A2Billing Flask API Documentation

Release 1.0

Areski Belaid

Mar 14, 2017

Contents

Contents:

CHAPTER 1

Overview

Source https://github.com/areski/a2billing-flask-api/

Keywords a2billing, api, flask

Flexible & Fast Restful APIs framework for A2Billing powered by Flask & Peewee. A2Billing-Flask-API comes with some tools for exposing your A2Billing models via a RESTful API.

Each RestFul APIs exposed supports the following:

/api/<model name>/ – GET and POST requests

/api/<model name>/<primary key>/ – GET, PUT and DELETE requests

Also, you can filter results by columns on the model. For example:

/api/cardgroup/?name=Some%20Blog

.._installation:

Installation

An install shell script is provided at https://github.com/areski/a2billing-flask-api/tree/master/install

The install script is intended to run on Debian 8.

Usage:

```
wget https://raw.githubusercontent.com/areski/a2billing-flask-api/master/install/
↪install-a2b-flask-api.sh
bash install-a2b-flask-api.sh
```

Requirements

This Application is build using Flask and Peewee:

- Python 2.5 or greater
- Flask : http://flask.pocoo.org/
- Peewee : http://peewee.readthedocs.org/en/latest/
- Gunicorn : http://gunicorn.org/
- WTForms : http://wtforms.readthedocs.org/en/latest/
- MySQL-python : MySQL-python
- Flask-HTTPAuth : https://pypi.python.org/pypi/Flask-HTTPAuth

See the file requirements.txt for the full list of requirements.

.._admin-panel:

Admin Panel

An Admin Panel is provided which can be accessed at http://<ip_address>:8008/admin/

You will need an admin username and password to login, see the section below on how to create an admin user.

View resources:

Edit resources:

Stress Test

Use ab, the Apache HTTP server benchmarking tool

Usage:

```
ab -c 100 -n 1000 -p test/post.txt -T application/x-www-form-urlencoded http://
↪localhost:8008/api/cardgroup/
```

Install & Deployment

There are many ways to deploy a Flask Application, we will describe the Apache Method here as this is the one more suitable for A2Billing users.

Reference: https://www.digitalocean.com/community/articles/how-to-deploy-a-flask-application-on-an-ubuntu-vps

Security

Edit a2billing_flaskapi.py and change the secret key and keep this really secret:

```
app.secret_key = 'ssshhhh-and-changeme-when-deploying'
```

Create an Admin User

We now have a functioning admin site, you can login with user / password: admin / admin

Change immediately the default password by a strong password!!!

You might want to create an other admin user from shell, to do so open up an interactive python shell in the directory alongside the app and run the following:

```
$ cd /usr/share/a2billing-flask-api/
$ workon a2billing-flask-api
$ python
```

Then in Python interpreter, type the following:

```
from a2billing_flask_api import auth
auth.User.create_table(fail_silently=True)  # make sure table created.
admin = auth.User(username='admin', email='', admin=True, active=True)
admin.set_password('admin')
admin.save()
```

Documentation

Check out the documentation on ‘Read the Docs’: http://a2billing-flask-api.readthedocs.org/en/latest/index.html

Contributing

If you’ve found a bug, add a feature or improve the project and think it is useful then please consider contributing. Patches, pull requests or just suggestions are always welcome!

Source code: https://github.com/areski/a2billing-flask-api/

If you don’t like Github and Git you’re welcome to send regular patches.

Bug tracker: https://github.com/areski/a2billing-flask-api//issues

License

A2Billing-Flask-API is licensed under MPLv2.

CHAPTER 2

List of APIs

This is the list of Restful APIs supported.

CardGroup - Method [GET/POST/PUT/DELETE]

Get list of card-groups, create new card-group, Update/Delete existing card-group.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/cardgroup/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/cardgroup/?name=DEFAULT'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/cardgroup/1
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/cardgroup/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{"name": "mygroup", "description": ""}' http://localhost:8008/api/cardgroup/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{"name": "mygroup-updated", "description": ""}' http://localhost:8008/api/
↪cardgroup/3/
```

Card - Method [GET/POST/PUT/DELETE]

Get list of cards, create new card, Update/Delete existing card.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/card/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/card/?username=1321546'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/card/1/
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/card/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{"name": "mygroup", "description": ""}' http://localhost:8008/api/card/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{"name": "mygroup-updated", "description": ""}' http://localhost:8008/api/
↪card/3/
```

Call - Method [GET/POST/PUT/DELETE]

Get list of calls, create new calls, Update/Delete existing calls.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/call/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/call/?field=1321546'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/call/1/
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/call/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{...}' http://localhost:8008/api/call/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{...}' http://localhost:8008/api/call/3/
```

CallerID - Method [GET/POST/PUT/DELETE]

Get list of CallerIds, create new CallerIds, Update/Delete existing CallerIds.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/callerid/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/callerid/?field=1321546'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/callerid/1/
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/callerid/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{...}' http://localhost:8008/api/callerid/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{...}' http://localhost:8008/api/callerid/3/
```

LogRefill - Method [GET/POST/PUT/DELETE]

Get list of Refills, create new Refills, Update/Delete existing Refills.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/logrefill/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/logrefill/?field=1321546'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/logrefill/1/
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/logrefill/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{...}' http://localhost:8008/api/logrefill/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{...}' http://localhost:8008/api/logrefill/3/
```

LogPayment - Method [GET/POST/PUT/DELETE]

Get list of Payments, create new Payments, Update/Delete existing Payments.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/logpayment/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/logpayment/?field=1321546'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/logpayment/1/
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/logpayment/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{...}' http://localhost:8008/api/logpayment/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{...}' http://localhost:8008/api/logpayment/3/
```

Country - Method [GET/POST/PUT/DELETE]

Get list of Countries, create new Countries, Update/Delete existing Countries.

METHODS:

GET ALL:

```
curl -u username:password http://localhost:8008/api/country/
```

GET ALL FILTER:

```
curl -u username:password 'http://localhost:8008/api/country/?field=1321546'
```

GET ONE:

```
curl -u username:password http://localhost:8008/api/country/1/
```

DELETE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X␣
↪DELETE http://localhost:8008/api/country/4/
```

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{...}' http://localhost:8008/api/country/
```

UPDATE:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X PUT --
↪data '{...}' http://localhost:8008/api/country/3/
```

Refill - Method [POST]

This API will refill an Account/Card for a given credit amount (value: Decimal). A logpayment and a logrefill will also be added to log the refill.

In the result, the current balance will be returned with the VAT/Tax from the Account/Card, and the created logpayment ID and logrefill Id will also be returned.

METHODS:

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{"credit": 5}' http://localhost:8008/custom_api/refill/1
```

Extra Charge - Method [POST]

This API will decrement an Account/Card for a given amount (value: Decimal), then a charge will also be added to log the transaction.

In the result, the current balance will be returned and the created Charge Id will also be returned.

METHODS:

ADD:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST -
↪-data '{"amount": 5}' http://localhost:8008/custom_api/extra_charge/1
```

CHAPTER 3

APIs Detailed Documentation

Usage API - Card

Cards are A2Billing Users on the A2Billing Platform, this regroups credentials and specific information related to the users, such as names, address, balance, etc..

GET ALL

$ curl -u username:password http://localhost:8008/api/card/

Result:

```
{
    "meta": {
      "model": "card",
      "next": "",
      "page": 1,
      "previous": ""
    },
    "objects": [
      {
        "email_notification": "areski@gmail.com",
        "status": 1,
        "expiredays": null,
        "loginkey": "4654",
        "lock_pin": "0",
        "useralias": "312224525577965",
        "uipass": "18314euvyzix7spr1eew",
        "activated": "f",
        "currency": "USD",
        "tag": "ok",
        "initialbalance": 0.0,
        "voicemail_activated": 0,
        ...,
            ...
        }
    ]
}
```

GET ONE

$ curl -u username:password http://localhost:8008/api/card/1/

Result:

```
{
  "email_notification": "areski@gmail.com",
  "status": 1,
  "expiredays": null,
  "loginkey": "4654",
  "lock_pin": "0",
  "useralias": "312224525577965",
  "uipass": "18314euvyzix7spr1eew",
  "activated": "f",
  "currency": "USD",
  "tag": "ok",
  "initialbalance": 0.0,
  "voicemail_activated": 0,
  "redial": "0",
  "id": 1,
  "sip_buddy": 1,
  "city": "Barcelona",
  "id_group": 1,
  ...,
}
```

DELETE

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X DELETE http://localhost:8008/api/card/4/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 18
Server: Werkzeug/0.9.4 Python/2.7.5+
Date: Thu, 17 Apr 2014 18:50:43 GMT

{
  "deleted": 1
}
```

ADD

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X POST –data ‘{“username”: “1234567890”, “useralias”: “0554654648”, “lastname”: “Belaid”, “firstname”: “Areski”, “uipass”: “6546456”, “credit”: “5”, “tariff”: “1”}’ http://localhost:8008/api/card/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1257
Server: Werkzeug/0.9.4 Python/2.7.5+
Date: Thu, 17 Apr 2014 23:33:14 GMT

{
  "email_notification": "",
  "status": 1,
  "expiredays": null,
  "loginkey": "",
  "lock_pin": null,
  "useralias": "0554654648",
  "uipass": "6546456",
  "activated": null,
  "currency": "USD",
  "tag": "",
  "initialbalance": 0.0,
  "voicemail_activated": 0,
  "redial": "",
  "id": 7,
  "sip_buddy": 0,
  "city": "",
  "id_group": 1,
  "notify_email": 0,
  ...
}
```

UPDATE

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X PUT –data ‘{“lastname”: “Belaid”}’ http://localhost:8008/api/card/7/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1290
Server: Werkzeug/0.9.4 Python/2.7.5+
Date: Thu, 17 Apr 2014 23:36:10 GMT

{
  "email_notification": "",
  "status": 1,
  "expiredays": "",
  "loginkey": "",
  "lock_pin": null,
  "useralias": "0554654648",
  "uipass": "6546456",
  "activated": "f",
  "currency": "USD",
  "tag": "",
  "initialbalance": 0.0,
  "voicemail_activated": 0,
  "redial": "",
  "id": 7,
  "sip_buddy": 0,
  "city": "",
  "id_group": 1,
  "notify_email": 0,
  ...
}
```

Usage API - Card Group

Card Group allows to regroup Card per entity and define agents associated to them, as well as user permissions when accessing the Customer UI.

GET ALL

$ curl -u username:password http://localhost:8008/api/cardgroup/

Result:

```
{
    "meta": {
        "model": "cardgroup",
        "next": "",
        "page": 1,
        "previous": ""
    },
    "objects": [
        {
            "id_agent": null,
            "description": "This group is the default group used when you create a␣
↪customer. It's forbidden to delete it because you need at least one group but you␣
↪can edit it.",
            "users_perms": 262142,
            "id": 1,
            "name": "DEFAULT"
        },
        {
            "id_agent": 0,
            "description": null,
            "users_perms": 0,
            "id": 2,
            "name": "NewGroup"
        }
    ]
}
```

GET ONE

$ curl -u username:password http://localhost:8008/api/cardgroup/1/

Result:

```
{
    "id_agent": null,
    "description": "This group is the default group used when you create a customer.␣
↪It's forbidden to delete it because you need at least one group but you can edit it.
↪",
    "users_perms": 262142,
    "id": 1,
    "name": "DEFAULT"
}
```

DELETE

$ curl -u username:password –dump-header - -H "Content-Type:application/json" -X DELETE http://localhost:8008/api/cardgroup/4/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 18
Server: Werkzeug/0.9.4 Python/2.7.5+
Date: Thu, 17 Apr 2014 16:11:03 GMT

{
  "deleted": 1
}
```

ADD

$ curl -u username:password –dump-header - -H "Content-Type:application/json" -X POST –data '{"name": "mygroup", "description": ""}' http://localhost:8008/api/cardgroup/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 96
Server: Werkzeug/0.9.4 Python/2.7.5+
Date: Thu, 17 Apr 2014 16:08:55 GMT

{
  "id_agent": 0,
  "description": "",
  "users_perms": 0,
  "id": 3,
  "name": "mygroup"
}
```

UPDATE

$ curl -u username:password –dump-header - -H "Content-Type:application/json" -X PUT –data '{"name": "mygroup-updated", "description": ""}' http://localhost:8008/api/cardgroup/3/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 104
Server: Werkzeug/0.9.4 Python/2.7.5+
Date: Thu, 17 Apr 2014 16:12:31 GMT

{
  "id_agent": 0,
  "description": "",
  "users_perms": 0,
  "id": 3,
  "name": "mygroup-updated"
}
```

Usage API - Callerid

This entity is the CallerIDs associated to a customer (Card)

GET ALL

$ curl -u username:password http://localhost:8008/api/cardgroup/

Result:

```
{
  "meta": {
    "model": "callerid",
    "next": "",
    "page": 1,
    "previous": ""
  },
  "objects": [
    {
      "id_cc_card": 1,
      "activated": "t",
      "id": 2,
      "cid": "45454565456456"
    }
  ]
}
```

GET ONE

$ curl -i -u username:password http://localhost:8008/api/cardgroup/1/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 79
Server: Werkzeug/0.11.2 Python/2.7.9
Date: Fri, 27 Nov 2015 21:27:56 GMT
```

```
{
  "id_cc_card": 1,
  "activated": "t",
  "id": 2,
  "cid": "45454565456456"
}
```

DELETE

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X DELETE http://localhost:8008/api/callerid/6/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 18
Server: Werkzeug/0.11.2 Python/2.7.9
Date: Fri, 27 Nov 2015 21:29:18 GMT

{
  "deleted": 1
}
```

ADD

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X POST –data ‘{“id_cc_card”: 1, “cid”: “9501234657”}’ http://localhost:8008/api/callerid/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 75
Server: Werkzeug/0.11.2 Python/2.7.9
Date: Fri, 27 Nov 2015 21:31:19 GMT

{
  "id_cc_card": 1,
  "activated": "t",
  "id": 7,
  "cid": "9501234657"
}
```

UPDATE

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X PUT –data ‘{“cid”: “9501234658”}’ http://localhost:8008/api/callerid/7/

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 75
```

```
Server: Werkzeug/0.11.2 Python/2.7.9
Date: Fri, 27 Nov 2015 21:32:30 GMT

{
  "id_cc_card": 1,
  "activated": "t",
  "id": 7,
  "cid": "9501234658"
}
```

Usage API - Logrefill

This is used to track the refill made into the A2Billing platform.

GET ALL

$ curl -u username:password http://localhost:8008/api/logrefill/

Result:

```
{
  "meta": {
    "model": "logrefill",
    "next": "",
    "page": 1,
    "previous": ""
  },
  "objects": [
    {
      "description": "CREATION CARD REFILL",
      "refill_type": 0,
      "agent": null,
      "credit": 5.00000,
      "date": "2014-04-16 01:11:45",
      "id": 1,
      "card": 1,
      "added_invoice": 0
    },
    {
      "description": "4654",
      "refill_type": 0,
      "agent": null,
      "credit": 6456.00000,
      "date": "2014-06-04 14:56:36",
      "id": 2,
      "card": 1,
      "added_invoice": 0
    },
  ]
}
```

GET ONE

TODO: Not documented!

DELETE

TODO: Not documented!

ADD

TODO: Not documented!

UPDATE

TODO: Not documented!

Usage API - Logpayment

This is used to track the payment made into the A2Billing platform.

GET ALL

$ curl -u username:password http://localhost:8008/api/logpayment/

Result:

```
{
  "meta": {
    "model": "logpayment",
    "next": "",
    "page": 1,
    "previous": ""
  },
  "objects": [
    {
      "added_refill": 1,
      "description": "4654",
      "added_commission": 0,
      "id": 1,
      "payment_type": 2,
      "agent": null,
      "date": "2014-06-04 14:56:36",
      "id_logrefill": 2,
      "payment": 6456.00000,
      "card": 1
    },
    {
      "added_refill": 0,
      "description": null,
      "added_commission": 0,
      "id": 2,
      "payment_type": 0,
      "agent": null,
      "date": null,
      "id_logrefill": 12,
      "payment": 5.89000,
      "card": 2
    },
  ]
}
```

GET ONE

TODO: Not documented!

DELETE

TODO: Not documented!

ADD

TODO: Not documented!

UPDATE

TODO: Not documented!

Usage API - Call

This entity is the Call, also known as CDR.

GET ALL

$ curl -u username:password http://localhost:8008/api/call/

Result:

```
{
  "meta": {
    "model": "call",
    "next": "",
    "page": 1,
    "previous": ""
  },
  "objects": [
    {
      "calledstation": "7987944994",
      "id_did": 0,
      "id_tariffplan": 1,
      "id": 1,
      "id_ratecard": 1,
      "terminatecauseid": 5,
      "destination": 132487987,
      "dnid": "61984644",
      "starttime": "2015-11-27 22:36:02",
      "id_card_package_offer": 0,
      "nasipaddress": "127.0.0.1",
      "id_trunk": 2,
      "sipiax": null,
      "sessionid": "13564654984",
      "stoptime": null,
      "sessiontime": 40,
      "uniqueid": "654654981615",
      "src": "source",
      "buycost": 0.10000,
      "card_id": 1,
      "id_tariffgroup": 2,
      "real_sessiontime": 50,
      "sessionbill": 40.0
    }
  ]
}
```

GET ONE

TODO: Not documented!

DELETE

TODO: Not documented!

ADD

TODO: Not documented!

UPDATE

TODO: Not documented!

Usage API - Country

List of countries.

GET ALL

$ curl -u username:password http://localhost:8008/api/country/

Result:

```
{
  "meta": {
    "model": "country",
    "next": "/api/country/?page=2",
    "page": 1,
    "previous": ""
  },
  "objects": [
    {
      "countryname": "Afghanistan",
      "id": 1,
      "countrycode": "AFG",
      "countryprefix": "93"
    },
    {
      "countryname": "Albania",
      "id": 2,
      "countrycode": "ALB",
      "countryprefix": "355"
    },
  ...
  ]
}
```

GET ONE

TODO: Not documented!

DELETE

TODO: Not documented!

ADD

TODO: Not documented!

UPDATE

TODO: Not documented!

Usage API - Refill

This API will refill an Account/Card for a given credit amount (value: Decimal). A logpayment and a logrefill will also be added to log the refill.

In the result, the current balance will be returned with the VAT/Tax from the Account/Card, and the created logpayment ID and logrefill Id will also be returned.

ADD

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X POST –data ‘{“credit”: 5}’ http://localhost:8008/custom_api/refill/1

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 169
Server: Werkzeug/0.11.2 Python/2.7.9
Date: Fri, 27 Nov 2015 22:04:31 GMT

{
  "card_id": 1,
  "credit_without_vat": 5.0,
  "credited": 5.0,
  "current_balance": 6511.0,
  "logpayment_id": 9,
  "logrefill_id": 19
  "vat": 0
}
```

Usage API - Extra Charge

This API will decrement an Account/Card for a given amount (value: Decimal), then a charge will also be added to log the transaction.

In the result, the current balance will be returned and the created Charge Id will also be returned.

ADD

$ curl -u username:password –dump-header - -H “Content-Type:application/json” -X POST –data ‘{“amount”: 5}’ http://localhost:8008/custom_api/extra_charge/1

Result:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 82
Server: Werkzeug/0.11.2 Python/2.7.9
Date: Fri, 27 Nov 2015 22:46:36 GMT

{
  "amount": 5.0,
  "card_id": 1,
  "charge_id": 8,
  "current_balance": 6496.0
}
```

CHAPTER 4

Deploy A2Billing-Flask-API

There are many ways to deploy a Flask application, here we will focus on deploying on Apache2 webserver using mod_wsgi. This should be the easiest way for A2Billing users without installing too many applications on their server.

Installing mod_wsgi

If you don't have mod_wsgi installed yet you have to either install it using a package manager or compile it yourself.

If you are using Ubuntu/Debian you can apt-get it and activate it as follows:

```
apt-get install libapache2-mod-wsgi
```

WSGI Application

To run your application you need an app.wsgi file. Mod_wsgi is executing this file on startup to get the application object.

The a2billing_flask_app.wsgi is located at the root of this repository.

Configuring Apache

The last thing to do is to create an Apache configuration file for your application.

Apache config:

```
<VirtualHost *>
    ServerName example.com

    WSGIDaemonProcess a2billing_flask_app user=user1 group=group1 threads=5
    WSGIScriptAlias / /usr/share/a2billing-flask-api/a2billing_flask_app.wsgi
    <Directory /usr/share/a2billing-flask-api>
        WSGIProcessGroup a2billing_flask_app
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

CHAPTER 5

Resources

Extra tools

- pwiz, a model generator:

  pwiz is a little script that ships with peewee and is capable of introspecting an existing database and generating model code suitable for interacting with the underlying data. If you have a database already, pwiz can give you a nice boost by generating skeleton code with correct column affinities and foreign keys.

  Documentation: http://docs.peewee-orm.com/en/latest/peewee/playhouse.html#pwiz-a-model-generator

CHAPTER 6

Indices and tables

- genindex
- modindex
- search