Module Assembly - Topcoder NodeJS Create Billing API v1.0 Deployment Guide

Revision History

Author	Revision Number	Date
kurtrips	1.0	2014/03/02

Deployment Instructions

1. Organization of Submission

The folder structure (the root directory of the submission is referred as the <base> throughout the document)

Name	Description
tc-api/	Main application source code of this assembly. Contains only new or changed files.
tc-api/doc/Module Assembly - TopCoder NodeJS Create Billing API v1.0.doc	This file.

2. Application Setup

2.1. Preconditions

- Linux 64bit System (preferably the Ubuntu 10.04), which is used by Heroku stack for Nodejs), with development tool, such as g++, maker.
- NodeJS 0.10.21 http://nodejs.org/
- Git
- Heroku Account and Toolbelt
  - Signup a Heroku account in https://api.heroku.com/signup/devcenter with your email address.
  - Download the Heroku Toolbelt from https://toolbelt.heroku.com, and make sure the Heroku CLI is working in your environment by typing the “heroku help”.
  - Login your Heroku account by typing the “**heroku login**”.

  This requires you to input your email and password, and then finally it will upload your public SSH key to the server so that you don’t have to provide your credential during the subsequent commands.

  If you have a “Permission denied (publickey)” error during deployment, you can use “heroku keys:add” to add the public key to the remote repo as described here.

2.2. Build Script Setup

Two BASH scripts are used to ease the deployment.

- deploy/development.sh

  It helps to launch the application locally.

It simply exports the configured environment variables and then launch the Node.js application.

- deploy/heroku.sh

  It helps to deploy the application onto the Heroku cloud.

  It has command line switches.

Switch	Description
clean	It cleans the current repository information by simply removing the .git folder.
commit	It initializes the local repository and commits all the current project files, if the current directory is not a GIT repo yet; or commit all the changes to the local repository, if the current directory is already a GIT repo.
create	It creates a Heroku app for the current logged-in Heroku account. This app will only be used if it is created when the current local GIT repo has not yet setup the remote repo of name “heroku”. If your local repo has already setup “heroku” remote for an existing app, this app will not be picked as the target app for other Heroku commands. If you want to create an new Heroku app and use it as the target app in this case, you can clean and re-initialize the current local repo first, or you explicitly rebase it to the new Heroku app.
config	It exports all the configured environment variables to the target Heroku app.
push	It pushes the current committed version to the “heroku” remote.
launch	It launches one Heroku dyno to run the target app.
all	It will execute all other steps in the order: “clean”, “commit”, “create”, “config”, “push”, and then “launch”.
deploy	It only executes the steps to deploy the projects to the Heroku cloud: “config”, “push”, and then “launch”. It assumes the code has been committed to the local repo, and the local repo has setup the “heroku” remote for the target app.

3. Database setup

1. Upload the submission to /mnt/shared/temp_files/
2. Login vm as ‘tc’ account
3. cd /mnt/shared/temp_files
4. unzip submission.zip
5. su informix -l
6. cd /mnt/shared/temp_files/submission/tc-api/db_scripts

7. Add the missing column *subscription_number* to the *project*

table. This is required only if the column is missing on your VM/machine.

```
dbaccess < time_oltp_main_schema.sql
```

8. Insert the one-time setup data using the time_oltp_data_setup.sql script

```
dbaccess < time_oltp_data_setup.sql
```

9. Insert test data into DB using '**create_billing.insert**' script. Use the following command:

```
dbaccess < create_billing.insert
```

There is also a '**create_billing.delete**' to clean the data if needed.

4. Configurations

There are two BASH scripts used to ease the deployments and launching of the application.

The configurations are defined in these scripts to export the environment variables.

One of them is for local development "tc-api/deploy/development.sh", where the configurations are defined.

Please set VM_IP variable with your VM IP

```
VM_IP=50.17.156.219
```

No changes are required to tc-api/deploy/heroku.sh.

5. Deployment Instructions

1. Setup heroku as described in section 2
2. Setup database as described in section 3.
3. Run **git clone https://github.com/cloudspokes/tc-api.**

The commit to use is 67c6270d17417a3c9fe936eae5279fdf5b6b97fe

4. Copy all files from **submission/tc-api** to checked out **tc-api** and override all files
5. Replace VM_IP as described in section 4.
6. Run **npm install**
7. To deploy application in heroku environment run:
   a. chmod +x deploy/heroku.sh
   b. ./deploy/heroku.sh all
   c. After running the script, you can use **"git remote -v"** to see the "heroku" remote repo. The remote repo will be like **git@heroku.com:<app-name>.git**, you can know the create app name here.
8. To deploy application in local environment run:
   a. chmod +x deploy/development.sh
   b. . deploy/development.sh

c. **npm start**
d. Server is listening on port 8080.

6. Verification by tests

6.1.Js lint

Install the node-jslint in your local environment globally by “sudo npm install jslint -g".
Check the following files (which contain the application code) under the

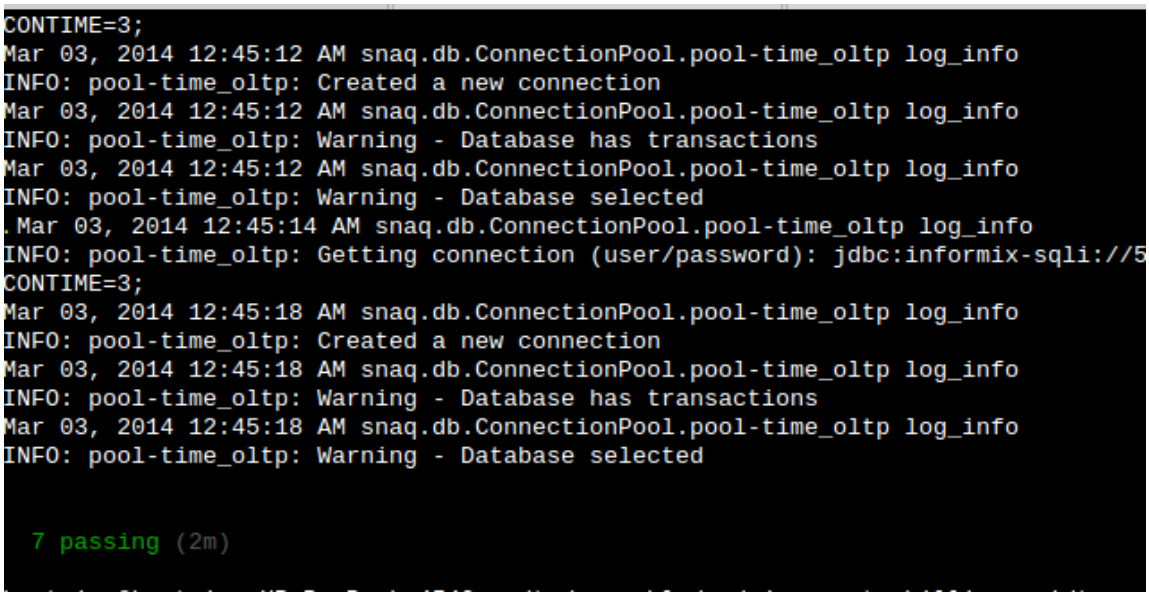
<base>/tc-api

```
jslint -nomen actions/billing.js
jslint -nomen initializers/idGenerator.js
jslint -nomen routes.js
jslint -nomen test/test.billingCreate.js
```

6.2.mocha

Run “**node_modules/.bin/mocha test/test.billingCreate.js**” from within the “<base>/tc-api/” directory.

There are 7 tests which should all pass.



7. Verification by pages

Please check apiary.apib for the updated docs.

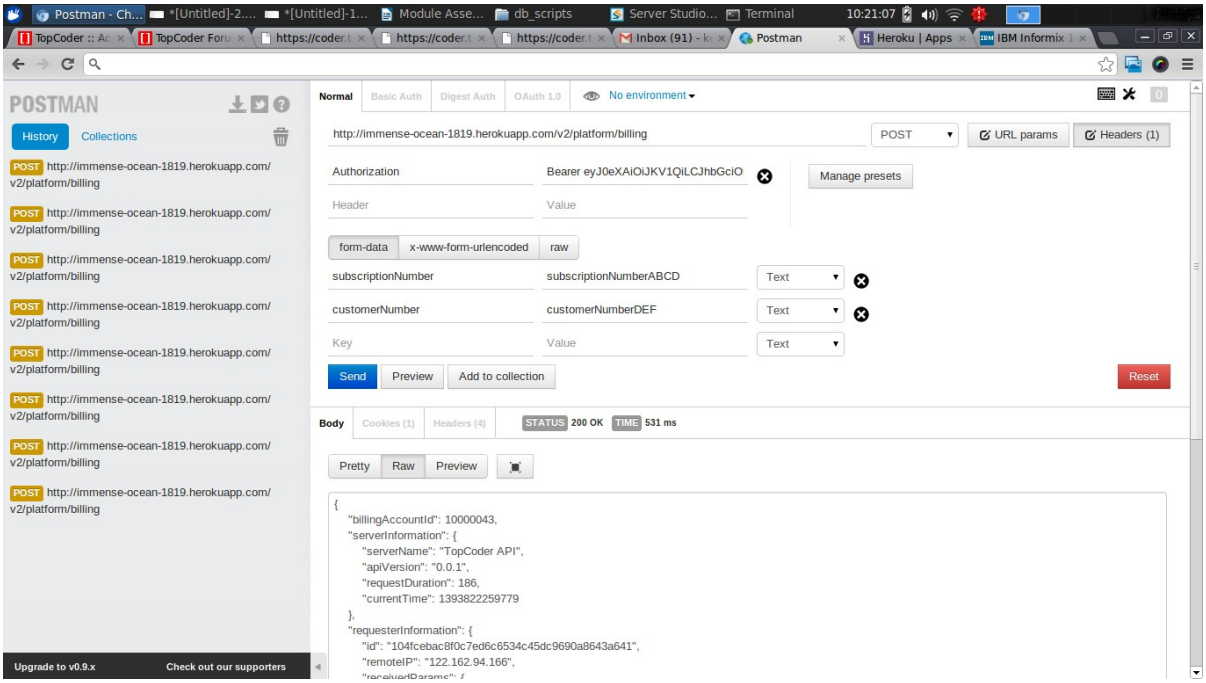
The example screenshots are taken from Google Chrome's POSTMAN client. It is used because we need to use custom Authorization header. It can be found at https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en

To generate the header, use **node test/helpers/manualJwt.js "facebook|fb132456"** This will return a token like:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJmYWNlYm9va3xmYjQwMDAxMSIsImV4cCI6MTM5MTI2MDM1MSwiYXVkIjoidG9wY29kZXIiLCJpYXQiOjEzOTEyMDAzNTF9.C4M3NVjzrMAYs2VTFX5d4chaNdIta388NhGEC3pkiOI
*The value of the Authorization header is **Bearer<space><token>***

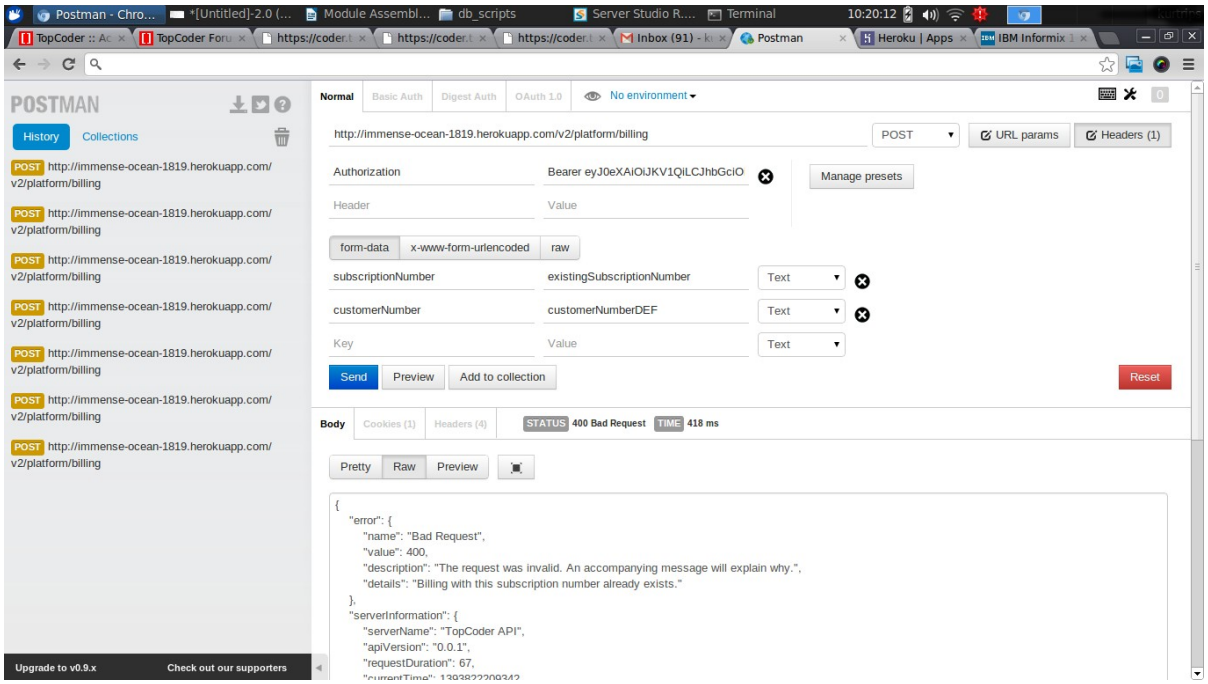
Create Billing

Example http://immense-ocean-1819.herokuapp.com/v2/platform/billing
with post parameters and authorization header properly set



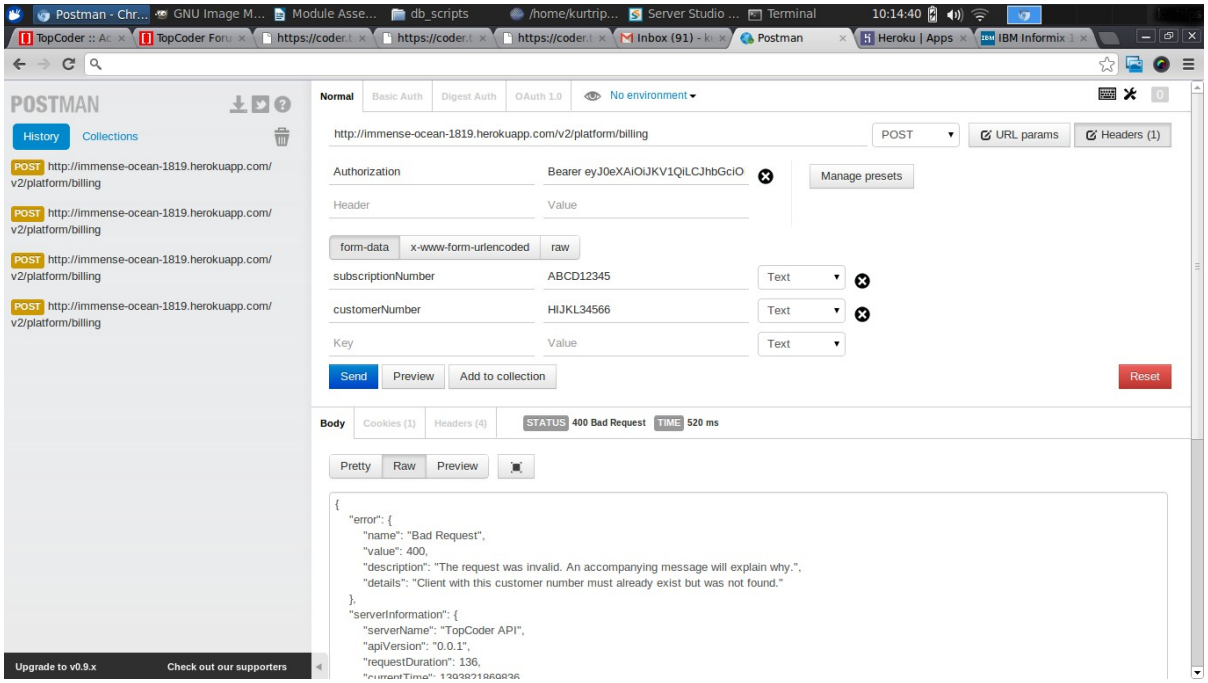
Create Billing – SubscriptionNumber already exists

Example http://immense-ocean-1819.herokuapp.com/v2/platform/billing
with same post parameters as last call and authorization header properly set



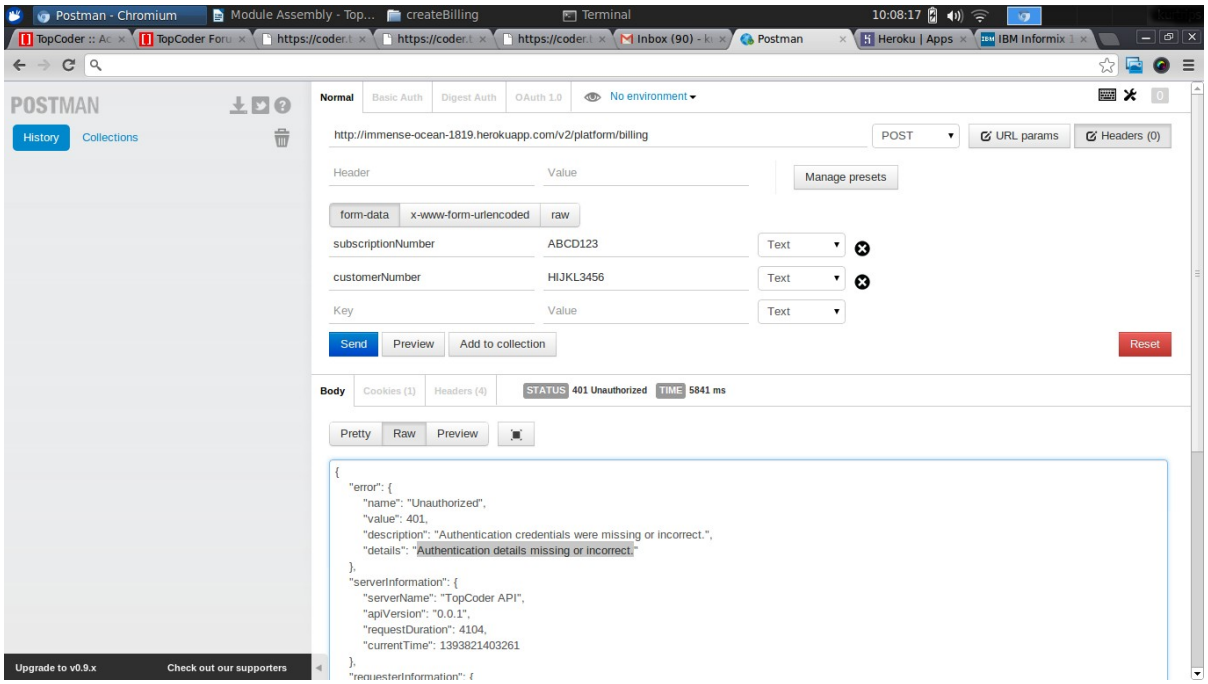
Create Billing – Client with customerNumber does not exist

Example http://immense-ocean-1819.herokuapp.com/v2/platform/billing
with post parameters set such that client with customerNumber does not exist and authorization header properly set



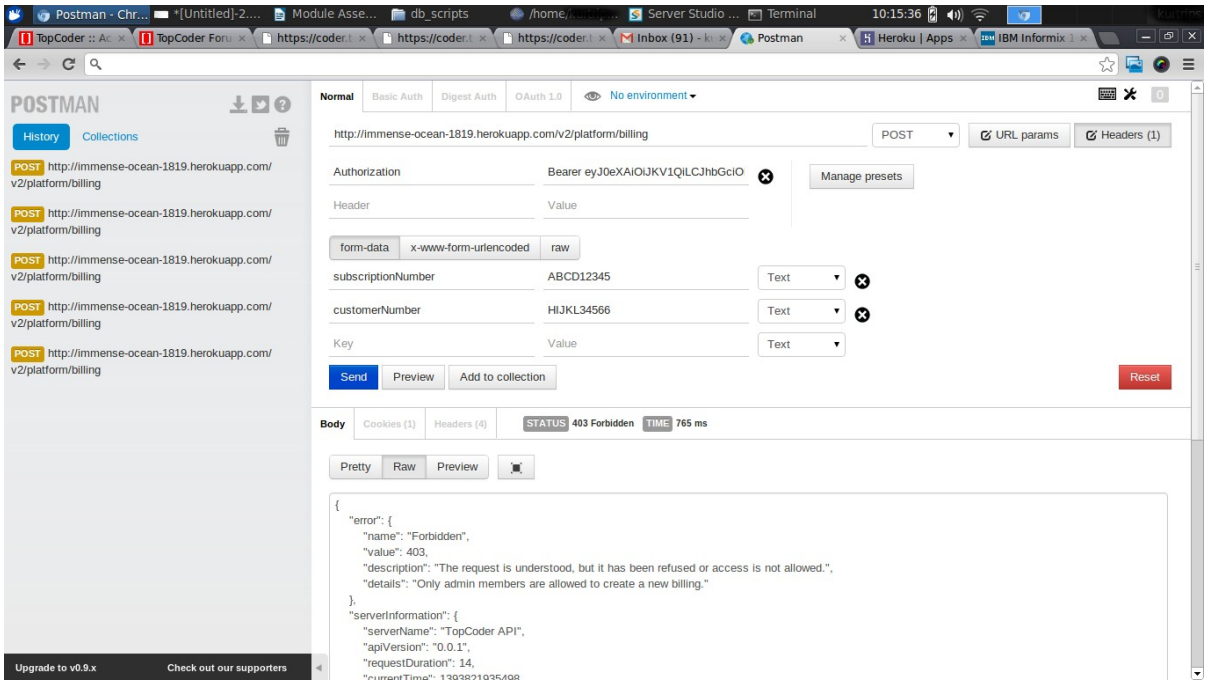
Create Billing – Not authenticated

Example http://immense-ocean-1819.herokuapp.com/v2/platform/billing
with authorization header missing



Create Billing – Not admin

Example http://immense-ocean-1819.herokuapp.com/v2/platform/billing
with post parameters and authorization header belonging to non admin user.
To generate authorization header for non-admin user use:
node test/helpers/manualJwt.js "facebook|fb124764"



8. Resource Contact List

Name	Resource Email
kurtrips	