

# Claims Management Platform (CMP) Data Management Portal (DMP) Documentation

By Carl Ngan

[Introduction](#)

[Related links](#)

[Last updated](#)

[Backlog](#)

[Usage](#)

[Authentication](#)

[Claims Manager](#)

[Create a claim](#)

[Update a claim](#)

[Delete a claim](#)

[Dealing with Errors](#)

[Developers](#)

[Running the project](#)

[Environments](#)

[Databases](#)

[Directory layout](#)

[Coding convention](#)

[Testing](#)

[Continuous Integration](#)

[Deployment](#)

[Final words](#)

## Introduction

This is the documentation for the Data Management Portal (DMP) of Claims Management Platform (CMP). This platform allows users (authorized personnel) to create, read, update, delete, and manage claims intelligently and efficiently. The purpose of the DMP is to provide a user interface for users so they do not have to directly interact with data, or the API. More concretely, users can manage claims through clicking buttons on a web page as opposed to learning how to make API calls.

## Related links

- The API - <http://api.cmp.carlNgan.com/>
- The Data Management Portal (DMP) - <http://api.dmp.carlNgan.com/>

## Last updated

March 23, 2016

## Backlog

Below are the features that would be nice to have (or things that I would do) if I were to make this a production ready product one day:

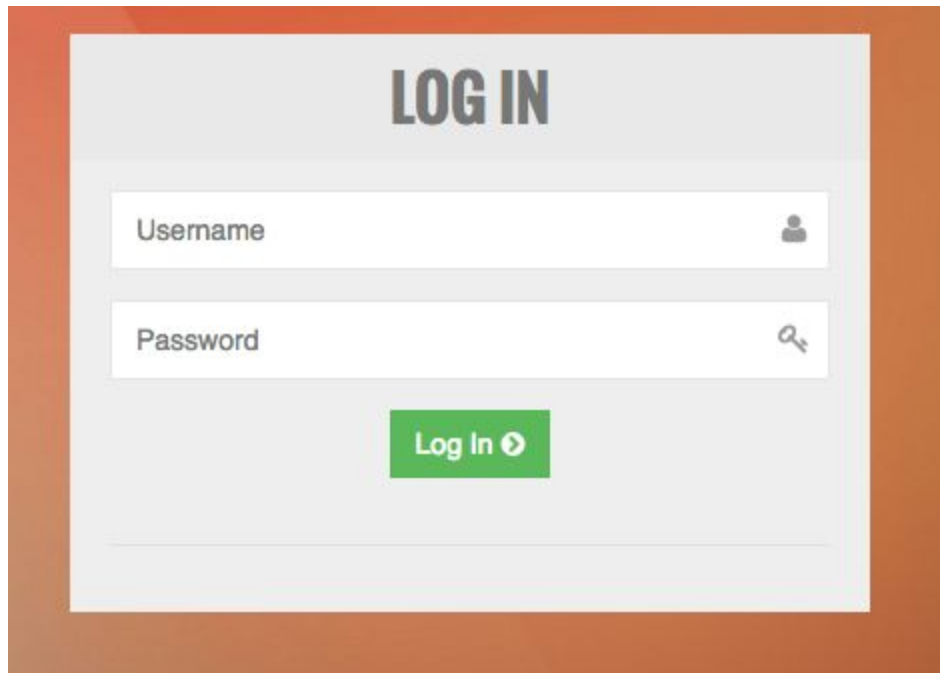
- Implement the ability to create and update claims by XML, JSON, or form. Currently we only support uploading an XML file.

## Usage

The DMP is more straightforward than the API because there is a user interface to interact with. Since this is not a production ready product, I will discuss some of our current limitations as well.

## Authentication

In order to access the portal, the DMP will need to make sure we are logged in. If not, we will be directed to the login page, and redirected back to where we were or where we wanted to go if the login is successful. If not, appropriate error messages will be displayed.



Once logged in, we will be taken directly to the Claims Manager

## Claims Manager

The Claims Manager:


The screenshot shows the "Claims Manager" interface. On the left is a sidebar with the "CMP" logo and a user profile for "Carl Ngan". The main area has a header "Claims Manager" with a "+ Create Claim" button. Below the header is a search bar "Search claim(s)..." and a section for filtering by "Loss Date Range" with "Start Date" and "End Date" fields. A table lists claims with columns: "Loss Date", "Name", "Claim Number", "Status", and "Action(s)". One claim is shown: "2014-07-10T00:19:13.631Z", "George Washington", "22c9c23bac142856018ce14a26b6c299", "OPEN". At the bottom are pagination controls: "First", "Previous", "1", "Next", "Last".

Loss Date	Name	Claim Number	Status	Action(s)
2014-07-10T00:19:13.631Z	George Washington	22c9c23bac142856018ce14a26b6c299	OPEN	

The claims manager allows us to see a grid view of the list of claims. The manager also allows us to filter and sort claims based on parameters we provide.

To search for claims, we start typing in:

Claims Manager

 Geor

Search By Loss Date Range (pref ISO String, but will automatically convert any format):

and the results should automatically populate. Currently it is only limited to searching claimants' first names.


To filter or search for claims with loss date between a date range, we can use:

Search By Loss Date Range (pref ISO String, but will automatically convert any format):

2012-01-01	to	2013-05-06
------------	----	------------

We can type in a full date, a year, year and month, year/month/day, with/without time. The system will automatically try to convert it to ISO Date Format(<https://www.w3.org/TR/NOTE-datetime>). If it really cannot, it will give you an error.

We can sort our results using:

Sort by ▾ Reset 

Loss Date (ascending)


Loss Date (descending)

**Name (ascending)**

Name (descending)

Claim Number (ascending)

Claim Number (descending)

The  button will clear all searches and filters, and bring us back to where we begun.


We can specify how many results(claims) we want in a page using:



If there are more than one pages of results, we can take advantage of the pagination tools at the bottom of the screen:



## Create a claim

To create a claim, we click on the  button from the Claims Manager and it should open up:

Create Claim

By XML

By Form

By JSON

Create Claim By XML

Upload XML

Choose File

No file chosen

XML Content

XML Content

Validate

Close

Create

We can choose to upload an XML file or copy and paste its content into the XML Content field.

It is highly recommended that we click on the 

Validate

 before 

Create

.

When we click on validate, the system will attempt to validate our xml against the provided xsd and prompt us for warnings or errors.

By XML

By Form

By JSON

## Create Claim By XML

Upload XML

 create-claim.xml

XML Content

```
<?xml version="1.0" encoding="UTF-8"?>
<cla:MitchellClaim xmlns:cla="http://www.mitchell.com/examples/claim">

  <cla:ClaimNumber>22c9c23bac142856018ce14a26b6c299</cla:ClaimNumber
>
  <cla:ClaimantFirstName>George</cla:ClaimantFirstName>
  <cla:ClaimantLastName>Washington</cla:ClaimantLastName>
  <cla:Status>OPEN</cla:Status>
  <cla:LossDate>2014-07-09T17:19:13.631-07:00</cla:LossDate>
  <cla:LossInfo>
```

CLAIMS: Error: Element '{http://www.mitchell.com/examples/claim}Vin': This element is not expected. Expected is ( {http://www.mitchell.com/examples/claim}ModelYear ). - You can continue to create for this demo purpose, but it is highly recommended to fix the xml and re-validate.



Validate

Close

Create

Create

If we try to without our xml being validated and well-formed, we will be warned with the message:

CLAIMS: Please validate your XML before trying to create -- you can force create anyway for the purpose of this demo.



Validate

Close

Create

Create Anyway

Although it is recommended to go back, check, and correct our xml, we can click on

Create Anyway

for the purpose of this demo and have the system create the claim anyway.

If the claim creation is successful, the pop up will close automatically and the claim will be added to the list of results.

## Update a claim

A claim appears as:

Loss Date	Name	Claim Number	Status	Action(s)
2014-07-10T00:19:13.631Z	George Washington	22c9c23bac142856018ce14a26b6c299	OPEN	 



To updated a claim, we click on the button and a pop up will show:



Update Claim

By XML

By Form

By JSON

Update Claim By XML

Upload XML

Choose File

No file chosen

XML Content

XML Content


Validate

Close

Update

Very similar to creating a claim, we should go through the validation procedure before clicking on **Update**.

## Delete a claim

Click on the  button to delete a claim. Before deletion, we will be prompted with a confirmation screen:

Are you sure you want to delete:

22c9c23bac142856018ce14a26b6c299 - George Washington

Delete

Cancel

## Dealing with Errors

Whether we miss validations, pass in bad values, or interact with the interface in unexpected ways, the DMP will intelligently return us errors specifying the message. The error message is usually a brief sentence saying what is wrong and we or the user should do to fix the error. The error code allows the user to lookup the error for a more detailed when our error documentation is built :)

## Developers

### Running the project

1. Navigate to a desired folder(it should be blank), or create a new folder for this project (i.e. cmp-dmp)
2. cd into the folder: ``cd cmp-dmp``
3. Clone the repo into this folder: ``git clone https://github.com/carlngan/cmp-dmp .``
4. ``npm install`` or ``sudo npm install``
5. ``bower install``
6. Make a file called ".env" -- ``vim .env``
7. Paste the following content:  
...

```
EXPRESS_SECRET=CARL
NODE_ENV=development
PORT=3002
```

...

8. ``npm start``

9. Follow the instructions to start the api here: <https://github.com/carlngan/cmp-api>

10. We can test locally by opening "localhost:3002"

Note: Make sure the API is running on localhost:3001

## Environments

Our environment needs to be defined because it is very dangerous to let the server infer. We use a different set of data for each environment. Specifically for the DMP, the database stores users sessions with the access token.

## Databases

We have three databases: one for production, one for development, and one for testing. Hence it is very important to set NODE\_ENV correctly. We use mongolab (<http://mlab.com>) for our databases. The credentials are found in ./config/database.json

## Directory layout

```
./bin/
- www //entry point to the app. Make server and call /app.js
./config/
- database.json //specify database credentials
- config.json //specify api url(s)
./modules/
- authentication/
  - public/
    - AuthenticationCtrl.js //angular file that is lazy loaded with controllers
  - routes/
    - routes.js //define routes for the auth module
  - src/
    - AuthMiddleware.js //middleware for authentication
  - views/
    - login.jade //file for the login screen
  - app.js //entry point for the authentication module
- claims/
  - public/
    - ClaimCtrl.js //angular file that is lazy loaded with controllers
  - routes/
    - templateRoutes.js //define routes for loading claim templates
  - views/ //jade files
    - modals/
      - create.jade
```

- update.jade
- delete.jade
- index.jade //template for the claims manager
- app.js //entry point for the claims module
- engine/
  - routes/
    - routes.js //define routes for the app, namely login, root, and logout
  - views/
    - layout.jade //layout for the whole admin panel
  - app.js //entry point for the engine module
- ./public/
  - css/ //css files
  - js/ //javascript files
    - angular-app.js //angular file for the project, contains all ui-states
  - images/ //public images
  - vendors/ //folder for vendor files, mostly css and js
- ./env //declare environment variables -- must have this file
- ./gitignore //specify which files and files to ignore for git
- ./app.js //entry point to the app after ./bin/www. Server level configurations should be defined here
- ./package.json //npm dependency manager
- ./README.md //project readme file

## Coding convention

I try to follow the airbnb javascript guide (<https://github.com/airbnb/javascript>). I can use ES6 features in the server-sided portion of this project and not for the client-sided code because not all browsers support ES6.

## Testing

None is written for the DMP -- though if I had more time, I would probably use Selenium (<http://www.seleniumhq.org/>) for front end testing.

## Continuous Integration

When pushed to master, Heroku (<http://heroku.com>) will automatically pick up and deploy to our production server.

## Deployment

For this project, I use Heroku (<http://heroku.com>) for deployment since it integrates well with circle and github. Personally, I use AWS (<https://aws.amazon.com/>) for my own projects.

## Final words

This was a fun project overall, though I wish I had more time to work on it. I was given the project about 2 weeks ago and had only gotten the chance to start working on it 2 days ago. Finishing up with school (finals) is not an excuse, but it did consume much more time than I had expected. Although I have not started coding until 2 days ago, I have been thinking about this project, from the end user's perspective. I think I have over engineered certain components and made it more complex than it needs to be for myself. If I had the chance to do this project again, I would really prioritize a list and do what is more important/critical first, and less on trying to design and build a production ready product.