LegSim Converge Payment Integration

Last Updated: August 26, 2020

Executive Summary

LegSim is a Rails application that uses ActiveMerchant to integrate with Converge / Elavon payment gateway. The integrate uses the Devise confirmable module to control access until payment has been received. A custom Rails model is used to store payment events.

Converge / Elavon Payment Gateway

The University of Washington provides a set of different payment gateways that may be used by projects like LegSim. Of that set, LegSim currently implements the Converge / Elavon payment gateway. Documentation from Converge can be found on their developer site: https://developer.elavon.com/. The gateway provides standard credit card processing for sale of goods and refunds against those sales.

Evalon Production/Testing Environments

Elavon provides both a Production and a Testing environment. However, the Testing environment is not perpetual and needs to be renewed each time testing needs to be performed. Requesting a renewed testing environment should be done by sending an email to techsupp@elavon.com referencing the production account number (provided below) and also provide any IP addresses that will need to talk to the testing server.

Elavon Credentials

Connecting to Elavon requires three different values to establish a connection: merchant_id, user_id, pin. These values are stored in LegSim's encrypted credentials file in `config/credentials.yml.enc`. Opening/editing that file requires a copy of the `config/master.key` file with the correct key value. Once the master.key file is in place, the credentials file can be opened by invoking:

```
bin/rails credentials:edit
```

In the credentials file there is an elavon section that breaks into **production** and **testing**. Within each section is a **login** (maps to **merchant_id**), **user** (maps to **user_id**), and **password** (maps to **pin**).

ActiveMerchant

LegSim does not directly interact with the Evalon API, but instead leverages ActiveMerchant to wrap the Evalon API and provide a set of standard interfaces that remain consistent regardless of payment gateway. The source code and documentation for ActiveMerchant can be found on GitHub: https://github.com/activemerchant/active_merchant/

The Elavon API wrapper can be found here: https://github.com/activemerchant/active_merchant/blob/master/lib/active_merchant/billing/gateways/elavon.rb

Rails Configuration

All global configuration for ActiveMerchant is located in `config/initializers/active_merchant.rb`. Currently the only setting there is whether to load into the testing mode or production mode. This is currently determined by the Rails environment. If the Rails environment is production, then ActiveMerchant loads in production. In all other environments, ActiveMerchant loads in test mode.

Payment Model

LegSim implements a Payment Model that stores relevant data about a payment attempt with Elavon. The model stores the following fields:

- **user_id** - foreign key to the User record of the user who attempted the payment
- **payment_type** - currently only stores the string “elavon” to indicate the payment was performed via Elavon, but could be set to other payment gateways.
- amount - the amount charged in *cents* (for a $10 charge the amount would be set to 1000)
- **prcoessed_at** - the datetime of the attempted transaction
- **transaction_id** - a string that identifies the transaction with the relevant gateway (needed for refunds)
- **approval_code** - no idea what this does, but we are storing it anyway
- **status** - strings the string “Paid”, “Failed”, or “Refunded”
- **cc_number** - the last four digits of the credit card used to help identify the transaction
- **test** - boolean for whether the transaction was run in a testing mode or not
- **details** - text that stores the error or validation messages that generated a failed status

Course Settings

When creating a new Course, the System Administrator must set the payment_option. Currently the only supported methods are **Elavon** or **Prepaid**. When a course is marked as **Prepaid**, all users will be considered licensed immediately upon creation. If it is set to **Elavon**, Member accounts will require payment before the User is considered licensed.

Devise Integration

LegSim leverages the Devise authentication framework to implement password authentication for users. To control access LegSim uses the confirmable module which denies access to secured parts of the application unless the user has been confirmed.

In the case of Prepaid courses, as soon as the user account is created (regardless of type) a confirmation email is sent to the user's email address containing a unique link. Once that link is clicked, the user is considered confirmed and may log into LegSim.

In all other cases (specifically Elavon) if a user is created of type Member, then the user is routed to a Elavon payment page (discussed later). Once payment information is received and successfully processed, a confirmation email is set to the user's email address containing a unique link. All non-Member users receive a confirmation email immediately without routing through the payment process.

Payment Controller

The payment process is handled via the PaymentsController at `app/controllers/payments_controller.rb`. Each payment gateway has a pair of actions. In the case of Elavon there is the **elavon** and **elavon_checkout** action. The **elavon** action renders a payment user interface. That page posts to **elavon_checkout** action which handles the logic:

1. If the user is considered to already have a license, render the has_license page
2. Connect to the payment gateway
3. Create an ActiveMerchant credit card object
4. Create a hash to store payment information in case it needs to be rendered into an error page
5. Check if the credit card inputs are valid
    a. If not: store a Failed Payment record with the validation errors and render the **evalon** page using the payment inputs (this saves the user from having to re-enter information)
6. Process the payment and see if it is a success

    a. If not: store a Failed Payment with the response message and render the **evalon** page using the payment inputs (this saves the user from having to re-enter information)
7. If successful, store a Paid Payment, send a confirmation email, and render the complete page

The controller also defines a **LICNESE_AMOUNT** value that defines how much to charge for a license in cents.

Payment Log

A complete log of all payment events for a user can be found in the /system interface. This will show all the information stored in the Payment model along with the details of why the payment failed. If the payment was successful it will also show a Refund button that, when pressed, will post to the refund action in the System::PaymentsController. The refund action creates a connection to the payment gateway and then uses the Payment's transaction ID to call a refund method on the gateway. When invoking refund LegSim passes **nil** for the amount which refunds the full amount of the transaction regardless of the amount. There is currently no interface to refund an amount other than the full amount.

When a refund is successful the Payment record's status is updated to Refunded.