research into this).

For all charge scenarios, the payment trigger will come from the consuming system. For example, when integrated into Direct, some events that will trigger a FPS charge to the client include:

- Starting a project.
- A contest beginning.
- A contest completing.
- A project completing.
- A bug race or task launching.
- A bug race or task completing.

It is up to you as the designer to come up with the best way to have external events such as these trigger FPS charges.

1.2.1 Charge Scenario - Charge Fixed Amount Immediately

Steps:

1. Charge a fixed amount now to the client.

1.2.2 Charge Scenario - Charge Fixed Amount, Authorize Fixed Future Amount

Steps:

1. Charge a fixed amount now to the client.
2. Authorize future charges.
3. On external trigger, charge client without re-authorization.

1.2.3 Charge Scenario - Charge Fixed Amount, Authorize Variable Future Charges Up to Some Threshold

Steps:

1. Charge a fixed amount now to the client.

2. Authorize maximum future charges.
3. On external trigger, charge client without re-authorization.
4. Lower threshold.

1.2.4 Support Canceling Reserved Transaction

External system may cancel a reserved future transaction. For example in Direct, if the client cancels a project future reserved transactions must be canceled.
See the Cancel Method.

1.2.5 Handle Rejected Charges

If a charge to the client fails the consuming system must be notified to handle the failure correctly.

1.1.6 Event Notification

The component must support notifying interested subscribers about payment events. For example, such a subscription feature will allow PACTs to be notified immediately about a client payment.

1.1.7 Amazon Sandbox

The component must support switching between Sandbox and Production mode:
http://docs.amazonwebservices.com/AmazonFPS/latest/FPSGettingStartedGuide/

1.1.8 Logging and Exception Handling

The component must implement good logging, both for debugging, development and production purposes. Log4j is recommended.

Exceptions must be gracefully handled and must not interfere with execution of consuming system.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be integrated into TopCoder Direct to allow clients to pay TopCoder directly for launching contests and projects.

1.5 Future Component Direction

In the future additional functionality will potentially support reporting.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None at this time. Bear in mind that the component will be integrated into TopCoder Direct. At points in the application flow where payment is required, clients will be redirected to Amazon Cobranded User UI Page to charge the customer with a fixed amount of money (e.g. 100$) and get customer approval for charging the customer in future. The component must support this scenario.

2.1.2 External Interfaces

The component will interface with external systems in the manner described in other sections of this specification.

2.1.3 Environment Requirements

- Development language: Java1.6
- Compile target: Java1.6
- Amazon FPS Advanced API

2.1.4 Package Structure

com.topcoder.payments.amazonfps

3. Software Requirements

3.1 References

The Amazon FPS Advanced API is to be used in for this component:
http://docs.amazonwebservices.com/AmazonFPS/latest/FPSAdvancedGuide/

Some other resources that you may find useful:
http://s3.amazonaws.com/awsdocs/FPS/latest/fps-qs-adv.pdf
http://s3.amazonaws.com/awsdocs/FPS/latest/fps-qs-act.pdf
http://s3.amazonaws.com/awsdocs/FPS/latest/fps-gsg.pdf

The following library may be a helpful reference and/or starting point: http://aws.amazon.com/code/Amazon-FPS/5090796688019801.

3.1.1 What elements of the application need to be configurable?

- Setting whether the component is hitting the FPS sandbox or production.
- AWS Credentials.
- If persistence is required (other than configuration values) then datasource must be configurable.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

Java
AWS - FPS

3.2.2 TopCoder Software Component Dependencies:

Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

AWS - FPS

3.2.4 QA Environment:

- If persistence is required, then Informix 10 and 11 must be supported.
- Component will be integrated into Direct.

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage.
- Component Specification must contain sample usage code.