e24PaymentPipe Documentation

Release 1.2.0

Burhan Khalid

Oct 30, 2017

Contents

Contents:

CHAPTER 1

e24PaymentPipe

This package provides a Python implementation for ACI's e24PaymentPipe Merchant Gateway

Note: For legacy reasons, the name of the module is kept the same as the one from ACI's toolkit. This does not conform to PEP-8 guidelines.

- Free software: BSD license
- Documentation: http://e24PaymentPipe.rtfd.org.

Features

- Written in Python from scratch (not an existing port)
- Supports both newer and legacy terminal resource (.cgn) file formats
- Reasonably well documented
- Proven code - running in production since 2011
- Compatible with Python 3.4, Python 2.7

Todo

- Add support for credit card payments, including refunds.
- Create comprehensive test suite

CHAPTER 2

Installation

At the command line:

```
$ easy_install e24PaymentPipe
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv e24PaymentPipe
$ pip install e24PaymentPipe
```

CHAPTER 3

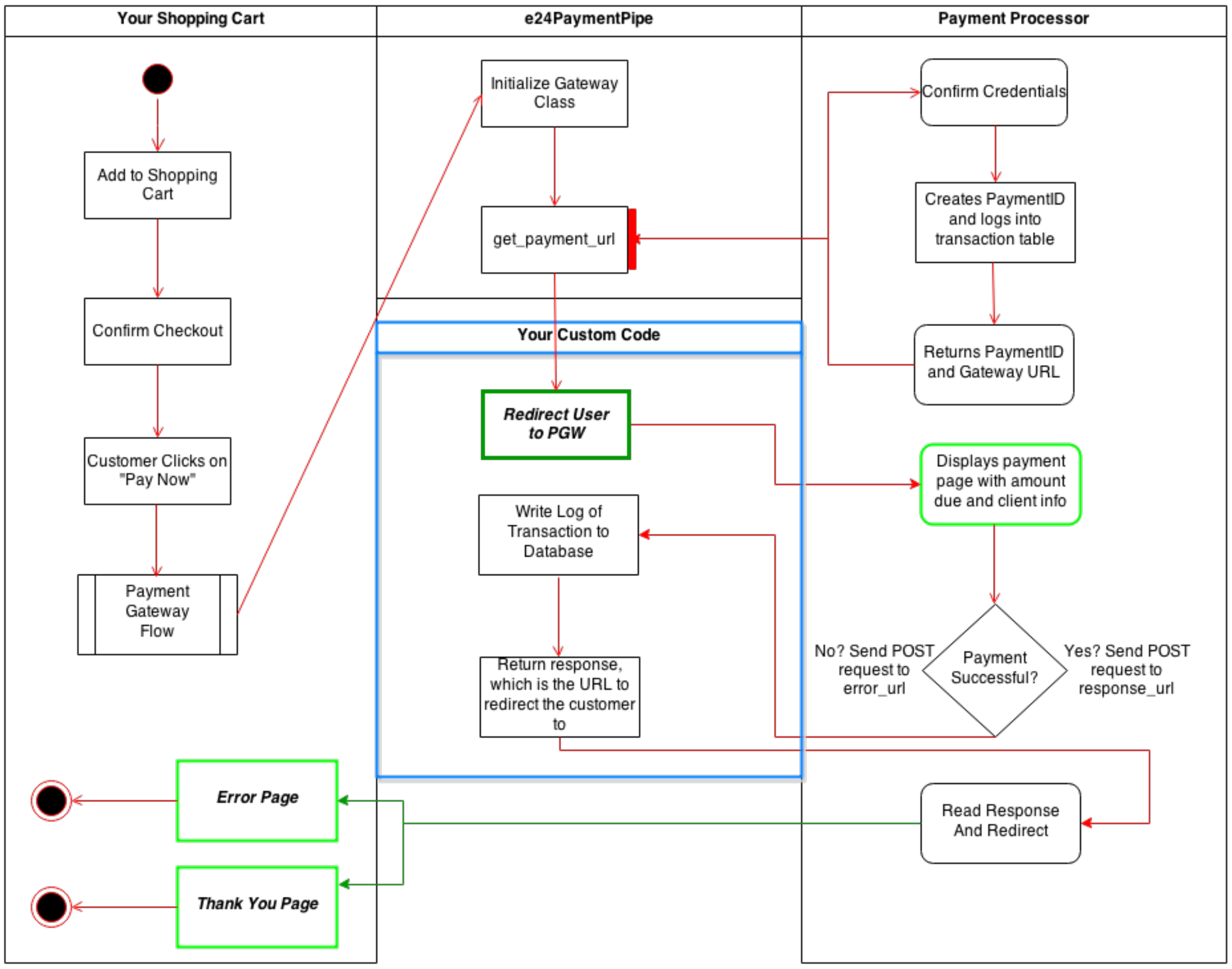
Quickstart

In order to use this module, you need to obtain a resource file and the alias for your terminal from your payment provider.

This file will allow the module to connect to the gateway server to initialize your transaction workflow.

Payment Flow Diagram

This diagram illustrates the typical flow for the e24PaymentPipe gateway, and how the various components fit together:


Your Shopping Cart
e24PaymentPipe
Payment Processor
Add to Shopping Cart
Confirm Checkout
Customer Clicks on "Pay Now"
Payment Gateway Flow
Initialize Gateway Class
get_payment_url
Confirm Credentials
Creates PaymentID and logs into transaction table
Returns PaymentID and Gateway URL
Your Custom Code
Redirect User to PGW
Write Log of Transaction to Database
Return response, which is the URL to redirect the customer to
Displays payment page with amount due and client info
Payment Successful?
No? Send POST request to error_url
Yes? Send POST request to response_url
Read Response And Redirect
Error Page
Thank You Page

This module provides the functionality contained in the block marked e24PaymentPipe. In a typical use case, you would write the functionality defined in the blue box in order to integrate the payment gateway with and existing system.

The green highlighted boxes show where the user will be redirected to.

For a sample implementation of this gateway, check out https://github.com/burhan/e24-sample-cart

Minimal Example

The quick demo for the impatient:

```
from e24PaymentPipe import Gateway

pgw = Gateway('resource.cgn', 'alias')
pgw.error_url = 'http://example.com/error'
pgw.response_url = 'http://example.com/response'
pgw.amount = 1.0
gw_info = pgw.get_payment_url()
```

The snippet above will create a transaction in KWD, the default currency for the system. To override this, see the module documentation.

The `get_payment_url()` method will return a dictionary with two keys paymentID and paymentURL. You need to store the paymentID in your database (as part of the transaction requirements) and then attach it at the end of the paymentURL to start the transaction initiation process.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/burhan/e24PaymentPipe/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

Write Documentation

e24PaymentPipe could always use more documentation, whether as part of the official e24PaymentPipe docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/burhan/e24PaymentPipe/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *e24PaymentPipe* for local development.

1. Fork the *e24PaymentPipe* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/e24PaymentPipe.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv e24PaymentPipe
$ cd e24PaymentPipe/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 e24PaymentPipe tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7 and for PyPy. Check https://travis-ci.org/burhan/e24PaymentPipe/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

Development Lead

- Burhan Khalid <burhan.khalid@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

1.2.0 (2015-12-29)

- Now compatible with Python 3.4!
- Added requests as a dependency

1.1.2 (2014-11-03)

- Fixed issue iterating over UDF dictionary keys
- Fixed issue where unicode was being passed to sanitize unicode strings are not allowed in UDF fields

1.1 (2014-08-12)

- Refactored class methods to *utils.py*
- PEP-8 cleanup
- Version bump

0.1.0 (2011-02-12)

- First release on Github.

0.1.1 (2011-02-14)

- Added custom exceptions

0.1.2 (2011-12-16)

- Optimized error checking
- Added character filters for UDF as per latest payment provider update

0.2.0 (2011-12-28)

- Version branch created with major changes in the codebase
- Cleaned up the API and interface

CHAPTER 7

Module Reference

If you haven't already done so, please read the *quickstart* for an overview of the payment process and a quick example on usage of this module.

Main Methods / Public Interface

class `Gateway` (*resource*, *alias*[, *currency=414*, *lang='ENG'*])

This is the main class that creates parses the terminal resource file and creates the gateway link in order to initiate the payment process.

Parameters

- **`resource`** (*`str`*) – The full path name the the resource file provided by the payment processor
- **`alias`** (*`str`*) – The alias for the terminal. See your payment processor for more information
- **`currency`** (*`int`*) – The ISO 4217 numeric code for the currency of the transaction. See ISO 4217 for more.
- **`lang`** (*`str`*) – The language supported by the gateway. See your payment processor's documentation for the languages supported. Defaults to 'ENG' for US English.

static `sanitize` (*s*) → string

Returns the string stripped of characters not allowed in the transaction id or the UDF (user defined fields)

The following characters are stripped:

Symbol	Hex	Name
~	x7E	TILDE
‘	x60	LEFT SINGLE QUOTATION MARK, GRAVE ACCENT
!	x21	EXCLAMATION POINT (bang)
#	x23	NUMBER SIGN (pound sign)
$	x24	DOLLAR SIGN
%	x25	PERCENT SIGN
^	x5E	CIRCUMFLEX ACCENT
	x7C	VERTICAL LINE (pipe)
	x5C	REVERSE SLANT (REVERSE SOLIDUS, backslash, backslant)
:	x3A	COLON
‘	x27	APOSTROPHE, RIGHT SINGLE QUOTATION MARK, ACUTE ACCENT (single quote)
“	x22	QUOTATION MARK, DIAERESIS
/	x2F	SLANT (SOLIDUS, slash)

Parameters `s` (*str*) – The string to be sanitized

Returns Sanitized string, with characters not allowed removed

Return type str

`get_payment_url()` → dict

This function returns a two element dictionary with the *paymentID* and the *paymentURL* that is needed to redirect the user to the gateway.

In order to call this method, the gateway needs to be correctly initialized and configured.

Raises `ValueError` – if the object is not configured correctly

Returns A dictionary of payment id and payment url from the gateway

Return type dict

Properties

Properties are used to set various gateway parameters, and to ensure that all parameters are correctly sanitized.

All properties are accessed from instances of the gateway object.

Property Name	Description	Required/Optional
udf	Set or get the user defined fields (UDF). Please note the following restrictions: 1. You can only send data for a maximum of 5 fields. 2. If passing a dictionary, the keys must be named 'UDF[1..5]' 3. When setting the field, if a tuple or list is passed, the keys are generated automatically. 4. All values are automatically processed by `sanitize()`	Optional
error_url	The fully qualified URL to the error handler for the gateway. See the *quickstart* for more information	Required
amount	The amount for this transaction, this should be a floating point number. The default and the minimum is 1.0. Invalid values will raise a *TypeError*.	Required
trackid	The tracking id for this transaction. It must be a unique value. Like the udf fields, it is also sanitized. A default value based on the current timestamp is generated if not provided.	Optional
response_url	The fully qualified URL for all affirmative responses from the gateway. See *quickstart*	Required

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index